SHIV: Reducing Supervisor Burden using Support Vectors for Efficient Learning from Demonstrations in High Dimensional State Spaces

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

***Abstract*— Online learning from demonstration algorithms, 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 respond to queries each time the robot encounters new states while executing its current best policy. Algorithms such as MMD-IL reduce supervisor burden by filtering queries with insufficient discrepancy in distribution and maintaining multiple policies. We introduce the SHIV algorithm (Svm-based reduction in Human InterVention), which converges to a single policy and reduces supervisor burden in non-stationary high dimensional state distributions. To facilitate scaling and outlier rejection, filtering is based on distance to an approximate level set boundary defined by a One Class support vector machine. We report on experiments in three contexts: 1) a driving simulator with a 27,936 dimensional visual feature space, 2) a push-grasping in clutter simulation with a 22 dimensional state space, and 3) physical surgical needle insertion with a 16 dimensional state space. Results suggest that SHIV can efficiently learn policies with equivalent performance requiring up to 70% fewer queries.**

I. Introduction

In model-free robot Learning from Demonstration (LfD), a robot learns to perform a task, such as driving or grasping an object in a cluttered environment, from examples provided by a supervisor, usually a human. In such problems, the robot does not have access to either the cost function that it should optimize, nor the dynamics model. The former occurs when it is difficult to specify how to trade-off various aspects that matter, like a car trying to drive on the road and dodge other cars [3]. The latter occurs when either the system or the interaction with the world is difficult to characterize, like when a robot is trying to grasp an object in clutter and does not have an accurate model of the objects dynamics.

Rather than explicitly learning the cost function (like in Inverse Reinforcement Learning [25]) and the dynamics model, and then using optimization to produce a policy for the task, in model-free LfD the robot learns the policy directly from supervisor examples, mapping states to controls [4]. Learning from demonstration has been used successfully in recent year for a large number of robotic tasks, including helicopter maneuvering [1], car parking [2], and robotic surgery [35].

LfD algorithms can be categorized as offline, where the robot only observes demonstrations, or online, where the robot interacts with the world and receives feedback from the supervisor. In offline LfD, 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, leading the robot away from the region of the state space where it was given examples, 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 eventually put the robot into states near the side of the road where its policy could fail [27]. Ross and Bagnell showed the number of errors made by the robot, in the worst case, can scale quadratically with the time horizon of the task [28].

Online LfD addresses this issue by iteratively gathering more examples from the supervisor in states the robot encounters [11], [28], [29]. One such algorithm, DAgger, learns a series of policies. At each iteration, the robot trains a policy based on the existing examples, then rolls out (executes)

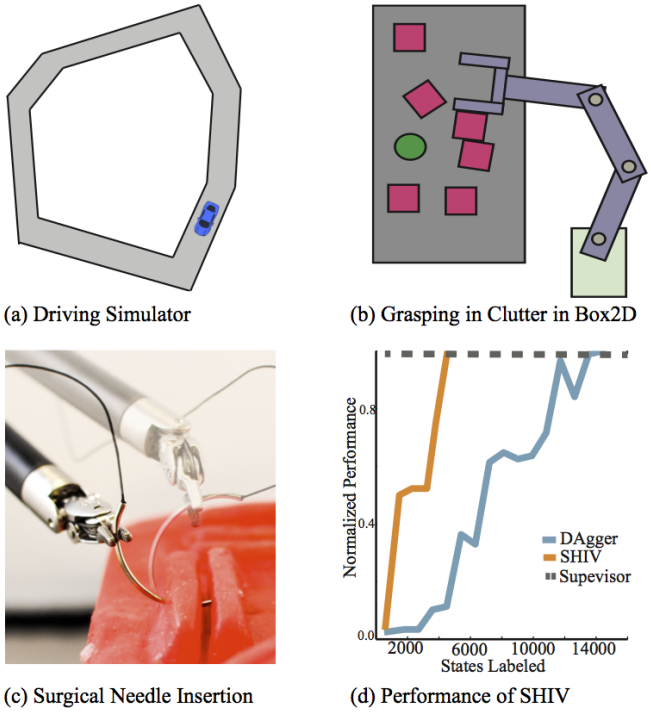


Fig. 1: SHIV reduces the number of queries to the supervisor in our three test domains: a) a driving simulator where the goal is to learn a controller on HOG features extracted from synthetic images to keep a car on a polygonal track; b) push-grasping in clutter in a physics simulator, where the goal 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 senior surgeon's demonstrations of correcting a needle before suturing in surgery; d) normalized performance of SHIV and DAgger's policy on the grasping in clutter example versus states labeled by the supervisor.

[1] Department of Electrical Engineering and Computer Sciences; {mdlaskey,iamwesleyhsieh,ftpokorny,anca}@berkeley.edu, staszass@rose-hulman.edu

[2] Department of Industrial Engineering and Operations Research; goldberg@berkeley.edu

[1−2] University of California, Berkeley; Berkeley, CA 94720, USA

Preprint submitted to 2016 IEEE International Conference on Robotics and Automation. Received September 15, 2015.

that policy, and the supervisor provides demonstrations for all states the robot visits. The new state/control examples are aggregated with the 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 [12], [10], [30]. In DAgger, under certain conditions, the number of errors scales only linearly with the time horizon of the task[29].

However one drawback is that DAgger significantly increases 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 performance.

Our key insight is that the robot only needs to request demonstrations for risky states. Risk arises because: (1) states are sufficiently different than the states previously encountered, i.e., they can be considered novel or outliers [13]; or (2) states are in a region that has been misclassified in previous training data.

We present an algorithm for online LfD that can actively decide whether it is necessary to ask the supervisor to label the new states that the robot encounters. 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, and tailored to high dimensional state spaces. We build on the One Class SVM method for approximating quantile level sets [31] 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 LfD and risk evaluation techniques.

Active Learning from Demonstration We formulate reducing supervisor burden in online Lfd as a stream-based active learning problem [5], [9]. 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 stream. In online LfD, this data stream is the states encountered when the robot is executing the current best policy.

Aside from the data stream, stream-based active learning has another ingredient: 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 [34], as in Fig. 2(a); or query by committee [6], which uses a committee of hypothesized estimators that are trained on different subsets of the training data. Risk in query by committee 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 LfD (see [8], [11] for the former and [15], [16] for the latter), online LfD 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 [7].

The problem lies in being estimator-centric. The estimator is a function of the current distribution. Therefore, it no longer provides a good measure of confidence when that distribution changes. And the distribution does change 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. Our experiments in Section VI-B.1 suggest that our measure of risk is more accurate in online LfD problems than estimator-centric measures.

Risk via Novelty Detection The main motivation for our risk definition is the notion that a trained model will be able to generalize within the distribution it is trained on [33], making states outside of this distribution risky. Novelty detection [13] is concerned with recognizing when this happens: recognizing that a sample is outside of this distribution.

Our work builds on Kim et al. [17] who apply novelty detection to safe online LfD to enable a supervisor to take over and prevent unsafe states. In their MMD-LI algorithm, risk is evaluated with Maximal Mean Discrepancy. For a single state, this is the sum of a kernel density estimate and the variance of the dataset, which works well in low dimensional state spaces. MMD-LI also reduces supervisor burden because the supervisor only takes over in risky states, as opposed to providing labels for all new states.

In contrast, we focus on active learning for online LfD in high-dimensional state spaces. Unlike safe learning, our problem does not require the supervisor to take over. This enables us to avoid the supervisor biasing the states sampled (which can in practice lead to a loss in performance [29]), and also avoid assuming the supervisor is always available. On the other hand, our problem does require novelty detection in high dimensional state spaces, where non-parametric density estimation does not scale very well: it can require data exponential in the dimension of the state space [22]. Our results (Section VI-B.1) suggests that in our high dimensional state spaces, SHIV achieves an error rate half of that of kernel density estimate-bases techniques like MMD.

An alternative to kernel density estimates is to measure distance to its nearest neighbors [20]. 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 [13].

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 [31]. 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 [36]. In [24], the One Class SVM has furthermore been used for novelty detection in high-

dimensional image data. To our knowledge, incorporating a query selection method for high-dimensional spaces and nonstationary distributions, such as the One Class SVM, into online LfD is a novel contribution.

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 dynamics occur when, given a state and a control, the probability of the next state does not change over time. Note this is different from the non-stationary distribution over the states the robot encounters during learning.

We model the initial state as sampled from a distribution over the state space. We assume a known state space and set of controls. We also 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 a control signal label. We additionally assume the supervisor can be noisy and imperfect, noting that the robot cannot surpass the level of performance of the supervisor.

Policies and State Densities. Following conventions from control theory, 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. 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 $t-1$. Following [29], we can compute the average density on states for any timepoint by $p(\mathbf{x}|\theta) = \frac{1}{T}\sum_{t=1}^{T} p(\mathbf{x}_t|\theta)$.

While we do not assume 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 $c : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ is typically user defined and task specific. For example, in the task of inserting a peg into a hole, a function on distance between the peg's current and desired final state is used [21].

In our problem, we do not have access to the cost function itself. Instead, we only have access to a supervisor that can achieve a desired level of performance on the task. The supervisor provides the robot an initial set 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 data set $\mathcal{D}$ of all state-control input pairs from the demonstrated trajectories.

We define a 'surrogate' loss function as in [29], $l : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$, which provides a distance measure between any pair of control values. In the continuous case, we consider $l(\mathbf{u}_0, \mathbf{u}_1) = ||\mathbf{u}_0 - \mathbf{u}_1||_2^2$, 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 surrogate loss relative to the supervisor's policy.

Following [29], our objective is to find 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:

$$\min_\theta E_{p(\mathbf{x}|\theta)}[l(\pi_\theta(\mathbf{x}), \tilde{\pi}(\mathbf{x}))] \tag{1}$$

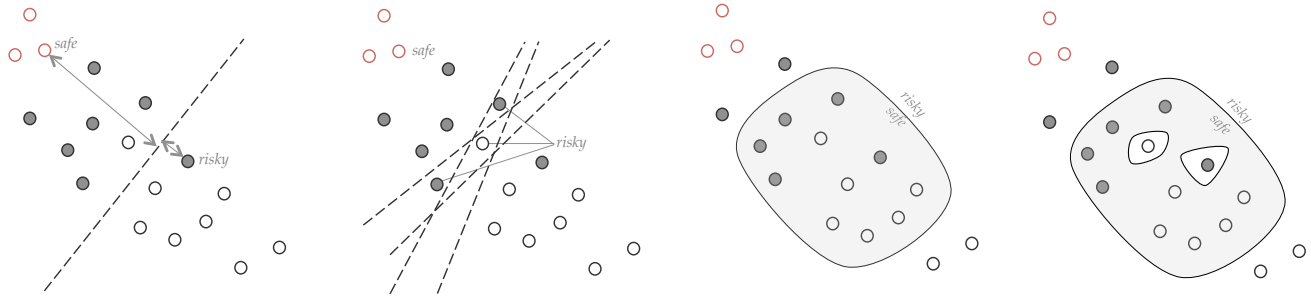
If the robot could learn the policy perfectly, this state density would match the one encountered in user examples. But if the robot makes an error, that error changes the distribution of states that the robot will visit, which can lead to states that are far away from any examples and difficult to generalize to [27]. This motivates iterative algorithms like DAgger, which iterate between learning a policy and 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. Risky and Safe States

Providing correct control inputs for (or "labeling") 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, SHIV uses a measure of risk to actively decide whether a label is necessary.

In contrast to the standard measure risk based purely on variance, we define a state as "risky" for 2 reasons: 1) it lies in an area with a low density of previously trained states, which can cause the current policy to mis-predict the supervisor and incur high surrogate loss [33], or 2) the surrogate loss, or training error, of the current policy at the state is high, so that the state is unlikely to model the supervisor's control inputs correctly. States that are not classified as "risky" are deemed "safe" .Our definition of risk can be visualized in Fig. 2(d).

Preprint submitted to 2016 IEEE International Conference on Robotics and Automation. Received September 15, 2015.



(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.

The amount of data needed to estimate the density, scales exponentially in the dimension of the state space [22]. Thus, to evaluate risk in high-dimensional state spaces, such as the HOG features in our driving simulator, Fig. 1(a), we use a modified version of the technique known as the One Class SVM that estimates a regularized boundary of a user defined quantile on the training data in $\mathcal{X}$ [31].

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

$$U(\gamma) = \inf\{\lambda(G) : P(G) \geq \gamma, G \in \mathcal{G}\}\ 0 < \gamma \leq 1 \qquad (2)$$

$\lambda : \mathcal{G} \to \mathbb{R}$ above denotes a volume measure. Suppose furthermore that $G : [0, 1] \to \mathcal{G}$ 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 $P(G(\gamma)) \geq \gamma$.

To handle distributions defined on high-dimensional spaces $\mathcal{X}$, work by Scholköpf et al. represents the class $\mathcal{G}$ via a kernel k as the set of half-spaces in the support vector (SV) feature space [31]. 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. 2.

Let $\Phi : \mathcal{X} \to \mathcal{F}$ denote the feature map corresponding to our exponential kernel, $k(\mathbf{x}_0, \mathbf{x}_1) = e^{-||\mathbf{x}_0 - \mathbf{x}_1||^2/2\sigma^2}$, 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 [31] determines a hyperplane in feature space $\mathcal{F}$ maximally separating the input data from the origin:

$$\underset{w \in \mathcal{F}, \xi \in \mathbb{R}, \rho \in \mathbb{R}}{\text{minimize}} \ \frac{1}{2}||w||^2 + \frac{1}{vn}\sum_i^n \xi_i - \rho \qquad (3)$$
$$\text{s.t } \langle w, \Phi(x_i) \rangle \geq \rho - \xi_i,\ \xi_i \geq 0.$$

Here, the parameter ν controls the penalty or 'slack term' and is equivalent to γ [36] in the quantile definition, Eq. 2, as the number of samples increases. The decision function, determining point membership in the approximate quantile levelset is given by $g(\mathbf{x}) = \text{sgn}(\langle w, \Phi(x) \rangle - \rho)$. 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 super-levelset and $g(x) = -1$ if x lies strictly in the quantile sub-levelset.

The dual form of the optimization yields a Quadratic Program that has worst case computational complexity of $O(n^3)$. However, Schölkopf et al. developed an improved optimization method that has empirically been shown to scale quadratically [31],which we use. In the dual, the decision function is given by $g(\mathbf{x}) = \text{sgn}(\sum_{i=1}^N \alpha_i k(\mathbf{x}_i, \mathbf{x}) - \rho)$ 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. 6 may still incur a large surrogate loss. To account for this, we propose a modification to the One Class SVM:

$$y_i = \begin{cases} 1 & : l(\pi_\theta(\mathbf{x}_i), \mathbf{u}_i) \leq \varepsilon \\ -1 & : l(\pi_\theta(\mathbf{x}_i), \mathbf{u}_i) > \varepsilon \end{cases} \qquad (4)$$

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: $\mathcal{D}_s = \{\{\mathbf{x}_i, \mathbf{u}_i\} \in \mathcal{D}_k, y_i = 1\}$ and those states incorrectly classified: $\mathcal{D}_r = \{\{\mathbf{x}_i, \mathbf{u}_i\} \in \mathcal{D}_k, y_i = -1\}$ 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:

$$g_\sigma(\mathbf{x}) = \begin{cases} 0 & : g_s(\mathbf{x}) == 1 \text{ and } g_r(\mathbf{x}) == -1 \\ -1 & : \text{otherwise} \end{cases} \qquad (5)$$

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. We treat σ as a "risk sensitivity" parameter (and study its implications in Section VI). For two reasons: 1)The expected number of examples, after a policy roll out, the supervisor can be asked is $T * \int_{\mathbf{x}} \mathbf{1}(g_\sigma(\mathbf{x}) == 0) p(\mathbf{x}|\theta) d\mathbf{x}$. Thus, smaller σ corresponds to asking for more examples. 2) A relation exists between how smooth the supervisor's policy,

Preprint submitted to 2016 IEEE International Conference on Robotics and Automation. Received September 15, 2015.

$\tilde{\pi}$ and how many examples are needed to learn it. Thus, a large σ can be dangerous for policies with sharp variation because it will treat points as safe that are really risky.

V. SHIV:SVM-BASED REDUCTION IN HUMAN INTERVENTION

Both SHIV and DAgger [29] solve the minimization in Eq. 1 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):

$$\theta_k = \arg\min_{\theta} \sum_{i=1}^{M} l(\pi_\theta(\mathbf{x}_i), \mathbf{u}_i). \quad (6)$$

This sub-problem is a supervised learning problem, solvable by estimators like a support vector machine or a neural net. Performance can vary though with the selection of a the estimator [32]

B. Step 2

The second step SHIV and DAgger rolls out their policies, π_{θ_k}, to sample states that are likely under $p(\mathbf{x}|\theta_k)$.

What happens next, however, differs between SHIV and DAgger. 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 $\tilde{\mathbf{u}}_t \sim \tilde{\pi}(\mathbf{x}_t) + \epsilon$, where ϵ is a small zero mean noise term, for $t \in \{0, \ldots, T\}$. The states and labeled controls are then aggregated into the next data set of demonstrations $\mathcal{D}_{k+1}$:

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

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

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

Steps 1 and 2 are repeated for K iterations or until the robot has achieved sufficient performance on the task[1].

VI. EXPERIMENTS

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 trained using Scikit-Learn [26]. 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.

We compare SHIV and DAgger in three domains: a driving simulator, push-grasping in clutter with a 4DOF arm, and surgical needle insertion using demonstrations from Dr. Douglas Boyd, a surgeon and professor at UC Davis. Each domain test different aspects of our algorithm: the driving simulator has a high dimensional visual state space, grasping in clutter has a human demonstrator provide the labels and is a challenging manipulation problem, surgical needle insertion uses data from a real robot.

We then 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 confidence, query-by-committee based methods, Maximum Mean Discrepancy, and the One Class SVM without the carving out misclassified holes modification. 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. 5).

A. Comparing SHIV with DAgger

We compare SHIV with DAgger on three domains: the driving simulator from above, push-grasping in clutter, and needle insertion. We hypothesize that SHIV achieves the same performance as DAgger, but by asking for fewer examples and thus reducing supervisor burden.

For each algorithm and budget of states labeled, we measure the normalized performance (e.g. 1 corresponds to matching the supervisor's performance on a task). Performance is a domain specific term, such as number of times the car crashes in the driving simulator.

1) Driving Simulator: Our first domain is a common benchmark in Reinforcement Learning: learning a policy for a car to drive around a track [4], [28], [29]. We implemented a driving simulator where the car must follow a polygonal track. We generate a polygonal track 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. 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 steering is $\mathcal{U} = \{-15°, 0, 15°\}$. A control input instantly changes the angle 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 is only allowed to search a finite amount a time ahead in the game and is prone to error, thus on averages crashes 5.9 times per lap.

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

[1]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 [12], [29]

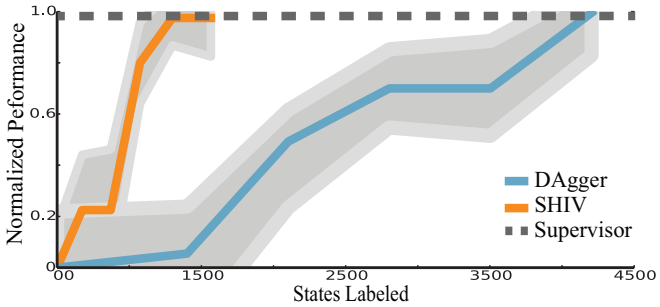


Fig. 3: We compare normalized performance (i.e. matching the performance of the supervisor) for the Driving Simulator , performance is defined as the number of times the car left the track versus the number of queries made to the supervisor. We plot the performance of DAgger and SHIV, the teal and orange lines. We also plot the performance of our supervisor, who makes non-optimal decisions and has an average performance of 5.9 crashes. Initial results, which are run for 6 iterations each and are averaged over 40 levels, shown in Fig. 3 suggest an 71% reduction in the number of queries needed for SHIV compared to DAgger. Error bars in grey measure the standard error on the mean

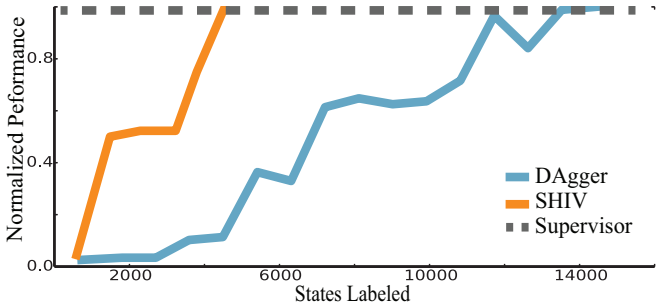


Fig. 4: We compare normalize performance (i.e. mathcing the performance of the supervisor) for the Grasping in Clutter domain, performance is defined as the sum of the number of objects knocked off the table plus 10 times the binary value indicating if the object is grasped or not. Initial results, which are averaged over 8 different trials suggest a 64% reduction in the number of queries needed for SHIV compared to 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 $\sigma_r = 200$, which are set via the grid search defined above.

In Fig. 3, 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 V of each algorithm over 40 different randomized polygonal tracks. Figure 3 presents averaged results from these experiments, suggesting a 71% reduction in the number of queries needed for SHIV compared to DAgger, in order to reach approximately the same performance as our non-optimal supervisor.

2) Grasping In Clutter in Box2D: We investigate having a human demonstrator control a simulated 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 [19], [18]. 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.

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 game 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 performance 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 performance 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 3% of the width of the table.

In Fig. 4 , we show the normalized performance 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 examples needed.

3) Surgical Needle Insertion: Suture tying in surgery, on a Robotic Surgical Assistant, 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 [23]. In this experiment, we are interested in learning to correct bad initial poses to the proper pose as shown in Fig. 2. Dr. Douglas Boyd, a surgeon and professor at UC Davis, provided us with a collection of demonstrations on a Intuitive Surgical Da Vinci Research Kit [14].

Dr. Boyd demonstrated a set 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

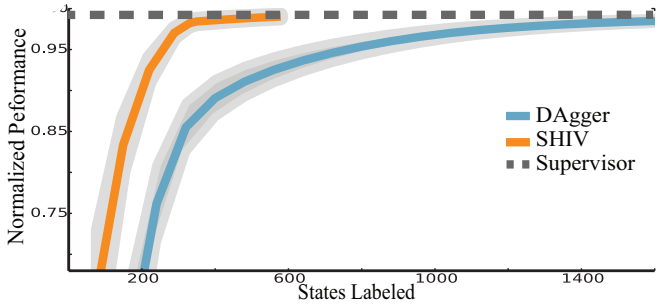


Fig. 5: We compare normalized performance (i.e. matching the performance of the supervisor), performance is defined as the euclidean distance between translation in centimeters. In Fig. 3, we plot the performance of DAgger and SHIV. Initial results, which are run for 20 iterations each and are averaged over 40 different initial starting positions, shown in Fig. 3 suggest an 67% reduction in the number of queries needed for SHIV compared to DAgger. Error bars in grey measure the standard error on the mean

lie algebra vector $c \in \mathbb{R}^6$ for $SE(3)$ and normalize it to the average magnitude of the control, $||\bar{c}_D||$, Dr. Boyd applied.

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. Performance 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.

In Fig. 4 , we show the normalized performance (i.e. 1 corresponds to matching the supervisor) 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.

B. SHIV Analysis

1) Comparison to active learning approaches.: We compare five active learning methods on high dimensional non-stationary state distributions. We first 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 then compare against another novelty detection method as well called Maximal Mean Discrepancy (MMD) from MMD-IL [17], which evaluates how close a point is to the distribution using kernel density estimate and adds the variance of the distribution in the dataset $\mathcal{D}$. MMD has been shown to work well in low dimensional online LfD tasks. We set bandwidth of the kernel as the same as our modified One Class SVM and the α decision boundary is set by sorting the MMD value of the states and picking an α such that the lowest 10% are marked as risky. This is in similar spirit to $\nu = 0.1$.

Risk Sensitivity (σ)	Normalized Perform.	States Labeled
1	1.0	4122
50	1.0	3864
150	1.0 9	1859
200	0.96	1524
250	0.94	1536
350	0.45	521

TABLE I: An analysis of the sensitivity risk sensitivity parameter,σ, of Eq. 5, or the decision function for risky or safe. SHIV was run 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 decreased performance.

We also compare against two baselines typically used in active learning. The first is confidence based on distance from the classification hyperplane [34] (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 last baseline is Query By Committee (Fig. 2(d)), which has a committee of different hypothesis estimators and points are marked risky if the committee disagrees. Our committee which was trained via bagging [6]. To obtain a committee, 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 in the driving simulator domain. 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.

Fig. 6 plots the performance for each query selection method, averaged over 50 tracks. We observe a significant performance improvement with methods based on classification based novelty detection compared to confidence,query by committee and MMD. Furthermore, using the combined measure of risk performs better than relying solely on the One Class SVM.

2) Sensitivity Analysis to Risk Sensitivity Parameter: To analyze the sensitivity of our method we varied the risk sensitivity parameter or σ of the decision function g_σ, or Eq. 5. 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. For each σ, we measure the final normalized performance of the policy π_θ after 6 iterations and the number of examples SHIV requested from the supervisor. 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. However, σ values between 150 and 250 all achieve similarly good performance, suggesting that SHIV is robust to the choice of a particular σ.

VII. Discussions and Future Work

Online LfD has been successful on an array of tasks [12], [29], however it can place significant burden on a supervisor. Our algorithm, SHIV, implements stream-based active learning with a query selection method that evaluates risk in a

Preprint submitted to 2016 IEEE International Conference on Robotics and Automation. Received September 15, 2015.

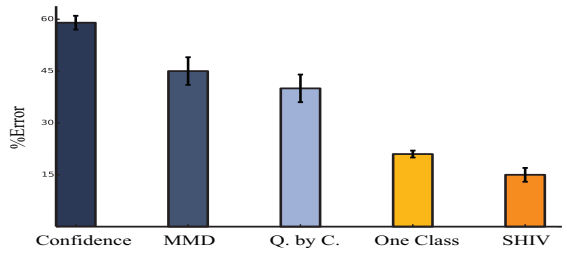


Fig. 6: A comparison of different active learning approaches in terms of the percentage of risky states that are estimated to be safe by the active learning technique during the first policy roll out. We compare against a confidence estimate of distance to hyperplane, Maximal Mean Discrepancy, query by committee for 3 hypothesis classifiers, the One Class SVM, and our modified One Class SVM, marked as SHIV. Results are averaged over 50 tracks and the policy π_θ is represented as a Linear SVM. Error bars measure the standard error on the mean.

manner tailored to non-stationary and high-dimensional state distributions. We empirically evaluated our method on three different domains: a driving simulator, grasping in clutter and surgical needle insertion. Results suggests up to a 70% reduction in the number of queries to the supervisor.

SHIV has several limitations. For instance, the selection of the value of σ can affect performance and a poor choice could result in the algorithm becoming offline imitation learning, as shown in Table I. Future work will look at better understanding the relationship of σ to the function π_θ and how to automatically select it.

Furthermore, to update the decision function g_σ requires solving a quadratic program, which has an upper bound computationally complexity of $O(n^3)$ in the number of data points. Fortunately, Scholköpf et al. provides an efficient optimization that empirically scales quadratically [31], which we use. Future work will also look at using techniques such as random features or PCA to allow for even better scaling [32].

Further analysis, video and code can be found at `http://berkeleyautomation.github.io/shiv`.

REFERENCES

[1] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng, "An application of reinforcement learning to aerobatic helicopter flight," *NIPS*, vol. 19, p. 1, 2007.

[2] P. Abbeel, D. Dolgov, A. Y. Ng, and S. Thrun, "Apprenticeship learning for motion planning with application to parking lot navigation," in *IROS 2008. IEEE/RS*. IEEE.

[3] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 1.

[4] 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.

[5] L. E. Atlas, D. A. Cohn, and R. E. Ladner, "Training connectionist networks with queries and selective sampling," in *NIPS*, 1990, pp. 566–573.

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

[7] 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.

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

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

[10] F. Duvallet, T. Kollar, and A. Stentz, "Imitation learning for natural language direction following through unknown environments," in *ICRA, 2013 IEEE*. IEEE, 2013, pp. 1047–1053.

[11] D. H. Grollman and O. C. Jenkins, "Dogged learning for robots," in *ICRA, 2007 IEEE*. IEEE.

[12] 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 *NIPS*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 3338–3346.

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

[14] Intuitive Surgical, "Annual report 2014," 2014.

[15] 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.

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

[17] B. Kim and J. Pineau, "Maximum mean discrepancy imitation learning." Citeseer.

[18] J. E. King, J. A. Haustein, S. S. Srinivasa, and T. Asfour, "Nonprehensile whole arm rearrangement planning on physics manifolds."

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

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

[21] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," 2015.

[22] H. Liu, J. D. Lafferty, and L. A. Wasserman, "Sparse nonparametric density estimation in high dimensions using the rodeo," in *International Conference on Artificial Intelligence and Statistics*, 2007, pp. 283–290.

[23] T. Liu and M. C. Cavusoglu, "Optimal needle grasp selection for automatic execution of suturing tasks in robotic minimally invasive surgery," in *ICRA, 2015*. IEEE, 2015, pp. 2894–2900.

[24] W. Liu, G. Hua, and J. R. Smith, "Unsupervised one-class learning for automatic outlier removal," in *CVPR, 2014 IEEE*. IEEE, 2014, pp. 3826–3833.

[25] A. Y. Ng, S. J. Russell *et al.*, "Algorithms for inverse reinforcement learning."

[26] 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.

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

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

[29] S. Ross, G. J. Gordon, and J. A. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," 2010.

[30] 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 *ICRA, 2013 IEEE*. IEEE.

[31] 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.

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

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

[34] 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.

[35] 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.

[36] 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.