Statistical Data Cleaning for Deep Learning of Automation Tasks from Demonstrations

Caleb Chuck[1], Michael Laskey[1], Sanjay Krishnan[1], Ruta Joshi[1], Roy Fox[1], Ken Goldberg[1,2]

***Abstract*— Automation using deep learning from demonstrations requires many training examples. Gathering this data is time consuming and expensive, and human demonstrators are prone to inconsistencies and errors that can delay or degrade learning. This paper explores how characterizing supervisor inconsistency and correcting for this noise can improve task performance with a limited budget of data. We consider a planar part extraction task (separating one part from a group) where human operators provide demonstrations by teleoperating a 2DOF robot. We analyze $30,000$ image-control pairs from 480 trajectories. After error corrections, trained CNN models show an improvement of 11.2% upon the baseline in mean absolute success rate.**

I. Introduction

Learning from Demonstrations (LfD) is a robot learning setting where the robot learns to perform a task using examples provided by a human supervisor [1]. By gathering human directed controls associated with sensor inputs as training data, one can fit a function to map sensor inputs to controls. Learning from demonstrations has applications in other domains, including navigation of unmanned vehicles [31], car driving [12], cart-pole swing-up [33] and peg-in-hole insertion [34].

One challenge for robotic leaning from demonstrations is that human demonstrations may be error prone. The literature has proposed several definitions for supervisor inconsistency, such as errors arising from human limitations or misinformed intent [4]. In this paper, we take supervisor inconsistency to mean to human actions which do not advance the agent towards completing the task, such as overshooting, or do so ineffectively, such as back-and-forth motion.

We study this challenge in a planar part extraction task. Planar part extraction involves the separation of one part from a group of parts, so that it can be individually identified or grasped. In contrast with part extraction by grasping and removing the individual part from within the group [27], pushing and separating the part from the group is useful in cases when grasp locations are difficult to determine, or the robot has constrained dynamics.

In machine learning, when confronted with large, possibly noisy datasets, various techniques, known as "data cleaning," are used to enforce the consistency and quality of the data. Algorithms such as ActiveClean [21] do this by iteratively locating outliers and providing corrections which enforce semantic, logical or physical constraints.

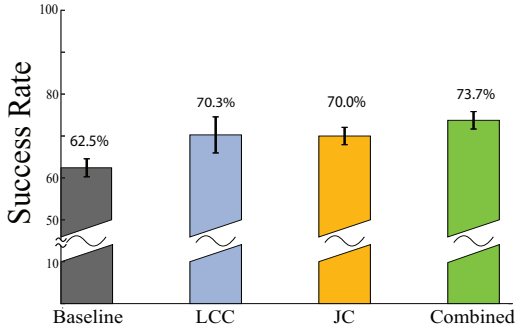

Fig. 1: Performance of the baseline, and the different data cleaning treatments. In order to highlight the differences in performance, a cut was made in the graph from 10%-50% on the Y axis. The baseline has 62.5% success rate, while after cleaning the data with the combined methods, the policy has a 73.7% success rate. LCC = low-confidence correction procedure. JC = jitter correction procedure.

In this paper, we study data cleaning techniques for learning from demonstrations. We use a dataset of human demonstrations of part extraction from part groups arranged in controlled patterns in a laboratory environment [3]. We propose data cleaning methods for two forms of inconsistency. After applying both methods, policies trained with the corrected data attain a 73.75% average success rate, an 11.2% absolute increase from the baseline of 62.5%.

II. Related Work

Prior literature in robotic LfD has proposed a variety of definitions of supervisor inconsistency. One work in human demonstration of game playing used overall task performance as a proxy for the inconsistency of human demonstrators [35]. In a driving domain, supervisor inconsistency has been described in [36] as oversteering and unstable steering motion. In the context of brain control interfaces, supervisor inconsistency has been measured by the delayed human response in a human-machine interface [37]. Human demonstrators can exhibit behavior where they misalign the robot effector with the task they are performing, and perform self-corrections [38].

Maximum-entropy inverse reinforcement learning [7] is a popular method for learning from demonstrations, which regularizes the inferred human intentions, thereby reducing overfitting to noise in human behavior. However, it requires complete knowledge of the system dynamics. This paper

All authors are with the AUTOLAB at UC Berkeley (automation.berkeley.edu). [1] EECS, [2] IEOR, UC, Berkeley, CA USA; {calebchuck, mdlaskey, sanjaykrishnan, rjoshi, royf, goldberg}@berkeley.edu

takes a Behavioral Cloning approach which does not assume any knowledge of the dynamics. We do assume full observability of the state, and use supervised learning on the state-action pairs in the demonstrations to regress directly from camera states to controls using a neural network.

Other methods for dealing with inconsistency include manual cleaning, where a human looks over the dataset and removes states that seem inconsistent [8], and real-time corrections, where a human provides corrective feedback to incorrect robot states so that the robot can learn from these states as well [9]. Alternatively, it is possible to train a probabilistic or predictive model of action confidence to identify action choices of low confidence and provide corrections [10], [12], or to smooth over multiple similar trajectories using clustering and averaging to reduce inconsistencies [13]. Another approach is to treat human demonstrations as inherently suboptimal and build into the robot policy cleaning heuristics that are resistant to this suboptimality [11]. However, to our knowledge, these methods are limited to low-dimensional inputs, and are not applicable to vision-based control.

In robotics, data cleaning and statistical techniques typically correct for constraints based on the physical limits of the robot, such as frequency response, voltage, and current [14]. Similarly, by data cleaning systems reports, one might reduce power usage by identifying and removing subsystem redundancy [15], or input data might need to be cleaned to use robotic sensors at a low level to perform better controls [16]. While in these cases the source of error is the sensing equipment, modeled as physical phenomena, our approach to data cleaning addresses scenarios where the human supervisor is the source of error.

In the field of data cleaning, several techniques consider human supervisors. These methods iteratively extract outliers, querying the human and learning a new model, until desired performance is achieved [17]. This process often involves outlier removal, state deduplication, and conversion of raw or incorrect featured data. Outliers and malformed data can be identified by pattern recognition and, when ambiguous, displayed to a user for correction [18]. Alternatively, data can be queried based on the value of information for correction [20], or sequentially by taking a gradient on the clean data to determine improvements [21]. In addition, Kubica et. al. suggests using a generative model to amend incorrect features [22].

In robotic LfD, many works reduce supervisor inconsistency by having the human provide only a small number (<10) demonstrations [39]. Other recent work has involved estimation algorithms for supervisor inconsistency in eye tracking [40] and pose estimation [41], where the sensing is expected to be fairly noisy. We are not aware of any studies that present data cleaning methods on sizable datasets.

III. Problem Formulation

Data cleaning studies the problem of efficiently identifying abnormal data, and when possible, automatically generating a correction. [17]. In this study, we wish to clean the dataset of human demonstrations to select actions that are more consistent with the task of part extraction, and we think of this consistency as the constraint to satisfy.

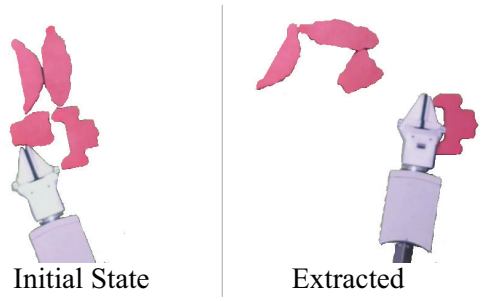

Fig. 2: Left: One of the initial states, which vary in position, order of parts, and pose. Right: the robot separates one part a significant distance from the group, at least 13cm from the center of the nearest part.

In our formulation of LfD, we want the robot to imitate the supervisor's control on the distribution over states with which the supervisor visits them. We model the system dynamics as Markovian, stochastic, and stationary. Stationary dynamics occur when the probability of the next state, given the current state and the control, does not change over time.

For our planar part extraction task, we use $250 \times 250 \times 3$ pixel RBG camera images as the states $x \in \mathcal{X}$, where $\mathcal{X} \subseteq \mathbb{R}^{d_x}$ is the set of possible such images, parametrized as a vector of dimension $d_x = 250^2$. Neural networks have been shown to successfully learn lower-dimensional featurizations, even of high-dimensional inputs [30].

As the control space, we use robot pose deltas (intended difference in pose): base rotation in the range $[-1.5°, 1.5°]$, and arm extension in the range $[-1\text{cm}, 1\text{cm}]$. We denote these deltas by the control $u \in \mathcal{U}$, and normalize it to lie in the box $\mathcal{U} \subseteq [-1, 1]^{d_u}$, parametrized by a vector of dimension $d_u = 2$. We use PID for pose control, after computing the new pose after applying the delta. This pose control is sufficient for pushing individual parts without significant error with 0.03cm extension error and 0.02 degree rotation error. When pushing multiple parts simultaneously, The force from the PID for a single time step may not be sufficient, but added over a few time steps, the robot can simultaneously push all four parts. The time step between deltas is sufficiently large such that oscillation due to the PID is negligible. A trajectory then consists of a sequence of image-delta pairs: $\tau_i = \{(x_t, u_t)\}_{t=1}^{T_i}$, where T_i is the duration of the trajectory. This duration is selected by the demonstrator as the time of task completion, but is capped at 100 time steps. A trajectory begins at some initial state from the initial state distribution defined below, and ends upon success in a state where a part is extracted.

Given this environment with fixed dynamics $p(x_{t+1}|x_t, u_t)$, the dataset D of trajectories is collected using the supervisor policy, which is a mapping from states to actions: $\tilde{\pi} : \mathcal{X} \to \mathcal{U}$. The dataset used in this paper is described in more detail in Section 4. The robot then fits a policy $\pi_{\theta|D} : \mathcal{X} \to \mathcal{U}$ for the dataset. Here θ represents

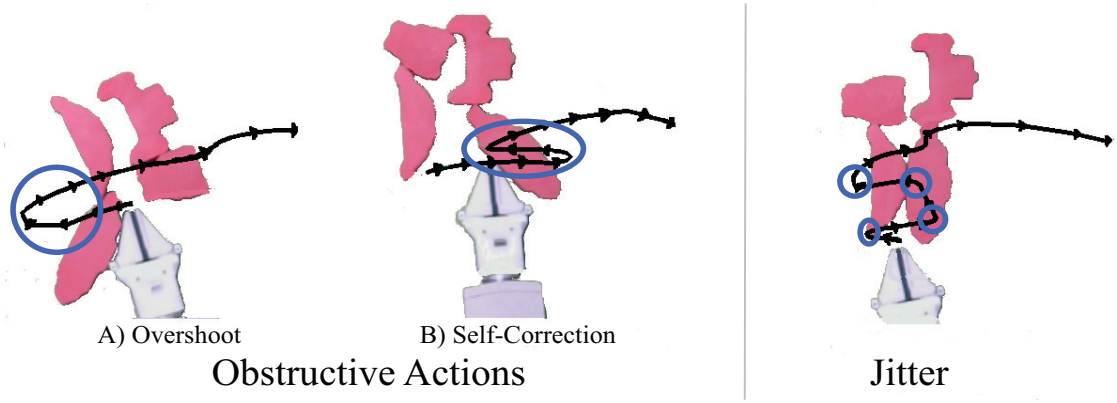

Fig. 3: Two categories of supervisor inconsistency. The goal is to move the **bottom-right part** to the right side of the workspace, apart from the other parts. On the trajectories, the parts of the trajectory **circled in blue** represent parts of the trajectory which we characterize as inconsistent. The left two images display examples of supervisor inconsistency from obstructive actions, while the right image represents jitter. The black line represents the path of the demonstrated trajectory, with supervisor inconsistency. The states shown are those with high cross validation error or jitter correction error, respectively. For cross-validation, notice that the first image contains overshoot, where the demonstrator moves left excessively. At the state shown, the task could be completed more succinctly by simply moving right. The other image contains self corrections, where the human moved for some time in one direction, but decided to backtrack after finding that would not be likely to succeed. Notice that in these cases, some motions move the robot further from completing the task. For the jitter correction case, the human makes multiple sudden changes resulting in an overly complex demonstration.

the weights of the neural network learned from the dataset D. We use the squared L^2 norm $\|u_R - u_H\|_2^2$ as the loss of choosing control $u_R = \pi_{\theta|D}(x)$ when the demonstrator chose $u_H = \tilde{\pi}(x)$.

The initial state distribution $p(x_0)$ is sampled by first drawing the translation of the cluster of parts from a Gaussian distribution centered 3 inches in front of the robot, with variance 20cm^2, then rotating it uniformly from the range $[-15°, 15°]$. The relative positions of the parts are arranged uniformly randomly. To guide a human operator in placing parts in their correct pose, we used a virtual overlay over a webcam image.

The distribution of trajectories induced by the robot policy is $p(\tau|\pi_{\theta|D})$, and distribution of actions given state and policy is: $p(u_t|x_t, \pi_{\theta|D}) = \delta_{u_t=\pi_{\theta|D}(x_t)}$. The transition probability is: $p(x_{t+1}|x_t, u_t)$, that is, the probability of getting to the following image given delta pose and current image.

We formalize the behavioral cloning problem for part singulation, starting with the probability of a trajectory given a policy π:

$$p(\tau|\pi) = p(x_0) \prod_{t=0}^{T-1} p(x_{t+1}|x_t, u_t) p(u_t|x_t, \pi_{\theta|D})$$

We wish to minimize the expected loss under the robot's policy:

$$\min_{\theta} \; E_{p(\tau|\pi_{\theta|D})} \sum_{t=0}^{T-1} \|\tilde{\pi}(x_t) - \pi_{\theta|D}(x_t)\|_2^2. \qquad (1)$$

This minimization is difficult because the trajectory distribution is coupled with the loss through the robot's policy, and the two need to be estimated jointly. Instead, as a common practice approximation [3], we can minimize the expected loss under the supervisor's policy:

$$\min_{\theta} \; E_{p(\tau|\tilde{\pi})} \sum_{t=0}^{T-1} \|\tilde{\pi}(x_t) - \pi_{\theta,D}(x_t)\|_2^2. \qquad (2)$$

When the supervisor's control is inconsistent with the task, optimization of Eq. III.2 becomes difficult [23], because the occasional inconsistent actions are at odds with the typically productive behavior towards the goal, providing different controls for similar states. In following sections, we characterize some of these inconsistencies that can be found in teleoperation systems, and introduce two methods for repairing them.

IV. Robotic Planar Part Extraction System

We used the dataset of demonstrations described in [3]. These trajectories were performed via teleoperation by 8 different human supervisors, each completing 60 episodes of planar part extraction.

We define success in the planar part extraction task as the separation of one part from its neighbors by a distance of at least 13cm between their centers, as illustrated in Figure 2, described in prior work [3], with a video at https://youtu.be/mMo4VSoM9S0. The human supervisors were instructed to move the part toward the upper right of the workspace, increasing the homogeneity of their behavior and rendering it easier to learn.

We used a planar 2-DOF robot arm with rotational and extensional control. The human teleoperated the robot using an xBox 360 controller, which applied end effector pose deltas through the right analog stick. This setup is a simplification of a factory environment, where planar part extraction might be one of several sub-tasks in a larger object manipulation task. Our robot was fitted with a 2-fingered gripper, that could potentially be used for grasping after part extraction.

The state images were captured by an overhead Logitech C270 camera, with sufficient field of view to observe the

parts and the robotic arm. The 4 parts were red extruded polygons, made of Medium Density Fiberboard with an average 4" diameter and 3" height. The trained policy was a deep neural network with the same architecture as defined in [3], [2]: a convolutional layer followed by two fully connected layers, separated by rectified linear units. The network was trained using TensorFlow [43].

We sampled 60 different initial states from the initial state distribution defined in Section III. Each of the 8 demonstrators performed 60 episodes with these same initial states. This data was repaired using the methods described in Section VI below, and a neural network with the same architecture as described before was trained on the repaired data. After training the network, we evaluated the robot policy performance on a fixed set of 30 initial states sampled from the same distribution as the training initial states. Execution performance was measured by rolling out a trajectory controlled by the neural network policy. We recorded the frequency of test episodes that terminated in successful planar part extraction.

In the original dataset, execution performance on the test states had a 62.5% success rate. Some common failure modes involve minimal motion, where the sweeping motion occurred with the arm not extended far enough, and exaggerated motion, where extension or approach overshot the desired part, pushing multiple parts without separating any one.

V. Two Error Mode Categories

In the dataset of human trajectories, we categorize two forms of supervisor inconsistency: obstructive actions and jitter. These error modes were common and pervasive in the dataset. They also relate to those observed in related work, with delayed human response [37] and misalignment [38] being related to obstructive actions, and unstable hand movements and oversteering [36] being related to jitter.

Obstructive actions: When a human performs a large number of demonstrations, occasionally his or her intended control is misaligned with his or her intended goal, and chooses actions that do not advance toward task completion (i.e. successful planar part extraction). This is often the case the human performs unproductive actions, such as overshooting the preferred trajectory, or starting a behavior too early, and is then forced to self-correct. If too many such actions are used, the human will fail at the task. Usually, however, the human will simply delay terminating the trajectory and perform self-corrective actions, eventually recording a successful demonstration. If the robot learns to imitate incorrect components of such behavior, its own behavior can be unsuccessful. See Figure 3 for some examples.

Jitter Another form of supervisor inconsistency involves human actions which are intended towards the goal, but are misperformed, over-complicating the trajectory. In contrast to obstructive actions, these inconsistencies involve shorter behaviors that do not necessarily make it harder for the human to complete the task, or extend the length of the trajectory. Instead jitter describes high-frequency motions that are harder to learn. Such actions often involve turns, changes in direction, or back-and-forth motions, which are overall aligned with the success of the policy, but contribute toward it inefficiently. These behaviors include short sequences of control steps which differ greatly from other controls demonstrated in similar state, which can impede learning.

VI. Data Cleaning

We introduce two algorithms for reducing overall noise, in correspondence with the two types of inconsistencies in Section V. Low-Confidence Correction aims to correct parts of the demonstration where the human performed obstructive actions, and Jitter correction attempts to reduce jitter.

A. Low-Confidence Correction

We expect the human to be mostly consistent, and obstructive actions to be outliers and thus low-confidence. However, not all low-confidence data points are due to inconsistency. In low-confidence correction, we use cross-validation to determine the confidence of the robot policy in its output in any given state, and query a human expert to determine corrections for low confidence states. We begin by breaking the dataset into $K+1$ subsets or "folds" of equally sized partitions of the data. We reserve one fold as the test set to evaluate overall performance. We then operate on the remaining K training folds in a manner similar to cross-validation [28]. We hold out each fold in turn, train a regression model using the remaining $K-1$ folds, and evaluate this model on the holdout fold.

The evaluation is performed by using a regression model to choose an action u_R for each state in the held out trajectories. These actions are compared to the demonstrated control in these trajectories u_H, computing their L^2 loss $\|u_R - u_H\|_2^2$, which we call the cross-validation error. This scheme allows us to identify states with high cross-validation error. The controller trained in cross-validation differs significantly from the demonstrator on these states, which indicates lower confidence that the human demonstrated consistent actions. Having identified these low-confidence states, they are then displayed to the expert for correction. The threshold error above which a query is made trades off false negatives that leave the data inconsistent, with false positives that burden the supervisor. In our experiments, we selected the error threshold to consider only 5% of the highest-error data for cleaning. As shown in Figure 4, this data represents significant outliers in terms of cross-validation error.

To clean a data point (a state's demonstrated control), we evaluate four different methods: 1) always remove the data point, 2) remove the data point based on the supervisor's discretion, 3) have the supervisor provide a corrected control 4) use the control from the policy trained on the other k-1 folds. Methods 1 and 2 are simple and effective in preventing wrong data from affecting training. Method 3 can improve upon 1 and 2 by not missing the opportunity to

Algorithm 1: Low-Confidence Correction

Data: State-action pairs $(x \in \mathcal{X}, u \in \mathcal{U})$ of human demonstrations D, number of folds K, threshold τ
Result: Equivalent dataset with selected corrections from the human supervisor at low-confidence states

```
Separate D into K + 1 folds;
Designate (K + 1)th fold as test set;
Set S contains remaining K folds, where S(i) = i'th fold;
Initialize CONTROLS as empty set;
for i = 0 to K do
    Separate TRAIN = S/{S(i)};
    Train policy on TRAIN;
    Evaluate policy on S(i) as NEW_CONTROLS;
    Append NEW_CONTROLS to CONTROLS;
end
for l in S do
    if ||CONTROLS(l) − D(l)||_2^2 > τ then
        Ask human for correction;
        Replace D(l) with correction, if specified;
    end
end
```

Algorithm 2: Jitter correction

Data: State action pairs $(x \in \mathcal{X}, u \in \mathcal{U})$ of human demonstrations D, number of folds K, threshold τ
Result: Equivalent dataset with deltas filtered using Butterworth Filter

```
Separate D into trajectories, T;
for t ∈ T do
    Extract ordered sequence of controls for t as L_t;
    Filter L by 5th order Butterworth filter as NL_t;
    Replace controls for states in t with NL_t;
end
```

demonstrate on low-confidence states. Method 4 has been suggested in prior work as a way to reduce learning error [19]. We describe this correction method in Algorithm 1.

Corrections were provided by an expert familiar with the operation of the robot and the algorithms being applied. We chose to use 12 folds of 5 trajectories per fold, with the 12^{th} fold being used for test set assessment. We chose 0.65 normalized error as the threshold error value; error values greater than 0.65 are displayed to the expert. We assume that the expert can choose actions for individual frames (time steps in a trajectory), particularly since the time steps represent non-negligible time windows (0.35 sec). The threshold was chosen at a point where the error begins to increase drastically with the points' error-based rank. Figure 4 displays the choice of threshold compared to the cumulative frequency of error values. We did not attempt to optimize the threshold value beyond the observed cross-validation errors in the training sets. We also did not attempt to optimize the number of folds.

B. Jitter correction

For jitter correction, we apply a low-pass filter to the control sequence of each trajectory. For the planar part extraction task, we selected a Butterworth filter with filter order 5. For policy training, we replace the demonstrated controls with the output of the low-pass filter. We chose not to burden an expert with correcting high-frequency inconsistency, because we expect most jitter noise to be corrected by reduction to low-frequency terms, making a low-pass filter more reliable than a human supervisor. This algorithm is described in Alg. 2.

Our choice of filter, and of filter order, was done without knowledge of full stack performance set. We choose the Butterworth filter due to its flat bandpass response, recognizing that we have no clear frequency above which we can consider control noise. We chose the filter order without optimization, using a value which was numerically stable. We did not attempt to optimize the choice of filter, because we can make only limited a-priori claims about the frequency spectrum which the noise occupies, and the Butterworth filter smooths out the data in a way appropriate to our usage. The Butterworth filter has the frequency response [24]:

$$|H(e^{j\omega})|^2 = \frac{G_0^2}{\sqrt{1 + \left(\frac{j\omega}{j\omega_c}\right)^{2n}}},$$

where n is the filter order, G_0 is the DC gain, and ω_c is the cutoff frequency. We used a cutoff frequency of 0.2 Hz, and a DC gain of 1.

	Jitter correction	LCC	Combined	Baseline
Training Error	9.87%	18.92%	10.89%	20.31%
Standard Error	0.29 %	0.28%	0.42%	0.41%

TABLE I: Difference in supervised learning training error, averaged over the 8 supervisors and the standard error for the different treatments, with fixed training iterations. LCC stands for low confidence correction. Jitter correction resulted in the largest change in training error

C. Combination of Methods

We combined both low-confidence correction and jitter correction to gauge how much overlap the two techniques carried. To do this, we first applied low-confidence correction and then applied jitter correction. We chose this order because smoothing might increase the complexity of providing human corrections, since changes to state deltas would make the trajectory harder to visualize, which is not a problem for jitter correction. Nonetheless, we consider changing the order in future work. After acquiring corrections using low-confidence correction, we pass the corrected controls through the low-pass filter as defined by the jitter correction operation. This new dataset is used to train the neural network to perform planar part extraction.

VII. Experiments

We present the resulting end-to-end performances from our treatments on the planar part extraction dataset, as well as some additional supporting results which may clarify possible forms of human inconsistency. In the planar part extraction dataset, our operations achieve significant improvements, and the final performance suggests the effectiveness of our treatments in this domain.

A. End-to-End Assessment of Cleaning Algorithms

Our primary assessment involved end-to-end testing of the robot policy. We took the trained neural network, and applied it to the set of 30 initial states defined in Section 4. The robot would perform the change in pose dictated by the output of the neural network for each of the 100 time steps. If the robot successfully extracted a part, then this would be a success. Figure 1 contains the results of applying treatments, as percentage change from the baseline, as well as the standard error on the differences. Applying our jitter correction and low-confidence correction techniques together resulted in a 11.2% improvement on the test set over the baseline performance of average 62.5% successful extraction, with a 2% standard error. Notice that though using both methods together does the best of all methods, the average success rate of the data is still only 73.7%, probably due to the small size of the training set.

Drawing from the results of Figure 1, notice that jitter correction and low-confidence correction performed comparably. Jitter correction provided a 7.5% absolute improvement, with an average performance of about 70% and standard error of 2%, a sizable improvement with significant confidence over the mean. Applying the low-confidence correction method resulted in a 7.8% absolute improvement, with a standard error of 4%. While this standard error is slightly higher than other methods, it still implies a significant improvement. In addition, the low-confidence correction end-to-end performance demonstrates how a small number of changes, only about 5% of the data, can result in significant improvements to the overall planar part extraction policy. We suggest that this significant change occurs due to the high error incurred by these states, biasing the overall policy. This result also suggests that obstructive actions tends to result in a relatively small number of extremely high error states.

B. Noise Characterization

Observing Figure 3, notice that shown trajectories with high jitter correction error also contain significant back-and-forth motion. Intuitively, this redundant and non-reproducible data, which adds unneeded complexity to the trajectory, would be difficult to learn. This is supported in I, which shows the training errors of the different treatments. Jitter correction produces a 10% decrease in training loss for the same number of iterations of training, with a standard error of .4%.

Figure 3 also shows trajectories which contain significant numbers of states with high cross-validation error. In the figure, these states occur when the human performs a motion in the trajectory which goes opposite or overshoots the general rightward motion. This is consistent with our description of obstructive actions. We only observe in I a very slight decrease in training error. However, we suggest this is because a much smaller subset of states is corrected in the low-confidence correction, and corrections are intended to make the true policy more representative, rather than just improve learnability. We assess errors of this form by noticing the qualitative nature of errors, as obstructive actions, and the quantitative improvements to test set performance after correcting such errors.

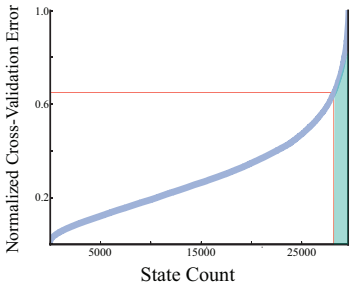

Fig. 4: Sorted normalized cross-valiation errors for all data points. The horizontal line indicates the threshold used for bias correction, which was .65 normalized error. The shaded region represents about 5% or 1,500 out of 29,610 data points, sent to the expert for corrections.

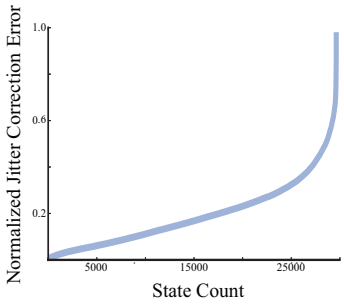

Fig. 5: Sorted normalized difference in control due to jitter correction. Notice that jitter correction alters a significant number of states, suggesting a good number of high frequency components. However, its correction is generally not as high as the cross-validation error.

To continue highlighting the proposed forms of human inconsistency, we demonstrate the cross-validation differences (as an estimate of prediction error). Figure 4 displays this estimated prediction error. That is, the graph shows the sorted L^2 difference between the cross-validation control and the original supervisor control, for all of the frames.

Figure 5 displays the amount of change to the control produced by jitter correction, that is, the loss value when comparing the control after jitter correction, with the control prior to jitter correction. Notice that a significant portion of states have greater than 0.5 normalized error, both for jitter correction and for cross validation. For jitter correction, this suggests human trajectories contain high frequency motions, which when corrected improves learnability. For cross-validation, this highlights high-error outliers that may indicate supervisor inconsistency.

C. Alternative low-confidence correction methods

In the results described by Figure 6, we notice that the alternative treatments for obstructive actions produce no significant improvements, especially when compared with the suggested low-confidence correction. Figure 6 also

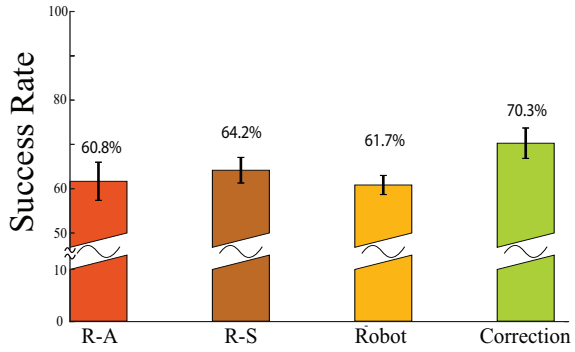

Fig. 6: Comparison of different procedures for correcting obstructive actions. In order to highlight the difference between low-confidence correction and the other techniques, a cut was made in the graph from 10%-50% on the Y axis. **R-A:** (remove-all) removal of all data points with cross-validation error above the threshold; **R-S:** (remove-selected) selective removal of data points, where the human decides which data points to remove; **Robot:** changing the data points to the ones determined by the robot policy; **Correction:** sending data points to the human for correction, which is the procedure we used for final analysis.

shows the difference between the suggested treatment, low-confidence correction, and other treatments. Our results are drawn by a smaller sample size and so this analysis only forms a preliminary study. We explored three alternative treatments: 1) Removing all states with low confidence that would have been queried, 2) Altering all high error states to the robot's policy 3) Having the human select which low-confidence states to remove. All three treatments had negligible effect on the execution performance, with none greater than 2% change from the baseline. The lack of significant change compared to the baseline for removing or altering the high error states to the robot's policy suggests that in fact high error states are not all misrepresented: some states with high prediction error are naturally low-confidece, and not due to human error, but simply because parts of the state space have higher variance. On the other hand, the fact that human-specified removal of certain states also had limited effect suggests that it is not sufficient to remove offending data to get significant improvements, as exhibited by low-confidence correction.

D. Demonstrator Experience

We are also interested in finding if experience has a significant effect on performance. To test this, we measured cross-validation and jitter correction error as a function of the number of demonstrations, to see if the human would have more learnable data as they gained more experience. The results demonstrated in 7 suggest that experience is not a particularly powerful factor in the number of erroneous actions, at least in our experiment. This is shown by the near-zero slope of both correlations.

VIII. Conclusion

To enhance LfD, this paper introduces several characterizations of supervisor inconsistency in a planar part

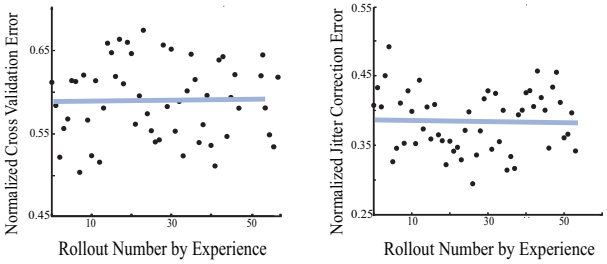

Fig. 7: Performance over each of the 60 example rollouts. Notice that there is almost zero correlation, which suggests that for this experiment, experience was not a confounding variable in performance measured with regards to jitter or obstructive actions. The left plot represents low-confidence correction, while the right plot represents jitter correction.

extraction task, as well as different techniques by which to determine and provide corrections for inconsistent human demonstrations. The dataset, coming from people inexperienced with operating the particular robot platform we used, certainly included many suboptimal cases, noticeable by inspection.

This work only assesses the supervisor inconsistency in planar part extraction. In future work, we will explore more complex tasks, using the intuitions and ideas from jitter correction and low-confidence correction. We hope to explore extending data cleaning techniques to other platforms and populations of robot trainers, at greater data scale. Some recent work has taken the applied ideas of jitter correction observed in this work, and applied them to train ball-scooping from teleoperation of a YuMi system, with significant success [42]. This suggests that these techniques may generalize to other tasks. As LfD involves more data, ensuring the integrity of this data will allow tasks to be learned with higher success rates.

IX. Acknowledgements

The research for this project was performed in UC Berkeley's Automation Sciences Lab under the UC Berkeley Center for Information Technology in the Interest of Society (CITRIS) "People and Robots" Initiative. The Authors were supported in part by the Berkeley Deep Drive (BDD) Program, as well as by the National Science Foundation (NSF) and the DOD through the National Defense Science And Engineering Graduate Fellowship (NDSEG) program. Special thanks to Steve McKinely, David Gealy and Meng Guo for help in initial construction and maintenance of the hardware setup.

References

[1] Argall, Brenna D., et al. "A survey of robot learning from demonstration." Robotics and autonomous systems 57.5 (2009): 469-483.
[2] Laskey, Michael, et al. "Robot Grasping in Clutter: Using a Hierarchy of Supervisors for Learning from Demonstrations."
[3] Laskey, Michael, et al. "Comparing Human-Centric and Robot-Centric Sampling for Robot Deep Learning from Demonstrations." arXiv preprint arXiv:1610.00850 (2016).

[4] Friedrich, Holger, Michael Kaiser, and RÃijdiger Dillmann. “What can robots learn from humans?.” Annual Reviews in Control 20 (1996): 167-172.

[5] Delson, Nathan, and Harry West. “Robot programming by human demonstration: The use of human inconsistency in improving 3D robot trajectories.” Intelligent Robots and Systems’ 94.’Advanced Robotic Systems and the Real World’, IROS’94. Proceedings of the IEEE/RSJ/GI International Conference on. Vol. 2. IEEE, 1994.

[6] Ross, Stéphane, Geoffrey J. Gordon, and Drew Bagnell. "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning." AISTATS. Vol. 1. No. 2. 2011.

[7] Ziebart, Brian D., et al. “Maximum Entropy Inverse Reinforcement Learning." AAAI. 2008.

[8] Friedrich, Holger, and Rudiger Dillmann. “Robot programming based on a single demonstration and user intentions.” 3rd European workshop on learning robots at ECML. Vol. 95. 1995.

[9] Argall, Brenna D. Learning mobile robot motion control from demonstration and corrective feedback. Diss. University of Southern California, 2009.

[10] Chernova, Sonia, and Manuela Veloso. “Interactive policy learning through confidence-based autonomy.” Journal of Artificial Intelligence Research 34.1 (2009): 1.

[11] Chernova, Sonia, and Manuela Veloso. “Learning equivalent action choices from demonstration.” 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2008.

[12] Chernova, Sonia, and Manuela Veloso. “Confidence-based policy learning from demonstration using gaussian mixture models." Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems. ACM, 2007.

[13] Aleotti, Jacopo, and Stefano Caselli. “Trajectory clustering and stochastic approximation for robot programming by demonstration.” 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2005.

[14] Williamson, James, et al. “Data sensing and analysis: Challenges for wearables." The 20th Asia and South Pacific Design Automation Conference. IEEE, 2015.

[15] Wang, Li, et al. “Data cleaning for RFID and WSN integration." IEEE Transactions on Industrial Informatics 10.1 (2014): 408-418.

[16] Mahler, Jeffrey, et al. “Learning accurate kinematic control of cable-driven surgical robots using data cleaning and gaussian process regression.” 2014 IEEE International Conference on Automation Science and Engineering (CASE). IEEE, 2014.

[17] Chu, Xu, et al. "Data cleaning: Overview and emerging challenges." Proceedings of the 2016 International Conference on Management of Data. ACM, 2016.

[18] Chu, Xu, et al. “Katara: A data cleaning system powered by knowledge bases and crowdsourcing." Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. ACM, 2015.

[19] He, He, Jason Eisner, and Hal Daume. "Imitation learning by coaching." Advances in Neural Information Processing Systems. 2012.

[20] Yakout, Mohamed, et al. “Guided data repair." Proceedings of the VLDB Endowment 4.5 (2011): 279-289.

[21] Krishnan, Sanjay, et al. "Activeclean: Interactive data cleaning while learning convex loss models." arXiv preprint arXiv:1601.03797 (2016).

[22] Kubica, Jeremy, and Andrew W. Moore. “Probabilistic noise identification and data cleaning." ICDM. 2003.

[23] Cohn, David A., Zoubin Ghahramani, and Michael I. Jordan. “Active learning with statistical models." Journal of artificial intelligence research (1996).

[24] Bianchi, G. (2007). Electronic Filter Simulation & Design. New York City: McGraw Hill Professional.

[25] Hubens, G., et al. "A performance study comparing manual and robotically assisted laparoscopic surgery using the da Vinci system." Surgical Endoscopy and other interventional techniques 17.10 (2003): 1595-1599.

[26] Howard, Ayanna M., and Chung Hyuk Park. "Haptically guided teleoperation for learning manipulation tasks." Georgia Institute of Technology, 2007.

[27] Kaipa, Krishnanand N., et al. "Automated plan generation for robotic singulation from mixed bins." Workshop on Task Planning for Intelligent Robots in Service and Manufacturing. 2015.

[28] Efron, Bradley, and Gail Gong. "A leisurely look at the bootstrap, the jackknife, and cross-validation." The American Statistician 37.1 (1983): 36-48.

[29] LeCun, Yann, et al. "Off-road obstacle avoidance through end-to-end learning." NIPS. 2005.

[30] Coates, Adam, Honglak Lee, and Andrew Y. Ng. "An analysis of single-layer networks in unsupervised feature learning." Ann Arbor 1001.48109 (2010): 2.

[31] Ross, Stéphane, et al. "Learning monocular reactive uav control in cluttered natural environments." Robotics and Automation (ICRA), 2013 IEEE International Conference on. IEEE, 2013.

[32] Laskey, Michael, et al. "Shiv: Reducing supervisor burden in dagger using support vectors for efficient learning from demonstrations in high dimensional state spaces." Robotics and Automation (ICRA), 2016 IEEE International Conference on. IEEE, 2016.

[33] Judah, Kshitij, et al. "Imitation Learning with Demonstrations and Shaping Rewards." AAAI. 2014.

[34] Dillmann, RÃijdiger, M. Kaiser, and Ales Ude. "Acquisition of elementary robot skills from human demonstration." International symposium on intelligent robotics systems. 1995.

[35] Taylor, Matthew E., Halit Bener Suay, and Sonia Chernova. "Integrating reinforcement learning with human demonstrations of varying ability." The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[36] Chen, Jessie YC, Ellen C. Haas, and Michael J. Barnes. "Human performance issues and user interface design for teleoperated robots." IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 37.6 (2007): 1231-1245.

[37] Parra, Lucas C., et al. "Response error correction-a demonstration of improved human-machine performance using real-time EEG monitoring." IEEE transactions on neural systems and rehabilitation engineering 11.2 (2003): 173-177.

[38] Delson, Nathan, and Harry West. "Robot programming by human demonstration: Adaptation and inconsistency in constrained motion." Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on. Vol. 1. IEEE, 1996.

[39] Havoutis, Ioannis, and Sylvain Calinon. "Supervisory teleoperation with online learning and optimal control." Proc. IEEE Intl Conf. on Robotics and Automation (ICRA), Singapore. 2017.

[40] Penkov, Svetlin, Alejandro Bordallo, and Subramanian Ramamoorthy. "Physical symbol grounding and instance learning through demonstration and eye tracking." Robotics and Automation, 2017 IEEE International Conference on, Singapore. 2017.

[41] Schultz, Christopher, et al. "Goal-Predictive Robotic Teleoperation from Noisy Sensors."

[42] J. Liang, J. Mahler, M. Laskey, P. Li, and K. Goldberg, âĂIJUsing dVRK Teleoperation to Facilitate Deep Learning of Automation Tasks for an Industrial Robot,âĂİ in Automation Science and Engineering (CASE), 2017 IEEE International Conference on. IEEE, 2016

[43] “Tensor flow,” https://www.tensorflow.org/