Constraint Estimation and Derivative-Free Recovery for Robot Learning from Demonstrations

Jonathan Lee[1], Michael Laskey[1], Roy Fox[1], Ken Goldberg[1,2]

***Abstract*— Learning from human demonstrations can facilitate automation but is risky because the execution of the learned policy might lead to collisions and other failures. Adding explicit constraints to avoid unsafe states is generally not possible when the state representations are complex. Furthermore, enforcing these constraints during execution of the learned policy can be challenging in environments where dynamics are difficult to model such as push mechanics in grasping. In this paper, we propose Derivative-Free Recovery (DFR), a two-phase method for generating robust policies from demonstrations in robotic manipulation tasks where the system comes to rest at each time step. In the first phase, we use support estimation of supervisor demonstrations and treat the support as implicit constraints on states. We also propose a time-varying modification for sequential tasks. In the second phase, we use this support estimate to derive a switching policy that employs the learned policy in the interior of the support and switches to a recovery policy to steer the robot away from the boundary of the support if it drifts too close. We present additional conditions, which linearly bound the difference in state at each time step by the magnitude of control, allowing us to prove that the robot will not violate the constraints using the recovery policy. A simulated pushing task in MuJoCo suggests that DFR can reduce collisions by 83%. On a physical line tracking task using a da Vinci Surgical Robot and a moving Stewart platform, DFR reduced collisions by 84%.**

I. Introduction

Robotic manipulation tasks are relevant in many industrial applications such as warehouse order fulfillment and flexible manufacturing where a robot must grasp or manipulate an object in environments with little structure. One method of approaching these problems is to construct an analytic model; however, doing so can often be difficult due to complex state spaces such as images, complicated mechanics such as pushing, and uncertainties in parameters such as friction. An alternative method is to use supervisor demonstrations to learn a policy. With learning from demonstrations, a robot observes a supervisor policy and learns a mapping from state to control via regression. This approach has shown promise for automation and robotics tasks such as grasping in clutter [16], robot-assisted surgery [33], and quadrotor flight [8].

Enforcing constraints on states, such as ensuring that a robot does not tension tissue above a certain level of force during a surgical task, remains an open problem in learning from demonstrations. Even if the demonstrated trajectories satisfy the constraints, there is no guarantee that the resulting learned policy will. For example, the robot may take a series of slightly sub-optimal actions due to approximation error of the learned policy and find itself in states vastly different from those visited by the supervisor. We desire to ensure the robot does not enter constraint-violating regions during execution. In this paper, we consider this problem for robotic manipulation in domains where the system comes to rest at each time step. This problem setting is inherent in many manipulation tasks in industrial and surgical settings with position control and has become increasingly important in automation [7], [25], [20].

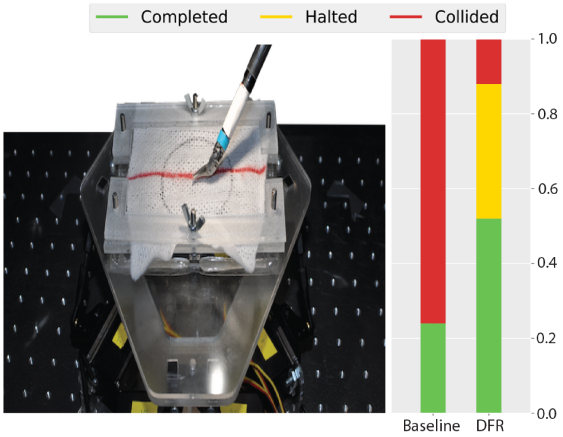

Fig. 1: The da Vinci Surgical Robot tracking a line drawn on gauze as the Stewart platform applies physical disturbances. The Baseline policy is compared with the policy with Derivative-Free Recovery (DFR) on the da Vinci line tracking task. Each segment depicts the fraction of "Completed," "Halted," and "Collided" trajectories. The results show that DFR significantly reduces collisions while also increasing the fraction of completed trajectories.

While techniques exist to enforce constraints on learned policies, they are often limited to operate in domains with known models [13], [22]. This can be challenging when dealing with robotic manipulation where interactions between objects can be fundamentally hard to model [31]. It can also be challenging to explicitly specify constraints. In a surgical task, objects such as tissue are often soft and deformable and observations often come from images from an endoscope. Additionally, specifying constraints such as the level of tension allowed on certain piece of tissue may require hard-coding rules that rely on complex models of these objects and noisy observations. However, the supervisor's demonstrated data provide not only information about the desired policy, but also information about the constraints. Intuitively, the robot should only visit states that the supervisor knows are safe to visit.

We propose leveraging the demonstration data to estimate the support of the supervisor's state distribution and treating the estimated support as a set of implicit constraints. The support is defined as the subset of the state space that the supervisor has non-zero probability of visiting. This subset is informative because it describes regions that must be safe

[1]Department of Electrical Engineering and Computer Science
[2]Department of Industrial Engineering and Operations Research
[1−2]The AUTOLAB at UC Berkeley; Berkeley, CA 94720, USA
jonathan_lee@berkeley.edu, laskeymd@berkeley.edu, royf@berkeley.edu, goldberg@berkeley.edu

since the supervisor visits those states. The complement of the support describes the region that may not be safe or include constraint-violating states. In the aforementioned surgical task, this would correspond to the robot recognizing that observations of heavily tensioned tissue are uncommon or nonexistant in the supervisor demonstrations and so it should try to avoid these states.

Various methods exist for density estimation which may be used to identify regions of support. In prior work, it was shown that the One Class SVM can be used effectively to estimate boundaries around the supervisor's demonstrations [18].

We use this support estimate to derive a switching policy that employs the robot's learned policy in safe states and switches to a recovery policy if the robot drifts close to the boundary of the estimated support. The recovery policy is posed as a derivative-free optimization (DFO) of the decision function of the support estimator, which provides a signal towards estimated safe areas. Because traditional DFO methods can be difficult to apply in dynamical systems, we propose a method to find likely directions toward safety by examining the outcome of applying small perturbations in the control signal, which we assumed lead to small changes in state. The recovery policy is designed to steer the robot towards safer regions in the best case or come to a stop if it cannot. We also present a condition, which bounds the change in state with respect to the magnitude of control, under which the robot will never enter the constraint-violating regions using the recovery policy.

In simulated experiments on the MuJoCo *Pusher* task [14], [30], we compared the proposed recovery control to a naive baseline and found that recovery reduced performance of the learned policy by 35% but also reduced the rate of collisions by 83%.

We also deployed the recovery strategy on a da Vinci Surgical Robot in a line tracking task under disturbances from a Stewart platform shown in Fig. 1(b) and found that the successes increased from 24% to 52% and collisions decreased from 76% to 12%.

This paper makes four contributions:

1) An implicit constraint inference method using support estimation on demonstrated data.
2) Derivative-Free Recovery, a novel model-free method for recovery control during execution of a learned policy.
3) Conditions under which the robot will not violate the constraints while using the recovery method.
4) Experimental results evaluating the proposed methods in simulation and on a physical robot.

II. Related Work

Learning from Demonstrations in Automation Tasks: Learning from demonstrations, sometimes also referred to as imitation learning, describes a broad collection of methods for learning to replicate sequential decision making. Specifically in automation and robotics, learning from demonstrations often makes use of kinesthetic or teleoperated demonstrations of control given by a human supervisor that is able to reason about the task from a high level. The learning system takes as input these demonstrations and outputs a policy mapping states to actions.

Prior work in automation has explored learning from demonstrations for highly unstructured tasks such as grasping in clutter, scooping, and pipetting [16], [19]. Past work has also addressed the specific problem of learning from demonstrations under constraints [4], [5]. A popular method for dealing with unknown constraints is to identify essential components of multiple successful trajectories based on variances in the corresponding states and then to produce a learned policy that also exhibits those components [6]. Despite early empirical success, constraint satisfaction is not guaranteed [22] and the machine learning model used to learn the policy must often be compatible with the variance estimator. We consider a method that is agnostic to the machine learning model.

C-LEARN [22] successfully incorporated motion planning with geometric constraints into keyframe-based learning from demonstrations for manipulation tasks, guaranteeing constraint satisfaction. However, constraints must be inferred from predetermined criteria, and an accurate model is required in order to satisfy those constraints using a motion planner.

Recent work has also dealt with learning constraint satisfaction policies from demonstrations when the constraints are unknown but linear with respect to the controls [3], [15]. There has also been recent work in guiding model-free policies towards states about which they are more confident, effectively trying to avoid certain unknown regions of the state space via temporal difference learning [28].

Significant literature exists on the topic of error detection and recovery (EDR) [9] with models. For example, Donald et al. [10] used EDR methods for planning with microrobots. In this paper we address this problem in the model-free domain.

Safe Learning to Control: Interest in learning-based approaches for control for under constraints has increased as a result of recent advances in learning and policy search, which have traditionally been studied without constraints due to their exploratory and unpredictable nature [1].

Assuming dynamics are known or can be estimated, Gillula and Tomlin [13] applied reachability analysis to address bounded disturbances by computing a sub-region within a predefined safe region where the robot will remain safe under any disturbance for a finite horizon. This region is referred to as the "discriminating kernel" by Akametalu et al. [2] and Fisac et al. [11] who extended this theory to obtain safe policies that are less conservative under uncertainty. In their work, the safety controller is applied only on the boundary of the discriminating kernel while the robot's controller is freely applied in the interior, resulting in a switching policy. Although our objectives are similar, there are several key differences in our assumptions. First, we do not require the model or constraints to be specified explicitly to the robot. Also, safe reinforcement learning aims to facilitate exploration for policy improvement while our approach addresses safe execution of policies after learning.

In surgical robotics, Yip and Camarillo [35] studied model-free control of continuum manipulators in constrained environments where the constraints are initially unknown. The authors proposed a combined position and force controller which actively estimates Jacobians. Continuum manipulators in surgical environments are in general designed to "conform" to obstacles constraints. In this paper, we consider manipulators

in general constrained environments where the manipulator may not have direct force feedback from interacting with constraints.

III. Problem Statement

Assumptions: We consider a discrete-time manipulation task with an unknown Markovian transition distribution and constraints specifying stay-out regions of the state space, such as collisions. The constraints are initially unknown to the robot. We further assume that the system comes to rest at each time step as in manipulation tasks with position control such as [19]. As in many applications of learning from demonstrations, we do not assume access to a reward function, meaning that there is no signal from the environment to indicate whether the robot is successfully completing the task. We assume a given set of observations of demonstrations from a supervisor that do not violate the constraints. The remainder of this section formalizes and elaborates these assumptions.

Modelling: Let the continuous state space and continuous control space be denoted by $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{U} \subseteq \mathbb{R}^d$, respectively. The unknown transition distribution is given by $p(x_{t+1}|x_t, u_t)$ with unknown initial state distribution $p_0(x)$. We define $\tau = \{(x_0, u_0), \ldots, (x_{T-1}, u_{T-1}), (x_T)\}$ as a trajectory of state-action pairs over T time steps. The probability of a trajectory under a stochastic policy $\pi : \mathcal{X} \mapsto \mathcal{U}$ is given by

$$p(\tau|\pi) = p_0(x) \prod_{t=0}^{T-1} p(u_t|x_t; \pi) p(x_{t+1}|x_t, u_t).$$

Additionally, we denote $p_t(x; \pi)$ as the distribution of states at time t under π, and we let $p(x; \pi) = \frac{1}{T} \sum_{t=0}^{T} p_t(x; \pi)$.

Although unknown, the dynamics of the system are assumed to leave the system at rest in each time step. For many practical discrete-time manipulation tasks, this property is common for example in settings where controls are positional and objects are naturally at rest such as in grasping in clutter [16].

Objective: This paper considers the problem of learning to accomplish a manipulation task reliably from observed supervisor demonstrations while attempting to satisfy constraints. We will only consider learning from demonstrations via direct policy learning, i.e. supervised learning.

Instead of a reward function, we assume that we have a supervisor that is able to demonstrate examples of the desired behavior in the form of trajectories. The robot's goal is then to replicate the behavior of the supervisor.

The goal in direct policy learning is to learn a policy $\pi : \mathcal{X} \mapsto \mathcal{U}$ that minimizes the following objective

$$\mathbb{E}_{\tau \sim p(\tau|\pi)} \, J(\tau, \pi^*) \tag{1}$$

where $J(\tau, \pi^*)$ is the cumulative loss of trajectory τ with respect to the supervisor policy π^*:

$$J(\tau, \pi^*) := \sum_{t=0}^{T-1} \ell(u_t, \pi^*(x_t)). \tag{2}$$

$\pi^*(x_t)$ indicates the supervisor's desired control at the state at time t, and $\ell : \mathcal{U} \times \mathcal{U} \mapsto [0, \infty)$ is a user-defined, non-negative loss function, such as the Euclidean norm of the difference between the controls. Note that in (1), the expectation is taken over trajectories sampled from π. Ideally, the learned policy minimizes the expected loss between its own controls and those of the supervisor on trajectories sampled from itself.

This objective is difficult to optimize directly because the trajectory distribution and loss terms are coupled. Instead, as in [18], [24], we formulate it as a supervised learning problem:

$$\min_{\pi \in \Pi} \quad \mathbb{E}_{\tau \sim p(\tau|\pi^*)} J(\tau, \pi). \tag{3}$$

Here, the expectation is taken with respect to the trajectories under the supervisor policy, rather than the robot's policy. This formulation decouples the distribution and the loss, allowing us to collect a dataset of training demonstrations $\{\tau_1, \ldots, \tau_N\}$ from the supervisor and minimize the empirical loss to obtain a learned policy $\hat{\pi}$:

$$\hat{\pi} = \arg\min_{\pi \in \Pi} \quad \frac{1}{N} \sum_{i=1}^{N} J(\tau_i, \pi). \tag{4}$$

This relaxation of the problem comes with a consequence. Because the training dataset is sampled from a different distribution (the supervisor distribution), it is difficult to apply traditional supervised learning guarantees about the learned policy. This problem is referred to as *covariate shift*. Prior work has considered learning recovery behavior during training [24], [17], but it is still not clear how errors may affect the robot or its environment, which motivates the need for increased robustness during execution.

Constraints: While prior work in learning from demonstrations has often dealt in the unconstrained setting, we consider learning in the presence of constraints that specify regions of the state space that the robot should actively avoid. Using the notation of [2], let $\mathcal{K}$ be a subset of $\mathcal{X}$ that is constraint-satisfying and let $\mathcal{K}^C$, the constraint-violating region, be its relative complement in $\mathcal{X}$. Note that this region is different from the support of the supervisor. The support is a subset of $\mathcal{K}$ that does not intersect $\mathcal{K}^C$. The supervisor, who is able to reason about the task at a high level, demonstrates the task robustly by providing constraint-satisfying trajectories during training time only. That is, $p(x; \pi^*) = 0$ for all $x \in \mathcal{K}^C$. Our objective is to have the robot learn this policy from demonstrations and perform it autonomously and reliably without entering the constraint-violating regions when it is deployed.

IV. Algorithms

A. Support Estimation

Given a set of sample states from supervisor demonstrations, $\{x_i\}_{i=1}^n \subset \mathcal{X}$, support estimation returns an approximate region of non-zero probability, $\{x \in \mathcal{X} : p(x; \pi^*) > 0\}$. Since the supervisor is always safely demonstrating the task, if $p(x; \pi^*) > 0$, then we know that $x \in \mathcal{K}$.

As presented by Schölkopf et al. in [26], a common objective in support estimation is to identify the set in the state space of least volume that captures a certain probability threshold α. For Lebesgue measure μ and probability space $(\mathcal{X}, \mathcal{B}, P)$ where $\mathcal{B}$ is the set of measurable subsets of $\mathcal{X}$ and

$P_{\pi^*}(B)$ is the probability of $B \in \mathcal{B}$ under the supervisor policy, the *quantile function* is

$$U(\alpha) = \inf_{B \in \mathcal{B}} \{\mu(B) \,:\, P_{\pi^*}(B) \geq \alpha\}.$$

The minimum volume estimator, $B(\alpha)$, is defined as the subset that achieves this objective for a given α [26]. To obtain the true support, we set $\alpha = 1$ since we would like to obtain the minimum volume estimator of the entire non-zero density region. In practice, there is no way to obtain the true minimum volume estimator with finite data and an unknown distribution. Instead, many methods for obtaining approximate support estimates have been proposed [12], [26]. For example, one might employ a kernel density estimator. In these cases, we often let $\alpha < 1$ to allow some tolerance for outliers, so that the estimator is more robust.

Despite prior use of support estimation in robotic and sequential tasks [18], estimators for which $\alpha < 1$ can be problematic when applied directly to observed states due to the time-variant nature of the state distribution. We provide a simple example where the minimum volume estimator fails to provide an accurate support estimate.

Consider two disjoint subsets of the state space B_0 and B_1, such that $p_0(x \in B_0; \pi^*) = 1$ and $p_t(x \in B_1; \pi^*) = 1$ for all $t > 0$. It is clear that $\lim_{T\to\infty} p(x \in B_0; \pi^*) = \lim_{t\to\infty} \frac{1}{T} \sum_{t=0}^{T} p_t(x \in B_0; \pi^*) = 0$ since states in B_0 are only possible as initial states. Therefore, if we simply draw examples from the distribution $p(x; \pi^*)$, the appropriate minimum volume estimate of any α-quantile will not include B_0 because the entire long-term probability density lies entirely in B_1.

This example reveals an important problem in the support estimation for tasks involving Markov chains: regions of the state space may be left out of the support estimate not because they are not relevant, but rather they are only relevant in a vanishing fraction of time steps. Thus, even if a region is known to surely be in the supervisor trajectories at some time step, it may be excluded from the estimated support. The example is not unrealistic. This problem may occur, albeit less severely, in any Markov chain where regions of the state space are revisited at different time steps.

Taking inspiration from [24], instead of using a single support estimator to encompass the entire distribution over states $p(x; \pi^*)$, we propose to use T estimators each for a corresponding distribution $p_t(x; \pi^*)$. By doing so, we limit each estimator to a single time step potentially reducing sample variance. When demonstrations are time-aligned, this can lead to improved support estimation. When they are not, we at worst increase the sample complexity T-fold.

In this paper, we use the One Class Support Vector Machine (OCSVM) to estimate the support [26], [27]. The estimator determines a small region of $\mathcal{X}$ where the fraction of examples within the region converges to an appropriate α-quantile as more data is collected [34]. Schölkopf et al. [26] present the primal optimization problem of the OCSVM as

$$\begin{aligned} \min_{w,\rho,\epsilon} \quad & \frac{1}{2}\|w\|_2^2 + \frac{1}{\nu m}\sum_{i=1}^{m} \epsilon_i - \rho \\ \text{s.t.} \quad & w^\top \phi(x_i) \geq \rho - \epsilon_i \quad i = 1, \ldots, m \end{aligned}$$

where m is the number of training examples, $0 < \nu < 1$ is a hyperparameter used to adjust the quantile level, and $\phi(\cdot)$ is a mapping from the state space to some implicit feature space.

At run time, we can determine whether each visited state lies in the estimated support by evaluating $\operatorname{sgn}\{g(x)\}$, where $g(x) = w^\top \phi(x) - \rho$ is the decision function. Positive values indicate that x is in the estimated support and negative values indicate otherwise. For the remainder of this paper, we will use the Gaussian kernel: $\phi(x)^\top \phi(x') = e^{-\gamma\|x - x'\|_2^2}$.

B. Derivative-Free Recovery Control

Once the support has been identified based on the supervisor demonstrations, the robot must learn a policy that minimizes the loss while staying within the boundaries of the estimated support to ensure it does not violate the constraints. To reconcile these potentially competing objectives, we propose using a switching policy at run time as in [2] that alternates between the learned policy $\hat{\pi}$ from (4) and a recovery policy π_R that attempts to guide the robot to interior regions of the support if it is close to the boundary.

The decision functions of the support estimators provide natural signed distance functions to the boundary of the estimated support. Thus as the robot rolls out, we can obtain reasonable online estimates of how "close" it is to the boundary. If the robot is in a state with a relatively high decision function value, it should apply its learned controls freely. However, if the decision function value at the robot's state is close to zero (i.e. near the boundary), the recovery should be activated to help the robot recover.

Formally, we may define a "close" distance as any distance from the boundary where the robot's learned policy could send it past the boundary in the next time step. Without a model of the dynamics, this cannot be known exactly. We introduce a tuneable hyperparameter λ, similar to a learning rate, which intuitively corresponds to a proportional relationship between the amount of change in the decision function and the magnitude of the applied control. We then propose a switching policy $\tilde{\pi}$ to incorporate the recovery behavior π_R:

$$\tilde{\pi} = \begin{cases} \hat{\pi} & g_t(x_t) > \lambda \|\hat{\pi}(x_t)\|_2 \\ \pi_R & \text{otherwise.} \end{cases}$$

The simplest recovery behavior is to apply zero control for the remaining time steps after the threshold has been crossed, potentially leaving the task incomplete. While this strategy will in principle reduce the risk of entering a constraint-violating state, it is overly conservative.

To increase the chance of completing the task while maintaining constraint satisfaction, we propose a best-effort recovery policy that leverages the decision function of the support estimator. When enabled, the recovery policy should drive the robot towards regions of the state space where the estimated decision value is higher, indicating the interior regions of the support. That is, we want to ascend on $g_t(x)$. If the dynamics model were known analytically, we could apply standard optimization techniques such as gradient ascent to obtain a local maximum of the decision function with respect to the controls. However, the model-free domain considered in this paper presents a challenge, as the decision function

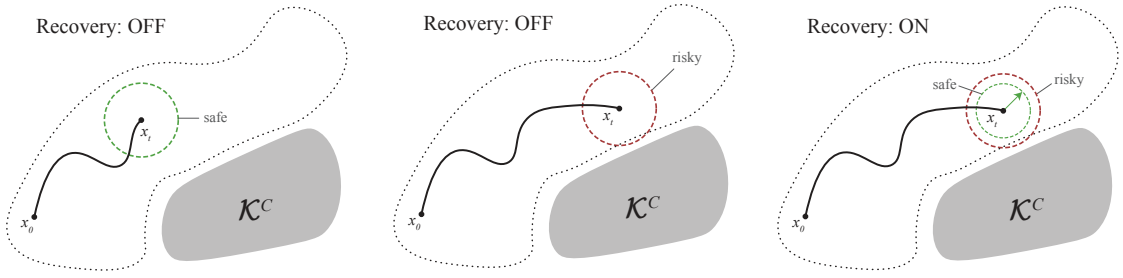

Fig. 2: The estimated support is represented as the dotted shape and the region of constraint-violating states is denoted by $\mathcal{K}^C$. At run time, the robot executes its learned policy starting at state x_0. The dashed circle around the current state x_t indicates the ball of states that the robot may enter in the next time step given its intended action. In (a), the ball is fully contained in the estimated support, so the robot uses its learned policy only. In (b), the ball overlaps with the boundary of the estimated support, indicating that the next state may be unsafe. In (c), as a result the recovery policy is activated, restricting the magnitude of control, as random perturbations are applied to find a direction of ascent.

Algorithm 1 Derivative-Free Recovery (DFR)

1: Initialize $t \leftarrow 0$, $x_0 \sim p_0(x)$
2: **while** $t < T$ **do**
3: $\quad \hat{u}_t \leftarrow \hat{\pi}(x_t)$
4: $\quad$ **while** $g_t(x_t) \leq \lambda \|\hat{u}_t\|_2$ **do**
5: $\quad\quad$ Sample random u_δ s.t. $\|u_\delta\|_2 \ll \frac{g_t(x_t)}{\lambda}$
6: $\quad\quad$ Apply u_δ and observe $x_\delta \sim p(\cdot|x_t, u_\delta)$
7: $\quad\quad$ **if** $g_t(x_\delta) \leq g_t(x_t)$ **then**
8: $\quad\quad\quad u_\delta \leftarrow -u_\delta$
9: $\quad\quad$ **end if**
10: $\quad\quad u_R \leftarrow \eta \frac{u_\delta}{\|u_\delta\|_2}$
11: $\quad\quad$ Apply u_R and observe $x \sim p\left(\cdot|x_\delta, u_R\right)$
12: $\quad\quad x_t \leftarrow x$
13: $\quad\quad \hat{u}_t \leftarrow \hat{\pi}(x_t)$
14: $\quad$ **end while**
15: $\quad$ Apply $\hat{u}_t$ and observe $x_{t+1} \sim p(\cdot|x_t, \hat{u}_t)$
16: $\quad t \leftarrow t + 1$
17: **end while**

with respect to the control is unknown. It is therefore not possible to use analytic derivative approaches to optimize the objective.

Additionally, conventional Derivative-Free Optimization (DFO) and finite difference methods [23], where multiple function evaluations of $g_t(x)$ would be made to find directions of ascent, are not suitable because we cannot directly manipulate the state x. Instead we may only control the state by applying input controls through the system, and we may only evaluate the effect of a control once it has been applied. Furthermore, because the system advances each time we apply a control, the objective function, which is a function of the current state, must change as well.

To address this problem, we propose a novel greedy derivative-free optimization approach, called Derivative-Free Recovery (DFR) Control, that employs a method similar to hill-climbing to make a best-effort recovery by applying conservative controls to ascend on the decision function. Consider the robot at state x_t. A small control perturbation u_δ is applied and yields a small change in state from x_t to x_δ. Consequently the perturbation also results in a small change in the decision function which indicates whether u_δ causes ascent or descent of the decision function at state x_t.

The full procedure for applying recovery controls online is shown in Algorithm 1. At any given time step, a control is obtained from the robot's policy. Using λ and the magnitude of the control, it is decided whether the robot's control is safe to use. If it is safe, then the control is executed without interruption. In the event that it is not safe, the recovery strategy is activated. A random but small control u_δ is then sampled, such that applying that control would still result in a positive decision function value. On lines 7 and 8, an approximate ascent direction is identified by executing the small random control and evaluating the decision function again. The recovery control u_R is then chosen as a vector in the direction of ascent with conservative magnitude η, where $0 < \eta < \frac{g_t(x)}{\lambda}$, limiting the risk of steering the robot out of the support and potentially into constraint-violating regions. Thus a larger choice of λ corresponds to a more conservative policy. While guaranteeing improvement of decision function may not be possible in all problems, improvements may be found in environments with locally nice and differentiable dynamics. A visual procedure is given in Fig. 2.

Furthermore, a fail-safe strategy naturally follows from this algorithm. In the event that recovery is not possible and the robot gets arbitrarily close to the boundary of the support, the magnitudes of the sample and recovery controls approach zero, effectively halting the robot to prevent it from failing. In the next section, we present conditions when we can guarantee constraint satisfaction for Algorithm 1 and formalize a worst-case choice for λ.

C. Conditions for Constraint Satisfaction

While it is not strictly necessary for good performance on many manipulation tasks as seen in the experiments, we introduce a condition on the dynamics model specific to some systems that formally characterizes a notion that the system comes to rest between time steps and allows us to guarantee that the robot will not violate constraints in systems where it is satisfied.

Assumption 4.1: For all $t \in \{0, \ldots, T-1\}$ there exists some constant K such that the following holds:

$$\|x_{t+1} - x_t\|_2 \leq K\|u_t\|_2. \tag{5}$$

This condition holds in stable manipulation systems where the amount of change from one state to the next is limited.

We now show that under the proposed algorithm and the above condition, it is guaranteed that the robot will not violate the constraints. Formally, let $\tilde{B}_t \equiv \{x : g_t(x) \geq 0\}$ be the estimated support of $p_t(x|\pi^*)$ with a corresponding

L-Lipschitz decision function $g_t(x)$. By (5) and the Lipschitz continuity of $g_t(x)$, $|g_t(x_{t+1}) - g_t(x_t)| \leq L\|x_{t+1} - x_t\|_2 \leq LK\|u_t\|_2$. This inequality formalizes a worst-case change in decision function value with respect to the magnitude of the robot's control, giving concrete meaning to the choice of $\lambda = LK$. Next, we guarantee constraint satisfaction for states in the estimated support:

Lemma 4.2: If at time t, the robot is in state x_t and $g_t(x_t) \geq 0$ and $\tilde{B}_t \cap \mathcal{K}^C = \emptyset$, then $x_t \in \mathcal{K}$.

Proof: This follows immediately from the condition that $\tilde{B}_t \equiv \{x : g_t(x) \geq 0\}$, which implies that $x_t \in \tilde{B}_t$. Thus, x_t must be in $\mathcal{K}$. ∎

Using this lemma, we are able to establish the following proposition:

Proposition 4.3: Under Algorithm 1 and the preceding conditions, the robot is never in violation of the constraints if $\tilde{B}_t \cap \mathcal{K}^C$ is empty.

Proof: The proof is by induction. Assume that the robot starts inside the estimated support. The induction assumption is that $g_t(x_t) \geq 0$, and we prove that this remains true after each step.

In the case where the learned policy $\hat{\pi}$ is constraint-satisfying, $\|\hat{u}_t\|_2 < \frac{1}{LK} g_t(x_t)$, we apply this control, and the next state satisfies

$$g_{t+1}(x_{t+1}) \geq g_t(x_t) - LK\|u_t\|_2 > 0.$$

The remaining case is where we switch to the recovery strategy, and we apply both u_δ and u_R with

$$\|u_\delta\|_2 = \tfrac{\epsilon}{LK} g_t(x_t)$$
$$\|u_R\|_2 = \eta \leq \tfrac{1-\epsilon}{LK} g_t(x_t)$$

for some $0 < \epsilon \ll 1$ splitting the difference between η and $\frac{g_t(x_t)}{LK}$. Then the state x after applying these controls satisfies

$$g_t(x) \geq g_t(x_t) - LK(\|u_\delta\|_2 + \|u_R\|_2) \geq 0.$$

We have shown that always $g_t(x_t) \geq 0$. If $\tilde{B}_t \cap \mathcal{K}^C = \emptyset$, then by Lemma 4.2 the robot is always constraint-satisfying. ∎

The intuition behind the proof of this proposition is that if we choose DFR controls with appropriately small magnitudes, applying those controls will never lead to a step that exceeds the boundary of the estimated support.

V. Experiments

We conducted manipulation experiments in simulation and on a physical robot to evaluate the proposed detection method and the reliability of various recovery strategies. Our experiments aim to answer the following questions:

1) Does support estimation provide a viable method for inferring safe regions given supervisor demonstrations when real constraint-violating regions exist but are not explicitly programmed by the supervisor? Is it viable even on systems where the conditions for constraint satisfaction do not necessarily hold?
2) Does DFR effectively climb the decision function?
3) How does DFR perform when varying the number of trajectories demonstrated?
4) How does DFR perform in response to small disturbances not seen during training time?

A. Pusher Simulation

Pusher (Fig. 4) is an environment simulated in MuJoCo [32] that considers the task of a one-armed robot pushing a light gray cylinder on a table to a green goal location. The initial state of the cylinder varies with each episode, preventing the robot from simply replaying a reference trajectory to succeed.

The robot has seven degrees of freedom controlling joint angle velocities. The state space consists of the joint angles, the joint angle velocities and the locations of the cylinder, end-effector, and goal object in 3D space. We modified the original task to allow control via direct changes in pose as opposed to velocity control of the joint angles. That is, the objects have no lasting momentum effects. We also introduced two regions marked in red representing the constraints of the task. The robot and the cylinder should not collide with these red regions. We stress that the robot does not know to avoid collisions with these states a priori, but the supervisor does. The robot must learn the support of the supervisor in order to recover if it approaches the collision states.

We generated an algorithmic supervisor using Trust Region Policy Optimization [29] to collect large batches of supervisor demonstrations. The learning model used a neural network with two 64-node hidden layers and $\tanh$ activations. 120 supervisor trajectories were collected for each trial. The learning models were also represented with neural networks optimizing (4). The models cannot match the supervisor exactly, which introduces the need for the recovery policy.

For the OCSVM, we set $\nu = 0.05$ as an arbitrary quantile of the observed data and then tuned the kernel scale $\gamma = 5.0$ on out-of-sample trajectories from the supervisor. To simplify the support estimation, we removed joint angles from the state space to include only those features relevant to the recovery behavior, as we found extraneous features often caused the OCSVM to require much more data.

For this task, we define a "Completed" trajectory to be any trajectory that reached the goal state without colliding. This includes trajectories where recovery was successful. A "Collided" trajectory is any trajectory that reached a collision state. Finally, a trajectory that "Halted" is any trajectory that neither reached the goal state nor entered a collision state in the allotted time. For example, the recovery policy may intentionally halt the task in high risk situations, resulting in a constraint-satisfying but incomplete trajectory. Trajectories that halted are strictly preferable to collisions. In many practical cases, they can also be reset, and the task may be attempted again. The ideal policy should minimize collisions while maintaining a high rate of completion.

We compared the proposed recovery strategy (DFR) in Algorithm 1 to a Baseline, which did not employ any recovery behavior, and an early stopping (ES) policy, which simply halted when it came close to the estimated support boundary. Fig. 3 illustrates the completed, halted, and collision rates for each method while varying the number of demonstrations of data. Across 10 trials with 60 evaluation samples per data-point per trial, DFR and ES significantly reduced the collision rate even with very little data compared to the Baseline, suggesting that staying within the estimated support is a viable method to avoid entering constraint violating regions. As more data was added, the completion rates of

Fig. 3: *Left:* The fraction of completed completed samples of the three methods (Baseline, Early Stopping (ES), DFR) is plotted as a function of the number of demonstrations. DFR achieves a comparable completion rate to Baseline. *Middle:* Halting rate which decreases for all methods as the learned policy acquires more data. Although Basline's halting rate decreases faster, it ultimately incurs more collisions without recovery. *Right:* The collision rate for Baseline is much higher than either ES or DFR, which both have consistently low collision rates even with very little data.

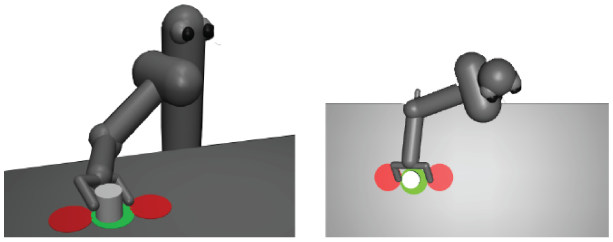

Fig. 4: *Left:* The *Pusher* task. The robot must learn to push the light gray object over the green circle without crossing over the red circles. *Right:* Top-down view.

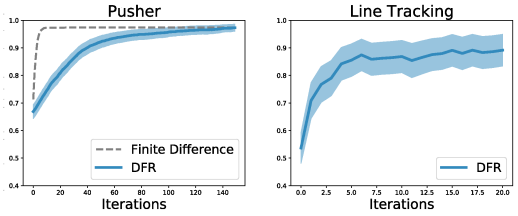

Fig. 5: *Left:* The average of 50 DFR optimization curves on *Pusher* is shown as a result of the recovery policy being activated during a trajectory. DFR is compared to a finite difference oracle. The decision function values were normalized between 0.0 and 1.0 where 1.0 represents the threshold of the switching policy. The normalized curves are capped at 1.0 because, by Alg. 1, the optimization stops once the threshold is reached. The few trajectories that do not reach 1.0 bring the average down slightly below 1.0 in both figures. *Right:* The average of 30 DFR curves on the da Vinci.

all three increased; however, DFR recovered from high risk situations allowing it to surpass ES and reach a comparable completion performance to the Baseline without significant collisions. DFR on average over all iterations achieved 83% fewer collisions compared to the Baseline. Additionally the completion rate of DFR was only 65% of that of the Baseline. Note that, due its conservative controller, DFR can prolong the wall clock time of a trajectory requiring an average of 1.50 seconds per trajectory while the baseline and ES required 0.13 seconds and 0.07 seconds, respectively.

Fig. 5 depicts the effectiveness of the derivative-free optimization technique on the decision function when the recovery strategy is activated. Note that the recovery strategy remains activated until the value of the decision function reaches the cutoff value $\lambda\|\hat{u}_t\|_2$ or until 500 iterations have elapsed. On 50 instantiations of the optimization algorithm on *Pusher*, each curve had nearly monotonic average improvement. We compared DFR with a finite difference oracle which was allowed to simulate controls before taking them in order to obtain numerical gradients with respect the controls.

B. Line Tracking on a da Vinci Surgical Robot

Robotic surgical procedures consist of safety-critical tasks that require robust control due to disturbances in environment and dynamics that are difficult to model. We consider learning positional control in a task that mimics disturbances that might be encountered in such environments. We applied support estimation and recovery policies to the task of tracking lines on gauze using the Intuitive Surgical da Vinci robot as shown in Fig. 1. The objective of the task was to deploy a learned policy from demonstrations to follow a red line drawn in gauze using the end-effector under disturbances that were not shown during training time. The gauze was mounted on a Stewart platform [21] which introduced random disturbances in the system during run time, but not during training. The robot used an overhead endoscope camera to observe images, which were processed to extract distances to the line and positions of the end-effector.

For this task, a "Collided" trajectory was defined as any trajectory where the end-effector deviated by more than 4 mm from the red line. A "Completed" trajectory was any trajectory that did not collide and tracked at least 40 mm of the gauze. All other trajectories were categorized as "Halted."

Over 50 demonstrations were given with an open-loop controller without disturbances. Thus the trajectories never deviated from the line. As a result no notion of feedback control was present in the demonstration data. The robot's policy was represented by a neural network. As in *Pusher*, we set the hyperparameters of the OCSVM by choosing a quantile level and validating on a held-out set of demonstrations.

The results are summarized in Fig. 1. The Baseline policy collided on the task repeatedly under random disturbances. The recovery was robust to the disturbances by attempting to keep the robot in the support. As in the *Pusher* task, an increase in trajectories that halted was observed with DFR, indicating the ability to detect constraint-violating areas and halt in the worst case. An increase in the rate of completion was also observed as DFR applied controls to mitigate deviations from the line and resume the original policies when the state was sufficiently far from the boundary.

VI. Discussion and Future Work

This paper presents Derivative-Free Recovery Control for robotic manipulation tasks. The results show that DFR can be used as an effective method of steering towards safe regions of a state space when a dynamics model is not known by ascending the decision function found by support estimation. Despite the promising asymptotic properties of the OCSVM, it can prove difficult in very high dimensional problems such as image space. This is a common trait of unsupervised learning the methods such as anomaly detection. Additionally the recovery procedure assumes the system comes to rest at each time step. In future work, we will extend DFR by addressing these problems with alternative support estimators and dimensionality reduction techniques and recovery planners that are less greedy.

VII. Acknowledgments

This research was performed at the AUTOLAB at UC Berkeley in affiliation with the Berkeley AI Research (BAIR) Lab, the Real-Time Intelligent Secure Execution (RISE) Lab, and the CITRIS "People and Robots" (CPAR) Initiative and with UC Berkeley's Center for Automation and Learning for Medical Robotics (Cal-MR). The authors were supported in part by donations from Siemens, Google, Cisco, Autodesk, Amazon, Toyota Research, Samsung, and Knapp and by the Scalable Collaborative Human-Robot Learning (SCHooL) Project, NSF National Robotics Initiative Award 1734633, and by a major equipment grant from Intuitive Surgical. We thank our colleagues who provided thoughtful feedback and suggestions, in particular Bill DeRose, Sanjay Krishnan, Jeffrey Mahler, Matthew Matl, and Ajay Kumar Tanwani.

References

[1] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *International Conference on Machine Learning (ICML)*, 2017.

[2] A. K. Akametalu, J. F. Fisac, J. H. Gillula, S. Kaynama, M. N. Zeilinger, and C. J. Tomlin, "Reachability-based safe learning with gaussian processes," in *IEEE Conference on Decision and Control (CDC)*, 2014.

[3] L. Armesto, V. Ivan, J. Moura, A. Sala, and S. Vijayakumar, "Learning constrained generalizable policies by demonstration," in *Robotics: Science and Systems (RSS)*, 2017.

[4] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Springer handbook of robotics*. Springer Berlin Heidelberg, 2008, pp. 1371–1394.

[5] S. Calinon, *Robot programming by demonstration*. EPFL Press, 2009.

[6] S. Calinon and A. Billard, "A probabilistic programming by demonstration framework handling constraints in joint space and task space," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2008.

[7] C. Chen, S. Krishnan, M. Laskey, R. Fox, and K. Goldberg, "An algorithm and user study for teaching bilateral manipulation via iterated best response demonstrations," in *International Conference on Automation Science and Engineering (CASE)*, 2017.

[8] A. Coates, P. Abbeel, and A. Y. Ng, "Learning for control from multiple demonstrations," in *International Conference on Machine Learning (ICML)*, 2008.

[9] B. R. Donald, *Error detection and recovery in robotics*. Springer-Verlag New York, 1989.

[10] B. R. Donald, C. G. Levey, I. Paprotny, and D. Rus, "Planning and control for microassembly of structures composed of stress-engineered mems microrobots," *The International Journal of Robotics Research*, vol. 32, no. 2, pp. 218–246, 2013.

[11] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. H. Gillula, and C. J. Tomlin, "A general safety framework for learning-based control in uncertain robotic systems," *arXiv preprint*, vol. abs/1705.01292, 2017.

[12] G. Gayraud, "Estimation of functionals of density support," *Mathematical Methods of Statistics*, vol. 6, no. 1, pp. 26–46, 1997.

[13] J. H. Gillula and C. J. Tomlin, "Guaranteed safe online learning via reachability: tracking a ground target using a quadrotor," in *IEEE International Conferece on Robotics and Automation (ICRA)*, 2012.

[14] K. Hausman, Y. Chebotar, S. Schaal, G. Sukhatme, and J. Lim, "Multi-modal imitation learning from unstructured demonstrations using generative adversarial nets," *arXiv preprint*, vol. abs/1705.10479, 2017.

[15] M. Howard, S. Klanke, M. Gienger, C. Goerick, and S. Vijayakumar, "A novel method for learning policies from variable constraint data," *Autonomous Robots*, vol. 27, no. 2, pp. 105–121, 2009.

[16] M. Laskey, J. Lee, C. Chuck, D. Gealy, W. Hsieh, F. T. Pokorny, A. D. Dragan, and K. Goldberg, "Robot grasping in clutter: Using a hierarchy of supervisors for learning from demonstrations," *Automation Science and Engineering (CASE), 2016 IEEE*, pp. 827–834, 2016.

[17] M. Laskey, J. Lee, R. Fox, A. Dragan, and K. Goldberg, "Dart: Noise injection for robust imitation learning," in *Conference on Robot Learning*, 2017.

[18] M. Laskey, S. Staszak, W. Y.-S. Hsieh, J. Mahler, F. T. Pokorny, A. D. Dragan, and K. Goldberg, "Shiv: Reducing supervisor burden in dagger using support vectors for efficient learning from demonstrations in high dimensional state spaces," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 462–469.

[19] J. Liang, J. Mahler, M. Laskey, P. Li, and K. Goldberg, "Using dvrk teleoperation to facilitate deep learning of automation tasks for an industrial robot," in *IEEE International Conference on Automation Science and Engineering (CASE)*, 2017.

[20] L. Lu and J. T. Wen, "Human-directed robot motion/force control for contact tasks in unstructured environments," in *International Conference on Automation Science and Engineering (CASE)*, 2015.

[21] V. Patel, S. Krishnan, A. Goncalves, and K. Goldberg, "Sprk: A low-cost stewart platform for motion study in surgical robotics," in *International Symposium on Medical Robotics (ISMR)*, 2018.

[22] C. Pérez-D'Arpino and J. A. Shah, "C-learn: Learning geometric constraints from demonstrations for multi-step manipulation in shared autonomy," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[23] L. M. Rios and N. V. Sahinidis, "Derivative-free optimization: a review of algorithms and comparison of software implementations," *Journal of Global Optimization*, vol. 56, no. 3, pp. 1247–1293, 2013.

[24] S. Ross and D. Bagnell, "Efficient reductions for imitation learning," in *International Conference on Artificial Intelligence and Statistics*, 2010, pp. 661–668.

[25] G. F. Rossano, C. Martinez, M. Hedelind, S. Murphy, and T. A. Fuhlbrigge, "Easy robot programming concepts: An industrial perspective," in *International Conference on Automation Science and Engineering (CASE)*, 2013.

[26] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[27] B. Schölkopf and A. J. Smola, *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[28] Y. Schroecker and C. L. Isbell, "State aware imitation learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 2915–2924.

[29] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning (ICML)*, 2015.

[30] A. Singh, L. Yang, and S. Levine, "Gplac: Generalizing vision-based robotic skills using weakly labeled images," *arXiv preprint*, vol. abs/1708.02313, 2017.

[31] B. Thananjeyan, A. Garg, S. Krishnan, C. Chen, L. Miller, and K. Goldberg, "Multilateral surgical pattern cutting in 2d orthotropic gauze with deep reinforcement learning policies for tensioning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[32] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[33] J. Van Den Berg, S. Miller, D. Duckworth, H. Hu, A. Wan, X.-Y. Fu, K. Goldberg, and P. Abbeel, "Superhuman performance of surgical tasks by robots using iterative learning from human-guided demonstrations," in *ICRA, 2010 IEEE*. IEEE, 2010, pp. 2074–2081.

[34] R. Vert and J.-P. Vert, "Consistency and convergence rates of one-class svms and related algorithms," *The Journal of Machine Learning Research*, vol. 7, pp. 817–854, 2006.

[35] M. C. Yip and D. B. Camarillo, "Model-less hybrid position/force control: a minimalist approach for continuum manipulators in unknown, constrained environments," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 844–851, 2016.