Statistical Data Cleaning for Deep Learning of Automation Tasks from Demonstrations

Caleb Chuck, Michael Laskey, Sanjay Krishnan, Ruta Joshi, Ken Goldberg

***Abstract*— Recent advances in learning can facilitate automation, at the cost of many training examples. Gathering this data is time consuming and expensive, and human demonstrators are prone to inconsistencies and errors that can delay or prevent learning. In this paper we explore how new methods for data cleaning can be applied to characterize and correct errors. We consider a planar part extraction task (separating one object from a group) and provide demonstrations using a 2DOF robot. We analyze almost 30000 data points in 480 trajectories from 8 human subjects to discover ways to characterize label noise. After providing corrections for the errors, our experimental results show an absolute improvement of 11.2% from the baseline which is noteworthy for the task.**

I. Introduction

One automation task of interest is planar part extraction. In the factory, planar part extraction involves the separation of the parts so that they can be disambiguated during a picking or transport phase. While some research has been done in planar part extraction by grasping and removing individual parts from their cluttered state [27], robot arm manipulators that must push around objects to exhibit behavior which separates one part from the rest may be quite complex. Such actions could be useful in cases where the objects are difficult to distinguish, or obscuring each other.

Learning from Demonstration (LfD), where the robot learns to perform a task using examples provided by a human supervisor [1], promises intuitive and robust methods to perform tasks like planar part extraction. By gathering sensor inputs like camera feed or relevant features, and corresponding human directed controls to the robot as training data, one can fit a regression function to map sensor inputs to human tele-operation controls during demonstration. Research has shown that applying this approach to various robot tasks has given good success rates for a wide variety of tasks [1]. In these and many other applications of learning from demonstration, tele-operation human controls were used, due to their widespread popularity for robot control, and their flexibility with equipment and design. In addition, supervised learning for learning from demonstration has seen applications in many other domains, including navigation of unmanned vehicles [32], simulated car driving [12], cart-pole [34] and peg in hole insertion [35].

One drawback of robotic leaning from demonstration is that human demonstrations may be flawed. Inconsistent data in robotic leaning from demonstration has been categorized in literature as unnecessary, ill intentioned, or unmotivated actions, where unmotivated actions are due to limitations in the robot state space [4]. While it is possible to enforce on the human some form of consistency in demonstrations, such as by creating a protocol for obstacle avoidance driving [30], it may still be necessary to perform algorithmic post processing to correct human inconsistency after the fact, rather than trying to enforce it in demonstrations. This is because humans may unintentionally neglect protocols. In addition, assuming consistency prevents the use of pre-existing datasets for learning.

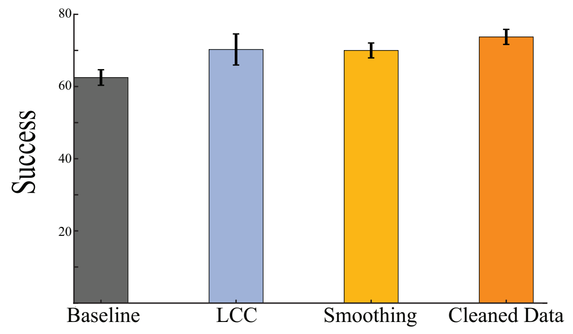

Fig. I.1. Performance of the baseline, and the different data cleaning treatments. The baseline has 62.5% success rate, while after cleaning the data, the policy (Combined) has a 73.7% success rate. LCC represents results from the low-confidence correction procedure.

In machine learning, when confronted with large, possibly noisy datasets, various techniques, known as "data cleaning," are used to ensure the consistency and quality of the database. This includes enforcing knowledge of the domain regarding physics, from the human control interface, or from work space limits. In robotic LfD, where data is expensive, we want to know whether these techniques can be used to improve task performance with a fixed budget of demonstrations. However, inconsistencies in a LfD dataset do not occur due to improper formatting, but human misinterpretation and error [3], [6]. Thus we extend data cleaning to the learning from demonstration domain.

In this paper, we draw from a dataset [3] of human demonstrations in a laboratory environment. This dataset has controlled arrangement and distribution of the parts, as well as processed perception systems. Our key contributions are: 1) characterizing different forms of human inconsistency as obstructive actions and jitter, 2) providing correction methods for each of these forms of human inconsistency. After we perform both treatments, we attain a 73.75% average success rate, which is a 11.2% absolute increase from the baseline, at 62.5%.

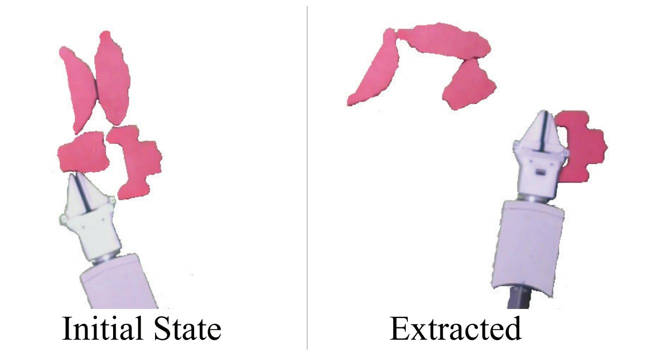

Fig. I.2. Left: an example initial state, with varied position, order of blocks, or pose. Right: the robot learns to separate one object a significant distance from the pile, at least 10cm from the center of the nearest block

II. Related Work

Human inconsistency in robotic teleoperation systems has been assessed by some metrics before. Previous works has used performance metrics to assess the level of human demonstrators, especially in game playing [36], which was indicative of final performance. Sources of human inconsistency has been specified in [37], where human loss of attention resulted in oversteering and difficulty in stabilizing hand movements. In addition, human-machine interfaces often result in delayed human response [38]. Finally, human demonstrators can exhibit behavior where they misalign with the task they are performing, and perform self-corrections [39]. In work with human demonstrators for teleoperation, human inconsistency has often been characterized as high frequency hand movements, self-correction and overshoot.

Maximum entropy inverse reinforcement learning [7] is a popular method in learning from demonstration used in order to protect the learner against human inconsistency. However, it relies upon some requirements of reinforcement learning, particularly full system dynamics which are made use of in planning. Thus we choose to represent a more simplified problem. We assume full observability, and regress directly from camera states to controls using a neural network, assuming that our states have no dependence on each other. Nonetheless, probabilistic methods for determining the degree of human demonstration suboptimality, which is used in maximum entropy inverse reinforcement learning, remains an interesting field.

Other methods for dealing with inconsistency include dataset reduction, where the human looks over the dataset and removes states that he thinks are inconsistent [8] and real-time corrections, where the 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 action choices with confidence or with a probabilistic model, to identify action choices of low confidence and provide corrections [10], [12], or smooth over multiple similar trajectories using clustering and averaging, thus reducing inconsistencies [13]. Another method treats human demonstrations as inherently suboptimal and performs cleaning heuristics that encode suboptimality resistance into the robot policy [11]. However, to our knowledge, these methods have not been extended to image-to-state regression. Their use case often involves classification of action choices with low-dimensional inputs, and thus they do not address the question of high dimensional image data, or create confidence with regression policies.

In robotics, data cleaning and statistical techniques to provide corrections generally involve the application of physically determined heuristics, such as frequency response, voltage, and current [14]. Additionally, robotic data cleaning might involve reductions in power usage through removal of redundancy [15]. Finally, input data might need to be cleaned to use robotic sensors at a low level to perform better controls [16]. However, in these cases, the source of error is from the sensing equipment, modeled by physical phenomena. In our application of data cleaning, we address questions where the human is inconsistent.

In the field of data cleaning, several techniques involve human supervisors. These methods involve iteratively extracting, querying the human and learning from the data, until desired performance is achieved [17]. This process often involves outlier removal, removal of duplicate states, and conversion of raw or incorrect featured data. Outliers and badly formed data can be located using patterns in the data 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 correct broken features [22]. Data cleaning tends to address the question of inconsistent data as violating some database. However, cleaning robotic demonstration data generally focuses on making demonstrations easier to learn by correcting inconsistent human demonstrations, and involving the human to disambiguate ambiguous cases.

We have not observed any studies that present data cleaning methods to identify hard-to-learn states or to receive corrections. However, it is also worth noticing that the techniques we developed for this project could be augmented with prior work in data cleaning and addressing inconsistency in LfD, to achieve stronger results.

III. Problem Formalism

We define data cleaning as adherence to a set of functional constraints on the human policy [17]. In this study, we enforce the efficiency of human actions toward successful extraction.

For our formulation of LfD, we want the robot to emulate the supervisor on the distribution over states which the supervisor visits, and we apply this directly to the planar part extraction problem. We model the system dynamics as Markovian, stochastic, and stationary. Stationary dynamics occur when the probability of the next state does not change over time, that is, with the robot and objects in a particular configuration, the next action should be consistent.

For our planar planar part extraction formulation, we use 250×250 pixel RBG camera images as the states $x \in \mathcal{X}$, where $\mathcal{X} \in \mathbb{R}^{d_x}$ is the set of all images possible when performing this task, parametrized as a vector in d_x.

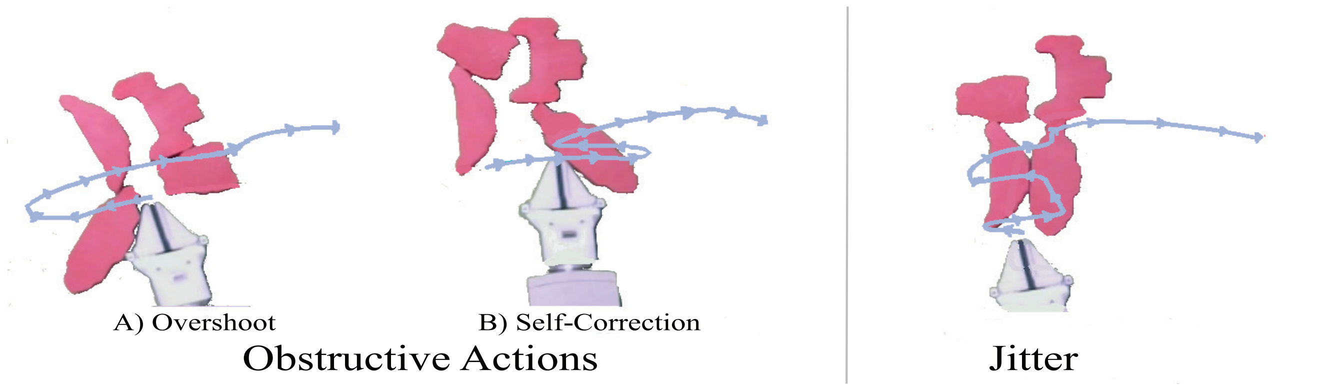

Fig. I.3. The images are example states of different categorizations of human inconsistency, which were taken from the training set of human demonstrations of planar part extraction, where the goal is to move the front-right object only to the right side of the workspace. The left two images display examples of human inconsistency from obstructive actions, while the right image represents jitter. The blue line represents the motion of the demonstrated trajectory, with human 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 excessively moves left, though 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 they would not be likely to succeed. Notice that in these cases, there is a significant component that puts the human further from completing the task. For the jitter correction case, the human makes multiple sudden changes to their trajectory, superfluously, resulting in an overly complex demonstration.

Because we use whole images, this space is large, though neural networks have been shown to learn lower dimensional features [31].

We have the label/control space of small delta poses controlling base rotation $[-1.5°, 1.5°]$ and arm extension $[-1cm, 1cm]$ of the robot, formalized as $u \in \mathcal{U}$, where $\mathcal{U} \in \mathbb{R}^{d_u}$ is the set of possible controls, where controls are parametrized by a vector of dimension $d_u = 2$. We normalized $\mathcal{U}$: $\mathcal{U} \in [-1, 1]^{d_u}$. We used PID pose control, which controls the position of the arm. The time step between deltas is sufficiently large such that "ringing" as a result of the PID is negligible. A trajectory then consists of a sequence of image-delta pose pairs: $\tau_i = \{(x_t, u_t)\}_{t=1}^{T_i}$, where T_i is the duration of the trajectory, which we capped at 100 time steps, though the demonstrator could have chosen to end earlier, if the demonstrator adequately performed the task. A trajectory begins at some initial state from the initial state distribution, and ends at an extracted state.

The dataset D of trajectories is collected based on the supervisor policy, which is a mapping from states to actions: $\tilde{\pi} : \mathcal{X} \rightarrow \mathcal{U}$. The dataset used for this paper is described in more detail in Section 4. The robot then fits a policy onto the dataset: $\pi_{\theta|D} : \mathcal{X} \rightarrow \mathcal{U}$, where $\pi_{\theta|D}(x) = u : \forall x \in \mathcal{X}$. Notice that θ represents the weights of the neural net learned from the dataset D. We use the l2 norm: $\|u_1 - u_2\|_2^2$ for the loss function $l : \mathcal{U} \times \mathcal{U} \rightarrow [0, 1]$.

The initial state distribution $p(x_0)$ samples the translation of the cluster as a Gaussian distribution centered 3 inches in front of the robot, with variance 20cm and rotation selected uniformly from range $[-15°, 15°]$. The relative positions of the objects are chosen randomly. To help a human operator place objects in the correct pose, we used a virtual overlay over the webcam.

The distribution of trajectories under the robot is $p(\tau|\pi_{\theta|D})$, and distribution of actions given state and policy is: $p(u_t|x_t, \pi_{\theta|D})$. 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.

In order to characterize what we hope our data cleaning will improve in the supervised learning for the planar part extraction problem, we hope to decompose the probability of failure to include a term dependent on label inconsistency. We begin with the probability of a trajectory given a policy:

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

From the probability above, we minimize the expected difference between the human demonstrator and the robot, which has the form:

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

This minimization is difficult because it is hard to approximate the distribution implied by the trained robot, while also optimizing the controls applied. It has been shown in literature that, we can instead minimize the expected loss over the distribution of the supervisor, as an upper bound [3]:

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

When the supervisor is inconsistent, optimization of Eq. III.1 becomes difficult [23]. Intuitively, when attempting to learn from multiple different labels for similar states, the desired behavior will be harder to learn. We introduce ways to reduce types of inconsistency found in a teleoperation system.

IV. Robotic Planar Part Extraction System

We used a dataset of teleoperated examples collected in [3]. These teleoperated demonstrations were performed by 8 different subjects, each performing 60 examples of planar part extraction with teleoperation.

We define the planar part extraction task as follows: separate an object from its neighbors by a least 10cm from the centers, as illustrated in Figure I.2, described in prior work [3]. In our training set, we instructed the demonstrators to move the object to the front right of the robot, toward the upper right of the workspace. This allowed demonstrations to be more consistent and thus easier to learn.

We used a planar 2-DOF robot arm with rotational and extensional control. The human teleoperated the robot using an xBox controller, which applied all pose deltas through the right analog stick. The state images were captured by an overhead Logitech C270 camera, with sufficient field of view to observe the objects and the robotic arm. The 4 objects are red extruded polygons, made of Medium Density Fiberboard with an average 10cm diameter and 7.5cm height. The policy is a deep neural network using the same architecture as defined in [3], [2], trained using TensorFlow [40].

From the initial state distribution defined in the Problem Statement we sampled 60 different initial states. All demonstrators then demonstrated on the same 60 initial states, and we assessed the full execution performance on a fixed set of 30 initial states sampled from the same distribution as the training states. Full execution performance was gathered by rolling out the neural network policy learned from the corrected data on the initial state. If the net produces a successful planar part extraction, then it is assigned success.

In the original dataset, execution performance on the test states had a 62.5% success rate. Some common failure modes in testing involved the robot performing a sweeping motion too early, and failing to successfully push the desired object, or pushing too late, and thus pushing too many objects to successfully singulate. Additionally, the robot's approach might have been too exaggerated, which resulted in failing to appropriately approach the object cluster, or too minimal, failing to push between objects.

V. Observed Human Behaviors

In the dataset of human trajectories, we observed the following two forms of human inconsistency: obstructive actions and jitter.

Obstructive actions: When a human performs a large number of demonstrations, some part of some trajectories will not advance them toward their desired result (i.e. successful planar part extraction). If too many such actions occur, the human will fail at the task. However, in many cases, the human will simply perform self-corrective actions or extend the length of the trajectory, and eventually record a successful demonstration. This often occurs if the human performs some other delaying action, possibly by overshooting the preferred behavior, or starts a behavior too early, and is forced to self-correct. See Figure I.3 for some examples.

Jitter Another form of noise that we observe involves human actions which over-complicate the trajectory. In some ways, this overlaps somewhat with obstructive actions, but it distinctly involves 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 inherent to the human which are hard to learn. Such actions often involve turns, changes in direction, or back-and-forth motion which provides little toward the success of the policy. Nonetheless, because these actions result in controls which differ greatly when the corresponding states differ by relatively little, they make it harder for the learner.

VI. Data Cleaning Algorithm

We introduce two treatments to reduce overall noise. Low-confidence correction corrects 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, so data due to obstructive actions will be outliers and thus hard-to-learn. However, not all hard-to-learn data points are due to inconsistency, and might represent genuinely high variance parts of the state space. Thus we use cross-validation to determine the confidence of the robot policy on particular data points, and a human expert to determine corrections for low confidence states. We call our technique low-confidence correction. In low-confidence correction we propose to take the training dataset and break it into $K+1$ "folds" or equally sized partitions of the data. We remove one fold as the test set as an assessment of overall performance. Then, we iterate through the following: 1) Choose one of the remaining K folds to be the holdout fold. 2) Train a regression model using $K-1$ of the folds, and assess the regression model against the holdout fold. Repeat until each fold has been used as holdout. Using this method for each of the K training folds, we can construct a set of labels L_R, which represent the labels likely to be given by the robot in similar, unseen states. This method is based on the common data cleaning technique of cross-validation, which is used as an indicator of predictive error [29]. When applied to the training set, cross validation verifies which states are likely to be incorrectly predicted given a trained regressor. States more likely to be mispredicted are those with lower confidence. We call this misprediction cross-validation error.

We compare these labels with the supervisor labels, L_H, and per-term error is l2 loss: $\|L_R(x)-L_H\|_2^2$, the same term used for loss. Then, we display states with low confidence to the expert for correction. This threshold is determined based on a reasonable value that trades off between encompassing significant portions of the loss, while not burdening the supervisor with too many re-labelings. We selected an error threshold to consider only 5% of the highest error data for cleaning. As shown in Figure VII.1, this data represents significant outliers in terms of cross-validation error.

To clean a data point, 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 apply a correction 4) have the current robot's policy apply the correction. Methods 1 and 2 are interesting to consider because the data point may be causing model error during training. Method 3 helps fix the inconsistent labels, which might improve upon 1 and 2 because it doesn't lose data. Method 4 is relevant to consider because prior work has suggested this is a way to reduce learning error [19]. VI.1. We describe this correction method in Algorithm 1.

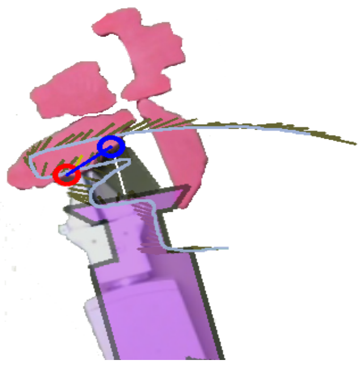

Fig. VI.1. An example state where the human provides a correction. Notice the overlay used to visualize where the robot might end up as a result of the human correction. The light green in the figure represents the trajectory taken by the demonstrator, and the dark green lines indicate the suggestions from the robot policy

Low-confidence correction

```
Data: State action pairs ($x \in X, u \in U$) of human
      demonstrations, $D$, number of folds $K$, Threshold $\tau$
Result: Equivalent dataset with selected corrections from the
        human supervisor at hard states
Separate $D$ into $K+1$ folds;
Designate $K+1$th fold as holdout;
; Set $S$ contains remaining folds, where S(i) = ith fold.;
Initialize LABELS 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_LABELS;
    Append NEW_LABELS to LABELS;
end
for $l$ = 0 to LABELS do
    if $\|LABELS(l) - D(l)\|_2^2 > \tau$ then
        Ask human for correction;
        Replace D(l) with correction, if specified.;
    else
        continue;
    end
end
```

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$^{\text{th}}$ fold being used for holdout set assessment. We chose 0.65 normalized error for a threshold error value; error values greater than 0.65 are displayed to the expert. We assume that individual frames (time steps in a trajectory) can be independently altered particularly because the time steps are safely far apart (0.35 sec), and because the states themselves are assumed to be Markovian. The threshold was chosen based on a suitable point where the error terms began to increase drastically. Figure VII.1 displays the choice of threshold against all error terms. Beyond this, we did not optimize the threshold value, and chose it based only on observed cross-validation differences in the training sets. We also did not optimize the number of folds.

B. Jitter correction

For jitter correction, we group by trajectory and apply a low pass to only the labels of that trajectory. For the planar

Jitter correction

```
Data: State action pairs ($x \in X, u \in U$) of human
      demonstrations, $D$, number of folds $K$, Threshold $\tau$
Result: Equivalent dataset with deltas filtered using
        Butterworth Filter
Separate $D$ into trajectories, $T$ for $t \in T$ do
    Labels for trajectory t, in order: $L_t$ New-Labels ($NL_t$):=
      Filter $L$ by 5th order Butterworth filter
end
New dataset $D'$ using states from $D$, and labels $NL_t$ for the
  corresponding labels
```

part extraction task jitter correction treatment, we selected a Butterworth filter with filter order 5. We did not optimize this value, due to constraints in rolling out a robot policy, and chose it based on reasonable values for Butterworth filters. Our choice of filter, and of filter order, was done without knowledge of full stack performance set. We replace the labels of the trajectory with those altered by the low pass filter, and filter the labels of each trajectory in the training set. We choose not to require an expert in correcting high frequency labels, because we expect most jitter noise to be corrected by reduction to low frequency terms. Thus, a low pass filter should capture this correction more completely than a human user. This algorithm is described in Alg. 2.

We choose the Butterworth filter due to its flat bandpass response, recognizing that we have no clear frequency above which we can consider label noise. We did not optimize the choice of filter, because we can make only limited 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.

C. Final Treatment

In our final treatment, 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 labels 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 datasetand some additional supporting results which may clarify possible

	Jitter correction	Low-confidence correction	Combined
Mean Training Error	9.87%	18.92%	10.89%
Standard Error	0.29 %	0.28%	0.42%

TABLE VI.1

DIFFERENCE IN TRAINING ERROR AND THE STANDARD ERROR FOR A FIXED BUDGET OF ITERATIONS, AVERAGED, FOR THE DIFFERENT TREATMENTS. JITTER CORRECTION RESULTED IN THE LARGEST CHANGE IN TRAINING ERROR.

forms of human noise. In the planar part extraction dataset, our operations achieve significant improvements, and the final performance suggests the effectiveness of our treatments in the planar part extraction dataset.

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 the 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 an object, then this would be a success. Figure I.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 size of the training set.

Drawing from the results of Figure I.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 actionss tend to result in a relatively small number of extremely high error states.

B. Noise Characterization

Observing Figure I.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 VI.1, 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 $.4\%$.

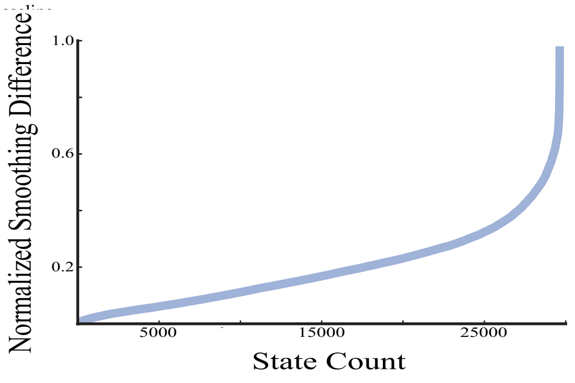

Fig. VII.1. Sorted normalized difference in label from jitter correction. Notice that jitter correction alters a significant number of states, suggesting a good number of high frequency components. However, cross-validation difference is still generally higher.

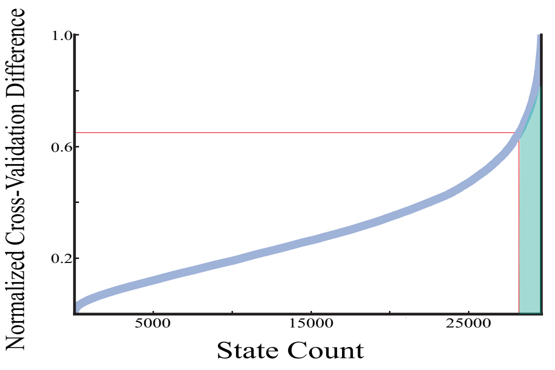

Fig. VII.2. Sorted normalized cross valiation errors of all data points in the dataset. The horizontal line indicates the threshold used for bias correction, which was .65 normalized error. The shaded region demonstrates the error sent to the expert for correction. Five percent, or 1,500 data points were sent to the human, of $29,610$.

Figure I.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 VI.1 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. Thus, 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.

To continue highlighting the proposed forms of human noise, we demonstrate the amount of deviation resulting from cross-validation (as an estimate of prediction error), and from the low pass filter. Figure VII.1 displays the estimated prediction error. That is, the graph shows the sorted l2 difference between the cross-validation label and the original supervisor label, for all of the frames.

Figure VII.2 displays the amount of change to the label produced by jitter correction, that is, the loss value when

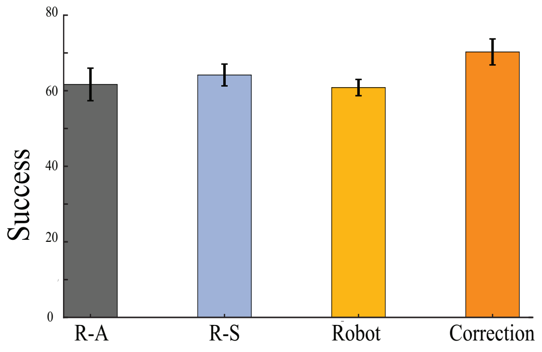

Fig. VII.3. The percentage change in performance from different methods to address obstructive actions. Remove-A is removal of all data points with cross-validation error over the threshold, Remove-S is selective removal of data points, where the human decides which data points to remove. Robot changes the data points to the ones determined by the robot policy. Correction describes sending data points to the human for correction.

comparing the label after jitter correction, with the label 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 human inconsistency.

C. Alternative low-confidence correction methods

In the results described by figure VII.3, we notice that the alternative treatments for obstructive actions produce no significant improvements, especially when compared with the suggested low-confidence correction. Figure VII.3 also 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. The 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 hard-to-learn, and not due to human error. 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.

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.

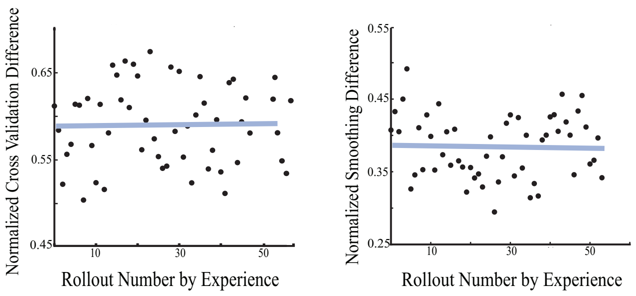

Fig. VII.4. Performance over each of the 60 example rollouts. Notice that there is almost zero correlation, which suggests that for this experiment, experience was not a confounding variable in performance measured with regards to jitter or obstructive actions. The right graph represents jitter correction, while the left represents low-confidence correction

The results demonstrated in VII.4 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

Interested in obtaining better results with limited data, this paper introduces several characterizations of human inconsistency in a planar part extraction task, as well as different techniques by which to determine inconsistent human demonstration behavior, and provide corrections. We provide a pilot user study to quantify our qualitative assessments of certain forms of noise in a planar part extraction task, and assess the results of our treatments. We have obtained promising results for applying cleaning techniques to teleoperation demonstrations. Certainly, we hope to gather more data to further evaluate this assessment. We would also like to perform further work in differentiating between improvements due to transforming data to be easier to learn for the model, without actually improving the integrity of the data and cleaning data to improve data quality.

One future step would be to increase our understanding of combining multiple treatments. We recognize that combining low-confidence correction and jitter correction could have some success because sudden changes to the label from low-confidence correction could be regularized by the subsequent jitter correction action. However, the opposite could also be true, where corrected actions get smoothed back to their erroneous state. In addition, data cleaning is often a fixed-point iteration, and we would hope to analyze multiple iterations of cleaning treatments, to see what kind of improvements would be possible. This is especially true for low-confidence correction, where after correcting some states, others, but hopefully fewer might be pushed to high error.

In future work, we would like to extend our low-confidence correction system beyond the planar part extraction setting, to test its effectiveness on other situations and setups. We also recognize also that the datasets which we collected most likely were small for learning the task, and we would like to test data cleaning techniques on datasets with higher accuracy. We also recognize that the dataset which we used, coming from people inexperienced with operating the particular robot platform we used, certainly included many suboptimal cases, noticeable by inspection, and we would like to apply our techniques to other platforms and populations of robot trainers.

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] Kaipa, Krishnanand N., Akshaya S. Kankanhalli-Nagendra, and Satyandra K. Gupta. "Toward estimating task execution confidence for robotic bin-picking applications." 2015 AAAI Fall Symposium Series. 2015.

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

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

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

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

[33] 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.

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

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

[36] 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.

[37] 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.

[38] 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.

[39] 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.

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