Zhu, Shibei; Kaushik, Rituraj; Kaski, Samuel; Kyrki, Ville

Imitation-guided Multimodal Policy Generation from Behaviourally Diverse Demonstrations

Published in:
2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)

DOI:
10.1109/IROS55552.2023.10341403

Published: 01/01/2023

Document Version
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

Imitation-guided Multimodal Policy Generation from Behaviourally Diverse Demonstrations

Shibei Zhu [1], Rituraj Kaushik [1], Samuel Kaski [1,2] and Ville Kyrki [1]

***Abstract*— Learning policies from multiple demonstrators is often difficult because different individuals perform the same task differently due to hidden factors such as preferences. In the context of policy learning, this leads to multimodal policies. Existing policy learning methods often converge to a single solution mode, failing to capture the diversity in the solution space. In this paper, we introduce an imitation-guided reinforcement learning framework to solve the multimodal policy learning problem from a limited number of state-only demonstrations. Then, we propose LfBD (Learning from Behaviourally diverse Demonstration), an algorithm that builds a parameterised solution space to capture the variability in the behaviour space defined by demonstrations. To this end, we define a projection function based on the state density distributions from demonstrations to define such space. Our goal is not only to learn how to solve the task as the human demonstrator but also to extrapolate beyond the provided demonstrations. In addition, we show that with our method, we can perform a post-hoc policy search in the built solution space to recover policies that satisfy specific constraints or to find a policy that matches a given (state-only) behaviour.**

I. INTRODUCTION

In many real-world tasks such as driving a car, humans often perform the same task in different ways due to the influence of their individual preferences such as driving style [1], [2]. The underlying policies are therefore multi-modal, where each mode represents a unique behaviour. Recent works show the advantages of having a diverse set of policies, for instance, rapid damage adaptation in robotics [3], [4], [5] and safe sim-to-real policy transfer [6]. However, Reinforcement Learning (RL) methods that learn policies by maximising a given task reward typically seek to converge to either a global or local optimum, thus leading to a fixed solution mode. How to explore diverse solutions remains an open problem for the community. Possible solutions include intrinsic motivation [7], [8] that encourages visitations of unseen states or noise injection [9] for better exploration. However, these works are typically focused on learning a (novel) single-mode policy rather than a multimodal policy.

Learning from demonstrations (LfD) [10] provides an alternative way to RL by learning policies that mimic human behaviour in a supervised manner. However, with multimodal demonstrations, typical LfD methods, such as Behaviour Cloning (BC) and Generative Adversarial Imitation Learning (GAIL), either learn a policy that converges to one of the modes resulting in *mode-seeking* behaviour or exhibit *mean-seeking* behaviour by trying to average across different modes [11], [12], [13]. The former recovers only one solution mode. The latter may cause undesirable out-of-distribution behaviours that fail to satisfy the task constraints (see Fig. 1). Moreover, none of these approaches is able to learn policies that generate behaviours for a wide range of individuals.

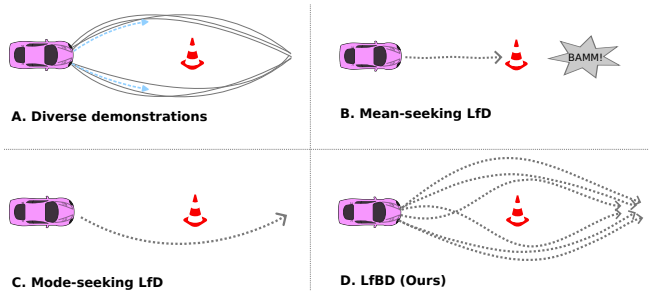

Fig. 1: Given the demonstrations from several individuals in **A**, the mean-seeking policy produces unseen behaviour that is unsafe as shown in **B**, the mode-seeking policy only recovers one mode as shown in **C**. We propose a new framework that recovers all the possible solution modes as shown in **D**. Example inspired by [11].

Unlike RL or LfD, our goal is not *how to perform a task* or *how to mimic humans*, but rather *how to perform a task in all possible ways* guided by the demonstrations as shown in Fig. 1D. Thus, we consider a setup where the task reward can be defined. However, the preference components cannot be defined mathematically to characterise multimodal policies. And we use demonstrations to solve this latter issue. Specifically, we consider Imitation Learning from Observations (ILfO) [14], [15] setup in low data regimes.

The contributions of this work are: 1) We formulate demonstration-guided multimodal policy generation as a constrained optimisation problem. 2) We propose an algorithm called Learning from Behaviourally diverse Demonstration (LfBD) to generate multimodal policies in a parameterised latent space, with each policy satisfying different constraints while being optimised for the task. 3) We propose a novel projection function that captures preferences as state-region visitations to define this parameterised latent space in an unsupervised manner. 4) We show that we can generate multimodal solutions from within and beyond the provided demonstrations. 5) We show that we can perform different types of *post-hoc* policy searches in the solution space: a) Given a (state-only) demonstration, find the closest policy capable of generating this demonstration. b) Search policies that have a high/low likelihood according to the provided demonstrations (*i.e.*, similar to the provided demonstrations).

[1]Shibei Zhu (corresponding author) and Samuel Kaski are with Department of Computer Science. Rituraj Kaushik and Ville Kyrki are with Department of Electrical Engineering and Automation, Aalto University, Finland. firstname.lastname@aalto.fi

[2] Samuel Kaski is also with Department of Computer Science, University of Manchester, United Kingdom.

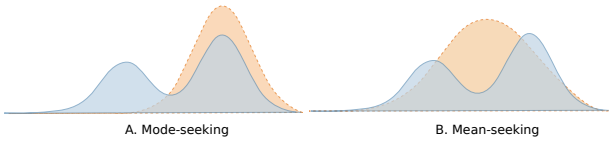

Fig. 2: Fitting a Gaussian to a multimodal distribution (blue) by minimising the reverse-KL (A, mode-seeking) and the forward-KL (B, mean-seeking)

c) Find policies that satisfy different constraints.

II. Background

A. Imitation learning as divergence minimisation

Prior works [13], [11], [12] show that Imitation Learning (IL) methods can be derived as a family of *f-divergence* minimisation methods, where a divergence of the state-action distributions of the expert $p_{\pi_{exp}}(s, a)$ and learning agent $p_{\pi_\theta}(s, a)$ is minimised, such as Kullback-Lebler (KL) divergences or Jensen-Shanon (JS) divergence. However, these divergences are not capable of dealing with multimodal distributions, as they either exhibit mean-seeking (Forward KL divergence) or mode-seeking (Reverse KL divergence and JS divergence) behaviours [12], [13], [16] (Fig. 2).

The state-of-the-art approaches are commonly based on *Generative Adversarial Networks* (GAN) [17] variants. For instance, *Adversarial Inverse Reinforcement Learning* [18] which optimises the reverse KL, and *Generative Adversarial Imitation Learning* (GAIL) [19] which optimises the JS-divergence. The generator is used as the policy network and the discriminator as a proxy of the reward function. And due to the min-max optimisation nature of GAN, it faces additional challenges from the adversarial learning such as training instability where the generator and the discriminator fail to converge optimally at the same time [20], or *mode collapse* [21], [22], where the generator collapses to produce only a small set of data samples (partial collapse) or even a single sample (complete collapse), which disregards the multi-modal nature of the distribution.

B. Imitation Learning from Observation only

Towards solving ILfO, [15] proposes a new approach called BCO, where an inverse dynamics model is learnt through interaction with the environment. The policy is learnt from state observations by using the action inferred by the inverse dynamics. However, as with most LfD methods, BCO does not handle multimodal demonstrations.

Similarly, ILPO [23] learns a forward dynamics model and a latent policy to circumvent the need for actions. However, it is limited to discrete action space. Latent policy learning from demonstrations has also been proposed by [24], [25], [26]. GAIfO [27] extends GAIL to state-only observations by using a discriminator that discriminates the state transitions instead of state-action pairs from the expert. *State-alignment based Imitation Learning* [28] uses state alignment to recover state sequences close to the demonstrations using local state alignment and global state alignment. While it shares some similarities with our method (*i.e.*, state distribution matching), it needs state-action pairs to pre-train the model.

The closest work is [12], where a multi-modal policy is recovered from state distributions using a modified reverse KL. However, this multimodality is due to the use of a stochastic policy (*i.e.*, outputs mean and variance of a Gaussian distribution). Unlike our approach, it does not allow the *post-hoc policy search* we propose. Therefore, it cannot obtain a specific policy deterministically. [29] uses a transformer to learn multimodal demonstrations; however, it does not deal with state-only data.

C. Combining RL with Demonstrations

Demonstrations have been used to overcome the difficulty of exploration and improve sample efficiency in RL [30], [31], [32], [33]. The common ways to use demonstrations are: 1) use BC to initialise the policies and/or regularise the policy loss with weighted BC loss [34], [33], [35], 2) integrate demonstrations in the reply buffer for off-policy learning [36], [31]. Our method differs from the first as we do not require pre-training nor a weighting constant for the BC loss. It differs from the second, as the demonstrations are not used during training. Instead, they are used to model the preference factors to define a parameterised solution space.

Existing approaches are interested in learning a generalisable (single-mode) policy from multi-task demonstrations or (single-mode) novel policy. Methods such as [37] that use GAIL and PPO to learn diverse visuomotor skills are essentially multi-task and therefore generate deterministic behaviours for each task. In contrast, we are interested in learning multimodal policies for the same task. We formulate the multimodal policy learning problem as a constrained optimisation problem. The existing works aim to find a single-mode solution with mode-seeking behaviour or multimodal policy with a stochastic policy. In this latter case, the stochastic policy might produce undesired results, such as the mean-seeking behaviour shown in Fig. 1. Our approach differs from these by aiming to generate a set of (deterministic) policies that satisfy different preference constraints while maximising the task reward.

III. Problem Setup

Consider a finite-horizon Markov Decision Process (MDP) $\langle \mathcal{S}, \mathcal{A}, p, p_0, R, \gamma, T \rangle$, where $\mathcal{S} \subseteq \mathbb{R}^{d_s}$ and $\mathcal{A} \subseteq \mathbb{R}^{d_a}$ are the continuous *state* and *action spaces*, $p(s'|s, a)$ is the state-transition probability, s and s' are the current and the next state, a is the applied action, p_0 the initial state distribution, $\gamma \in [0, 1]$ the *discount factor*, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ the *reward function*, and T the *time horizon*. We assume that this MDP has differentiable dynamics with a continuous states transition function $s_{t+1} = f_c(s_t, a_t)$, such that $d(s_t, s_{t+1}) < \delta$, with d as a distance metric, and δ a small constant.

Let $\mathbb{D} = \{\tau_0, \tau_1, \ldots, \tau_{n-1}\}$ be a small set of state-trajectories (demonstrations) collected from n human policies (demonstrators) $\{\pi_h^i(a|s)|i = 0 : n-1\}$, where $\tau_i = (s_0^i, s_1^i, \ldots, s_{T-1}^i), s_t^i \in \mathbb{R}^{d_s}, \tau_i \in \mathbb{R}^{T \times d_s}$. We hypothesise that while all human policies maximise the same task reward $R_{task}(s, a)$, each policy might be constrained by different

preference factors. Similar to prior methods such as *Max-margin planning* [38], we assume that different individuals prefer to visit different states while solving the task. Existing methods commonly interpreted this preference as discrete state-visitation counts. In contrast, we assume that it is defined by the state distribution. And human policies are the results of solving the following constrained optimisation problem:

$$\pi_h^i := \arg\max_{\pi} \mathbb{E}\Big[\sum_{t=0}^{T-1} \gamma^t R_{task}(s_t, a_t)|\pi\Big], \tag{1}$$

$$\text{subject to } \mathcal{D}_{KL}[p_\pi(s)||g_i(s)] < \epsilon \tag{2}$$

where $g_i(s) = p(s|\pi_h^i)$ is the state distribution preferred by the individual i, and $p_\pi(s)$ the state distribution of the learning agent. As different individuals optimise the task reward function with different preferred states, the demonstration distribution can have multiple solution modes.

Assuming that the general task reward can be defined, but the preference factors cannot be mathematically defined, *e.g.*, due to the lack of a universal mathematical form. Our goal is to learn a large set of policies capable of solving the same task in different ways without explicit hand-crafted definitions of preference factors.To this end, we use demonstrations with diverse behaviours that allow our learning agent to optimise its policies towards the unobserved (as it is not explicitly defined) preference components present within these demonstrations. This gives us the equation above that constrains the optimisation scheme of RL with the divergence.

IV. Policy generation from behaviourally diverse demonstrations

Our method consists of two steps: 1) latent space modelling from the demonstration, and 2) diverse policy generation on the latent space. In addition, our method allows us to perform *post-hoc solution searching* on this space in order to obtain solutions (policies) that satisfy different constraints.

To find out a large set of policies, one for each $g_i(\cdot)$ as per the KL constraint in Eq. 2, we first need to construct a latent space $\mathcal{Z}$, such that any $g_i(\cdot)$ can be mapped onto that space. Now, if we use a projection function that maps two similar $g_i(\cdot)$s closer to each other in the latent space $\mathcal{Z}$, the KL constraint in Eq. 2 can be approximated using a Euclidean constraint in this space $\mathcal{Z}$. More concretely, for some constant $\delta \in \mathbb{R}^+$, $z_i \in \mathcal{Z}$, and projection function $\mathrm{Enc}(\cdot)$ the problem can be reformulated on the latent space $\mathcal{Z}$ as

$$\begin{aligned} &\pi_i := \arg\max_{\pi} \mathbb{E}[(R_{task}|\pi)], \\ &\text{subject to } \; ||\,\mathrm{Enc}(\tau_\pi) - z_i||_2 < \delta \end{aligned} \tag{3}$$

Practically, we want to find one policy for each $z_i \in \mathcal{Z}$, optimising the objective in Eq. 3, where z_i are uniformly and densely distributed in the space $\mathcal{Z}$. In other words, we optimise policies for each z_i or *niche*, such that they maximise the task reward while producing state trajectories specified by its own niche only. This is a *quality-diversity optimization* problem [39]: finding high-quality solutions according to a cost/fitness function in a space that specifies the *behaviour* of the solutions. We solve this optimisation problem using a quality-diversity algorithm called MAP-Elites [40], [41], where we determine the *behaviour* of the solutions as the latent preferences extracted from our projection function.

A. Latent space modelling from density estimation

We first model the latent factors that explain the diversity in the behaviours from the demonstrations to define our projection function. As mentioned in Sec. III, we hypothesise that the diversity is caused by individual preference over the state visitations; *i.e.*, different individuals prefer visiting different regions in the state space. *Max-margin planning* [38] has proposed a similar hypothesis, where the demonstrations are results of optimising an unknown reward function that matches the state-visitation frequency for every single state and the state-action feature vector. However, it uses numerical state visitation counts for each state, which cannot handle large continuous state space. While, we approach this problem from a distribution matching perspective, where the *state region visitation frequencies*, instead of the single state visitation counts, are captured by density modelling using Gaussian Mixture Model (GMM).

Given the demonstrated state distributions $p(s)$ as defined in Sec III, we use a GMM of k mixture components to model the density as $p(s) = \sum_i^K \phi_i \mathcal{N}(s|\mu_i, \sigma_i)$, where ϕ_i is the mixture weight for each mixture component. This state density $p(s)$ models the state observations of all the demonstrations. Under the assumption of a continuous state transition,*i.e.*, states close in time will be in the same state region (see Sec III), each mixture component will cover a state region in the state space. The state distribution of each individual, g_i, belongs to a subset of the entire state distribution. As a mixture component can be seen as a state region, different trajectories would have different state region visitations or assigned mixture components. Given a fitted GMM model, we can estimate the state distribution of each $g_i(s)$ given its observations by a subset of mixture components according to their corresponding posterior probability or *responsibilities* $p(k|s)$. That is, given the sequence of states, we can estimate the responsibilities $p(k|s)$ for these states using the fitted GMM, and use the corresponding mixtures to approximate its state distribution. Given the assigned mixture components $\{k_j\}$, and their assignment frequencies $\frac{f_j}{T}$(*e.g.*, number of assignment for each component w.r.t the total number of states T) for $g_i(s)$, we can approximately define the density of $g_i(s)$ using a new GMM model composed of these mixture components as $g_i(s) \approx \sum \frac{f_j}{T} \mathcal{N}(s|\mu_{k_j}, \sigma_{k_j})$, where the assigned mixture components correspond to state region visitations, and the new mixture weight is readjusted based on their assignment frequencies $\frac{f_i}{T}$. An illustration can be found in Fig .3.

Now, we can define our projection function, or encoder function $\mathrm{Enc} : \mathbb{R}^{T \times d_s} \rightarrow \mathbb{R}^{d_z}, d_z \ll (T \times d_s)$, based on the assignment of the mixture components. Given a

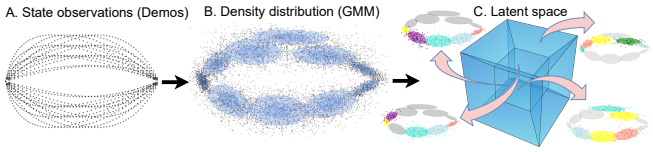

Fig. 3: Generation of the latent space from the demonstration. The state observations (A) are modelled by a GMM (B). The latent space (C) is built based on different combinations of mixture components.

set of states from a policy, the GMM is used as an unsupervised clustering method to classify these states into different mixture components. There are different ways to represent this information. For instance, we can have an encoding of size K and give the frequency values of each component $[f_1, f_2, \dots f_k]$. However, to limit the size of the encoding, we choose to take the indices of the m most frequently assigned components without explicitly referring to the actual frequencies (see Algorithm 1). The precision of the mapping depends on our choice of m as we will have information loss with a small value of m. (further discussion in Appx. VI-A.2).

Algorithm 1 State region visitation frequency

```
function ENCODE(gmm, state_sequence, size_descriptor)
    list_comp=gmm.predict(state_sequence)
    unique_comp, n_counts = unique(list_comp)
    comp_indices = top_comp(n_counts, unique_comp, size_descriptor)
    return comp_indices
end function
```

Implicitly, we are defining the feature vector of a trajectory as the mixture components responsible for its states. The visitation frequency is explicitly taken into account by taking the top d_z most assigned mixture components in descending order (see Fig. 4). In addition, the use of GMM allows us to evaluate the likelihood of a given trajectory according to the density distribution modelled from demonstrations. For instance, in Fig. 4, we can evaluate whether these trajectories belong to in-distribution data or out-of-distribution based on their likelihood (of the state sequence).

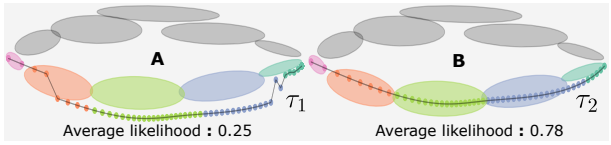

Fig. 4: Mixture components/clusters assignment (represented by **coloured** confidence ellipsoids) for the state distributions of trajectories $\tau_1 \in \mathbb{R}^{80}$ and $\tau_2 \in \mathbb{R}^{50}$. **A.** Out-of-distribution trajectory τ_1 with low average likelihood. **B.** In-distribution trajectory τ_2 with high average likelihood.

a) Solution matching: We can measure the similarity of 2 trajectories by measuring their associated mixture components. As we have the analytical form of each component (*i.e.*, Gaussian with its mean and variance), we can directly measure the distance between their distributions by using measurements as *Bhattacharyya distance* or *KL* divergence.

B. MAP-Elites (EA) as implicit conditional generative model

Given $\mathcal{Z}$ as the latent space defined by the projection function, we aim to generate a solution archive $\mathcal{P}_z$ parameterised by the latent encoding z, where each entry in this archive has a unique encoding z as shown in Eq. 3. The goal is to generate policies and map them into this space according to encoding values for their respective $g(\cdot)$. While the latent space contains all the possible state distributions defined by the mixture components of the GMM, the solution space is constrained by the system's dynamics, as the possible solutions are subject to the system constraints that cannot generate certain state distributions. Thus, $\mathcal{P}_z \subset \mathcal{Z}$. As the solution archive is the result of filling the latent space with policies, we may use latent space interchangeably to refer to the empty solution space.

EA can be seen as an implicit state density model over states with high fitness/reward [42]. In our case, the solution space is constrained by the state distribution and the generation of the policy in this space conditioned on the encoding. Thus, the use of EA can be seen as an implicit state density modelling over the solution space, conditioned on the latent encodings. In other word, our method can be regarded as a conditional generative model that generates policies with high reward. The workflow is shown in Fig. 5, and the detailed algorithm in Algorithm 2.

Algorithm 2 Solution archive generation

```
function ARCHIVE_GENERATE(gmm, size_descriptor)
    Initialise policy π, solution archive P, fitness archive R, number of iteration N, number of initial samples M
    while i ≤ M do
        θ = uni_sampling(0,1)
        π_θ = policy.set_parameter(π, θ)
        τ, r_task = simulate(π_θ)
        z = encode(gmm, τ, size_descriptor)
        P(z) ← θ, R(z) ← r_task
    end while
    while i ≤ N do
        θ = ramdom_solution_select(P)
        θ' = mutate(θ) # Gaussian noise injection
        τ, r_task = simulate(π_θ')
        z = encode(gmm, τ, size_descriptor)
        if P(z) = ∅ or R(z) < r_task then
            P(z) ← θ', R(z) ← r_task
        end if
    end while
    return P, R
end function
```

As mentioned in Sec. IV, we use MAP-Elites as the algorithm of our choice. It starts with sampling N random policy

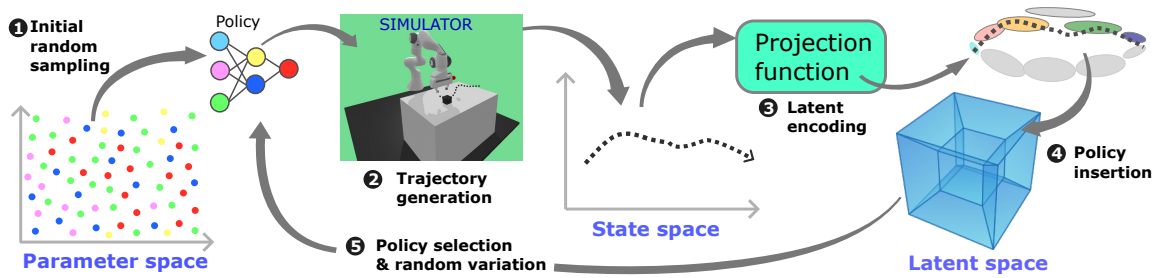

Fig. 5: Workflow of policy generation and allocation in the latent space. To initialise, the parameters of the policy (here we use a neural network) are sampled uniformly, and then the policies are executed on the simulator to get the corresponding trajectories (set of states), in order to allocate the policies into the latent space according to their encoding. Then, in a loop, policies are selected from the archive, small variations are added, and new encodings are obtained to insert them into the archive based on their reward.

parameters from a uniform distribution: $\theta_{i=0:N-1} \sim \mathcal{U}_{[a,b]}$. Then it evaluates the policies in simulation using the reward function $R_{task}(\cdot,\cdot)$, and at the same time generates the state trajectories $\tau_{i=0:N-1}$, and their corresponding encodings $z_{i=0:N-1}$. Then the policies are inserted into the closest cells in the solution archive $\mathcal{P}$. If two policies compete for the same cell, the one with a higher reward occupies the cell. Once this initialization is done, MAP-Elites randomly selects a policy from the archive, adds a small random noise (mutation) to the parameters, and evaluates the policy on the simulator for the reward, state trajectory, and the corresponding encoding. This new policy is inserted into the archive if either the cell is empty or the new policy has a higher reward; the policy is discarded otherwise. The selection, variation, and insertion continue until the maximum assigned policy evaluation count is reached.

As the highest rewarding policies occupy the nearest cells in the archive, those policies produce maximally rewarding behaviour staying close to the state distribution specified in the cell. In other words, the policies in the archive essentially optimise the objective in Eq. 3 for different z_i.

a) Post-hoc policy search: Having a set of solutions parameterised by latent encodings allows us to find solutions in a post-hoc manner that satisfy different constraints. For instance, a solution archive $\mathcal{G}$ built for a 2D navigation environment with one obstacle contains solutions for an equivalent environment $\mathcal{J}$ with two obstacles, as the latter space is more restricted, where $\mathcal{J} \subset \mathcal{G}$ (see example in Sec. V-.0.e). In addition, given a new demonstration, we can find the closest policy in our archive by finding a policy with the closest distribution distance (see Sec. IV-A.0.a).

V. Experiments

We test our method in 3 environments with continuous state and action spaces. We use 2 (motion) planning examples where the task is to generate a non-colliding path from two points end-to-end. The first is a 2D toy environment and the second the Franka Emika robot environment with Pybullet [43] as the physic simulator. Finally, we show a driving experiment modified on [44] as a close-loop control example, using neural networks for a step-wise decision-making policy. As results, we show that we can generate policies that optimise implicitly the density distribution from demonstrations. All the experiments are run 3 times. The average performance and further ablation study can be found in Appx. VI-A. The latent encoding is a vector of 6 dimensions defined by the top 6 most frequent mixture components responsible for the state sequence.

a) 2D path planning: We use 28 non-colliding trajectories with fixed lengths generated artificially using *Bézier curves* as shown in Fig. 6A. We use *B-spline* as the policy function, which takes control points $\boldsymbol{\theta} \in \mathbb{R}^6$ to generate a continuous curve as the path proposal. We use a GMM of 20 components for the multimodal distribution and 10 for the single-mode distributions. Under the exact same setup, we demonstrate that the control points are optimised towards state distributions modelled by their respective GMM. As shown in Fig. 6, the resulting control points from the single-mode setup are scattered mainly in the region within single-mode state distribution. While for the multi-modal setup, the control points are scattered in both regions of the state space. This comparison is even more noticeable after filtering out the solutions with a higher likelihood (according to their respective GMM models).

b) Comparison with VAE: VAE is commonly used as a feature encoding method as well as a (conditional) generative model, we show its capability of feature encoding and generation ability as comparison. We use the same encoding size and trained the model for 30000 iterations. As shown in Fig. 7A, for a small dataset, the generator suffers from a partial collapse that only generates samples from certain modes and fails to produce diverse samples. Moreover, the encoder fails to produce unique encoding values as shown in Fig. 7(C, D). While training the same model using the data generated from our solution, both its capability of reconstruction and encoding improves considerably (Fig. 7(C, E)). Unlike our projection function, it does not provide an interpretable encoding value (*e.g.*, closer distance in the encoding space does not guarantee a closer distance in the behaviour space), as shown in the post-hoc solution search in Fig. 10. And it cannot take advantage of the simulator that is available to interact with.

c) Franka arm manipulation: We collected 140 demonstrations using a motion tracker (MoCap) that tracks the movement of a pointer to draw the trajectories without considering the real dynamics of the robot (Fig. 8A). The length of demos spans between 240 and 726 time-steps. We changed the end coordinate for the policy optimisation in the

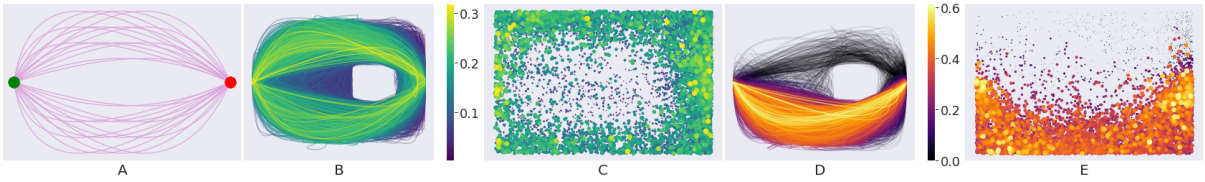

Fig. 6: Solution archives optimised for different GMM models fitted to single mode and multimodal demos. **A.** Multimodal demonstration. **B.** Trajectories with GMM from **multimodal demonstration** with colour bar as their corresponding average **likelihood**. **C.** Control points for policies from B shown with their colour and size re-adjusted according to their likelihood. **D.** Trajectories from solution space build for the **single-mode demonstrations** (*i.e.*, lower regions) with their average likelihood. **E.** Control points for the policies of Fig. D.

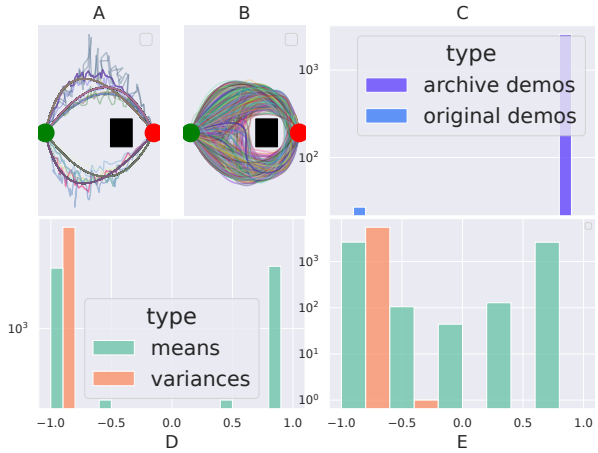

Fig. 7: VAE failed on the original demonstrations but can be fully trained using trajectories from our method. **A.** 1000 samples from VAE trained on the original demonstrations. **B.** 1000 samples from VAE trained on trajectories generated from our method. **C.** Histogram of the unique encodings from both VAEs. **D.** Histogram of the predicted encodings for the samples (*i.e.*, mean and variance for the isotropic Gaussian) from the VAE in **A**).**E** Histogram of the predicted encoding values from the VAE in **B**.

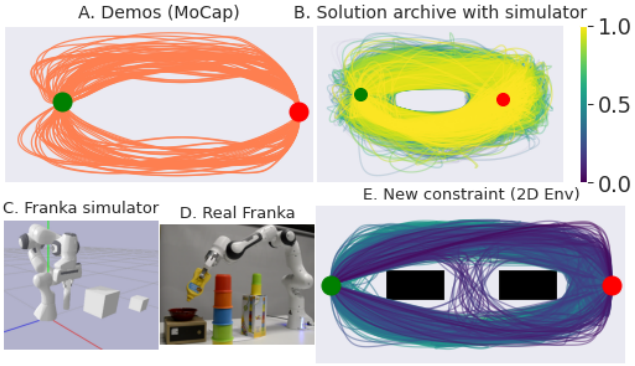

Fig. 8: **A**. Demonstrations collected using MoCap. **B**. Solution archive built by using the simulator in **C**. **D**. Trajectory that satisfies the constraint in the real robotic. **E**. Solutions that satisfy a different constraint without re-training in 2D environment.

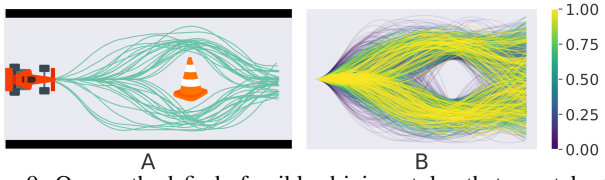

Fig. 9: Our method finds feasible driving styles that overtake the obstacle in different ways. **A.** Demonstration. **B.** Trajectories from the multimodal solution archive shown with their likelihood values.

simulator due to the range limitation of the robot. We use this to show that the solution space still optimises toward the given state distribution. We also tested the resulting trajectories in a real robot to show that the solution archive contains valid solutions with different constraints, as seen in Fig. 8D. As the demonstrations do not contain Z-axis information, we fix this as constant during training. We increase the mixture component of the GMM to 25 as more data are available. This environment contains a 3D obstacle, where the generated trajectories in the solution archive are plotted in 2D space for visualisation purposes in Fig. 8B.

d) Highway driving: We collected 40 demonstrations of 151 time-steps using a manual controller from the simulator. The observations contain both the position coordinates and the rotation angles of the vehicle, $s \in \mathbb{R}^4$. The action is the steering applied to the vehicle to overtake the obstacle, as shown in Fig. 9A. The resulting solution archive is shown in Fig. 9B which proves that our method is capable of optimising parameters of neural networks where $\boldsymbol{\theta} \in \mathbb{R}^{201}$.

e) Post-hoc solution search: Given that our archive contains a diverse range of solutions, we can still find valid solutions when the constraints in the environment change. As shown in Fig. 8E, the same solution archive in Fig.6B contains 1738 valid trajectories for a modified environment with different obstacles. Similarly, the solution archive built on the Franka simulator (Fig. 8C) contains trajectories valid for the real robot (Fig. 8D).

It is also possible to obtain a set of trajectories close to a given demonstration by matching the latent encodings (see Sec. IV-A.0.a) in the solution archive (see Fig. 10). Here, we compare the result of encoding matching using VAE trained on the trajectories generated from our method (see Fig. 7C) and the result from our proposed projection function. The results show that our projection function enables better matches, as we can measure the distances between the underlying state distributions based on the encoding values.

VI. CONCLUSIONS

In this paper, we propose a novel method to build a parameterised solution space to generate a diverse range of behaviours (*i.e.*, multimodal policies) from state-only observations in low data regimes. We show empirically that the generated policies are optimised toward different state density distributions to satisfy preference constraints. In addition, we can perform different post-hoc policy searches in the solution space to obtain the desired policies.

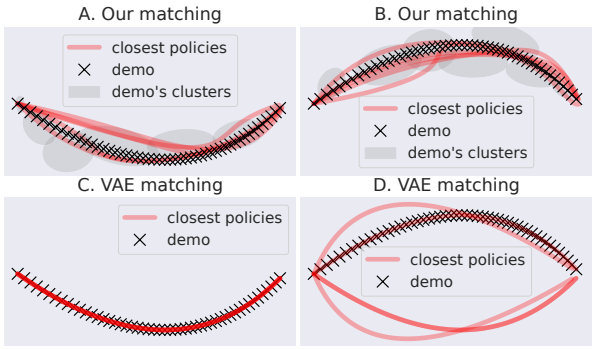

Fig. 10: **Post-hoc policy searching in the solution space to find policies for a given demonstration**. **A/B**: 5 closest policies in the solution archive using our projection function with GMM. **C/D**: 5 closest policies from samples in Fig 7B using VAE encodings.

APPENDIX

A. Numerical quantification of solution and ablation studies

For 2D path planning, we use Intel Core i7-6850K CPU @3.60GHz with 6 cores and 12 parallel threads. For Franka, we use a single thread process with 1 CPU core of Xeon Gold 6148 @2.40GHz, and 16GB of RAM without multi-threading. For the highway environment, we use 8 cores of Intel Xeon Gold 6248 @2.50GHz with 12 parallel threads and 8GB of RAM.

As the performance indicator, we show the total number of policies generated and the total valid solutions that reach the given task reward, *i.e.*, reaching the goal while preventing collisions. We report the results in Table I with 3 independent runs and record their respective means and standard deviations. We also show the average training time used to obtain these statistics in Table II.

TABLE I: Number of demonstrations vs archive solutions. $^{+}$M: multimodal, U: upper (single mode), L: lower. *Number components for the encoding / total number of mixture components.

Experiments^{+}	Demos	Encoding*	# solutions	# valid
Planning (M)	28 × 50	6/20	7703 ± 20	6584 ± 38
Planning (U)	14 × 50	6/10	2329 ± 47	1999 ± 27
Planning (L)	14× 50	6/10	3920 ± 29	3314 ± 34
Highway (M)	40 × 151	6/20	3900 ± 59	1009 ± 42
Highway (U)	20 × 151	6/10	1657 ± 33	464 ± 16
Highway (L)	20 × 151	6/10	1187 ± 14	261 ± 9

TABLE II: Training time

Experiments	# iterations	Time duration
Planning (multimodal)	10^5	3m49s ± 1m6s
Planning (upper)	10^5	2m33s ± 0m12s
Planning (lower)	10^5	2m27s ± 0m44s
Franka	10^5	6h49m ± 11m57s
Highway (multimodal)	5×10^5.	2h46m ± 4m19s
Highway (upper)	5×10^5.	1h54m ± 0m56s
Highway (lower)	5×10^5.	1h26m ± 11m57s

1) Encoding space dimensionality: Given a GMM with d_{gmm} components and an encoding size of d_z. The number of possible encodings is defined by different permutations of d_{gmm} with a theoretical upper bound of $\frac{d_{gmm}!}{(d_{gmm}-d_z)!}$. In practice, the real upper bound depends on the system's dynamics, *e.g.*, it may be impossible to have a state distribution with only 2 different state regions that are far away from each other as the state transition is continuous.

2) Comparison of different choices for GMM: The size of the latent (encoding) space depends on the number of mixture components and the size of the descriptor. Here we show the result of using 2 different GMMs with 10 and 5 mixture components respectively. As shown in Fig. 11, the size of solution archives reduces with respect to the number of mixture components.

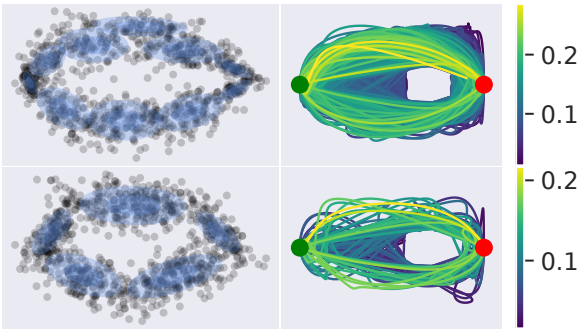

Fig. 11: Comparison of solution archives optimised with different sizes of GMM mixture components in the projection function. **Top**: 10 mixture components with 7-dimensional encoding components. **Bottom**: 5 mixture components with 4-dimensional encoding.

B. Comparison with GAN

We show the result of GAN in the 2D path planning environment in Fig. 12. As GAN cannot deal with a small dataset, we trained it using the trajectories generated by our archive (Fig. 6). The discriminator has 3 hidden layers with 128, 32, and 8 neurons respectively, and the generator has 3 hidden layers with 128, 64, and 32 neurons respectively. The model is trained for 23000 iterations. The results are consistent with mode-seeking behaviour as discussed in Sec II-A.

REFERENCES

[1] J. Fürnkranz and E. Hüllermeier, "Preference learning and ranking by pairwise comparison," in *Preference learning*. Springer, 2010, pp. 65–82.

[2] M. Babes, V. Marivate, K. Subramanian, and M. L. Littman, "Apprenticeship learning about multiple intentions," in *Proceedings of the 28th international conference on machine learning (ICML-11)*. Citeseer, 2011, pp. 897–904.

[3] R. Kaushik, P. Desreumaux, and J.-B. Mouret, "Adaptive prior selection for repertoire-based online adaptation in robotics," *Frontiers in Robotics and AI*, p. 151, 2020.

[4] K. Chatzilygeroudis, V. Vassiliades, and J.-B. Mouret, "Reset-free trial-and-error learning for robot damage recovery," *Robotics and Autonomous Systems*, vol. 100, pp. 236–250, 2018.

[5] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521, no. 7553, pp. 503–507, 2015.

[6] R. Kaushik, K. Arndt, and V. Kyrki, "Safeapt: Safe simulation-to-real robot learning using diverse policies learned in simulation," *IEEE Robotics and Automation Letters*, 2022.

[7] P.-Y. Oudeyer and F. Kaplan, "What is intrinsic motivation? a typology of computational approaches," *Frontiers in neurorobotics*, p. 6, 2009.

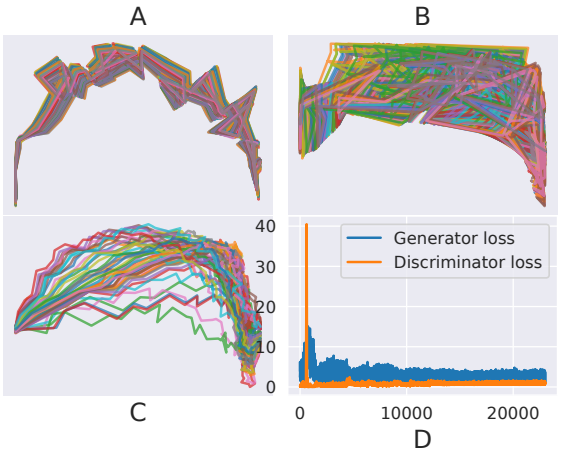

Fig. 12: **A**: Samples from GAN after 3000 iterations. **B.** Samples from GAN after 15000 iterations. **C.** Samples from GAN after training. **D.** Training loss of the discriminator and generator.

[8] N. Chentanez, A. Barto, and S. Singh, "Intrinsically motivated reinforcement learning," *Advances in neural information processing systems*, vol. 17, 2004.

[9] M. Plappert, R. Houthooft, P. Dhariwal, S. Sidor, R. Y. Chen, X. Chen, T. Asfour, P. Abbeel, and M. Andrychowicz, "Parameter space noise for exploration," in *International Conference on Learning Representations*, 2018.

[10] S. Schaal, "Learning from demonstration," *Advances in neural information processing systems*, vol. 9, 1996.

[11] L. Ke, S. Choudhury, M. Barnes, W. Sun, G. Lee, and S. Srinivasa, "Imitation learning as f-divergence minimization," in *International Workshop on the Algorithmic Foundations of Robotics*. Springer, 2020, pp. 313–329.

[12] S. K. S. Ghasemipour, R. Zemel, and S. Gu, "A divergence minimization perspective on imitation learning methods," in *Conference on Robot Learning*. PMLR, 2020, pp. 1259–1277.

[13] 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*, vol. 33, pp. 12 805–12 815, 2020.

[14] W. Sun, A. Vemula, B. Boots, and D. Bagnell, "Provably efficient imitation learning from observation alone," in *International conference on machine learning*. PMLR, 2019, pp. 6036–6045.

[15] F. Torabi, G. Warnell, and P. Stone, "Behavioral cloning from observation," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2018, pp. 4950–4957.

[16] L. Theis, A. van den Oord, and M. Bethge, "A note on the evaluation of generative models," in *International Conference on Learning Representations (ICLR 2016)*, 2016, pp. 1–10.

[17] 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*, vol. 27, 2014.

[18] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adverserial inverse reinforcement learning," in *International Conference on Learning Representations*, 2018.

[19] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Advances in neural information processing systems*, vol. 29, 2016.

[20] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[21] A. Srivastava, L. Valkov, C. Russell, M. U. Gutmann, and C. Sutton, "Veegan: Reducing mode collapse in gans using implicit variational learning," *Advances in neural information processing systems*, vol. 30, 2017.

[22] D. Bau, J.-Y. Zhu, J. Wulff, W. Peebles, H. Strobelt, B. Zhou, and A. Torralba, "Seeing what a gan cannot generate," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4502–4511.

[23] A. Edwards, H. Sahni, Y. Schroecker, and C. Isbell, "Imitating latent policies from observation," in *International conference on machine learning*. PMLR, 2019, pp. 1755–1763.

[24] A. Ren, S. Veer, and A. Majumdar, "Generalization guarantees for imitation learning," in *Conference on Robot Learning*. PMLR, 2021, pp. 1426–1442.

[25] H. Fujiishi, T. Kobayashi, and K. Sugimoto, "Safe and efficient imitation learning by clarification of experienced latent space," *Advanced Robotics*, vol. 35, no. 16, pp. 1012–1027, 2021.

[26] T. Wang, N. Karnwal, and N. Atanasov, "Latent policies for adversarial imitation learning," *arXiv preprint arXiv:2206.11299*, 2022.

[27] 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*, 2019, pp. 2229–2231.

[28] F. Liu, Z. Ling, T. Mu, and H. Su, "State alignment-based imitation learning," in *International Conference on Learning Representations*, 2019.

[29] N. M. Shafiullah, Z. Cui, A. A. Altanzaya, and L. Pinto, "Behavior transformers: Cloning k modes with one stone," *Advances in neural information processing systems*, vol. 35, pp. 22 955–22 968, 2022.

[30] K. Subramanian, C. L. Isbell Jr, and A. L. Thomaz, "Exploration from demonstration for interactive reinforcement learning," in *Proceedings of the 2016 international conference on autonomous agents & multiagent systems*, 2016, pp. 447–456.

[31] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," *arXiv preprint arXiv:1707.08817*, 2017.

[32] C. Gulcehre, T. Le Paine, B. Shahriari, M. Denil, M. Hoffman, H. Soyer, R. Tanburn, S. Kapturowski, N. Rabinowitz, D. Williams *et al.*, "Making efficient use of demonstrations to solve hard exploration problems," in *International Conference on Learning Representations*, 2019.

[33] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 6292–6299.

[34] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," *Robotics: Science and Systems XIV*, 2018.

[35] H. Zhu, A. Gupta, A. Rajeswaran, S. Levine, and V. Kumar, "Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3651–3657.

[36] K. Pertsch, Y. Lee, Y. Wu, and J. J. Lim, "Guided reinforcement learning with learned skills," in *Conference on Robot Learning*. PMLR, 2022, pp. 729–739.

[37] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas *et al.*, "Reinforcement and imitation learning for diverse visuomotor skills," *arXiv preprint arXiv:1802.09564*, 2018.

[38] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich, "Maximum margin planning," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 729–736.

[39] A. Cully and Y. Demiris, "Quality and diversity optimization: A unifying modular framework," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 2, pp. 245–259, 2017.

[40] J.-B. Mouret and J. Clune, "Illuminating search spaces by mapping elites," *arXiv preprint arXiv:1504.04909*, 2015.

[41] V. Vassiliades, K. Chatzilygeroudis, and J.-B. Mouret, "Using centroidal voronoi tessellations to scale up the multidimensional archive of phenotypic elites algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 4, pp. 623–630, 2017.

[42] K. P. Murphy, *Probabilistic machine learning: Advanced topics*. MIT press, 2022.

[43] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," 2016.

[44] E. Leurent *et al.*, "An environment for autonomous driving decision-making," 2018.