Proceedings of International Joint Conference on Neural Networks, Atlanta, Georgia, USA, June 14-19, 2009

Improving the performance of ANN training with an unsupervised filtering method

Sekou Remy, Chung Hyuk Park and Ayanna M. Howard

Abstract— **Learning control strategies from examples has been identified as an important capability for many robotic systems. In this work we show how the learning process can be aided by autonomously filtering the training set provided to improve key properties of the learning process. Demonstrated with data gathered for manipulation tasks, the results herein show the improved performance when autonomous filtering is applied. The filtration method, with no prior knowledge of the task, was able to partition the training sets into sets almost equal to expertly labeled sets. In the case where the filter did not produce the same groupings as the expert user, the method still permitted a controller to be trained which demonstrated a success rate of 92%.**

I. Introduction

In a large number of fields there has been interest, effort, and progress in the battle to uncover knowledge from data. It is often the case that such knowledge once uncovered is then applied to enable autonomous (or semi-autonomous) control of some system. In a narrow view of this broad domain, learning specific control strategies based on observations of data is a common challenge for many types of practitioners.

A plethora of methods exist to implement control strategies but due to their flexible structure and ability to map non linear functions [1], neuro-controllers [2] have become a commonly applied method for robotics and other complex systems. Several approaches have been devised to generate the weights used for these controllers, each with their relative benefits.

Whether considering evolutionary techniques, back propagation, or any other method, the quality of the training sets applied have a strong influence on the determined weights. Concern for the quality of the training sets is also elevated by the growing trend to use human examples to generate training sets. The process of gathering human examples is often the most readily accessible method of acquiring useful data.

One quality of learning from human examples that we are interested in developing is the ability of the system to identify similar subsets in the provided training sets. With this capability, the system can determine what are likely different behaviors, and then learn from them separately. Learning from appropriately partitioned sets can have beneficial outcomes regardless of the learning method.

The issue of how to partition training sets, especially considering the design goal of minimizing human involvement, is a challenge, but it is a key facet for solutions of interest to the "learning from examples" community.

Sekou Remy, Chung Hyuk Park and Ayanna M. Howard are with the Human-Automation Systems (HumAnS) Lab, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta GA 30332, USA {sekou, chung.park, ayanna.howard}@ece.gatech.edu

II. Related Work

For this work we consider the domain of manipulation and learning generated from human examples. Many researchers have built frameworks to transfer human skills to a manipulation platform. Among others, Michael Nechyba [3] designed a system in which a robot works as both a learner and a teacher in transferring dynamic control skills. More recent studies of manipulation learning shows a tendency to rely primarily on vision. A. Saxena et al. presented research in grasping real-world objects based on grasping point classification using monocular vision [4]. And K. Yamazaki et al. tried to learn a simple task in the visual domain by using direct instruction guiding [5]. The common limitation in these studies, however, lies in the need to hand-code the task behaviors at some level.

Learning from demonstration (also known as Imitation Learning [6]) is widely studied in the robotic learning field, as it provides methods similar in nature to the ones by which humans learn from others. However, it requires a high level of visual recognition and physical modeling, and current learning from demonstration is typically limited to a restricted area equipped with expensive camera systems or motion capture devices (e.g., [7]). Also, a transformation between the human model and a robotic model is usually required to institute robot control. In order to improve the robots' abilities to function as an integral part of our daily lives, we need to design the robots such that they are both easier to control and to interact with.

Instead of mapping a human model to a robot's model there have been several works including [8], [9] who have utilized teleoperation interfaces to capture examples of target behaviors. These methods utilize knowledge of the kinematics of the robotic system and map the motion of salient portions of the system to some Human Interface Device (HID). The human teacher's role is then to control manipulation via the HID and other parts of the user interface provided by the robot designer(s).

While this approach simplifies control in many ways, collecting examples in this manner does not remove the need to screen bad examples of behavior, or isolate different behaviors prior to the application of learning techniques.

The work in [10] presents an approach which partially addresses this issue. In this work the researchers demonstrate successfully the functionality of identifying success or failure from unlabelled examples of the activity. Where this work falls short is that it requires previous training exposure of successful (and non-successful) actions and is only able to classify future unlabelled examples based on this a priori information. What is needed is a method to provide some level of differentiation based only on the examples provided. Removing the burden of labeling examples a priori is a design goal since it would remove a critical barrier in systems that learn from human examples. Equipping the robot to recognize differences and adapt accordingly would enable a much more fluid learning process as well as permit the teacher greater degrees of freedom. More significantly, it would also reduce the need for the robot teacher to have the knowledge level of a robot designer. It is for these reasons that we propose the method presented in this work.

III. Methodology

When considering human-in-the-loop robotic systems which incorporate learning, one challenge is the process of acquiring training data. For example, even under the best of circumstances, the human teacher will display a level of variability while demonstrating a task; and in many cases will also demonstrate more than one task during normal robot operation. The multiplicity caused by such a scenario often results in training sets which include poor examples of behaviors which must be screened prior to the application of the learning process. To reduce the burden on the (human) instructor, we present a method to permit automated filtration/grouping of behaviors prior to the learning process. This two phase process is implemented via coherence based filtering and then learning.

Coherence based filtering (CBF) is a behavior filtration method that is predicated on the assumption that the human teacher possesses knowledge about the behavior they would like to teach the robot. Each time they demonstrate a target behavior it is expected that they demonstrate an **instance** of the behavior. Whether due to natural human variability or error, instances of the behavior may vary but in the mind of the teacher there still exists some model of that behavior. The teacher, knowing this model, has the ability to generate additional examples of the model (other instances).

A. Differentiating Behaviors

In CBF it is then the challenge of the automated system to apply a method to determine if two or more data sets are likely to be instances of the same behavior. Said more plainly, if provided with three examples A, B, and C, determine the likelihood that A, B, and C are examples of the same behavior (e.g. obstacle avoidance). If it can be determined that example C is not an instance of the same behavior as A and B, then it can be excluded from the training data. If it is not possible to infer that C is an instance of a different behavior from A and B, while it may indeed be the case, there would not be enough evidence to rule this instance out. In such a scenario, grouping A, B, and C then using these as instances of the same behavior by incorporating them into the same learning process is a defensible approach to filtering instances of the behaviors.

To make the capability available, we apply a variant of the Kolmogorov-Smirnoff (K-S) test [11], one of a family of offerings of statistical tests including Pearson Chi-squared test and the Anderson-Darling test. The basis of this test is the null hypothesis that in its two sample form states that *the samples are drawn from the same distribution*. To test this hypothesis, a K-S statistic ($D_{n,m}$) is calculated based on empirical distributions F_n and F_m of the two samples defined over range x (See (1)). $D_{n,m}$ is then compared to a derived value $D_{1-\alpha}$, which is typically extracted either from standard or custom tables. The hypothesis is rejected if $D_{n,m}$ is sufficiently large.

$$D_{n,m} = \sup_x |F_n(x) - F_m(x)| \tag{1}$$

B. Representing Behaviors

For CBF, we take the approach that a behavior is a potentially non-linear mapping from sensing to action. For most interesting scenarios, this mapping is defined over large sensing and action spaces which make the study of the underlying mapping computationally challenging. To mitigate this challenge, dimensionality reduction which preserves the local geometry of the high-dimensional space can be applied. We are interested in methods which can effectively reduce the dimensionality of the spaces to one (i.e. one dimension in sensing and one dimension in action). Both linear and non-linear techniques exist which accomplish this task, each with its own merit, but principal components analysis, principal curves, self-organizing maps are often applied approaches.

This is how the values of a and s are respectively derived from the actuator values and sensor values. For a given example of a behavior B_i, the sensor state s_j and the associated action state a_j are collected during teleoperation. N_i is the total number of sensor-action pairs that exist for the behavior B_i (See 2).

$$B_i = \{a_j, s_j\}_{j=1}^{N_i} \tag{2}$$

With this two dimensional representation of a behavior, we must apply a two dimensional variant of the K-S test. For this work, we use the implementation of the two-dimensional two-sample K-S (2D2SK-S) presented in [12] to calculate the values of $D_{n,m}$ This implementation is based on the work of Peacock[13], and Fasano and Franceschini [14] to extend the K-S test to two dimensions.

For behaviors defined in this manner, the K-S statistic is calculated for pairwise combinations of all presented instances. Instead of using $D_{1-\alpha}$ from tables, the median $D_{n,m}$ value is used as the threshold. Such a decision seeks to capitalize on the likely scenario that two or more instances of a behavior will be provided in a training set.

C. Summary

This is how instances of behaviors are filtered. This method is called coherence based because it exploits the consistent relationships between sensing and action for each instance of a behavior. There is however one major assumption with this approach. CBF assumes that the sensing provided contains all that is necessary to make the decision about the required action. If this is not the case then the link between sensing and action can appear random. While this is significant, no learning method performs well if the basis for the learning is incomplete, no matter how carefully it is gathered.

D. Learning

When performing a manipulation task via teleoperation, it is necessary to sense the target object and understand the action(s) required relative to the target. Since cognition and reaction take place over time, we focus on the temporal sequences of sensing and actuation data. We utilize feed-forward artificial neural networks (ANN) trained by the back-propagation algorithm ([15]) for learning relationships within the temporal sequences of teleoperation commands from the human. We design two different architectures for this temporal data based learning, one happening in joint space and the other in Cartesian space. Our ANN structure is conventional three-layer network with one hidden middle layer. Basically, we set the learning rate to 0.10, with error limit (tolerance) applied with 0.01, 0.05, and 0.1, and choose the best networks in terms of error. More details on this approach are presented in the following section.

IV. Experimental Setup

A. Teleoperative Manipulation System

Our teleoperative manipulation system depicted in Fig. 1 consists of a Pioneer3AT mobile robot, 5-DOF Pioneer Arm, a USB camera, a laptop for master interface program (Fig. 2), and a force-feedback joystick. Pioneer3AT is a four-wheel drive, skid-steer mobile robot. The Pioneer Arm is a relatively low-cost robot arm that is driven by six open-loop servo motors, providing five degrees-of-freedom with an end-effector capable of grasping objects up to 150 grams in weight.

For acquisition of the visual data, we mount a small USB webcam on the gripper, so our system can transmit the workspace view observed by the end-effector to the operator (i.e. as the arm approaches an object, the object in the view grows larger in size). The maximum frame rate of the camera is approximately 30 fps/sec with pixel resolution of 320x240. It also has a diagonal 54 degrees of field-of-view angle with focus range of 5cm to infinity.

Due to the difficulties in controlling the position and orientation at the same time caused by lack of degrees of freedom and awkward positioning of the pivot joint and rotational joint of the wrist, we utilized only four degrees of freedom with this arm and opted to keep the wrist orientation fixed. To incorporate the 2D joystick control with 5DoF arm movements, we mapped 2 dimensional control of joystick into x-axis and y-axis of arm's end-effector, and used the slider of joystick to control z-axis. The window size of the mapping between joystick and arm's workspace area is designed to change as it approaches the object.

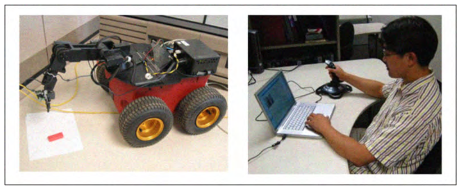
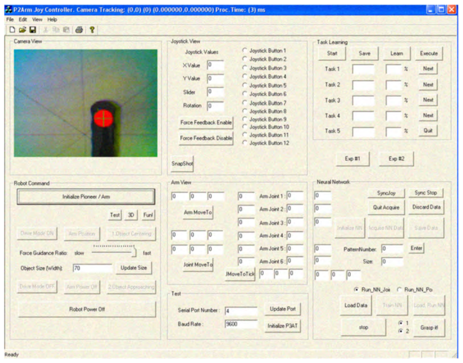


Fig. 1. Teleoperative manipulation system using Pioneer3AT and Pioneer Arm.



P2Arm Joy Controller. Camera Tracking: (0,0) (0) (0.000000,0.000000) Proc. Time: (3) ms

X Value: 0
Y Value: 0
Slider: 0
Rotation: 0
Force Guidance Ratio: slow – fast
Object Size (Width): 70
Serial Port Number: 4
Baud Rate: 9600
PatternNumber: 0
Size: 0
Arm Joint 1: 0
Arm Joint 2: 0
Arm Joint 3: 0
Arm Joint 4: 0
Arm Joint 5: 0
Arm Joint 6: 0
Ready

Fig. 2. GUI for Teleoperative Manipulation.

Due to dimension matching problem, haptic force feedback was minimized to creating holding forces to acquire stable manipulation at this time.

B. Task

For this work, multiple demonstrations of two tasks were considered. The aim of all cases was to teach our robotic system to reach down and grasp an object which was initially in the view of the eye-in-hand camera. Demonstrative trajectories were captured while a user picked up an object from distinct location on the surface. The overall training set would feature an object on a planar surface in a 15cm by 10 cm area (See Figure 3).

The second task was similar to the first except that the start position of the end effector was initialized near to the target object. This task posed slightly different challenges since each trajectory would contain a fewer number of items and errors early in the motion sequence could have greater impact on keeping the object in the portion of the workspace where it was visible to the camera.

For each task, an expert user (the system designer) provides nine examples. In each case the object is located within 5 - 7 cm from the center of the workspace, on the plane the robot rested. The selection of locations was not precise but they were disbursed over the space as indicated in Figure 3. Figure 4 shows the view from the camera just prior to activating the gripper.

During these "good" examples, the user demonstrated the skill expected of an expert user. In addition to a good examples, three bad examples were also captured. For these examples, the user intentionally introduced extra motion in the x, y, and z dimensions of the workspace. Examples of the end effector trajectories for both "good" and "bad" examples are shown in Figures 5 & 6 respectively. It should be noted

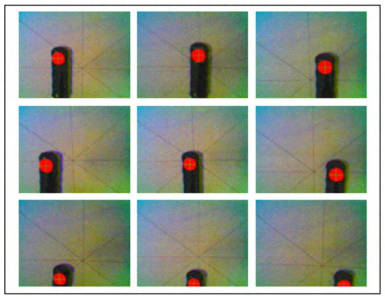


Fig. 3. Images of eye-in-hand view at the initial state of trajectories 1-9 of the training data.

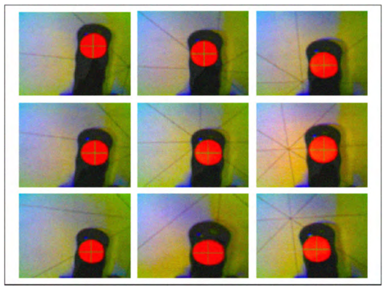


Fig. 4. Images of eye-in-hand view at the final (grasping) status of trajectories 1-9.

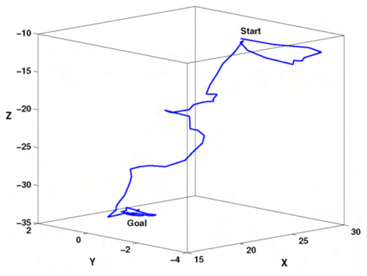


Fig. 5. End effector trajectory of a good example. Z coordinate is negative valued since the arm is attached on top of Pioneer and it is reaching down to grasp an object. 113 sequences (11.3sec) are represented in the graph. Units are in centimeters.

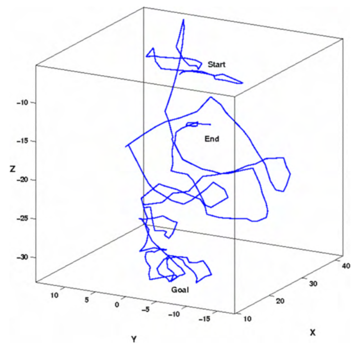


Fig. 6. End effector trajectory of a bad example. 271 sequences (27.1 sec) are depicted.

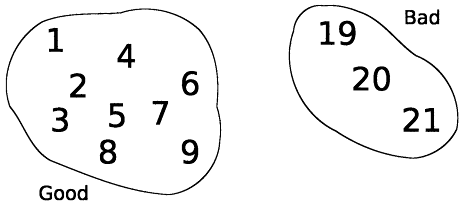


Fig. 7. Expert labelled grouping of trajectories for behavior 1.

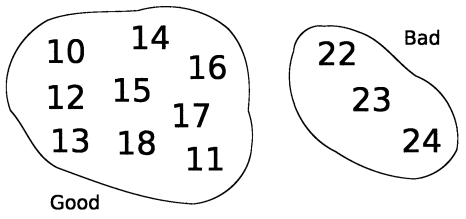


Fig. 8. Expert labelled grouping of trajectories for behavior 2.

that in all examples, whether good or bad, the user did grasp the target. The discrete data was collected every 100ms over the teleoperation time, and each trajectories consist of 90-150 sequences. For conciseness, the labels of the training set gathered are presented in Figures 7 and 8.

C. Applied Learning

1) Joint Space: The first structure we utilize for training on the temporal data sequences is associating independent neural networks to each degree of freedom. The main advantage of this approach lies in the ability to recover from errors by decomposing the dimensional load into several distributed ANNs. As shown in Figure 9, each neural network for the i-th joint has inputs of object position in image domain (XY plane - See Figure 10), object size from the camera viewpoint (W - in pixels), and joint values from the base joint to the i-th joint in the kinematic chain. It also has one hidden layer of neurons and has one output for differential value for the joint that corresponds to the commanded actuation value. So five ANNs are in charge of our 5-DoF Pioneer Arm (used in our experimental setup), and these ANNs take the current manipulator state and the current sensor (camera) status as inputs, and trains itself to generate proper output for the next actuated movement. Since the differential values are generated by subtracting the robot's current arm state from the human operator's command, this structure facilitates learning the behavior taught by the human operator, in the domain of robot's joint coordinates. With this neural network structure, we seek to decouple the linkage inherent in the kinematic chains of the manipulator and learn each joint's independent contribution to the task.

2) Cartesian Space: The second approach we utilize for building the neural net structure is to consider the actions of the end-effector in Cartesian space. As depicted in Figure 11, only three independent ANNs are in control of our 5-DoF manipulator, and each ANN is trained to generate the proper differential value for the next movement in each of X, Y, Z direction, based on the input values of the current

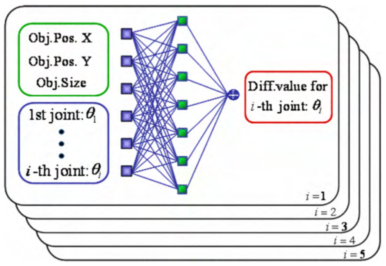


Fig. 9. Artificial neural network (ANN) structure of learning in joint space. Each joint has its independent ANNs, with input from the camera sensor and joint angles, and with output of incremental joint angles for next movement.

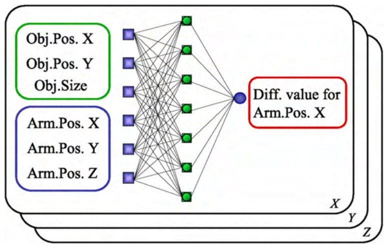


Fig. 11. Artificial neural network (ANN) structure of learning in Cartesian space. Each joint has its independent ANNs, with inputs from camera sensor and current position of the end-effector, and with output of incremental value for x-,y-,z-coordinates for next movement.

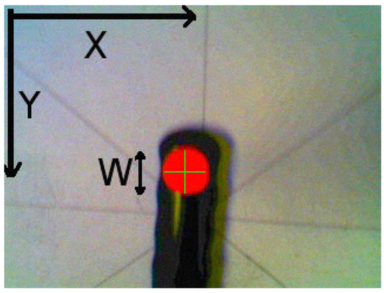


Fig. 10. XY-W plane for camera image. The W dimension is derived from the diameter of the marker of known size, and is thus related to depth.

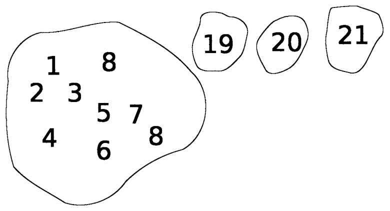


Fig. 12. CBF labelled grouping of trajectories for behavior 1.

end-effector X, Y, Z positions and the object data from the camera (X, Y, W). Since the coordinates of the arm position are in the arm's base frame while the coordinates of the object are in the camera frame, we do not combine them directly but provide as independent inputs to the ANN so that the correlations can be trained over two different space domain.

In this structure, we disregard the previous consideration on the status of arm posture, and decide to learn the direct mapping between the sensor condition (environment) and actuation of the end-effector (result in action), based on assumed prior knowledge of the global inverse kinematics of the manipulator system. This approach is designed to create a direct mapping between the dimensions of sensing and the dimensions of actuation, and to simplify the learning problem for a manipulation task. We are also composing these three ANNs independently in order to avoid the potential coupling of motion in the X, Y, and Z axis.

V. Results

To briefly recap, the goal of this work was to present a method to permit training sets provided from human examples to be filtered in an unsupervised manner prior to applying the set to a learning mechanism.

A. Coherence Based Filtering

Each of the trajectories captured is treated as an instance of a behavior. The sensor values with temporal markers were mapped from four dimensional space into a one dimensional sensor state with 100 values and the actuator values were mapped from a four dimensional space to a one dimensional action state with 25 values. These states captured a representation of sensing and action at different times.

The null hypothesis was considered for pairwise comparisons of behaviors B_i and $B_j \forall i \neq j$. If the null hypothesis is not rejected for the K-S statistic of B_i and B_j, these behaviors are placed in the same behavior candidate set. If the null hypothesis is rejected and the behavior is not already in a candidate set, a new candidate set is created for the behavior. If any behavior is in more than one candidate set then these sets are replaced by their union.

After all comparison have been made, the resulting behavior candidate sets contain groups of instances which are likely instances of statistically similar behaviors. Learning from these candidate sets separately instead of all together is expected to produce improved learning performance.

The results of applying CBF to the 24 trajectories considered in this work are presented in the candidate sets shown in Figures 12 & 13.

1) Behavior Candidate Sets: Figure 12 shows the behavior candidate sets uncovered for Behavior 1. This figure groups trajectories 1-9 into a single candidate set and trajectories 19, 20, and 21 into separate candidate sets. Such groupings provide support for training a single ANN with trajectories 1-9, separately from the other examples. It is interesting to note that these were all the examples of "good" behaviors. Without prior knowledge about the training set, CBF permitted the good behaviors to be grouped into the same behavior candidate set. It is also significant that each of the bad trajectories is relegated to its own candidate set. This is not alarming since it was unlikely that each of these trajectories would be distorted in the same manner. A high degree of connectivity was observed in the graph which captured the relationship between trajectories 1-9. This indicates to some degree the confidence that each of these trajectories are examples of the same behavior.

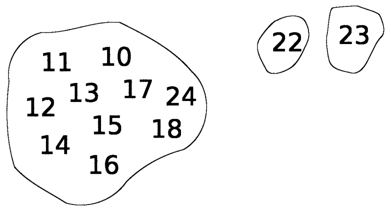


Fig. 13. Expert labelled grouping of trajectories for behavior 2.

TABLE I
ERROR VALUES BY CASE

Behavior	Type I (α)	Type II (β)
1	0/9	0/3
2	0/9	1/3

A similar story is told in Figure 13. In this figure we consider the trajectories presented for Behavior 2. The behavior candidate sets group trajectories 10-18, and 24 into a single candidate sets and trajectories 22 and 23 are relegated into their own candidate sets.

Unlike the case with the examples of Behavior 1, a single behavior from the user defined "bad" set is included with the "good" trajectories. The errors in classification for each of these tasks are presented in summary form in Table I.

2) Evaluation: To test the effects of the candidate set groupings, the following sets of trajectories were considered as training sets for ANNs:

A = {1,..,9}
B = {1,..,9, 19,..,21}
C = {10,..,18}
D = {10,..,18, 22,..,24}
E = {10,..,18, 24}

Comparing the differences between learning from sets A and B will show the expected benefit of learning from the CBF suggested examples of the behavior over learning from the entire set of examples of that behavior.

Unlike the case for Behavior 1, none of the candidate sets for Behavior 2 are equivalent to perfect knowledge of good examples. This provides an opportunity to demonstrate what is likely to be useful information to evaluate this approach. Since an expert user presented the trajectories for C, it would be expected that the ANN trained with that set would outperform the sets D and E. The outcome of a successful test of CBF would show that the performance of a network trained with E exceeds that of one trained with D. Such an outcome would be better only if the performance after learning with E is equivalent to that of learning with C.

B. Learning

To evaluate learning, we consider the effects of two different parts of the learning process: 1) learning and error rates, and 2) evaluation of the trained ANN.

1) Learning and Error Rates: For each of the sets previously defined an ANN was trained. The table II shows the observed learning rates and errors. When comparing the learning rates for the networks trained with A and B, it is

TABLE II
LEARNING AND ERROR RATES BY CASE.

Group	Tolerance	Learn Rate (%)			Error (MSE)		
		X	Y	Z	X	Y	Z
A	0.01	40.6	83.2	58.6	0.153	0.040	0.111
	0.02	66.6	97.0	79.5	0.154	0.040	0.113
	0.05	90.7	100.0	95.4	0.155	0.066	0.115
	0.1	98.7	100.0	97.9	0.149	0.066	0.112
B	0.01	28.3	62.4	35.3	0.377	0.129	0.326
	0.02	41.0	80.3	57.2	0.393	0.132	0.323
	0.05	70.2	94.5	84.1	0.394	0.131	0.322
	0.1	88.1	99.1	92.7	0.386	0.129	0.312
C	0.01	36.6	66.9	46.7	0.259	0.099	0.221
	0.02	56.3	83.4	64.4	0.258	0.098	0.215
	0.05	79.5	97.4	85.9	0.255	0.098	0.217
	0.1	95.0	99.8	96.8	0.257	0.098	0.221
D	0.01	19.3	34.6	28.9	0.581	0.485	0.530
	0.02	32.7	43.6	43.5	0.578	0.510	0.513
	0.05	57.4	70.5	71.0	0.558	0.517	0.493
	0.1	76.9	87.5	85.5	0.543	0.482	0.535
E	0.01	33.0	60.0	37.9	0.304	0.110	0.255
	0.02	48.6	80.7	55.8	0.308	0.108	0.254
	0.05	74.4	96.0	79.9	0.310	0.107	0.260
	0.1	92.4	99.7	94.4	0.313	0.108	0.255

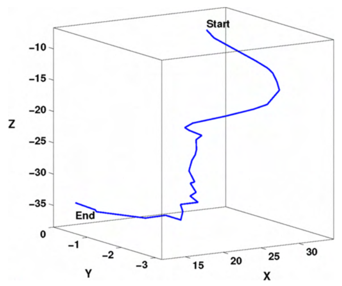


Fig. 14. Trajectory of ANN performance trained with set C. Execution time was 3.4 sec.

clear that the ANN produced with A performs better in every axis. Similar observations are evident when considering the mean square error. The error is observed during the training process are higher for B. These comparisons show that CBF provides advantages over blindly applying all the examples in the training set.

Further comparisons between ANNs trained with C, D, and E serve to confirm the advantages of performing CBF. This method, without external expert knowledge about the training set, performs better than the case where it was not applied. In each dimension, the CBF method produced a training set that was within 80 percent of the performance with respect to learning rates and error rates.

While these comparisons do not yet show the performance of the trained ANN, they show that the ANNs were trained faster and better when CBF was applied to the training sets. They also show that in the case where the candidate set was not equivalent to the "good" set that the ANN's training specs rivaled those of the "good" set.

2) Task Performance: For evaluation of the trained ANNs, we do not use the hold-out sets since the objective of the

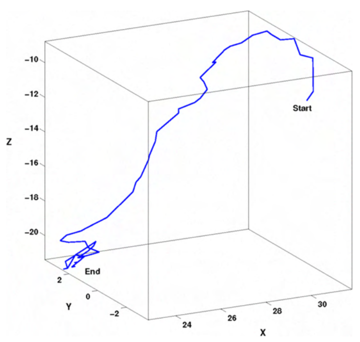


Fig. 15. Trajectory of ANN performance trained with set E. Execution time was 9.5 sec.

TABLE III
SUCCESS VALUES BY CASE

Dataset	Success rate	percentage
A	7/9	78%
B	4/9	44%
C	12/12	100%
D	7/12	58%
E	11/12	92%

task is to generate proper trajectories toward the target and calculating the errors of 3-dimensional trajectories may not be sufficient to tell if it is a good trajectory or not. So instead, we physically test the trained ANNs on the actual robotic platform, with randomly distributed target points over the workspace. nine tests are performed with ANNs generated from A, B, while 12 were performed for C, D, and E.

In case of Behavior 1, the ANNs from the good dataset A shows 78% of success rate, lower than expected due to the characteristic of the dataset containing small spatial distribution. However, A shows definitely better result than B. For Behavior 2, dataset C which contained only the good example trajectories generates ANNs with 100% success rate, while D has a 58% success rate and E a 92% success rate. These numbers confirm our claim that CBF is an effective method to improve the performance of these learning techniques. The networks are trained faster and better and now these results confirm that they also help produce controllers which are more

One notable point is that ANNs from C shows reliability with 0.1 error boundary, while ANNs from E is reliable with lower boundary such as 0.05. And still the ANNs from set E generated more noisy trajectories than C, as shown in the following figures

One minor failing point is that ANNs trained over the joint space did not show good results, due to the unexpected reason that each ANN for the joint actually interferes with other joints' movements, making the arm's movement unstable. It conflicted with our intent to use independent ANN for each joints to dissolve the kinematic linkages, but increases our knowledge on composing ANN structures for robotic manipulation.

As a result we can only present these results for the networks trained with data for the Cartesian space. This is not a limitation of the filtration approach.

VI. Summary and Future Work

In this work we have presented a process to autonomously filter training sets provided from teleoperation. It is well known that better training sets provide opportunities for better learning and our process removed the burden of selecting/grouping the examples manually. In this work we have shown that when applied, Coherence Based Filtering enables better controllers to be learned faster and confirmed the improvement in performance with on-robot tests.

In the future we seek to study the sensitivity of this approach and also investigate whether its performance is contingent on the learning method applied.

Acknowledgment

This research was supported in part by the National Science Foundation's Division of Information & Intelligent Systems (award number IIS-0705130)

References

[1] I. W. Thomas Miller, R. S. Sutton, and P. J. Werbos, Eds., *Neural networks for control*. Cambridge, MA, USA: MIT Press, 1990.

[2] M. T. Hagan, H. B. Demuth, Orlando, and D. Jess, "An introduction to the use of neural networks in control systems," *International Journal of Robust and Nonlinear Control, John Wiley & Sons*, vol. 12, pp. 959–985, 2002.

[3] M. Nechyba and Y. Xu, "Human skill transfer: Neural networks as learners and teachers," vol. 3, August 1995, pp. 314–319.

[4] A. Saxena, J. Driemeyer, J. Kearns, C. Osondu, and A. Y. Ng, "Learning to grasp novel objects using vision," in *10th International Symposium of Experimental Robotics (ISER*, 2006.

[5] K. Yamazaki, T. Tsubouchi, and M. Tomono, "Modeling and motion planning for handling furniture by a mobile manipulator," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, August 2007, pp. 1926–1931.

[6] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in Cognitive Sciences*, vol. 3, no. 6, pp. 233–242, June 1999.

[7] R. Pardowitz, M. Dillmann, "Towards life-long learning in household robots: the piagetian approach," in *6th IEEE International Conference on Development and Learning, Proceedings*, 2007.

[8] A. Howard and C. H. Park, "Haptically guided teleoperation for learning manipulation tasks," June 2007.

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

[10] O. C. Jenkins, R. A. Peters, and R. E. Bodenheimer, "Uncovering success in manipulation," in *Robotics: Science and Systems Workshop on Manipulation in Human Environments*, Philadelphia, PA, USA, Aug. 2006.

[11] A. M. Law and D. W. Kelton, *Simulation Modeling and Analysis*. McGraw-Hill Higher Education, 2000.

[12] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical recipes in C (2nd ed.): the art of scientific computing*. New York, NY, USA: Cambridge University Press, 1992.

[13] J. A. Peacock, "Two-dimensional goodness-of-fit testing in astronomy," *Monthly Notices of the Royal Astronomical Society*, vol. 202, pp. 615–627, 1983.

[14] G. Fasano and A. Franceschini, "A multidimensional version of the kolmogorov-smirnov test," *Monthly Notices of the Royal Astronomical Society*, vol. 225, pp. 155–170, 1987.

[15] V. V. Phansalkar and P. S. Sastry, "Analysis of the back-propagation algorithm with momentum," *IEEE Transactions on Neural Networks*, vol. 5, pp. 505–506, May 1994.