Learning Solution Manifolds for Control Problems via Energy Minimization

Miguel Zamora[1], Roi Poranne[1,2], and Stelian Coros[1]

***Abstract*— A variety of control tasks such as inverse kinematics (IK), trajectory optimization (TO), and model predictive control (MPC) are commonly formulated as energy minimization problems. Numerical solutions to such problems are well-established. However, these are often too slow to be used directly in real-time applications. The alternative is to learn solution manifolds for control problems in an offline stage. Although this distillation process can be trivially formulated as a behavioral cloning (BC) problem, our experiments highlight a number of significant shortcomings arising due to incompatible local minima, interpolation artifacts, and insufficient coverage of the state space. In this paper, we propose an alternative to BC that is efficient and numerically robust. We formulate the learning of solution manifolds as a minimization of the energy terms of a control objective integrated over the space of problems of interest. We minimize this energy integral with a novel method that combines Monte Carlo-inspired adaptive sampling strategies with the derivatives used to solve individual instances of the control task. We evaluate the performance of our formulation on a series of robotic control problems of increasing complexity, and we highlight its benefits through comparisons against traditional methods such as behavioral cloning and Dataset aggregation (Dagger).**

I. Introduction

Optimization-based control is one of the key components in the motion pipeline of some of the most advanced robots of today [1], [2], [3].

It can treat many highly nonlinear problems ranging from simple inverse kinematics to full-blown trajectory optimization. However, optimization algorithms are still generally too demanding for real-time applications. These applications require fast reaction times to input changes, and even simply changing an IK target for example, would require a slow re-optimization.

Our goal is to make optimization-based control instantaneous using machine learning. One possible approach is to first generate a dataset of solutions for different inputs using any optimization algorithm, and then train a neural network via supervised learning. In other words, optimal solutions can be *distilled* into high capacity functions approximators that can be queried in real time. This is in fact a common approach known as *behavioral cloning* (BC). However, BC is known to produce policies that generalize poorly in the presence of conflicting samples in the dataset, which manifests as an erratic approximation landscape. This is shown for a simple 2-link IK problem in Fig. 1. Conflicting samples in turn are prone to happen when there are multiple solutions for the same task, i.e. *multimodality*, and indeed, almost every IK target admits two solutions (Fig 1). As an alternative, we propose to formulate the distillation process as the minimization of the problem objective over the *set of all inputs*, simultaneously. Put differently, we minimize the *integral* of the objective function over the entire input domain. This approach does not rely on sampling solutions, and thus is more robust and results in more consistent solutions manifolds, as shown in Fig. 1.

In this paper, we focus on *trajectory-based policies* [4], that is, policies that produce the entire sequence of actions at once. This is in contrast to one-step policies that are evaluated iteratively, and generate only one action at a time. We begin by recalling first-order and second-order update rules (Newton's method) necessary to minimize the objective for a *single* problem instance. We then show how a *sample* of the inputs can be optimized in a reformulation of the problem as a BC problem, but one that can leverage the same first and second order information. We examine the performance of our approach compared to standard BC, and investigate different sampling and dataset aggregation strategies. We then tackle multimodal objectives that might generate conflicting data, by introducing detect-and-reject mechanism, which exhibits smoother solution manifolds and a better performance overall. Throughout the paper we use a 2-Link planar robot as a guiding example, and demonstrate the generality of our method in the results section.

II. Related work

Leveraging optimization-based control to train neural network policies has been of great interest for both the robotics and the learning communities [5], [6]. Behavioral cloning, an instance of imitation learning [4], is one of the simplest ways to train policies, using expert demonstrations generated by well-established optimization algorithms [7], [8]. However, its performance can be hindered by data conflicts that often occur due to multimodality (see e.g. [9], [10] and Fig. 1).

Our main focus is training trajectory policies for problems that can be formulated as one-shot decision making problems, such as simple IK or more involved kinematic trajectory optimization. We ground our formulation on gradient-based optimization of an energy function using second order information. Other gradient-based formulations to train policies are presented in [11], [12] in a differentiable simulation framework, and in [13], [6] under the umbrella

This research was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (Grant No. 866480).

[1] The authors are with the Department of Computer Science, ETH, Zurich, Switzerland. (miguel.zamora; roi.poranne; stelian.coros)@inf.ethz.ch

[2]Roi Poranne is with the Department of Computer Science, University of Haifa, Haifa, Israel. roiporanne@cs.haifa.ac.il

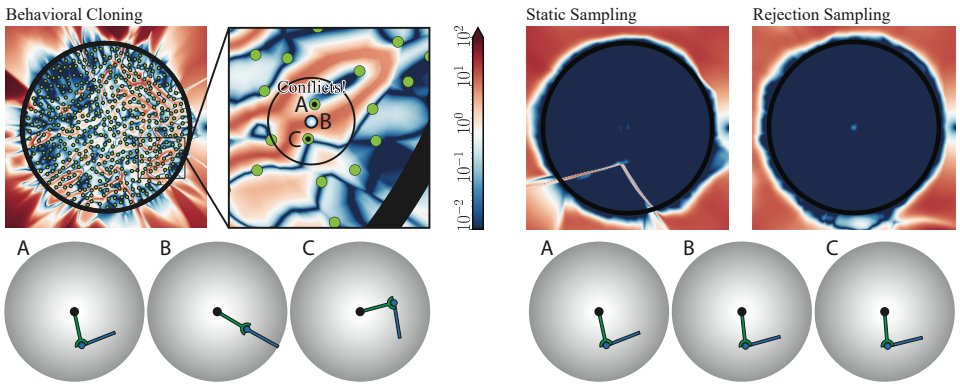


Fig. 1: Comparison of objective landscapes (Eq. 2) of a policy trained using Behavioral Cloning (BC) and our approaches. The dataset for BC was obtained by numerically solving 500 individual instances of the IK problem for a 2-Link mechanism with a maximum reach of 0.3m. The landscape was generated by evaluating the trained policy on 60k samples in a squared region of $0.3m \times 0.3m$. The black circle denotes the sampling region for training with a radius of 0.25m. The blow-up highlights regions with high energy values that are caused by conflicting samples, such as those denoted by A and C, where B is their interpolated mid-point. The bottom row shows the respective state. As can be seen, the state B indeed does not reach the target. In contrast, our sampling scheme exhibits no conflicts and generates low objectives. See the accompanying video for a visualization of the evolution of the energy landscapes for different methods as training progresses. Note that the elbow-shaped conflicting regions depend on the random initialization of the policy.

of guided policy search methods. However, their focus is on training reactive one-step policies. Other approaches that are concerned with trajectory policies include [14], [15], [16] in the context of Motion Primitives, [17] where the policy defines directly a trajectory distribution, [18] learns policies that modulate trajectory generators for robot locomotion, and [19] trains a neural network to output weighting matrices to mix different basis functions that represent full trajectories.

As mentioned, one of the challenges arise due to multi-modality. Generative Adversarial imitation learning (GAIL) [20] enables learning multimodal solutions, but [21] argues that GAIL and similar approaches still struggle with the problem of interpolating modes. The paper also suggests that learning one type of solutions should suffice for robotics applications, and we adopt this viewpoint in this paper as well. Other approaches include [22] which deals with conflicts by introducing an advantage weighted behavior model to regularize the policy, [10] by using an adaptive auxiliary cost weighting, and [21] by minimizing an f-divergence metric. Our approach is to detect and remove conflicting samples from the training set, as discussed in the following sections.

Further recent approaches that display good performance in the presence of multimodal data are Soft-Q Imitation Learning (SQIL) [23], which reformulates BC as an reinforcement learning problem by assigning rewards when the policy matches a demonstrated action in a demonstrated state, and Implicit Behavioral Cloning (IBC) [24], which we discuss in the results section. Our approach can also be interpreted as an algorithm for imitation learning using a converging supervisor [25]. However, our focus is on trajectory-polices in contrast to the reactive policies used in [25]. Finally, we share the goal of accelerating optimization-based control with [26]. But, their focus is on kinematic planning, while we apply our method to both kinematic and dynamic problems.

III. Method

In the following section we first recall the standard second-order gradient-based approach, e.g. Newton's method, used to solve individual problem instances. We then formulate the problem of learning the space of optimal solutions as the minimization of an integrated objective over an input domain, which can be approximated by a finite sum. While this problem can be optimized directly, we show that it can be solved via BC steps that use targets derived from first and second-order information (Converging supervisor). We show how these targets can enjoy standard optimization techniques such as line-search or the Gauss-Newton approximation. We conclude this section by presenting different sampling a strategies to discretize the integration domain, and introduce a method to detect and reject conflicting samples in a dataset.

A. Solving individual problem instances

Consider an optimal control problem of the form

$$\bar{\boldsymbol{a}}^{*} = \arg\min_{\bar{\boldsymbol{a}}} E(\bar{\boldsymbol{s}}(\bar{\boldsymbol{a}}, \boldsymbol{p}), \bar{\boldsymbol{a}}, \boldsymbol{p}), \tag{1}$$

where E is an energy function, $\bar{\boldsymbol{s}} := [\boldsymbol{s}_1, \ldots, \boldsymbol{s}_t]$ and $\bar{\boldsymbol{a}} := [\boldsymbol{a}_0, \ldots, \boldsymbol{a}_{t-1}]$ are the state and control trajectories, and $\boldsymbol{p}$ is a set of input parameters. The notation $\bar{\boldsymbol{s}}(\bar{\boldsymbol{a}}, \boldsymbol{p})$ indicates that the state trajectory depends on the actions and the inputs. For the simple IK problem we discuss in this section, E is defined by

$$E = \|\mathcal{F}(\bar{\boldsymbol{a}}) - \boldsymbol{p}\|^2 + w_0 \|\bar{\boldsymbol{a}} - \bar{\boldsymbol{a}}_{ref}\|^2 \tag{2}$$

where $\bar{\boldsymbol{a}}$ represents the joint angles, $\boldsymbol{p}$ is the desired end-effector position, $\mathcal{F}$ is the forward-kinematics map, and $\bar{\boldsymbol{a}}_{ref}$

serves as a regularization term with a small weight w_0. Note that $\boldsymbol{p}$ can include information about a particular initial state, a target position, or a physical property of the system. The action trajectory $\bar{\boldsymbol{a}}$ can be optimized with update rules, based on gradient descent and Newton's method:

- First order update rule

$$\bar{\boldsymbol{a}} \leftarrow \bar{\boldsymbol{a}} - \alpha_{\bar{\boldsymbol{a}}} \left[\frac{\mathrm{d}E(\bar{\boldsymbol{s}}(\bar{\boldsymbol{a}}), \bar{\boldsymbol{a}})}{\mathrm{d}\bar{\boldsymbol{a}}} \right] \tag{3}$$

- Second order update rule

$$\bar{\boldsymbol{a}} \leftarrow \bar{\boldsymbol{a}} - \alpha_{\bar{\boldsymbol{a}}} \left[\frac{\mathrm{d}^2 E(\bar{\boldsymbol{s}}(\bar{\boldsymbol{a}}), \bar{\boldsymbol{a}})}{\mathrm{d}\bar{\boldsymbol{a}}^2} \right]^{-1} \left[\frac{\mathrm{d}E(\bar{\boldsymbol{s}}(\bar{\boldsymbol{a}}), \bar{\boldsymbol{a}})}{\mathrm{d}\bar{\boldsymbol{a}}} \right] \tag{4}$$

where we neglect $\boldsymbol{p}$ for conciseness. The terms $\frac{\mathrm{d}E(\bar{\boldsymbol{s}}(\bar{\boldsymbol{a}}),\bar{\boldsymbol{a}})}{\mathrm{d}\bar{\boldsymbol{a}}}$ and $\frac{\mathrm{d}^2 E(\bar{\boldsymbol{s}}(\bar{\boldsymbol{a}}),\bar{\boldsymbol{a}})}{\mathrm{d}\bar{\boldsymbol{a}}^2}$ are the gradient and Hessian, and $\alpha_{\bar{\boldsymbol{a}}}$ is a learning rate, which can be determined using a line search procedure to ensure monotonic improvements. We note that in practice, it is preferable to use the Gauss-Newton approximation of the Hessian since it is positive definite, which guarantees a descent direction [27], [28].

B. Learning the optimal solution space

We formulate the learning process as a minimization of the integral of an energy function over a space of input parameters P as follows:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \int_P E(\bar{\boldsymbol{s}}(\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}, \boldsymbol{p}), \bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}, \boldsymbol{p}) \; d\boldsymbol{p}, \tag{5}$$

where $\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}} := [\boldsymbol{a}_0, \ldots, \boldsymbol{a}_{t-1}]$ is a neural network policy that produces a one-shot sequence of actions given a set of input parameters $\boldsymbol{p}$ and a set of weights $\boldsymbol{\theta}$. The integral can be approximated via *Monte-Carlo integration*, using a finite sum of random samples:

$$\boldsymbol{\theta}^* \approx \arg\min_{\boldsymbol{\theta}} \frac{1}{M} \sum_{m=1}^{M} E(\bar{\boldsymbol{s}}(\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}, \boldsymbol{p}^m), \bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}, \boldsymbol{p}^m). \tag{6}$$

We immediately obtain a first order update rule for the sum,

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha_{\boldsymbol{\theta}} \frac{1}{M} \sum_{m=1}^{M} \frac{\mathrm{d}E(\bar{\boldsymbol{s}}(\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}, \boldsymbol{p}^m), \bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}, \boldsymbol{p}^m)}{\mathrm{d}\boldsymbol{\theta}}. \tag{7}$$

The summands can be evaluated using the chain-rule:

$$\frac{\mathrm{d}E(\bar{\boldsymbol{s}}(\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}), \bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}, \boldsymbol{p}^m)}{\mathrm{d}\boldsymbol{\theta}} = \frac{\mathrm{d}E(\bar{\boldsymbol{s}}(\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}), \bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}, \boldsymbol{p}^m)}{\mathrm{d}\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}} \frac{\mathrm{d}\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}}{\mathrm{d}\boldsymbol{\theta}}. \tag{8}$$

We can interpret this optimization in the parameter space $\boldsymbol{\theta}$ as BC by considering the supervised learning loss

$$L = \frac{1}{M} \sum_{m=1}^{M} \frac{1}{2} || \bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}^m - \mathrm{T}^m ||^2,$$

where $\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}^m := \bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}(\boldsymbol{p}^m)$ and T^m is a target policy. An update rule for $\boldsymbol{\theta}$ that minimizes L is

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha_L \sum_{m=1}^{M} (\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}^m - \mathrm{T}^m)^T \frac{\mathrm{d}\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}^m}{\mathrm{d}\boldsymbol{\theta}}. \tag{9}$$

If we replace T^m by

$$\mathrm{T}^m = \bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}^m - \alpha_{\bar{\boldsymbol{a}}} \frac{\mathrm{d}E(\bar{\boldsymbol{s}}(\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}^m), \bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}^m)}{\mathrm{d}\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}^m} \tag{10}$$

and substitute into the supervised update rule (9) we obtain

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha_L \alpha_{\boldsymbol{a}} \frac{1}{M} \sum_{m=1}^{M} \frac{\mathrm{d}E(\boldsymbol{s}(\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}^m), \bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}^m)}{\mathrm{d}\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}^m} \frac{\mathrm{d}\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}^m}{\mathrm{d}\boldsymbol{\theta}},$$

which matches the first-order update rule for energy minimization (7) when setting α_L appropriately. Note that while it is counterproductive to apply a line-search procedure to (7), we can use line-search on $\alpha_{\bar{\boldsymbol{a}}}$ to guarantee $E(\bar{\boldsymbol{s}}(\mathrm{T}^M), \mathrm{T}^m) < E(\bar{\boldsymbol{s}}(\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}^m), \bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}^m)$, which stabilizes the training process.

Taking this idea further, we define a second-order target

$$\mathrm{T}^m = \bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}^m - \alpha_{\boldsymbol{a}} \mathbf{H}^{-1} \frac{\mathrm{d}E(\bar{\boldsymbol{s}}(\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}^m), \bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}^m)}{\mathrm{d}\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}^m} \tag{11}$$

where $\mathbf{H} = \frac{\mathrm{d}^2 E(\bar{\boldsymbol{s}}(\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}^m), \bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}^m)}{\mathrm{d}(\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}^m)^2}$ is the Hessian. Substituting the second-order targets in (9) yields the update rule

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha_L \alpha_{\boldsymbol{a}} \sum_{m=1}^{M} \frac{\mathrm{d}E(\boldsymbol{s}(\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}^m), \bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}^m)}{\mathrm{d}\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}^m} \mathbf{H}^{-1} \frac{\mathrm{d}\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}^m}{\mathrm{d}\boldsymbol{\theta}}. \tag{12}$$

This can also be viewed as an update rule that minimizes the weighted squared error loss.

$$L = \frac{1}{M} \sum_{m=1}^{M} \frac{1}{2} (\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}^m - \mathrm{T}^m)^T W^m (\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}^m - \mathrm{T}^m), \tag{13}$$

with $W^m = \mathbf{H}^{-1}$. In practice, it is better to use the Gauss-Newton approximation. Algorithm 1 presents our suggested approach together with a sampling strategy described below.

Algorithm 1 Energy Minimization

```
for k = 1 to K do
    Sample M_k input parameters p^m
    // Define converging targets
    for m = 1 to M_k do
        T^m = π̄^m_θ − α_a H^{-1} dE(s̄(π̄^m_θ), π̄^m_θ)^T / dπ̄^m_θ
    end for
    // Find non-conflicting samples
    Dataset = ((p^0, T^0), ..., (p^NC, T^NC))
    // Improvement in PS
    for n = 1 to N do
        θ ← θ − α_L Σ_{d=1}^{NC} [ (π̄^d_θ − T^d)^T dπ̄^d_θ / dθ ]^T
    end for
end for
```

C. Strategic Sampling for policy learning

Using (12) with a static set of samples from P, already exhibits better behaved energy landscapes compared to BC (Fig. 1). We attribute this to the targets being dynamic and monotonically improving. Additionally, BC cannot resolve conflicting data, which might be present during the entire training process. While conflicts may still exist in dynamic

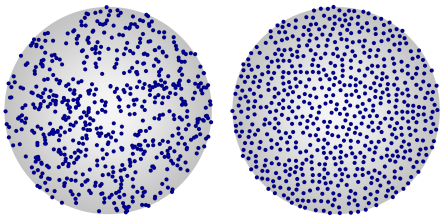


Fig. 2: **Left to right;** Uniform, and Poisson-Disk sampling

targets, they tend to vanish during optimization. Nevertheless, a fixed set of *samples* is still limited, and some conflicts might remain due to multimodality of the solution space. Indeed, the choice of samples in (6) has a significant impact in both performance and generalization. In the following we propose strategies to mitigate that.

Static Sampling. The samples $\boldsymbol{p}^m$ are selected once and are kept fixed. Instead of the common uniform or Gaussian sampling, we propose to employ Poisson-disk sampling (PDS) [29]. PDS spreads samples more evenly (Fig. 2), which results in a better coverage and easy detection of conflicting data as explained below.

Dynamic Sampling. There is in fact no reason to keep the parameter sample $\boldsymbol{p}^m$ fixed at every iteration. To the contrary, resampling allows for better coverage of the input domain, for the same computational budget. This can be viewed as SGP with batches taken from a continuous distribution.

Incremental sampling. We can also gradually increase the size of the input sample set. Specifically, we begin with a seed sample that is chosen by sampling a batch of inputs and selecting the sample that produces the lowest energy with the default weights of the policy. We then sample again increasing the number of samples, but keep only the ones that are close to the seed sample. Once a maximal number of samples is reached, we stop expending the domain.

We found that this sampling method can induce uniform solution manifolds. We attribute such behavior to a bootstraping effect that the overall training process presents. To illustrate this, consider the case of a single sample. After one iteration, the output of the policy will match the first target that leads to a lower energy region. Using a second sample that is close to the seed sample to query the policy is likely to result in a target that is non-conflicting with the target of the seed sample. In contrast to Dagger methods, we are not aggregating samples labeled with fully optimal solutions, but we grow a dataset of samples labeled with converging 2nd order targets that change dynamically at every iteration.

Rejection sampling As mentioned, interpolating two conflicting data points and their corresponding labels leads to high energy regions. We can encourage non-conflicting datasets by searching and removing samples that when averaged with their nearest neighbours result in high energy (Fig 1). Algorithm 2 describes the process, which relies on a *rejection* measure D. The measure can be the energy itself, but for interpretability, it can also be any other function that describes discrepancy between two averaged samples. For the IK examples we used the position error as a metric, which allowed us to set the threshold ϵ in an intuitive way. Furthermore, using PDS we can compute the Poisson disk radius, which denotes half the distance between the closest pair of samples, to set the search radius r in Algorithm 2 equals to twice the Poisson disk radius.

Algorithm 2 Detect and reject conflicts

Input: $Dataset = ((\boldsymbol{p}^0, \mathrm{T}^0), \ldots, (\boldsymbol{p}^M, \mathrm{T}^M))$
for $m = 0$ **to** M **do**
  Find neighbours of $\boldsymbol{p}^m$ within a radius of r
  Interpolate all neighbours to get $\boldsymbol{p}^{m_{avg}}$
  Interpolate associated targets $\mathrm{T}^{m_{avg}}$
  Evaluate and Store metric $D^m(\boldsymbol{p}^{m_{avg}}, \mathrm{T}^{m_{avg}})$
end for
Compute average $D^{m_{avg}}$
for $m = 0$ **to** M **do**
  if $D^m > D^{m_{avg}} + \epsilon$ **then**
    Reject sample m and all its neighbouring samples.
  end if
end for

Note that for the $Dataset$ in Algorithm 2, T^m is a converging target and not a final solution obtained by performing K steps of energy minimization. In practice, we apply this rejection mechanism at every iteration once the targets T^m have been updated, as noted in Algorithm 1, which result in a uniform energy landscape (Fig. 1).

To illustrate the effectiveness of the conflict detection mechanism, we applied Algorithm 2 on testing sets with different sizes, for a policy trained using our method with static sampling for the 2-Link IK problem. Figure 3 shows how even for narrow regions in the space of input parameters, we can detect conflicting data using a relatively sparse number of samples, which we consider an important feature as sampling a higher dimensional space has a sparser nature. Furthermore, notice that samples around the origin are also highlighted as conflicting samples, which is reasonable as there exist infinite solutions to the IK problem for the origin.

IV. Experimental setups

In this section we describe in more detail the series of kinematic problems that we used to evaluate our approach. Details concerning the network architectures that we used are presented in Appendix VI-B.

Planar IK for serial n-link mechanisms:
The goal for this family of problems is to learn the IK solutions of planar serial mechanisms with different number of links (Fig. 4, top row). The inputs $\boldsymbol{p}$ to the network are 2D end effector targets, and the output is a set of n joint angles $\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}(\boldsymbol{p})$. This experiment shows the performance on increasingly complex systems. Note that as the number of link n increases, so does the dimensionality of possible IK solutions, which in turn increases the chances for conflicts.

Considering the iterative nature of the optimization approach, we denote the current output of the policy as $\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}_i}(\boldsymbol{p})$.

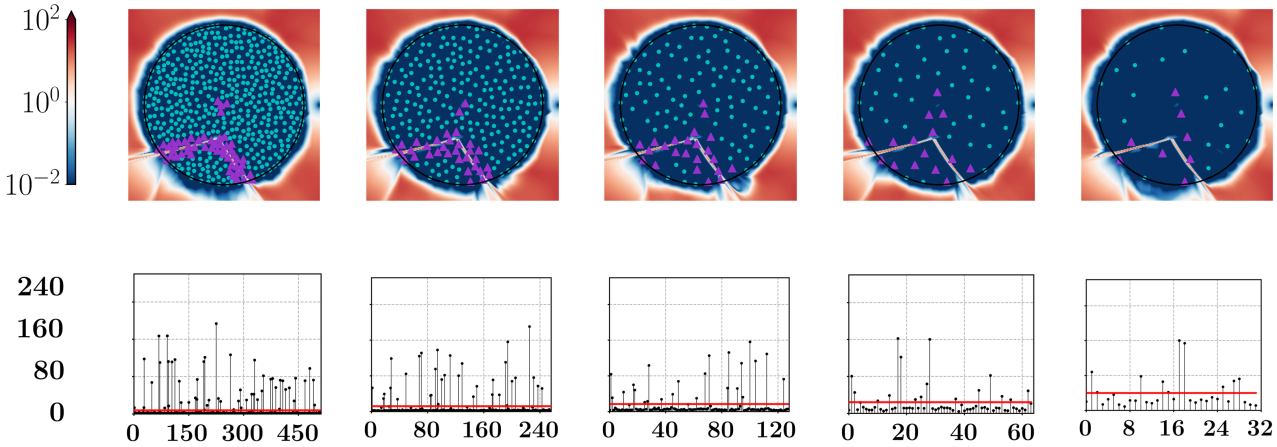


Fig. 3: **Top row**: Detecting conflicts using different testing set sizes (512, 256, 128, 64, 32) for the energy landscapes of policies trained using our method with static samples. Purple and cyan markers denote conflicting and non-conflicting samples respectively. **Bottom row**: Position error of the samples averaged with the neighbours within twice the PSD radius. Horizontal axis enumerates samples in the testing set. The red line is the average test error of the interpolated samples.

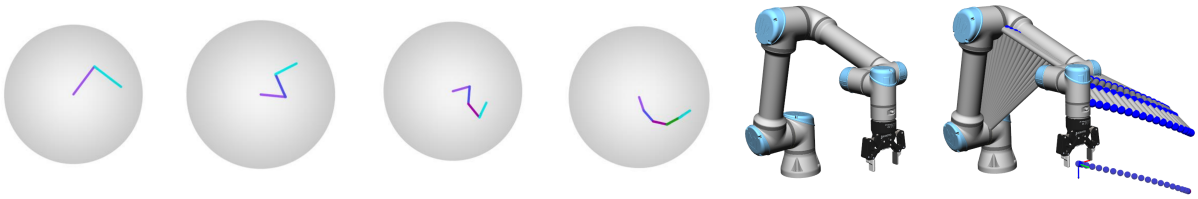


Fig. 4: **Left to right:** Planar IK for n-link serial robots $n = 2$ to 5. IK for UR5, Kinematic TO for UR5

The energy function is then:

$$E(\boldsymbol{s}, \bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}_i}(\boldsymbol{p}), \boldsymbol{p}) \;=\; w_0||\boldsymbol{s}_0 - \boldsymbol{s}_{Pos}(\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}_i}(\boldsymbol{p}))||^2 + w_1||\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}_i}(\boldsymbol{p}) - \boldsymbol{a}_{ref}||^2 \;+\; w_2||\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}_i}(\boldsymbol{p}) - \bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}_{i-1}}(\boldsymbol{p})||^2$$

where $\boldsymbol{s}_{Pos}(\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}_i}(\boldsymbol{p}))$ represents the Forward Kinematics mapping from joint angles $\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}_i}(\boldsymbol{p})$ to Cartesian coordinates $\boldsymbol{s}_{Pos}$, and $\boldsymbol{a}_{ref}$ is a reference set of joint angles.

IK for UR5 serial robot with self collision:
Here we consider a UR5 robot, with a fixed gripper orientation pointing downwards, and a collision avoidance term following [30] (see Appendix VI-A). The inputs are 3D Cartesian target and the outputs are sets of 6 joint angles. The energy function is then:

$$E(\boldsymbol{s}, \bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}_i}(\boldsymbol{p}), \boldsymbol{p}) \;=\; E_{joint\ limits}(\boldsymbol{a}_i) + E_{collision}(\boldsymbol{a}_i) + w_0||\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}_i}(\boldsymbol{p}) - \boldsymbol{a}_{ref}||^2 \;+\; w_1||\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}_i}(\boldsymbol{p}) - \bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}_{i-1}}(\boldsymbol{p})||^2 + w_2||\boldsymbol{s}_0 - \boldsymbol{s}_{Pos}(\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}_i}(\boldsymbol{p}))||^2 + w_3||\boldsymbol{s}_{Rot_0} - \boldsymbol{s}_{Rot}(\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}_i}(\boldsymbol{p}))||^2$$

Kinematic trajectory optimization:
Here we learn feasible trajectories that travel from one point to another. As above, the gripper orientation remains fixed, and we include additional objectives to ensure the smoothness and feasibility of the trajectory. In this case, the one-shot policy $\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}_i}(\boldsymbol{p})$ is defined as a sequence of joint angles $\boldsymbol{a}_j$, that is $\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}_i}(\boldsymbol{p}) = [\boldsymbol{a}_0^T \dots \boldsymbol{a}_{t-1}^T]^T$. Moreover, the input parameters $\boldsymbol{p} = [\boldsymbol{a}_{ref}^T \; \boldsymbol{s}_{Pos_t}^T]^T$ contain a set of initial joint angles $\boldsymbol{a}_{ref}^T$ that describe the initial robot configuration and serve also as a regularizer state, and $\boldsymbol{s}_{Pos_t}$ a target Cartesian position. The energy function is then:

$$E(\boldsymbol{s}, \bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}_i}(\boldsymbol{p}), \boldsymbol{p}) \;=\; E_{feasibility}(\boldsymbol{s}, \bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}_i}(\boldsymbol{p}), \boldsymbol{p}) + E_{smoothness}(\boldsymbol{s}, \bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}_i}(\boldsymbol{p}), \boldsymbol{p}) \;+\; E_{collision}(\boldsymbol{s}, \bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}_i}(\boldsymbol{p}), \boldsymbol{p}) + w_0||\boldsymbol{s}_{Pos_t} - \boldsymbol{s}_{Pos}(\boldsymbol{a}_0)||^2 + w_1||\boldsymbol{s}_{Rot_t} - \boldsymbol{s}_{Rot}(\boldsymbol{a}_0)||^2 + w_2 \sum_{j=0}^{t-1} ||\boldsymbol{a}_j - \boldsymbol{a}_{ref}||^2 \;+\; w_3||\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}_i}(\boldsymbol{p}) - \bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}_{i-1}}(\boldsymbol{p})||^2$$

The objectives $E_{feasibility}$, $E_{smoothness}$, $E_{collision}$, presented in more detail in Appendix VI-A, encourage the feasibility in terms of joint limits at the position, velocity, acceleration, and jerk levels, the smoothness of the control trajectory in terms of small velocities, accelerations and jerks, and the feasibility in terms of collisions.

V. Results

We evaluate the performance of our method with different sampling strategies against BC and Dagger. For the sake of interpretability and despite the fact that we minimize energy functions that weight different objectives, we report the average position error for the different training methods that we compare (Figure 5). We used training and testing sets

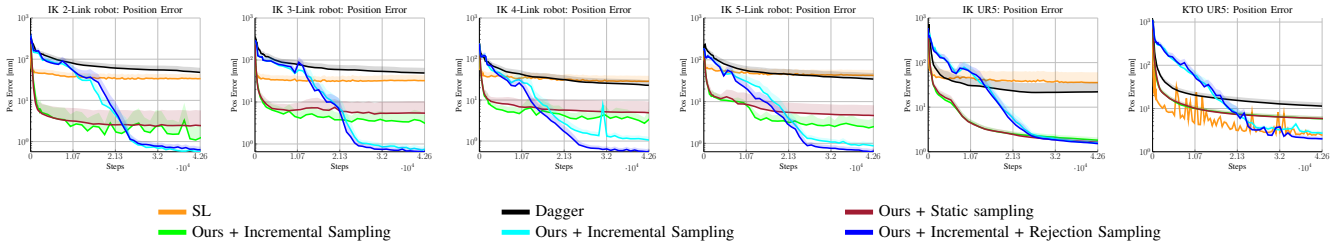
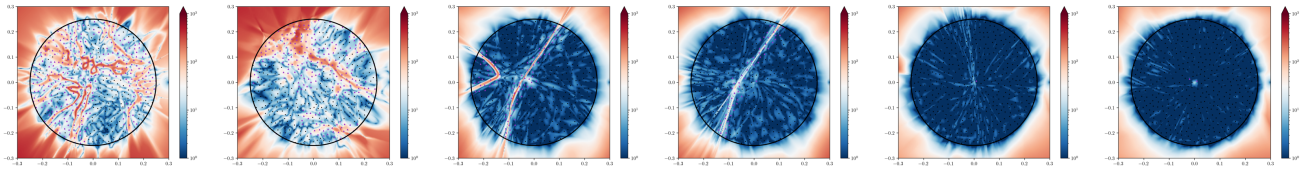


Fig. 5: Comparison of position error on a testing set for different training methods. Testing sets for the planar n-Link problems and the UR5 problems contained 500 and 2000 samples respectively. (For clarity we only plot the upper side of the shaded region that represents the standard deviation).



(a) (b) (c) (d) (e) (f)

Fig. 6: Position Error Landscape for a 5-Link planar robot. (a) SL. (b) Dagger. (c) Ours + static sampling. (d) Ours + Dynamic sampling. (e) Ours + Incremental sampling. (g) Ours + Incremental + Rejection Sampling

of 500 and 2000 samples for the planar n-Link problems, and the problems involving the UR5, respectively.

To illustrate the fairness of our comparisons, consider that in the planar IK problems, we used 500 input samples for BC (the computational cost of input samples is almost negligible), and then we applied the Gauss-Newton update rule (Eq. 4) $\mathbf{K}$ times, for each input sample, to generate a fully labeled dataset, with an associated computational budget of $\mathbf{500K}$. For our method with static or dynamic sampling, it's straightforward to see that doing $\mathbf{K}$ iterations of Algorithm 1 and given the same number of input samples, the computation budget is roughly the same as the one associated to BC. For the case of incremental sampling or incremental + rejection sampling, we keep track of the number of times that Eq. 11 is called to make sure it does not exceed the computational budget of $\mathbf{500K}$. Furthermore, while Dagger is traditionally used in the context of reactive policies, we applied it to our trajectory-based policy setting. At every Dagger iteration we get 1 new input sample using PDS, we obtain a trajectory from $\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}$ and we relabel such trajectory by applying $\mathbf{K}$ times the Gauss-Newton update rule (Eq. 4). We aggregate the relabeled trajectories and we train $\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}}$ on the the aggregated dataset. We iterate until the computational budget that was allocated for BC is reached.

The axis of Figure 5 represents only the gradient steps effectively performed on $\boldsymbol{\theta}$, as we consider it is most standard. Our method with rejection sampling shows a slower initial convergence than BC, which can be explained by the fact that BC has already access from the beginning to both a set of samples that discretize the entire domain of interest and full labels obtained via optimization based control. In contrast, the rejection sampling method includes only a small region of the space of input parameters during the early stages, while the test set spans the entire domain of integration. Furthermore, we show the corresponding position error landscape of the learned policies for the 5-Link example (Figure 6) to highlight how our method results in lower and more uniform landscapes. Note that, while for some tasks, our method with incremental MC sampling achieves uniform landscapes, using rejection sampling achieves even lower error with lower variance, in general.

A. *Warm-starting optimization*

Using the learned policies in a similar fashion to [26], and warm-starting the KTO problem for the UR5 robot (Section III-A), on a test set of 1000 input samples, yields a reduction of optimization time from 47ms to 3ms, on average, with a position and orientation error of 0.05 mm and $0.05°$, while the trained policy alone evaluated on the same test set, achieves an average error of 2.04 mm and $0.06°$.

B. *Broader class of systems*

The optimization scheme presented in the Method section is very general. When applied to kinematic trajectory optimization problems, it can be seen as direct transcription method. Furthermore, when the map $\bar{\boldsymbol{s}}(\bar{\boldsymbol{a}}, \boldsymbol{p})$ represents a dynamical system, such formulation can be seen as a direct shooting method. To exemplify such generality we applied our method to the problems shown in Fig 7. The goal of the first two examples is to drag a point mass or rigid body towards the origin over a surface with a differentiable friction model as implemented by [31]. For the point mass example the controls represent the position of the handle that is attached to the point mass via a spring. For the rigid body box, in the second example, the controls are the forces applied to the center of mass and a torque applied along the vertical axis to achieve a target orientation. The goal of the third experiment is to reorient a box that is connected to a rigid body via a flexible attachment, and the controls are the position and orientation of the smaller rigid body.

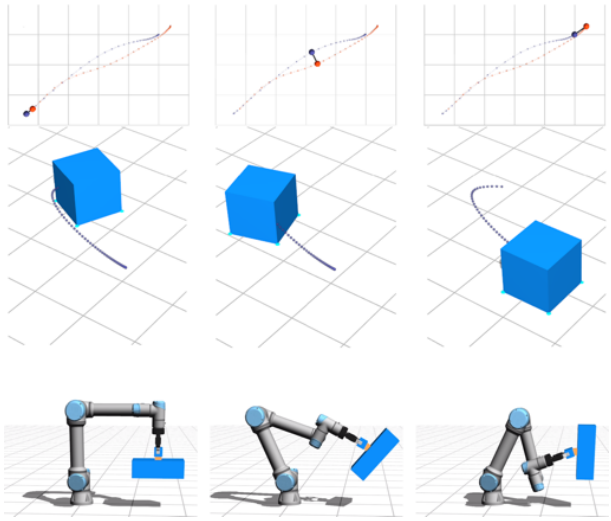


Fig. 7: **Top Row:** Dragging a point mass (purple) towards the origin. **Middle Row:** Repositioning and reorienting a rigid body box sliding over a frictional surface. **Bottom Row:** Reorienting a rigid body box with a flexible attachment.

C. Multimodal behaviour

We trained Implicit Behavioral Cloning (IBC) [24] to solve the planar n-Link IK problems. IBC used a dataset of 500 samples (500 input samples each fully labeled with a possible IK solution). We trained IBC using an InfoNCE loss with batches of 256 positive samples and 256 random uniform negative samples per positive sample. At inference time we used a derivative free optimization method with 16384 samples to optimize the energy landscape the IBC model represents. While IBC preserves multimodality, our method performs better in terms of position error and compute time as shown in table I. Additional experiments show that the performance of IBC improves with access to larger datasets.

TABLE I: Performance on a test set of 512 samples

IK	OURS		IBC	
	ERROR [MM]	TIME [MIN:SEC]	ERROR [MM]	TIME [MIN:SEC]
2-LINK	0.63	3:09	2.2	82
3-LINK	0.65	3:03	7.1	104
4-LINK	0.56	2:33	26.5	108
5-LINK	0.64	3:01	76.6	103

VI. CONCLUSION AND FUTURE WORK

We presented a gradient-based framework to learn trajectory-policies via the minimization of an energy integral over a domain of interest. We showed how to reformulate such minimization into a sequence of small BC problems by using first and second order targets. Furthermore, we investigated different sampling strategies to discretize the domain of integration and we introduced a simple mechanism to detect and reject conflicting samples. Such strategies allowed us to learn consistent solution manifolds despite the multimodality of the solution space. However, the sample strategies that worked best rely on KNN which might prevent the scalability of the approach to larger datasets. Extending our method to enable learning multiple solutions manifolds is also another exciting research direction. Further ways to leverage our mechanism to detect conflicts should be explored. An interesting direction would be to use the detection mechanism to cluster different solutions. This can enable extending our work into a multimodal setting. Furthermore, detecting conflicting areas in the domain can also be used to indicate regions that required sampling more densely.

Further sampling strategies should be explored. Extending our work to use weighted Poisson disk sampling will enable coarse and fine sampling, in different regions of the domain, depending on the nature of problem. We believe better rejections mechanisms should be explored. In this work, we reject all the samples associated to a conflict. Such rejection rule, however, might be too conservative. Finally, we believe this work, with all its exciting future venues of research, sets a good starting point to enable robust and instantaneous queries from well-stablished optimization-based control algorithms.

APPENDIX

A. Additional objectives

Collision

Following the approach presented in [30], we define a set of collision primitives using spheres and capsules, as shown in figure 8. The energy is then defined using unilateral barrier function [32] to ensure that the minimum distance between collision primitives is larger than a safety threshold.

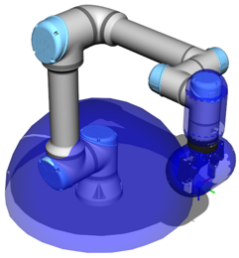


Fig. 8: Collision primitives

Feasibility

$$E_{feasibility}(\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}_i}(\boldsymbol{p}), \boldsymbol{p}) = E_{joint\ pos\ limits}(\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}_i}(\boldsymbol{p})) + E_{joint\ vel\ limits}(\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}_i}(\boldsymbol{p})) + E_{joint\ acc\ limits}(\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}_i}(\boldsymbol{p})) + E_{joint\ jerk\ limits}(\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}_i}(\boldsymbol{p}))$$

The energy associated to joint limits is implemented using bilateral barrier functions as presented in [32].

Smoothness

$$E_{smoothness}(\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}_i}(\boldsymbol{p}), \boldsymbol{p}) = E_{small\ vel}(\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}_i}(\boldsymbol{p}), \boldsymbol{p}) + E_{small\ acc}(\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}_i}(\boldsymbol{p}), \boldsymbol{p}) + E_{small\ jerk}(\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}_i}(\boldsymbol{p}), \boldsymbol{p})$$

where each individual term is simply the squared norm of the corresponding quantity computed using a second order backward finite difference [33] and assuming the robot stands still in its initial configuration $\boldsymbol{a}_{ref}$ e.g.

$$E_{small\ vel}(\bar{\boldsymbol{\pi}}_{\boldsymbol{\theta}_i}(\boldsymbol{p}), \boldsymbol{p}) = ||\frac{3}{2}\boldsymbol{a}_0 - 2\boldsymbol{a}_{ref} + \frac{1}{2}\boldsymbol{a}_{ref}||^2 + ||\frac{3}{2}\boldsymbol{a}_1 - 2\boldsymbol{a}_0 + \frac{1}{2}\boldsymbol{a}_{ref}||^2 + \sum_{j=2}^{t-1} ||\frac{3}{2}\boldsymbol{a}_j - 2\boldsymbol{a}_{j-1} + \frac{1}{2}\boldsymbol{a}_{j-2}||^2$$

B. Network Architecture

For the experiments described in this paper, we used two hidden layers with relu activation functions and 512 nodes, and an output layer with tanh activations.

As we deal with periodic variables for the joint angles of a robot, we encoded the output of the network such that each each angle θ_i is represented by tuple $(\sin\theta_i, \cos\theta_i)$. As done by [34]. A theoretical justification can be found in [35] (Pages 105-110). We found such encoding to alleviate some of the problems related to conflicting data, and to improve performance as well. Table II summarizes the dimensionality of the problems described in section IV. Note that because of the encoding we used, the output dimension doubles.

TABLE II: Dimensionality of problems.

PROBLEM	INPUT DIM	OUTPUT DIM
IK N-LINK MECHANISM	2	2*N
IK UR5	3	2*6
KTO UR5	9	2*6*30 = 180

ACKNOWLEDGMENT

We thank Vittorio Megaro for the initial explorations of energy minimization, Ramon Witschi for the plotting tools, and Kiran Doshi for the help running comparisons.

REFERENCES

[1] V. S. Medeiros, E. Jelavic, M. Bjelonic, R. Siegwart, M. A. Meggiolaro, and M. Hutter, "Trajectory optimization for wheeled-legged quadrupedal robots driving in challenging terrain," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4172–4179, 2020.

[2] S. Zimmermann, R. Poranne, and S. Coros, "Go fetch! - dynamic grasps using boston dynamics spot with external robotic arm," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 4488–4494.

[3] S. Dafarra, S. Bertrand, R. J. Griffin, G. Metta, D. Pucci, and J. E. Pratt, "Non-linear trajectory optimization for large step-ups: Application to the humanoid robot atlas," *CoRR*, vol. abs/2004.12083, 2020.

[4] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, and J. Peters, "An algorithmic perspective on imitation learning," *Foundations and Trends in Robotics*, vol. 7, no. 1-2, p. 1–179, 2018.

[5] I. Mordatch and E. Todorov, "Combining the benefits of function approximation and trajectory optimization," 07 2014.

[6] W. Montgomery and S. Levine, "Guided policy search as approximate mirror descent," 2016.

[7] J. Song, W. Jiang, A. Yazdanbakhsh, E. Songhori, A. D. Goldie, N. Jaitly, and A. Mirhoseini, "Efficient imitation learning with local trajectory optimization," 2020.

[8] G. Kahn, T. Zhang, S. Levine, and P. Abbeel, "Plato: Policy learning using adaptive trajectory optimization," 2017.

[9] D. Gaddy, A. Kouzemtchenko, P. K. Reddy, P. Kolhar, and R. Shah, "Overcoming conflicting data for model updates," *CoRR*, vol. abs/2010.12675, 2020.

[10] C. Pinneri, S. Sawant, S. Blaes, and G. Martius, "Extracting strong policies for robotics tasks from zero-order trajectory optimizers," in *International Conference on Learning Representations*, 2021.

[11] L. Fussell, K. Bergamin, and D. Holden, "Supertrack: Motion tracking for physically simulated characters using supervised learning," *ACM Trans. Graph.*, vol. 40, no. 6, dec 2021.

[12] M. A. Z. Mora, M. Peychev, S. Ha, M. Vechev, and S. Coros, "Pods: Policy optimization via differentiable simulation," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 7805–7817. [Online]. Available: http://proceedings.mlr.press/v139/mora21a.html

[13] S. Levine and V. Koltun, "Learning complex neural network policies with trajectory optimization," in *ICML '14: Proceedings of the 31st International Conference on Machine Learning*, 2014.

[14] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, "Learning movement primitives," in *11th International Symposium on Robotics Research (ISRR2003)*. Springer, 2004, pp. 561–572, clmc.

[15] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Computation*, vol. 25, no. 2, pp. 328–373, 2013.

[16] G. Maeda, G. Neumann, M. Ewerton, R. Lioutikov, O. Kroemer, and J. Peters, "Probabilistic movement primitives for coordination of multiple human-robot collaborative tasks," *Autonomous Robots*, vol. 41, no. 3, pp. 593 – 612, March 2017.

[17] S. Choi and J. Kim, "Trajectory-based probabilistic policy gradient for learning locomotion behaviors," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 1–7.

[18] A. Iscen, K. Caluwaerts, J. Tan, T. Zhang, E. Coumans, V. Sindhwani, and V. Vanhoucke, "Policies modulating trajectory generators," 2019.

[19] T. Osa, "Motion planning by learning the solution manifold in trajectory optimization," *CoRR*, vol. abs/2107.05842, 2021.

[20] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016.

[21] L. Ke, S. Choudhury, M. Barnes, W. Sun, G. Lee, and S. Srinivasa, "Imitation learning as f-divergence minimization," 2020.

[22] N. Siegel, J. T. Springenberg, F. Berkenkamp, A. Abdolmaleki, M. Neunert, T. Lampe, R. Hafner, N. Heess, and M. Riedmiller, "Keep doing what worked: Behavior modelling priors for offline reinforcement learning," in *International Conference on Learning Representations*, 2020.

[23] S. Reddy, A. D. Dragan, and S. Levine, "{SQIL}: Imitation learning via reinforcement learning with sparse rewards," in *International Conference on Learning Representations*, 2020.

[24] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson, "Implicit behavioral cloning," in *5th Annual Conference on Robot Learning*, 2021.

[25] A. Balakrishna, B. Thananjeyan, J. Lee, F. Li, A. Zahed, J. E. Gonzalez, and K. Goldberg, "On-policy robot imitation learning from a converging supervisor," in *Proceedings of the Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, L. P. Kaelbling, D. Kragic, and K. Sugiura, Eds., vol. 100. PMLR, 30 Oct–01 Nov 2020, pp. 24–41.

[26] J. Ichnowski, Y. Avigal, V. Satish, and K. Goldberg, "Deep learning can accelerate grasp-optimized motion planning," *Science Robotics*, vol. 5, no. 48, p. eabd7710, 2020.

[27] S. Zimmermann, R. Poranne, J. M. Bern, and S. Coros, "Puppetmaster: Robotic animation of marionettes," *ACM Trans. Graph.*, vol. 38, no. 4, Jul. 2019. [Online]. Available: https://doi.org/10.1145/3306346.3323003

[28] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York, NY, USA: Springer, 2006.

[29] C. Yuksel, "Sample elimination for generating poisson disk sample sets," *Computer Graphics Forum (Proceedings of EUROGRAPHICS 2015)*, vol. 34, no. 2, pp. 25–32, 2015.

[30] S. Duenser, J. M. Bern, R. Poranne, and S. Coros, "Interactive robotic manipulation of elastic objects," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 3476–3481.

[31] M. Geilinger, D. Hahn, J. Zehnder, M. Bächer, B. Thomaszewski, and S. Coros, "Add: Analytically differentiable dynamics for multi-body systems with frictional contact," *ACM Trans. Graph.*, vol. 39, no. 6, nov 2020.

[32] J. M. Bern, K.-H. Chang, and S. Coros, "Interactive design of animated plushies," *ACM Trans. Graph.*, vol. 36, no. 4, jul 2017.

[33] B. Fornberg, "Generation of finite difference formulas on arbitrarily spaced grids," *Mathematics of Computation*, vol. 51, no. 184, pp. 699–699, 1988.

[34] T. von Oehsen, A. Fabisch, S. Kumar, and F. Kirchner, "Comparison of distal teacher learning with numerical and analytical methods to solve inverse kinematics for rigid-body mechanisms," *CoRR*, vol. abs/2003.00225, 2020.

[35] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.