SHIV: Reducing Supervisor Burden using Support Vectors to Efficiently Learn Robot Control Policies from Demonstrations [v18]

Michael Laskey[1], Sam Staszak[3], Wesley Yu-Shu Hsieh[1], Jeffrey Mahler[1],
Florian T. Pokorny[1], Anca D. Dragan[1], and Ken Goldberg[1,2]

***Abstract*— Online algorithms for robot model-free robot learning from demonstration (such as DAgger) can learn policies for problems where the system dynamics and the cost function are unknown. However, during learning they impose a burden on supervisors to provide labels (control signals) as the robot encounters new states when iteratively executing its current best policy. We introduce the SHIV algorithm (Svm-based reduction in Human InterVention), which reduces this burden by enabling the robot to *only request supervision for risky states*. SHIV uses stream-based active learning with a query selection method that evaluates risk in a manner tailored to non-stationary high dimensional state distributions. To facilitate scaling and outlier rejection, risk is based on distance to an approximate level set boundary defined by a One Class support vector machine. We compare SHIV with DAgger in three policy learning experiments: 1) a driving simulator with a 27936D visual HOG feature state representation, 2) 4DOF robot arm push-grasping in clutter (simulation with Box2D) with a 22D state representation of joint angles and the pose of each object, and 3) surgical needle insertion with a 16D state represented spatial transform. Results suggest that SHIV can reduce the number of queries up to 70%.**

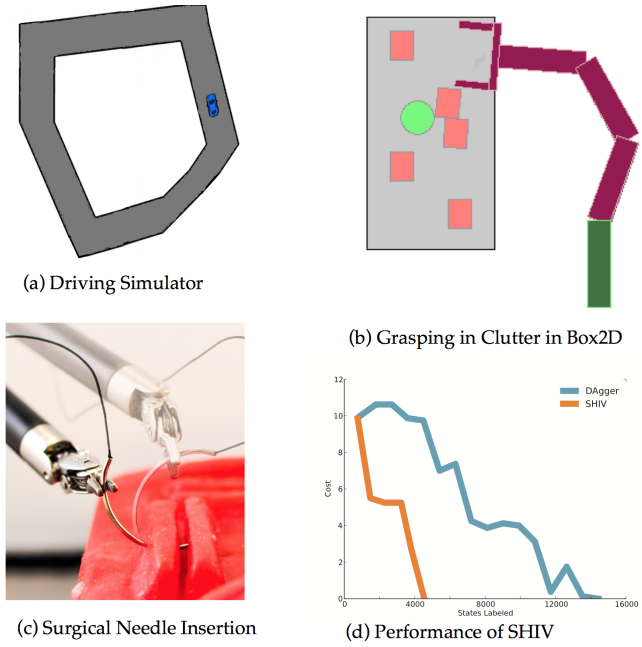
I. Introduction

In model-free robot learning from demonstration, a robot learns to perform a task, such as driving or grasping an object in a cluttered environment (Fig. 1), from examples provided by a supervisor, usually a human. In such problems, the robot does not have access to neither the cost function that it should optimize, nor the dynamics model. The former happens when it is difficult for a human to specify how to trade-off various aspects that matter, like when the robot is reaching for a goal object while not pushing objects off a table(Fig. 1(b). The latter happens when either the system or the interaction with the world is difficult to characterize with the given observations, like in Fig. 1(a) the robot must stay on the track but only has access to visual information from a bird's eye view.

Rather than learning the cost function and the dynamics model, and then using optimization to produce a policy for the task, the robot learns the policy directly from supervisor examples mapping states to controls [1].

In the offline learning from demonstration setting, the robot learns the policy based on a batch of examples, and then executes it to achieve the task. During execution, a small error can accumulate and push the robot away from the region of the state space where it has seen examples. The learned policy might not be helpful in unfamiliar states, leading to unrecoverable failures. For example, a robot driving may be trained on examples driving safely down the center of a lane, but even slight deviations will put the robot into states near the side of the road where its policy could fail [22]. The number of errors made by the robot, in the worst case, can scale quadratically with the time horizon of the task [23].

Online learning from demonstration approaches such as DAgger address this issue by iteratively gathering more examples from the supervisor in states the robot is likely to visit [9], [23], [24]. DAgger learns a series of policies. At each iteration, the robot trains a policy based on the existing examples, then rolls out (executes) that policy and the supervisor provides demonstrations for all states the robot visits. The new examples are aggregated with the



(a) Driving Simulator (b) Grasping in Clutter in Box2D (c) Surgical Needle Insertion (d) Performance of SHIV

Fig. 1: SHIV reduces the number of queries to the supervisor in our three test domains: a) a driving simulator where the goal is to learn a controller on HOG features extracted from synthetic images to keep a car on a polygonal track; b) push-grasping in clutter in a physics simulator, where the goal is to learn a controller from human demonstrations to grasp the green object while not letting the red square objects fall off the gray boundary representing a table; c) learning a controller for the Da Vinci Research Kit from a master surgeon's demonstrations of correcting a needle before suturing in surgery; d) cost of SHIV and DAgger's policy on the grasping in clutter example versus states labeled.

[1] Department of Electrical Engineering and Computer Sciences; {mdlaskey,iamwesleyhsieh, ftpokorny, anca}@berkeley.edu
[2] Department of Industrial Engineering and Operations Research; goldberg@berkeley.edu
[1−2] University of California, Berkeley; Berkeley, CA 94720, USA

old examples for the next iteration. DAgger and related algorithms have been applied in a wide range of applications, from quadrotor flight to natural language to Atari games [10], [8], [25]. In DAgger, the number of errors scales only linearly with the time horizon of the task.

To achieve better performance guarantees, online algorithms can significantly increase the burden on the supervisor, who now labels all states that the robot visits during training. Our goal is to reduce supervisor burden without degrading robot performance.

Our key insight, to reduce burden, is that the robot only needs to request demonstrations for risky states. Risk arises because: (1) states are different than the states previously encountered and used to train the current policy, i.e., they can be considered novel or outliers [11]; or (2) states are in a region that has been misclassified, in previous training data.

We contribute an algorithm for online learning from demonstration that can actively decide whether it is necessary to ask the supervisor to label the new states that the robot encounters. Our algorithm, SHIV (Svm-based reduction in Human InterVention), reduces supervisor burden by only requesting supervision for risky states. SHIV uses stream-based active learning with a query selection method that evaluates risk in a manner tailored to the non-stationary state distributions the robot encounters as it iteratively executes learned policies. We build on the One Class SVM method for approximating quantile level sets [26] to provide a classification method for deciding if a state is risky or safe.

II. Related Work

Below we summarize related work in active approaches to robot learning from demonstration and risk evaluation techniques..

Active Learning from Demonstration Online learning from demonstration can be formulated as a stream-based active learning problem [2], [6]. In stream based active learning, the decision of whether to query the supervisor (for a label or control signal) or not is not over the entire state space (like in traditional pool-based active learning), but on states drawn one at a time from some data source. In online learning from demonstration, the data source is the states encountered during learning.

Stream-based active learning has two ingredients: a stream, given here by executing the current best policy, and a query selection method deciding whether or not to query. Typical query selection methods are estimator-centric: evaluating risk using the estimator, e.g. distance from the classification hyperplane [29], as in Fig. 2(a). An alternative to distance-based measures of risk is query by committee [3], which uses a committee of hypothesized estimators that are trained on different subsets of the training data. Risk here is based on the level of agreement or disagreement among these hypotheses, with higher levels of disagreement for a state leading to higher chances of querying that state (Fig. 2(b)).

Both approaches implicitly assume a stationary state distribution — that the new data is sampled from the same distribution as the previous training data. Although such methods have been previously proposed for online learning from demonstration (see [5], [9] for the former and [13], [14] for the latter), online learning from demonstration violates the stationary distribution assumption because each new policy induces a new distribution of states. This can have negative consequences for learning: it has been shown that when training and test distributions are different, query by committee can perform worse than randomly selecting which states to query [4].

The estimator is a function of the current distribution. Therefore, it no longer provides a good measure of confidence when that distribution changes, as it does when the robot is rolling out a learned policy and starts encountering further away states. Instead of relying on the estimator, our measure of risk explicitly identifies when states are drawn from a different distribution, i.e. when they are novel.

Risk via Novelty Detection. One part of our risk definition is the notion that a trained model will be able to generalize within the distribution it is trained on [28], making states outside of this distribution risky. Novelty detection [11] is concerned with recognizing when this happens: recognizing that a sample is outside of this distribution.

One approach to novelty detection is to directly estimate the underlying probability density function from which the data is sampled. If the probability of that point is low with respect to the underlying density, it will be marked as novel. However, the amount of data needed to accurately do so scales exponentially in the number of dimensions [20].

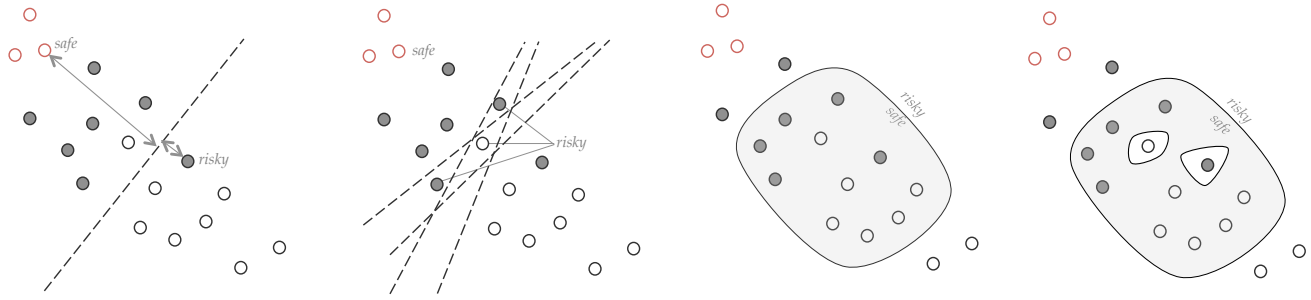
An alternative to evaluating the probability of a data point is to measure distance to its nearest neighbors [16]. However, this approach was shown to be susceptible to issues, since nearest neighbors incorporates only local information about the data. For example, a group of outliers can be close together, but significantly far from the majority of the data and nearest neighbors would mark them as not novel [11].

Our approach is based on the One Class SVM proposed by Scholköpf et al., which estimates a particular quantile level set for the training data by solving a convex quadratic program to the find support vectors [26]. The method has been theoretically shown to approximate the quantile levelset of a density estimate asymptotically for correctly chosen bandwidth settings and in the case of a normalized Gaussian kernel function [31]. In [19], the One Class SVM has furthermore been used for novelty detection in high-dimensional image data. To our knowledge the use of One Class SVM to online learning from demonstration has never been done before.

III. Problem Statement

The goal of this work is to learn a policy that matches that of the supervisor's while asking the supervisor for as few examples as possible.

Modeling Choices and Assumptions We model the system dynamics as markovian, stochastic, and stationary. Stationary meaning given a state and a control the probability of the next state doesn't change over time. We model the initial state as sampled from a distribution over the state space.



(a) Confidence (b) Query by Committee (c) Novelty (d) Combined Risk

Fig. 2: A comparison of different query selection strategies for active learning on a non-stationary distribution. The shaded and empty circles are training data from two classes, 1 and 0 respectively. The red empty circles are samples from a new distribution (produced by executing the learned policy), and belong to class 0. Typical strategies classify states close the decision boundary as risky(a), or for which a set of estimators disagree (b). Neither of these apply to our new samples in red. In contrast, we use a strategy that is amenable to non-stationary distributions by classifying novel states as safe (i.e not risky) (c) and states in historically mislabeled regions.

We assume a known state space and set of controls. We assume access to a robot or simulator, such that we can sample from the state sequences induced by a sequence of controls. Lastly, we assume access to a supervisor who can, given a state, provide the desired control signal label.

Policies and State Densities. We denote by $\mathcal{X}$ the set consisting of observable states for a robot task, consisting, for example, of high-dimensional vectors corresponding to images from a camera, or robot joint angles and object poses in the environment. We furthermore consider a set $\mathcal{U}$ of allowable control inputs for the robot, which can be discrete or continuous. We model dynamics, as Markovian, such that the probability of state $\mathbf{x_{t+1}} \in \mathcal{X}$ can be determined from the previous state $\mathbf{x}_t \in \mathcal{X}$ and control input $\mathbf{u}_t \in \mathcal{U}$:

$$p(\mathbf{x}_{t+1}|\mathbf{u}_t, \mathbf{x}_t, \ldots, \mathbf{u}_0, \mathbf{x}_0) = p(\mathbf{x}_{t+1}|\mathbf{u}_t, \mathbf{x}_t)$$

We assume a probability density over initial states $p(\mathbf{x}_0)$.

A trajectory $\hat{\tau}$ is a finite series of $T+1$ pairs of states visited and corresponding control inputs at these states, $\hat{\tau} = (\mathbf{x}_0, \mathbf{u}_0, ..., \mathbf{x}_T, \mathbf{u}_T)$, where $\mathbf{x}_t \in \mathcal{X}$ and $\mathbf{u}_t \in \mathcal{U}$ for $t \in \{0, \ldots, T\}$ and some $T \in \mathbb{N}$. For a given trajectory $\hat{\tau}$ as above, we denote by τ the corresponding trajectory in state space, $\tau = (\mathbf{x}_0, ..., \mathbf{x}_T)$.

A policy is a function $\pi : \mathcal{X} \to \mathcal{U}$ from states to control inputs. We consider a space of policies $\pi_\theta : \mathcal{X} \to \mathcal{U}$ parameterized by some $\theta \in \mathbb{R}^d$. Any such policy π_θ in an environment with probabilistic initial state density and Markovian dynamics induces a density on trajectories of length $T+1$:

$$p(\tau|\theta) = p(\mathbf{x}_0) \prod_{i=0}^{T-1} p(\mathbf{x}_{t+1}|\pi_\theta(\mathbf{x}_t), \mathbf{x}_t)$$

Let $p(\mathbf{x}_t|\theta)$ denote the value of the density of states visited at time t if the robot follows the policy π_θ from time 0 to time $t-1$. This density can be computed by marginalization: $p(\mathbf{x}_t|\theta) = \int_{\mathbf{x}_{t-1}} \ldots \int_{\mathbf{x}_1} p((\mathbf{x}_t, ..., \mathbf{x}_1)|\theta) d\mathbf{x}_{t-1} ... d\mathbf{x}_1$. Following [24], we can compute the average density on states *for any timepoint* by

$$p(\mathbf{x}|\theta) = \frac{1}{T} \sum_{t=1}^{T} p(\mathbf{x}_t|\theta) \tag{1}$$

While we do not assume analytic knowledge of the distributions corresponding to: $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$, $p(\mathbf{x}_0)$, $p(\mathbf{x}_t|\theta)$ or $p(\mathbf{x}|\theta)$, we assume that we have a stochastic real robot or a simulator such that for any state $\mathbf{x}_t$ and control $\mathbf{u}_t$, we can sample the $\mathbf{x}_{t+1}$ from the density $p(\mathbf{x}_{t+1}|\pi_\theta(\mathbf{x}_t), \mathbf{x}_t)$. Therefore, when 'rolling out' trajectories under a policy π_θ, we utilize the robot or a simulator to sample the resulting stochastic trajectories rather than estimating $p(\mathbf{x}|\theta)$ itself.

Objective. The objective of policy learning is to find a policy that minimizes some known cost function $C(\hat{\tau}) = \sum_{t=1}^{T} c(\mathbf{x}_t, \mathbf{u}_t)$ of a trajectory $\hat{\tau}$. The cost function $c : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ is typically user defined and task specific. For example, in the task of inserting a peg into a hole, a function on the distance between the peg's current and desired final state can be considered [17].

However in our learning from demonstration setup, the robot does not have access to the cost function itself. Instead, it only has access to a supervisor that we assume uses some $\tilde{\pi}$ to minimize $C(\hat{\tau})$, to an acceptable level. We are furthermore given an initial set of N stochastic demonstration trajectories $\{\tilde{\tau}^1, ..., \tilde{\tau}^N\}$. which are the result of the supervisor applying this policy. This induces a training dataset $\mathcal{D}$ of all state-control input pairs from the demonstrated trajectories.

We define a 'surrogate' loss function $l : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$, which provides a distance measure between any pair of control values. In the continuous case, we consider $l(\mathbf{u}_0, \mathbf{u}_1) = ||\mathbf{u}_0 - \mathbf{u}_1||^2$, while in the discrete case $l(\mathbf{u}_0, \mathbf{u}_1) = 1$ if $\mathbf{u}_0 \neq \mathbf{u}_1$ and $l(\mathbf{u}_0, \mathbf{u}_1) = 0$ otherwise.

Given a candidate policy π_θ, we then use the surrogate loss function to approximately measure how "close" the policy's returned control input $\pi_\theta(\mathbf{x}) \in \mathcal{U}$ at a given state $\mathbf{x} \in \mathcal{X}$ is to the supervisor's policy's control output $\tilde{\pi}(\mathbf{x}) \in \mathcal{U}$. The goal is to produce a policy that minimizes the surrgoate loss between relative to the supervisor's policy.

Following [24], our objective then consists of determining a policy π_θ minimizing the expected surrogate loss, where

the expectation is taken over the distribution of states induced by the policy across any time point in the horizon:

$$\min_{\theta} E_{p(\mathbf{x}|\theta)}[l(\pi_\theta(\mathbf{x}), \tilde{\pi}(\mathbf{x}))] \quad (2)$$

with $p(\mathbf{x}|\theta)$ from Eq. 1.

If the robot could learn the policy perfectly, this state density would match the one encountered in the user examples. But if the robot makes an error, that error changes the distribution of states that the robot will visit, which can lead to states that are far away from any examples and difficult to generalize to [22].Thus, motivating iterative algorithms like DAgger and now SHIV, which iterate between learning a policy and then the supervisor providing feedback. The feedback is in the form of control signals on states sampled from the robot's new distribution of states.

IV. SHIV

Both SHIV and DAgger [24] solve the minimization in Eq. 2 by iterating two steps: 1) compute a θ using the training data $\mathcal{D}$ thus far, and 2) execute the policy induced by the current θ, and ask for labels for the encountered states. However, instead of querying the supervisor for every new state, SHIV actively decides whether the state is risky enough to warrant a query.

A. Step 1.

The first step of any iteration k is to compute a θ_k that minimizes surrogate loss on the current dataset $\mathcal{D}_k = \{(x_i, u_i)|i \in \{1, \ldots, M\}\}$ of demonstrated state-control pairs (initially just the set $\mathcal{D}$ of initial trajectory demonstrations):

$$\theta_k = \arg\min_{\theta} \sum_{i=1}^{M} l(\pi_\theta(\mathbf{x}_i), \mathbf{u}_i). \quad (3)$$

This problem is a supervised learning problem, solvable by estimators like a support vector machine or a neural net [1]. Finding the right function representation and optimization method for a given problem is still an open question in machine learning. However a large number of advances have been made that make the use of these techniques very feasible [27]

B. Step 2

The second step starts in both SHIV and DAgger by rolling out the policy π_{θ_k} to sample states that are likely under $p(\mathbf{x}|\theta_k)$.

What happens next, however, differs. For every state visited, DAgger requests the supervisor to provide the appropriate control/label. Formally, for a given sampled trajectory $\hat{\tau} = (\mathbf{x}_0, \mathbf{u}_0, ..., \mathbf{x}_T, \mathbf{u}_T)$, the supervisor provides labels $\tilde{\mathbf{u}}_t$, where $\tilde{\mathbf{u}}_t \sim \pi(\mathbf{x}_t) + \epsilon$ for $t \in \{0, \ldots, T\}$. The states and labeled controls are then aggregated into the next data set of demonstrations $\mathcal{D}_{k+1}$:

$$D_{k+1} = \mathcal{D}_k \cup \{(\mathbf{x}_t, \tilde{\mathbf{u}}_t) \| t \in \{0, \ldots, T\}\}$$

Providing correct control inputs or "labeling" for all states encountered at each iteration can impose a large burden on the supervisor. Instead of asking the supervisor for labels at all visited states $\{\mathbf{x}_0, .., \mathbf{x}_T\}$, SHIV uses a measure of risk to actively decide whether a label is necessary. For every state encountered $\mathbf{x}_t$, it applies a decision function g_σ, which we introduce in the next section, parametrized by a risk-sensitivity parameter σ which captures how smooth we expect the policy function to be: smoother functions imply that training examples have larger support, and data nearby will be less risky.

SHIV only asks for supervision on states for which $g_\sigma(\mathbf{x}) \neq 0$:

$$D_{k+1} = \mathcal{D}_k \cup \{(\mathbf{x}_t, \tilde{\mathbf{u}}_t) \| t \in \{0, \ldots, T\}, g(\mathbf{x}_t) = -1\}$$

Steps 1 and 2 are repeated for K iterations or until the cumulative surrogate is smaller than some predefined threshold[2].

For a conservative σ, the SHIV performance is analogous to DAgger without asking the expert when it is absolutely not necessary. However, if σ is too high, then SHIV will not collect new data that is needed and revert back to the performance of offline methods. We analyze the dependence on σ in Sec. V-A, and our results suggest that SHIV is robust to this parameter selection.

C. Measuring Risk

We work under the assumption that a state can be risky for 2 reasons: 1) it lies in an area with a low density of previously trained states, which can cause our policy to mispredict the supervisor and incur high surrogate loss [28], or 2) the surrogate loss, or training error, of the current policy at a state in $\mathcal{D}$ is high, so that the state does not model the supervisor's control inputs correctly.

To evaluate risk in high-dimensional state spaces (such as HOG features on images, like in our driving simulator, Fig. 1(a)), we use a modified version of the technique known as the One Class SVM that approximately estimates a boundary of a user defined quantile of a density representing the training data in $\mathcal{X}$ [26].

We consider the problem of estimating the quantile level-sets of a distribution P on a set $\mathcal{X}$ by means of a finite set of independent and identically distributed samples $\mathbf{x}_1, ..., \mathbf{x}_n \in \mathcal{X}$. In most general terms, the quantile function for P and subject to a class of measurable subsets $\mathcal{G}$ of $\mathcal{X}$ is defined by

$$U(\gamma) = \inf\{\lambda(G) : P(G) \geq \gamma, G \in \mathcal{G}\} \; 0 < \gamma \leq 1 \quad (4)$$

[1]To handle the fact that the supervisor's policy can be noisy, a zero-mean noise term ϵ can be considered as present in the policy's output. A regularization technique in the optimization is used to control the smoothness of the function that is fit to the sampled data. In practice this regularization corresponds to a penalty term on either the L2 norm on the weights for regression based techniques or the slack coefficient for support vector machines [27].

[2]In the original DAgger the policy rolled out was stochastically mixed with the supervisor, thus with probability β it would either take the supervisor's action or the robots. The use of this stochastically mix policy was for theoretical analysis. In practice, it is recommended to set $\beta = 0$ to avoid biasing the sampling [10], [24]

$\lambda : \mathcal{G} \rightarrow \mathbb{R}$ above denotes a volume measure which most commonly is given by the Lebesgue measure. Suppose furthermore that $G : [0,1] \rightarrow \mathcal{G}$ assigns a set $G(\gamma) \in \mathcal{G}$ that attains the infinum measure (i.e. volume) for each $\gamma \in [0,1]$ (this set is in general not necessarily unique). $G(\gamma)$ denotes a set of minimum measure $G \in \mathcal{G}$ with $P(G(\gamma)) \geq \gamma$. Note in particular that $G(1)$ is the support of the density p corresponding to P, if p exists.

To handle distributions defined on high-dimensional spaces $\mathcal{X}$, work by Scholköpf et al. looked at representing the class $\mathcal{G}$ via a kernel k as the set of half-spaces in the support vector (SV) feature space [26]. By minimizing a support vector regularizer controlling the smoothness of the estimated level set function this work derives an approximation of the quantile function described in Eq. 4, which in particular has rigorous convergence guarantees asymptotically when a normalized Gaussian kernel k is chosen and bandwidths decay at an appropriately chosen rate as the number of samples tends to infinity [31]. This approach can be thought of as employing $\lambda(G) = ||w||^2$, where $G_w = \{x : f_w(x) \geq \rho\}$, $f_w(\mathbf{x}) = \sum_i w_i k(\mathbf{x}_i, \mathbf{x})$ and (w, ρ) denote the weight vector and offset parameterizing a hyperplane in the feature space associated with a Gaussian kernel $k(\mathbf{x}_0, \mathbf{x}_1) = e^{-||\mathbf{x}_0 - \mathbf{x}_1||^2/2\sigma^2}$.

Let $\Phi : \mathcal{X} \rightarrow \mathcal{F}$ denote the feature map corresponding to our exponential kernel, mapping the observation space $\mathcal{X}$ into a Hilbert space $(\mathcal{F}, \langle , \rangle)$ such that $k(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle$.

The One Class SVM proposed by [26] determines a hyperplane in feature space $\mathcal{F}$ maximally separating the input data from the origin:

$$\underset{w \in \mathcal{F}, \xi \in \mathbb{R}, \rho \in \mathbb{R}}{\text{minimize}} \frac{1}{2}||w||^2 + \frac{1}{vn}\sum_i^n \xi_i - \rho \qquad (5)$$
$$\text{s.t } \langle w, \Phi(x_i) \rangle \geq \rho - \xi_i,\ \xi_i \geq 0.$$

Here, the parameter ν controls the penalty or 'slack term' and is equivalent to γ [31] in the quantile definition, Eq. 4, as the number of samples increases. The decision function, determining point membership in the approximate quantile levelset is given by

$$g(\mathbf{x}) = \text{sgn}(\langle w, \Phi(x) \rangle - \rho). \qquad (6)$$

Here, for $x \in \mathcal{X}$, $g(x) = 0$ if x lies on the quantile levelset, $g(x) = 1$ if x is strictly in the interior of the quantile superlevelset and $g(x) = -1$ if x lies strictly in the quantile sublevelset. The dual form of the optimization yields a Quadratic Program that can be solved efficiently [26]. In the dual the decision function is given by

$$g(\mathbf{x}) = \text{sgn}(\sum_{i=1}^{N} \alpha_i k(\mathbf{x}_i, \mathbf{x}) - \rho). \qquad (7)$$

where α_i corresponds to the dual variables. The novelty detection method can be visualized in Fig. 2(c). However, even when sufficient data is available, the associated control inputs may be inconsistent or noisy and a resulting policy optimizing Eq. 3 may still incur a large surrogate loss. To account for this, we propose a modification to the One Class SVM:

$$y_i = \begin{cases} 1 & : l(\pi_\theta(\mathbf{x}_i), \mathbf{u}_i) \leq \varepsilon \\ -1 & : l(\pi_\theta(\mathbf{x}_i), \mathbf{u}_i) > \varepsilon \end{cases} \qquad (8)$$

Where, in the case when l denotes discrete $0-1$ loss, we set $\varepsilon = 0$, while in the continuous L_2 loss case, ε is a user defined threshold specifying allowable surrogate loss. We use y_i to modify the One Class SVM decision function as follows:

We divide up our data in to two sets those correctly classified:

$$\mathcal{D}_s = \{\{\mathbf{x}_i, \mathbf{u}_i\} \in \mathcal{D}_k, y = 1\}$$

and those states incorrectly classified:

$$\mathcal{D}_r = \{\{\mathbf{x}_i, \mathbf{u}_i\} \in \mathcal{D}_k, y = -1\}$$

A separate One-Class SVM is then trained on each set of states, (D_s and D_r) and providing measures of the level sets, g_s and g_r. Specified by parameters (ν, σ) and (ν_r, σ_r), respectively.

We then define the overall decision function as:

$$g_\sigma(\mathbf{x}) = g_r(\mathbf{x}) + g_s(\mathbf{x}) \qquad (9)$$

points are deemed risky if $g_\sigma(\mathbf{x}) \neq 0$. Practically, this modification corresponds to 'carving out holes' in the estimated quantile super-levelset such that neighborhoods around states with $y_i = -1$ are excluded from the super-levelset. An illustration of this can be seen in Fig. 2(d).

The decision function parametrization consists of the kernel bandwidth σ in g_s. Changing sigma corresponds to how much influence each point in the training set has on the measure of risk. Large σ corresponds to points nearby being safe. For a policy that is a smooth function σ should be large.

Our experiments analyze the performance of risk as a query selection strategy, with or without the addition of the second risk criteria, i.e. regions of space where the surrogate loss is high.

V. Experiments

We begin our experiments in an in-depth analysis of SHIV with a driving simulator. Here, we compare our query selection method with those typically used in active learning. We show that for a non-stationary state distribution like ours, the notion of risk based on novelty and misclassified regions performs better than a confidence and query-by-committee based methods. We continue with a sensitivity analysis, which suggests that the performance of SHIV is robust to the choice of how risky the robot is allowed to be (the σ parameter from Eq. 9).

We then compare SHIV and DAgger on three domains: a driving simulator, push-grasping in clutter with a 4DOF arm, and surgery needle insertion using demonstrations from Dr. Douglas Boyd, a master surgeon at UC Davis.

All experiments were run on a machine with OS X with a 2.7 GHz Intel core i7 processor and 16 GB 1600 MHz

memory in Python 2.7. The policies, π_θ are either the linear SVM or kernelized ridge regression classes in Scikit-Learn [21].

Our modified One Class SVM contains two different ν parameters, ν and ν_r. We set $\nu = 0.1$ and $\nu_r = 10^{-3}$ for all experiments. We tuned σ and σ_r by performing a grid search over different values on the surrogate loss for a single trial of SHIV for 3 iterations for each domain.

A. SHIV Analysis

We analyze our algorithm in a driving domain.

Driving Simulator Domain. Our first domain is a common benchmark in Reinforcement Learning: learning a policy for a car to drive around a track [1], [23], [**?**]. We implemented a driving simulator where the car must follow a polygonal track. We generated polygonal tracks by repeatedly sampling from a Gaussian with mean that is the center of the video game workspace centered in the middle of the workspace, and computing the convex hull of the sampled points.Then a convex hull is computed on the sampled points. In general this produces tracks composed of five to seven edges, an example is shown in Fig. 1(a). If the car accidentally leaves the track, it is placed back on the center of the track at a nearby position. The car's control input space is given by $\mathcal{U} = \{-15°, 0, 15°\}$. A control input instantly changes the angle of the car which drives at a unit speed. The internal state space of the car is given by the xy-coordinates and the angle it is facing. In our experiments, the supervisor is provided by an algorithm that uses state space search through the driving simulator to plan the next control.

The supervisor drives around the track twice. We collect raw images of the simulation from a 2D bird's eye view and use Gaussian Pyramids to down-sample the images to 125×125 RGB pixels and then extract Histogram of Oriented Gradients (HOG) features using OpenCV. This results in a 27926 dimensional state space description. For both DAgger and SHIV, we use a Linear Support Vector Machine (SVM) to parameterize allowable policies π_θ, with $\gamma = 0.01$ as a regularization term on the slack variables, which was set via cross validation on the initial training examples. We set SHIV's parameters of the exponential kernel's bandwidth as $\sigma = 200$ and $\sigma_r = 200$.

1) Comparison to active learning approaches.:

Independent Variables: We compare four query selection methods. We compare our combined notion of risk (Fig. 2(d)), with risk based on novelty alone (Fig. 2(c)) in order to test whether carving out regions that have been misclassified previously is valuable.

We also compare against two baselines, typically used in active learning. The first is confidence based on distance from the classification hyperplane [29] (Fig. 2(a)). We set the threshold distance to the average distance from the hyperplane for the mis-classified points in $\mathcal{D}_0$, which consisted of two demonstrations from our solver.

The second baseline is Query By Committee (Fig. 2(d)), which was trained via bagging [3]. To obtain a committe, we divided the training data into 3 overlapping subsets, each with 80% of the data. We trained a Linear SVM on each subset. If the three classifiers agreed with each other the point was determined low risk and if they disagree it was determined high risk.

We run each query selection method over 50 different car tracks.

Dependent Measures. We measured the percentage of truly risky states, encountered during the first policy roll out, that are estimated to be safe by the active learning technique. The active learning techniques are trained on the initial demonstrations $\mathcal{D}_0$, which is different then the distribution being sampled from, $p(\mathbf{x}|\theta_0)$, in this experiment.

Results. Fig. 3 plots the performance for each query selection method, averaged over 50 tracks. We observe a significant performance improvement with methods based on novelty compared to confidence and query by committee. Furthermore, using the combined measure of risk performs better than relying solely on novelty.

2) Sensitivity Analysis: .

Independent Variables. To analyze the sensitivity of our method we varied the σ parameter of the risk estimation function g_σ. σ is a measure of how much risk we allow, with smaller σs leading to more risk-adverse behavior. SHIV was ran for 6 iterations and averaged over 40 different randomized polygonal tracks.

Dependent Measures. For each σ, we measure the final cost function for the policy π_θ after 6 iterations and the number of examples SHIV requested from the supervisor.

Results. As shown in Table I, small σ, $\sigma = 1$, corresponds to always asking for help (many states labeled) and very large sigma, $\sigma = 350$, relates to less data being used, but worse performance similar to the traditional (offline) supervised learning approaches. However, σ values between 150 and 250 all achieve similarly good performance, suggesting that SHIV is robust to the choice of a particular σ.

Risk Sensitivity (σ)	Final Cost	States Labeled
1	6	4122
50	6	3864
150	6	1859
200	6	1524
250	6	1536
350	13	521

TABLE I: An analysis of the sensitivity of σ. SHIV was ran for 6 iterations and averaged over 40 different randomized polygonal tracks. Small σ, $\sigma = 1$, corresponds to always asking for help and very large sigma, $\sigma = 350$, relates to a lot less data being used, but worst performance similar to the traditional supervise learning approach.

B. Comparing SHIV with DAgger

We compare SHIV with DAgger on three domains: the driving simulator from above, push-grasping in clutter, and needle insertion.

Independent Variables. We manipulate two variables. First, we manipulate the online learning from demonstration algorithm we use: SHIV vs. DAgger. Second, we manipulate how many examples the algorithms are allowed to ask for: we set a budget of labeled states, and analyze performance as this budget increases.

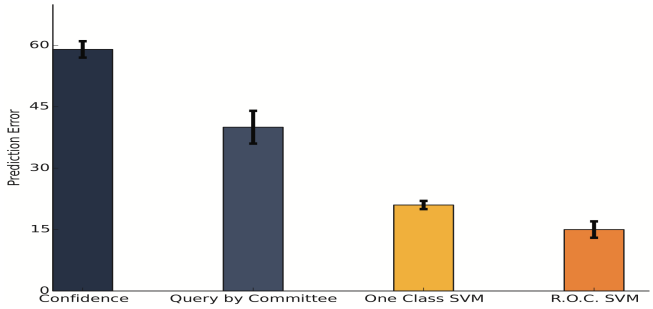


Fig. 3: A comparison of different active learning approaches in terms of the percentage of risky states that are estimated to be safe by the active learning technique during the first policy roll out. We compare against a confidence estimate of distance to hyperplane, query by committee for 3 hypothesis classifiers, the One Class SVM, and our modified One Class SVM. Results are averaged over 50 tracks and the policy π_θ is represented as a Linear SVM.

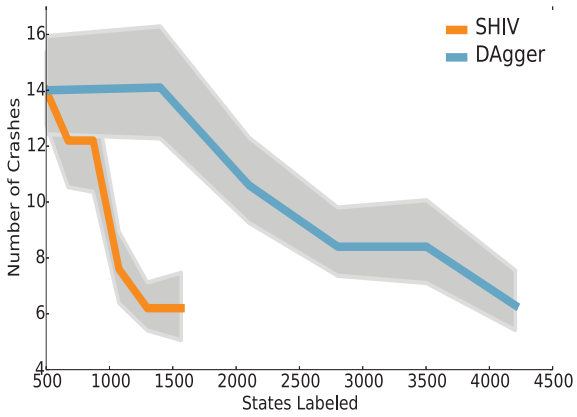


Fig. 4: We compare performance for the Driving Simulator in terms of minimization of the underlying cost function $c(\mathbf{x}, \mathbf{u})$, which is the number of times the car left the track versus the number of queries made to the supervisor. In Fig. 4, we plot the performance of DAgger and SHIV. Initial results, which are run for 6 iterations each and are averaged over 40 levels, shown in Fig. 4 suggest an 71% reduction in the number of queries needed for SHIV compared to DAgger,

Dependent Measures. For each algorithm and budget of states labeled, we measure the real cost that the learned policy attains (e.g. number of crashes in the driving domain).
Hypothesis: We expect SHIV to consistently achieve a lower cost that DAgger for the same budget of supervisor examples. Equivalently, we expect that SHIV achieves the same performance as DAgger, but by asking for fewer examples and thus reducing supervisor burden.

1) Driving.: **Domain.** Our first domain is the driving domain described above and illustrates SHIV's performance in high dimensions. We compare the policy's performance in terms of minimization of an underlying cost function $c(\mathbf{x}, \mathbf{u})$, which is defined to be total number of times the car left the track in relation to the number of queries made to the supervisor.
Results. In Fig. 4, we visualize the performance of DAgger and SHIV. We run 6 iterations, which is a completion of both Step 1 and Step 2 described in Section IV of each algorithm over 40 different randomized polygonal tracks. Figure 4 presents averaged results from these experiments, suggesting a 71% reduction in the number of queries needed for SHIV compared to DAgger.

2) Grasping In Clutter: **Domain.** We investigate having a human demonstrator control a robot arm in 2D to reach a target object with out knocking other objects off a table. Grasping an object in a cluttered environment is a common task for a robot in an unstructured environment and has been considered a benchmark for robotic manipulation [15], [**?**]. The task is difficult because modeling the physics of pushing an object is non-trivial and requires knowing the shape, mass distribution and friction coefficient of all objects on the table. We are interested in learning such a policy via human demonstrations.

In lieu of real robot demonstrations, we used Box2D a physics simulator to model a virtual world. We simulate a 4 DOF robot arm with three main joints and a parallel jaw gripper as displayed in Fig. 1(b). SHIV and DAgger do not have access to the underlying dynamics of the simulator and must learn a policy from only demonstrations.

For input the human demonstrator provides controls through an XBox controller. The right joystick was used to provide horizontal and vertical velocity inputs for the center of the end-effector which were then translated into robot arm motions by means of a Jacobian transpose controller for the 3 main joint angles. The left 'bumper' button on the joystick was used to provide a binary control signal to close the parallel jaw gripper. The control inputs are hence modeled by the set $\mathcal{U} = \{[-1, 1], [-1, 1], \{0, 1\}\}$.

A state $\mathbf{x} \in \mathcal{X}$ consisted of the 3 dimensional pose of the six objects on the table (translation and rotation), the 3 joint angles of the arm and a scalar value in the range $[0, 1]$ that measured the position of the gripper, 1 being fully closed and 0 being opened. For our representation of π_θ, we used kernelized ridge regression with the radial basis function as the kernel with the default Sci-Kit learn parameters. We defined the cost functions, $C(\mathbf{x}, \mathbf{u})$, as the sum of the number of objects knocked off the table plus 10 times the binary value indicating if the object is grasped or not. The order of magnitude difference in cost for grasping the object is to place emphasize on that portion of the task. The bandwidth parameters for SHIV were set to $\sigma_r = 5$ and $\sigma = 6$. For the ϵ term in the our risk method, we used the median in regression error which was the L2 distance between the predicted control and the true supervisor's control.

In our experiment, a human demonstrator provided one demonstration and then iterated until the cost function was zero during the policy roll out. At each iteration, we sampled the pose of the target object from an isotropic Gaussian with a standard deviation that is equal to 3% of the width of the table.
Results. In Fig. 5 , we show the cost function $c(\mathbf{x}, \mathbf{u})$ averaged over 8 rounds for SHIV and DAgger. Supporting our hypothesis, our results suggests that SHIV can achieve the same performance with a 64% reduction in the number of examples needed.

3) Surgical Experiment: **Domain.** Robotic Surgical Assistants (RSAs) are frequently used for procedures such

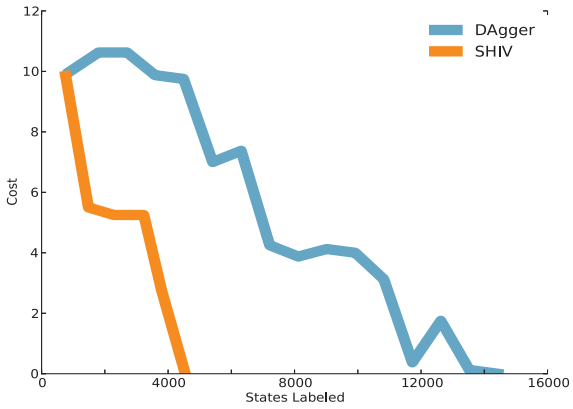


Fig. 5: We compare performance for Grasping in Clutter in terms of minimization of the underlying cost function $c(\mathbf{x}, \mathbf{u})$, which is the sum of the number of objects knocked off the table plus 10 times the binary value indicating if the object is grasped or not. Initial results, which are averaged over 8 different trials suggest a 64% reduction in the number of queries needed for SHIV compared to DAgger,

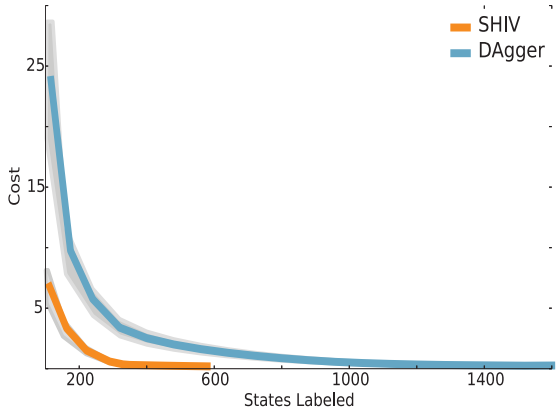


Fig. 6: We compare performance in terms of minimization of the underlying cost function $c(\mathbf{x}, \mathbf{u})$, which is euclidean distance between translation in centimeters. In Fig. 4, we plot the performance of DAgger and SHIV without the modification. Initial results, which are run for 20 iterations each and are averaged over 40 different initial starting positions, shown in Fig. 4 suggest an 67% reduction in the number of queries needed for SHIV compared to DAgger,

as: prostectomy, uterectomy, and tumorectomies within the abdominal and thoracic cavities with high success rates [30], [7]. Currently, these devices are controlled by surgeons via physical tele-operation at all times; introducing autonomy of surgical sub-tasks has the potential to reduce cognitive load and facilitate supervised autonomy for remote tele-surgerical procedures.

Suture tying in surgery is a manually intensive task that can occur frequently through out a surgery. One important step in suture tying is properly placing a needle in an initial configuration for insertion. Misplacement of this the needle can lead to suture failure and potentially rupture the surrounding tissue [18]. In this experiment, we are interested in learning a policy to correct bad initial poses to the proper insertion pose as shown in Fig. 2. Dr. Douglas Boyd, a surgeon at UC Davis, provided us with a collection of demonstrations on a Intuitive Surgical Da Vinci Research Kit [12].

Dr. Boyd demonstrated a series of trajectories that each started at an initial attempted needle insertion pose P_0 and applied the necessary corrections to achieve a goal pose P_G. The time horizon, T of each trajectory was on average 80. We used three of these demonstrations as our initial dataset $\mathcal{D}_0$, thus $|\mathcal{D}_0| = 240$. In order to study convergence of both SHIV and DAgger, we chose to create a synthetic expert for online learning part. The expert controller computed the transformation between the desired insertion pose and the current pose by calculating the inverse of a transformation matrix, $C = P_0 P_G^{-1}$. Then converted C to the associated lie algebra vector $c \in \mathbb{R}^6$ for $SE(3)$ and normalize it to the average magnitude of the control, $||\bar{c}_D||$, Doug Boyd applied to the robot.

The policy π_θ was represented as kernel ridge regression with the default values given in Sci-Kit learn. The state $\mathcal{X}$ was a 16 dimensional vector consisting of the elements in the pose P vector. The control space $\mathcal{U}$ was a $\mathbb{R}^6$ vector representing the Lie algebra.

For the ϵ term in the our risk method, we used the median in regression error which was the L2 distance between the predicted control and the true supervisor's control. regions respectively. The bandwidth, σ, in the rbf kernel was set to 2 and $\sigma_b = 1$.

For trials we sample a start position from a Gaussian on the translational component of P_0 with isotropic variance of 0.1 cm. The distance between initial P_0 and P_G is roughly 3 cm. Our cost function $c(\mathbf{x}, \mathbf{u})$ is measured in terms of Euclidean distance in translational component, which is in centimeters. We run DAgger and SHIV both for 20 iterations and average over 40 different starting positions.

Results. In Fig. 5 , we show the cost function $c(\mathbf{x}, \mathbf{u})$ averaged over 40 rounds for SHIV and DAgger. Supporting our hypothesis, our results suggests that SHIV can achieve the same performance with a 67% reduction in the number of examples needed.

VI. Discussions and Future Work

VII. Acknowledgments

This work is supported in part by the U.S. National Science Foundation under Award IIS-1227536 and NSF-Graduate Research Fellowship.. We thank UC Berkeley and our colleagues who gave feedback and suggestions, in particular Sanjay Krishnan, Siddarth Sen, Steve McKinley, Sachin Patil and Sergey Levine.

References

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[2] L. E. Atlas, D. A. Cohn, and R. E. Ladner, "Training connectionist networks with queries and selective sampling," in *Advances in neural information processing systems*, 1990, pp. 566–573.

[3] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[4] R. Burbidge, J. J. Rowland, and R. D. King, "Active learning for regression based on query by committee," in *Intelligent Data Engineering and Automated Learning-IDEAL 2007*. Springer, 2007, pp. 209–218.

[5] S. Chernova and M. Veloso, "Interactive policy learning through confidence-based autonomy," *Journal of Artificial Intelligence Research*, vol. 34, no. 1, p. 1, 2009.

[6] D. Cohn, L. Atlas, and R. Ladner, "Improving generalization with active learning," *Machine learning*, vol. 15, no. 2, pp. 201–221, 1994.

[7] A. Darzi and Y. Munz, "The impact of minimally invasive surgical techniques," *Annu. Rev. Med.*, vol. 55, pp. 223–237, 2004.

[8] F. Duvallet, T. Kollar, and A. Stentz, "Imitation learning for natural language direction following through unknown environments," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 1047–1053.

[9] D. H. Grollman and O. C. Jenkins, "Dogged learning for robots," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 2483–2488.

[10] X. Guo, S. Singh, H. Lee, R. L. Lewis, and X. Wang, "Deep learning for real-time atari game play using offline monte-carlo tree search planning," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 3338–3346.

[11] V. J. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artificial Intelligence Review*, vol. 22, no. 2, pp. 85–126, 2004.

[12] Intuitive Surgical, "Annual report 2014." [Online]. Available: http://investor.intuitivesurgical.com/phoenix.zhtml?c=122359&p=irol-IRHome

[13] K. Judah, A. Fern, and T. Dietterich, "Active imitation learning via state queries," in *Proceedings of the ICML Workshop on Combining Learning Strategies to Reduce Label Cost*, 2011.

[14] K. Judah, A. Fern, and T. G. ietterich, "Active imitation learning via reduction to iid active learning," *arXiv preprint arXiv:1210.4876*, 2012.

[15] N. Kitaev, I. Mordatch, S. Patil, and P. Abbeel, "Physics-based trajectory optimization for grasping in cluttered environments."

[16] E. M. Knox and R. T. Ng, "Algorithms for mining distancebased outliers in large datasets." Citeseer.

[17] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *arXiv preprint arXiv:1504.00702*, 2015.

[18] T. Liu and M. C. Cavusoglu, "Optimal needle grasp selection for automatic execution of suturing tasks in robotic minimally invasive surgery," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 2894–2900.

[19] W. Liu, G. Hua, and J. R. Smith, "Unsupervised one-class learning for automatic outlier removal," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*. IEEE, 2014, pp. 3826–3833.

[20] E. A. Nadaraya, "On estimating regression," *Theory of Probability & Its Applications*, vol. 9, no. 1, pp. 141–142, 1964.

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[22] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," DTIC Document, Tech. Rep., 1989.

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

[24] S. Ross, G. J. Gordon, and J. A. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," *arXiv preprint arXiv:1011.0686*, 2010.

[25] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert, "Learning monocular reactive uav control in cluttered natural environments," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 1765–1772.

[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] S. T. Tokdar and R. E. Kass, "Importance sampling: a review," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 1, pp. 54–60, 2010.

[29] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *The Journal of Machine Learning Research*, vol. 2, pp. 45–66, 2002.

[30] M. H. van der Pas, E. Haglind, M. A. Cuesta, A. Fürst, A. M. Lacy, W. C. Hop, H. J. Bonjer, C. cancer Laparoscopic or Open Resection II (COLOR II) Study Group *et al.*, "Laparoscopic versus open surgery for rectal cancer (color ii): short-term outcomes of a randomised, phase 3 trial," *The lancet oncology*, vol. 14, no. 3, pp. 210–218, 2013.

[31] 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.