Inferring Versatile Plans from Demonstrations by Matching Geometric Features

Anonymous Author(s)
Affiliation
Address
email

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 plans 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 generic geometric representations 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 representation greatly helps in generalizing to new configurations and that combining this representation 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, causing these approaches to often generalize poorly to human demonstrations [8]. Additionally, while humans excel at transferring learned behavior to previously unseen tasks or contexts, IL approaches often need immense amounts of data to achieve generalization [9, 8]. Such data-efficient generalization is crucial for applications that involve dynamically changing or unstructured environments. We tackle these challenges by a feature matching approach to IL, matching context-insensitive geometric representations to facilitate generalization to novel contexts. Here, we use the term context as a synonym for 'task descriptor', e.g., as goal locations to reach. More precisely, we match the features of planned trajectories with features of the long-term behavior of the expert. The resulting approach, Versatile Imitation from Geometrically Observed Representations (VIGOR), models planned trajectories using Gaussian Mixture Models (GMMs) of Probabilistic Motion Primitives (ProMPs) [10] which capture the versatility in the demonstrated behavior. We enable generalization to new contexts by using geometric descriptors instead of directly matching the agent's and expert's trajectory distributions. An example of such a descriptor would be using the distance to a target

Submitted to the 6th Conference on Robot Learning (CoRL 2022). Do not distribute.

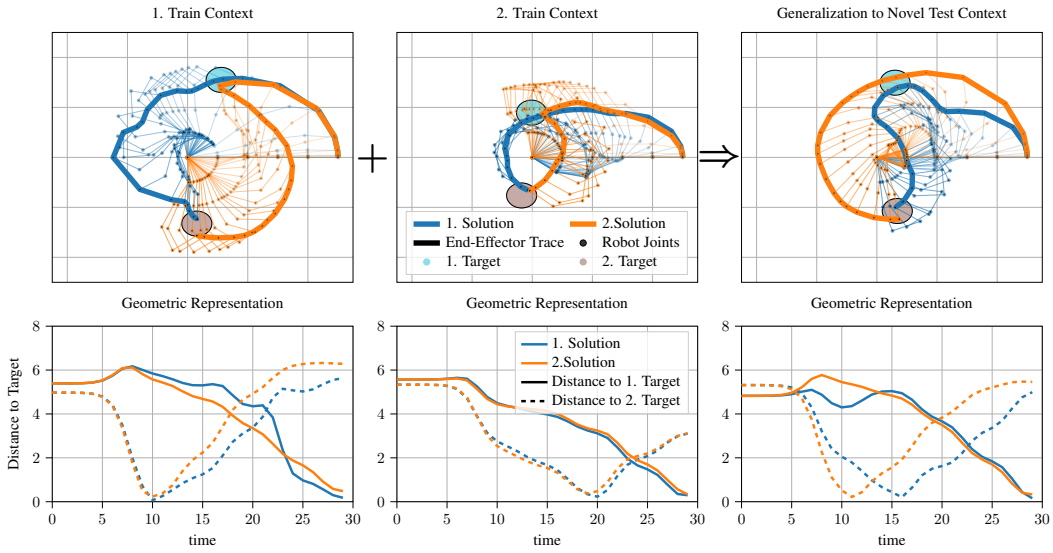

Figure 1: End-effector traces for a planar point-reaching task on 3 different contexts. (Left, Middle) Human experts solve the task by reaching the dotted grey target and ending their trajectory in the solid one. (Right) By matching context-insensitive representations of these demonstrations under a variational objective, VIGOR produces versatile behavior on unseen contexts, such as new target positions. Depicted are the component means for a Gaussian Mixture Policy with $n = 5$ components. For each context, we show the joint configurations for the first/second demonstration/sample in blue/orange.

instead of its absolute position. The exact form of this representation is task-dependent and allows the user to include domain knowledge. Figure 1 shows how this works on an exemplary reaching task. For training the GMMs, we rely on recent approaches [11, 12, 13] for distribution matching and variational inference with GMMs. For inference, we train individual GMMs for each training context and an arbitrary number of unseen test contexts. VIGOR generalizes well already from a small number of demonstrated configurations and is capable of reproducing the versatility inherent to the expert demonstrations.

We evaluate VIGOR on a suite of versatile robot coordination tasks, where demonstrations are collected from human experts through teleoperation. We compare VIGOR to standard Behavioral Cloning [14], Generative Adversarial Imitation Learning (GAIL) [5] and Inverse Reinforcement Learning (IRL) techniques [7]. We find that these methods either can not extract goal-directed behavior from the versatile demonstrations as they try to average over multiple solutions or that they can not capture the full versatility of the demonstrations. Contrary to this, VIGORextracts behavior of the highest performance that is also highly versatile. Expert demonstrations and code will be provided for the camera-ready version.

2 Related Work

Imitation Learning for Skills. A common way to represent skills over trajectories is via Movement Primitives (MPs) [15, 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) [16], both [17] and [18] fit a single GMM over 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. To alleviate this problem, recent work [19] instead learns a mixture over contextualized MPs using Gaussian Mixture Regression. This has been extended to non-linear relations between MP parameters and context [20] by using Mixture Density

Networks (MDNs) [21]. While the above approaches work for tasks with few modes, they tend to scale poorly to versatile demonstrations with a high number of modes. Moreover, optimizing an MDNs can be challenging in the case of many modes in the demonstrations. Osa et al. [22] 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. Distribution matching for IL has been used in more classical approaches [23], and recently gained popularity with the advent of Generative Adversarial Nets [24]. Starting from GAIL [5], a whole class of distribution matching based IL approaches was developed [4, 25, 26, 27, 28, 29]. These approaches commonly work in a step-based setting, minimizing some divergence between the state-action occupancy or state marginals. They generally do not consider versatile behavior, and additionally have the major disadvantage that they need to interact with the environment during training in order to generate states to match. We instead work in a trajectory-based setting, abstracting away the system's dynamics. This abstraction allows us to learn entirely offline, i.e., without any environment interactions. So far, distribution matching approaches have only found limited applicability in this setting, arguably due to the simplicity of maximum likelihood approaches for skills. Yet, as previously discussed, maximum likelihood methods often perform poorly on versatile data. As a solution to this problem, Becker et al. [13] propose using the Information-Projection [30] 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 in [31]. Here, we build on these methods, generalizing them to sequential data and geometric representations.

Further Relevant Approaches. Brown et al. [6] perform IRL from ranked demonstrations by utilizing a comparison-based loss. While originally requiring access to these rankings, Disturbance-based Reward Extrapolation (D-REX) [7] has extended this approach to instead automatically generate rankings by fitting a Behavioral Cloning (BC) policy on the demonstrations and subsequently drawing samples with different levels of noise from this policy. Here, we adapt this method to our trajectory-based setting to serve as a baseline.

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 $\tau_t = \boldsymbol{\Phi}(t)^T \boldsymbol{w}$. 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.

Variational Inference for Gaussian Mixture Models. We build on a recent class of distribution matching approaches for GMMs using the Information-Projection [30], i.e., these approaches minimize the reverse Kullback-Leibler divergence (KL) [32] $\mathrm{KL}\big(q(\boldsymbol{w})||p(\boldsymbol{w})\big)$ 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. In variational inference (VI), $p(\boldsymbol{w})$ is assumed to be given up to a normalizer. Efficient methods exist to use VI for optimizing GMMs, which allow modeling versatile behavior [13, 11]. Arenz et al. [11] introduced an upper-bound objective based on a variational decomposition. The resulting approach, Variational Inference by Policy Search (VIPS) [11, 12], repeatedly solves the objective

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

where $\hat{q}$ denotes the model from the previous iteration. This objective decomposes into individual optimizations for the components and gating distributions, which Arenz et al. [11] solve using trust-region methods from policy search [33, 34].

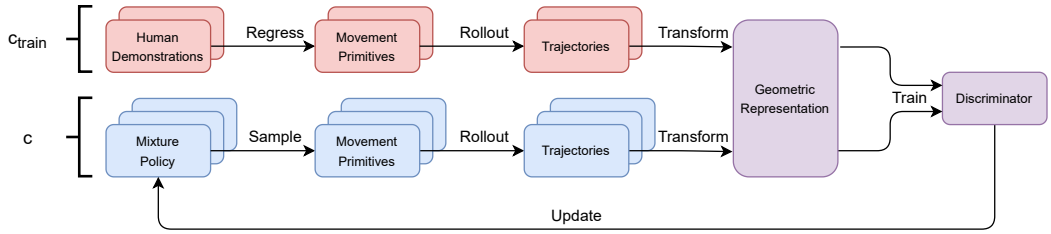

Figure 2: Schematic of VIGOR. First, we fit movement primitive parameters for each human demonstration. We use the resulting parameters to generate fixed-length desired trajectories. For every step, a context-insensitive geometric representation is computed and fed into a discriminator. A similar process is repeated for the learned policies, which are represented as mixture distributions over ProMP parameters. We then train the discriminator to distinguish between human and policy samples and use it to update the mixture policies for training and test contexts.

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. [13] propose using density ratio estimation [35] 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 [24] where the discriminator effectively also estimates a density ratio.

4 Inferring Versatile Plans by Matching Geometric Behavioral Descriptors

Our approach estimates versatile plans in form of GMM distributions over ProMP weights for all training and test configurations of the given task. To obtain a good match with the demonstrator's distribution, we utilize full-covariance GMM components. For each plan, we compute geometric features such as distances to key-points, like via-points the agent has to reach, or the edges of an object to manipulate. We directly build on EIM [13] for inferring GMMs from demonstrations by matching the agent's and demonstrator's trajectory distributions in the given 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[1] are hard to learn and we assume a relatively small amount of training contexts, 6 in all our experiments, with multiple demonstrations per train context, we resort to maintaining a single GMM-Policy $q_{\boldsymbol{c}}(\boldsymbol{w})$ for each context $\boldsymbol{c}$. 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 [11, 12].

Distribution Matching of Behavioral Descriptors We assume access to a behavioral descriptor $\boldsymbol{O} = f(\boldsymbol{w})$. For simplicity, we will assume that the mapping f between $\boldsymbol{w}$ and $\boldsymbol{O}$ is deterministic, such that we can easily evaluate the behavioral descriptor $\boldsymbol{O}$ 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 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

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

setup with the difference that the discriminator is trained using $\boldsymbol{O}$ instead of $\boldsymbol{w}$. Moreover, while we learn different distributions $q_{\boldsymbol{c}}(\boldsymbol{w})$ for each context $\boldsymbol{c}$, we use the same discriminator for all contexts. This way, we can infer distributions $q_{\boldsymbol{c}_{\text{test}}}(\boldsymbol{w})$, i.e. inferring versatile plans for unseen scenarios.

Density Ratio Estimation for Sequential Behavioral Descriptors As in our case, $\boldsymbol{w}$ represents the desired trajectory, the behavioral descriptors are typically computed per time-step, i.e., $\boldsymbol{O} = (\boldsymbol{o}_1, \dots, \boldsymbol{o}_T)^T$. To enable the classifier to deal with sequential data, we consider a sequential neural network $\boldsymbol{\phi}(\boldsymbol{O}) = \boldsymbol{\phi}((\boldsymbol{o}_1, \dots, \boldsymbol{o}_T)^T) = (y_1, \dots, y_T)^T$ that 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 depends on multiple o_j depending on the size of the receptive field of the network. 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. In this case, we can consider the summed output $h(\boldsymbol{O}) = \sum_{i=0}^{T} \phi_i(\boldsymbol{O})$ with $\phi_i(\boldsymbol{O}) = y_i$ of the sequential network for discrimination. The discriminator can therefore 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

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

w.r.t. $\boldsymbol{\phi}(\boldsymbol{O})$, recovering the log density ratio $\boldsymbol{\phi}(\boldsymbol{O}) = \log \boldsymbol{O}^{(p)} - \log \boldsymbol{O}^{(q)}$ at convergence [35]. 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

$$\text{BCE}_{\text{step}}\big(\boldsymbol{\phi}(\boldsymbol{O}), \boldsymbol{O}^{(p)}, \boldsymbol{O}^{(q)}\big) = -\mathbb{E}_{\boldsymbol{O}^{(q)}}\left[\sum_{t=0}^{T} \log\left(\sigma\left(y_t\right)\right)\right] - \mathbb{E}_{\boldsymbol{O}^{(p)}}\left[\sum_{t=0}^{T} \log\left(1 - \sigma\left(y_t\right)\right)\right] \quad (2)$$

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 $1d$ convolutional neural networks ($1d$-CNNs) work best in our experiments. We investigate this choice in Section 5.

Geometric Behavioral Descriptors VIGOR utilizes geometric descriptors that are insensitive to the task configuration to generalize to new configurations. To this end, we encode the current state along the desired trajectory with respect to geometric, context-independent features rather than absolute values. Such a descriptor space can often be straightforwardly constructed from the geometry of the task. For example, in object manipulation, the behavioral descriptor can be composed of distances of the end-effector to key-points if the object to manipulate, e.g., the corners of a box to push.

5 Experiments

All experiments use 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. It also comes with the benefit of compressing all demonstrations to a fixed length in a principled fashion. Unless noted otherwise, all experiments use 6 training and 6 test contexts. We repeat each experiment for 10 random seeds. We train a mixture with 5 components for each configuration, method and environment. Unless noted otherwise, we do *not* train the weight of these components, using a uniform weighting instead. We evaluate this mixture using a number of samples for each component of each *test* context, and then rank these components according to their performance. We then average over the samples of the *best* component for each test context, and report statistics of the resulting value over the random seeds, which allows us to obtain statistics about the best to the fifth-best learned components. We report the best component in order to avoid outlier components disrupting our evaluations. For baselines that do not use mixture policies, we instead draw samples of the trained policies for each test context and average over the performance of all of them. Appendix E shows all hyperparameters and training details.

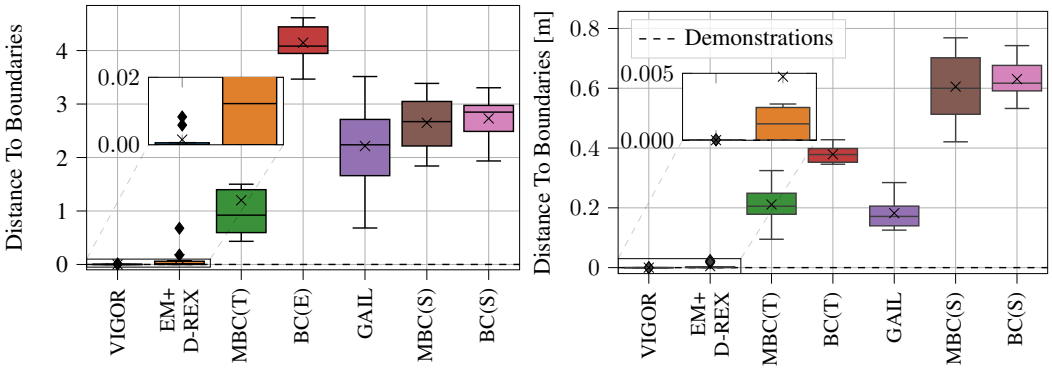

Figure 3: Mean target distance of samples from the best policy component on the Planar (left) and Panda (right) Reaching tasks for test contexts. Both VIGOR and EM+D-REX are able to achieve good performance, with VIGOR reliably hitting both targets. MBC(T) and GAIL pick up on the general task but fail at consistently reaching the targets. The other baselines fail to solve the task, presumably due to too few and too versatile demonstrations.

Baselines. We compare VIGOR to a number of IL baselines, using three trajectory-based methods, namely, standard behavioral cloning (BC(T)), behavioral cloning with a mixture model (MBC(T)), and a modified version of D-REX (EM+D-REX). Further, we also consider three state-action baselines. These are GAIL [5], as well as behavioral cloning with and without a multi-modal policy (BC(S), and MBC(S)). We refer to B for details on the baselines and to C for further details and results on all experiments described below.

Planar Reacher. We consider a planar robot with 5 Degrees of Freedom (DoF) and joints of length 1. Here, the goal is to reach an intermediate target area with the end-effector before ending the trajectory in a final target area. The left and middle panels of Figure 1 depict example demonstrations. The geometric representation for this task is 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. We explore other choices for the geometric representation in Section 5. 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.

7-D Panda Reacher. Next, we repeat the above point-reaching task using a 7 DoF Franka Emika Panda robot and 3-D intermediate and goal positions. Since we only care about the end-effector's position but not its rotation, we can ignore the last DoF, leading to an action dimension of 6. 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 representation as for the planar reacher task. The right of Figure 3 shows results on test contexts. Similar to the previous task, 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 the appendix.

Box Pusher. We also evaluate our approach on a complex 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 the trajectory, yielding versatile solutions as the box can be pushed from both inside and outside of the box. The geometric representation for this task considers 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

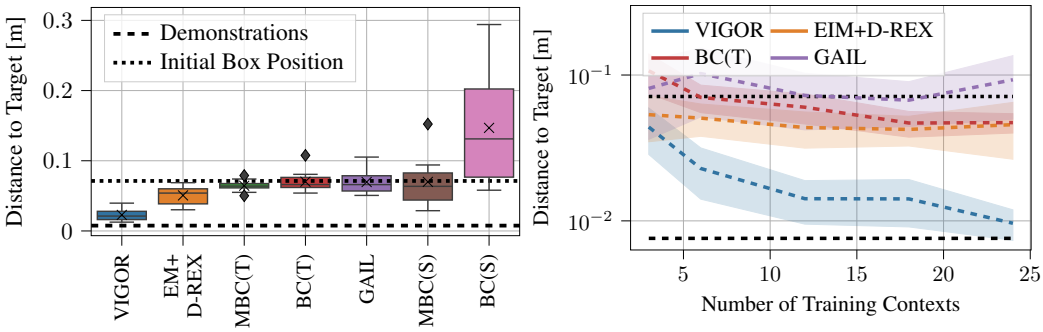

Figure 4: Mean target distance of samples from the best policy component on the Box Pushing task for test contexts. The dotted line denotes the average corner distance between the initial and the desired box position. The dashed line shows the performance of the human demonstrations. (Left) VIGOR is 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 samples, almost matching the performance of the demonstrator on unseen contexts for 24 training contexts.

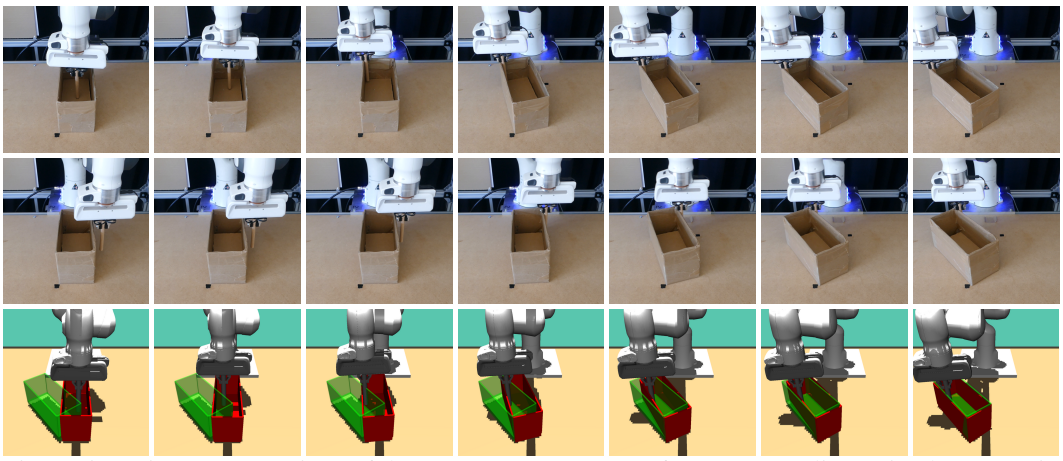

Figure 5: Trajectory executions of three component means of a VIGOR policy trained on 24 train contexts for the same test context. The first two rows show a real Franka Panda executing the planned end-effector trajectories. The last row shows the same setup in the simulation, where the target box is 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.

these distances such that the desired final box aligns with the origin. This geometric representation 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 additionally add end-effector velocity and acceleration as well as the time-step, resulting in 19 features per time-step. We use 3 demonstrations per train context. We evaluate the average distance of the corners of the final box to that of the *desired final* position. Figure 4 shows the results for our experiments on the left and the effects of varying numbers of train-contexts on the performance of VIGOR and several baselines on the right. 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 train contexts on an exemplary test context are shown in Figure 5. Videos can be found in the supplement.

Ablations. We conduct extensive ablation studies to investigate how our design choices affect the model. In Figure 6 we show some of these results for VIGOR on the Planar Reaching task. For

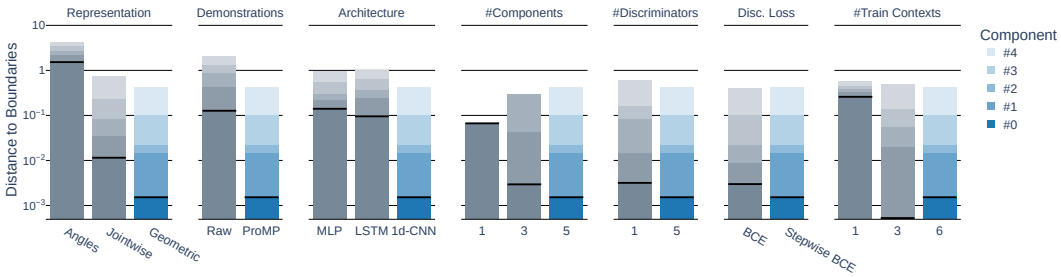

Figure 6: Ablation study for VIGOR on the Planar Reacher. Each bar is split into segments showing the ranked components of the learned mixture policies. In the main experiments, we evaluate the performance of the best component. This is highlighted by a black line. The blue bar shows the performance of VIGOR , the grey bars those of the ablations. We ablate the choice of geometric representation, trying concatenation of joint angles and context (*Angles*) as well as distances, velocities and accelerations for each robot joint (*Jointwise*). Additionally, we look at the performance of VIGOR when trained directly on sub-sampled human demonstrations (*Raw*) without fitting a ProMP on each demonstration. 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*) [36]. Further, we 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. 2), and the amount of train contexts (*#Contexts*).

more thorough explanations and extensive ablations of both VIGOR and EM+D-REX, we refer to Appendix D.

6 Limitations

Scale. We currently train and maintain a separate full-rank GMM for each training and test configuration. While we show that VIGOR can generalize well from a few configurations, it currently has linear space and time complexity w.r.t. the total number of configurations, impeding its use for settings with large or continuous numbers of contexts. One way to address this challenge would be to instead train a joint Mixture Density Network $q(\boldsymbol{w}|c)$ over all configurations, which is left for future work.

Geometric Representation. VIGOR facilitates generalization to novel task configurations by using geometric task representations. While this allows the user to include domain knowledge, it also requires a clear idea about which aspects of the task are important. One natural extension would thus be to instead extract a compact geometric representation of the task, for example by using NNs processing point clouds [37, 38].

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.

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] 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.

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

[13] 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.

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

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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

[20] 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.

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

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

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

[28] 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.

[29] 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.

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

[31] 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.

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

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

[34] 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.

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

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

[37] 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.

[38] 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.

[39] 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.

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

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.