Robotic Grasping in Clutter Using a Hierarchy of Supervisors for Deep Learning from Demonstrations

Michael Laskey[1], Jonathan Lee[1], Caleb Chuck[1], David Gealy[1], Wesley Hseih[1], Florian T. Pokorny, Anca D. Dragan[1], and Ken Goldberg[1,2]

***Abstract*— Recent progress in learning from demonstrations increasingly indicates that robots can acquire manipulation skills from large collections of training data. Online learning from demonstration algorithms such as DAgger can in particular learn policies for problems where the system dynamics and the cost function are unknown. Traditionally, these systems have been trained by skilled human supervisors providing feedback on the current learned policy. As the number of demonstrations increases, this approach can however incur a substantial time and labor cost. We study the grasping in clutter problem and propose a hierarchical approach to learning from demonstrations based on DAgger which also incorporates a deep learning architecture to learn policies directly from video sequences. In experiments with a 4-DOF Zymark Robot, we consider a hierarchy of three increasingly competent and costly supervisor types: an analytical grasp planner, a crowd-sourced human workers and a human expert supervisor. Our results indicate that training a policy hierarchically by bootstrapping with the analytical method, followed by crowd sourced and expert supervision improves test-time performance compared to training only with an expert supervisor by XX while reducing training cost by XX. We furthermore find that the addition of crowd-sourced supervisors compared to analytic pre-training followed by supervision by an expert improves performance significantly. [FP: Fine tune how we report main results here]**

I. Introduction

As illustrated by the recent Amazon Picking Challenge at ICRA 2015, the grasping in clutter problem, where a robot needs to grasp an object that might be occluded by other objects in the environment poses an interesting challenge to robotic systems. Two fundamental approaches to grasping in clutter include the analytic model driven approach [**?**], [**?**], [**?**], where the interaction dynamics between the robot and obstacles are formulated analytically. However, modeling all the physical properties of interaction poses a highly challenging problem due to uncertainty in modeling parameters such as inertial properties and friction.

An alternative approach, which we propose here, is to approach the grasping in clutter problem in a model-free manner, where the interaction behavior is learned directly from data within the learning from demonstrations framework [4]. Learning from demonstration (LfD) algorithms have been used successfully in recent years for a large number of robotic tasks, including helicopter maneuvering [2], car parking [3], and robot surgery [24]. Furthermore, deep learning, which we use to learn policies directly from raw video data, has emerged as a highly versatile technique used for this purpose [18].

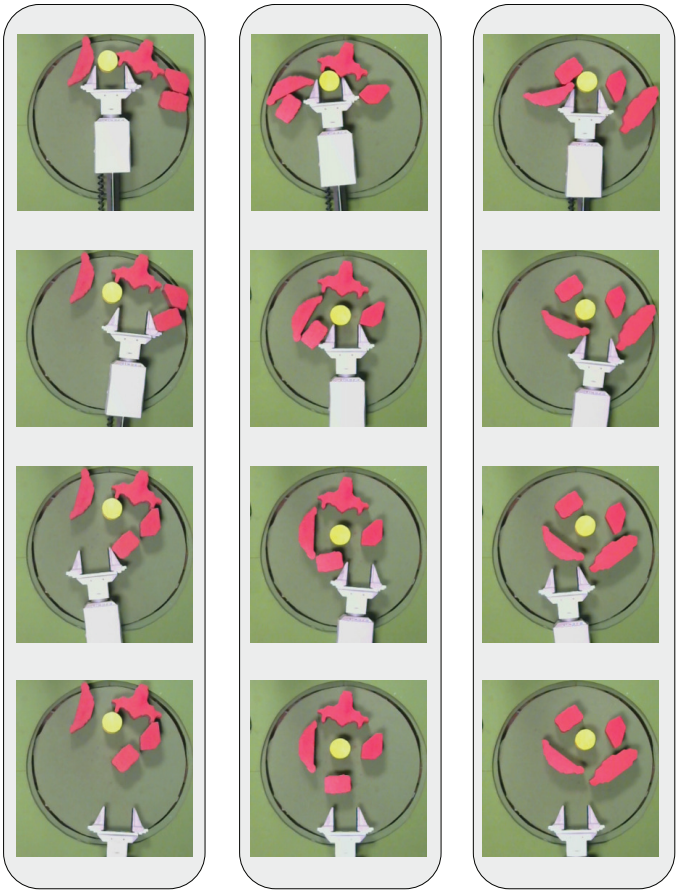


Fig. 1: Three roll-outs of a fully trained grasping in clutter policy (one per column, bottom to top) which was trained using a hierarchy of three supervisors consisting of analytical motion planning, crowd-sourcing and human expert. Red shapes indicate clutter objects and the robot is trained to reach the yellow circle. FP: how about we display the policy after each of the three training steps? One per column

Our approach is based on online Learning from Demonstrations (LfD) where a robot iteratively learns a policy and is provided by feedback on the current policy roll out by a supervisor [9], [20], [21]. We build on, DAgger, an online LfD algorithm which at each iteration, computes a policy based on prior demonstrations, and then rolls out that policy. A supervisor then provides control signals as feedback on the current policy and new state/control examples are aggregated with the prior examples for the next iteration. DAgger and related algorithms have been applied in a wide range of applications, from quadrotor flight to natural language to Atari games [10], [8], [22]. Ross et al. showed that DAgger,

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

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

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

under a no-regret assumption, can be guaranteed to deviate from the supervisor's policy with an error at most linear in the time horizon for the task [21].

One drawback is that DAgger imposes a substantial burden on the supervisor, who must label all states that the robot visits during training. Traditionally there has only been one supervisor and it is deemed an expert [20], [21], [22], [8]. In this work, we propose to instead utilize a hierarchy of supervisors that exhibit increasing cost and competence at a given task to incrementally bootstrap the learning process. At the lowest level of the hierarchy, we in particular consider an analytic motion planner which acts as an affordable supervisor.

II. Related Work

Below, we summarize related work in Robotic Grasiping in Clutter, the Online LfD setting and then the field of Curriculum learning, which is related to the concept of hierarchical supervisors.

Robotic Grasping in Clutter Robotic grasping is a well-established research topic in robotics that has been studied for several decades [6]. A significant amount of prior work focuses on planning grasps given a known object. However, in unstructured environments clutter poses a significant challenge for reaching the planned grasp [12]. Prior work has addressed integrated perception and grasping in clutter where the objective is to grasp objects in an unorganized pile [18], [**?**], but these methods do not specifically aim to grasp a single target object in clutter. Leeper et al. [] use a human operator for assistance to guide the robot through clutter with the objective of grasping a target object. However, the robot required the human to be present at all times and did not attempt to learn to operate autonomously. We are interested in a data-driven approach that only queries a supervisor at training time and which can operate autonomously thereafter.

Prior work has studied the problem of manipulating objects by performing pushing operations []. Berenson et al. [] in particular use a sampling-based planner that considers clearance from obstacles in the environment to plan a grasp approach trajectory in cluttered environments. Cosgun et al. [3] and King et al. [14] consider the problem of planning a series of push operations that move an object to a desired target location. Kiteav et al. planned a trajectory in a physics simulator using LQG based controllers [15]. However, all of these works assume a known model of of the object behavior is provided. We are interested in learning manipulation polices in a data-driven manner instead, leveraging a hierarchy of supervisors for training. **ML: We should discuss how to distinguish our work (Properties of our approach are we assume raw image data, real time and unknown dynamics).**

Online Lfd with an Expert Supervisor Successful robotic examples of Online Learning From Demonstration with an expert supervisor include applications to flying a quad-copter through a forest, navigating a wheel chair across a room and teaching a robot to follow verbal instructions and surgical needle insertion [22], [13], [8], [**?**].

However, to date, these approaches have used only one expert supervisor to provide training data for all parts of the state space. We propose to instead utilize hierarchy of supervisor of different skill level and cost to reduce the overall learning cost.

Reducing Supervisor Burden in Online LfD One approach that has been studied to reduce the number of supervisor supervision is to apply active learning to only ask for supervision when the robot is uncertain about the correct control to apply. Traditional active learning techniques like query-by-committee and uncertainty sampling have in particular be utilized for this purpose [7], [11], [9]

However, Kim et al. demonstrated that due to the non-stationarity of the distribution of states encountered during learning, the traditional active learning techniques may be suitable since the underlying state distribution changes. Thus the use of novelty detection was proposed [13]. Laskey et al. introduced SHIV, using an active learning approach tailored to high dimensional and non-stationarity state distributions and a modified version of the One Class SVM classifier. This enabled the authors to reduce the density estimation problem to a simpler regularized binary classification [**?**]. However the grasping in clutter problem exhibits a high amount of stochasticity requiring a large training data set which poses a significant computational challenge to these methods. In the present paper, we hence consider using Deep Learning, as an underlying scalable learning algorithm in combination with DAgger.

Curriculum Learning Our approach is closely related to ideas from *curriculum learning*, where a neural network is trained via incrementally, first on easier examples and then gradually on data of increasing difficulty [5].

Sanger et al. used curriculum learning in robotics to gradually train a neural network policy to learn the inverse dynamics of a robot manipulator. They then considered a collection scheme where easily learned trajectories where shown to the robot first and then gradually increased the difficulty [**?**].

Our approach is different from curriculum learning in that a supervisor may occasionally provide low quality examples in parts of the state space visited by the robot, which implies that it is now necessarily easier for the robot to learn one supervisor's policy versus another in the hierarchy. In future work, we intend to study potential combinations of hierarchical supervisors and curriculum learning which we believe could provide a promising avenue for research.

III. Problem Statement

Given a collection of increasingly able and costly supervisors $S_1, \ldots, S_M$, the goal of this work is to learn a policy that closely matches that of the most able supervisor S_M while minimizing the overall cost of training a policy. We formalize this approach as follows.

Assumptions and Modeling Choices We assume a known state space and set of controls. We also assume access to a robot or simulator, such that we can sample from the state sequences induced by a sequence of controls.Lastly, we assume access to a set of supervisors who can, given a state, provide a control signal label. We additionally assume the supervisors can be noisy and imperfect, noting that a lower cost supervisor also has lower quality.

We model the system dynamics as Markovian, stochastic, and stationary. Stationary dynamics occur when, given a

state and a control, the probability of the next state does not change over time. Note this is different from the non-stationary distribution over the states the robot encounters during learning. We model the initial state as sampled from a distribution over the state space.

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

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

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

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

A policy is a function $\pi : \mathcal{X} \to \mathcal{U}$ from states to control inputs. We consider a space of policies $\pi_\theta : \mathcal{X} \to \mathcal{U}$ parameterized by some $\theta \in \mathbb{R}^d$. Any such policy π_θ in an environment with probabilistic initial state density and Markovian dynamics induces a density on trajectories. Let $p(\mathbf{x}_t|\theta)$ denote the value of the density of states visited at time t if the robot follows the policy π_θ from time 0 to time $t-1$. Following [21], we can compute the average density on states for any timepoint by $p(\mathbf{x}|\theta) = \frac{1}{T}\sum_{t=1}^{T} p(\mathbf{x}_t|\theta)$.

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

FP: might make this and the above specific to grasping in clutter? **Objective.** The objective of policy learning is to find a policy that maximizes some known cumulative reward function $\sum_{t=1}^{T} r(\mathbf{x}_t, \mathbf{u}_t)$ of a trajectory $\hat{\tau}$. The reward $r : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ is typically user defined and task specific [**?**], [**?**], [**?**]. For example, in the task of inserting a peg into a hole, a function quantifying a notion of distance between the peg's current and desired final state can be used [16].

Since grasp success is typically considered as a binary reward which is observed only at a delayed final state [15], grasping in clutter poses a challenging problem for traditional reinforcement learning methods. We hence instead build upon DAgger which queries a supervisor for appropiate actions, to provide the robot a set of N stochastic demonstrations trajectories $\{\hat{\tau}^1, ...\hat{\tau}^N\}$. This induces a training data set $\mathcal{D}$ of state-control input pairs. We define a 'surrogate' loss function as in [21], $l : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$, which provides a distance measure between any pair of control values. We consider $l(\mathbf{u}_0, \mathbf{u}_1) = ||\mathbf{u}_0 - \mathbf{u}_1||_2^2$.

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

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

Instead of a single supervisor S, which classically is considered to be a skilled human teacher, we instead consider a hierarchy $S_1, \ldots, S_M$ of supervisors which we may be algorithms or humans and which follow policies $\pi_1, \ldots, \pi_M$ with associated expected cumulative rewards R_i satisfying $R_1 \leq R_2 \leq \ldots \leq R_M$. Furthermore, we assume that the cost associated to providing a state label for supervisor S_i is C_i with $C_1 \leq C_2 \leq \ldots C_M$, so that the ordering of supervisors is consistent with respect to both cost and skill level. We consider the problem of minimizing the expected surrogate loss of a trained policy with respect to the most skilled supervisor S_M in the hierarchy while minimizing the overall training cost. In particular, this paper provides an empirical study of greedy combinations of three types of supervisors S_1, S_2, S_3 for grasping in clutter which are applied in order to train a policy parameterized by a deep neural network. Here, S_1 is an analytical motion planning algorithm with a cost $C_1 = XX$, S_2 a supervisor consisting a crowd-sourced Amazon Mechanical Turk laborers with $C_2 = YY$, and S_3 a skilled human supervisor with $C_3 = ZZ$.

IV. Approach and Background

A. Details on DAgger: Dataset Aggregation

Since the cumulative expected reward of a policy is difficult to optimize directly, DAgger [21] instead solves the minimization in Eq. 1 by iterating two steps: 1) computing the policy parameter θ using the training data $\mathcal{D}$ thus far, and 2) execute the policy induced by the current θ, and ask for labels for the encountered states.

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

$$\theta_k = \arg\min_\theta \sum_{i=1}^{M} l(\pi_\theta(\mathbf{x}_i), \mathbf{u}_i). \tag{2}$$

This sub-problem is a supervised learning problem, solvable by estimators like a support vector machine or a neural net. Performance can vary though with the selection of a the estimator Selecting the correct function class depends on the task being consider and knowledge of the problem, see for a guide [23].

2) Step 2: The second step DAgger rolls out their policies, π_{θ_k}, to sample states that are likely under $p(\mathbf{x}|\theta_k)$. For every state visited, DAgger requests the supervisor to provide the appropriate control/label. Formally, for a given sampled trajectory $\hat{\tau} = (\mathbf{x}_0, \mathbf{u}_0, ..., \mathbf{x}_T, \mathbf{u}_T)$, the supervisor provides labels $\tilde{\mathbf{u}}_t$, where $\tilde{\mathbf{u}}_t \sim \tilde{\pi}(\mathbf{x}_t) + \epsilon$, where ϵ is a small zero mean noise term, for $t \in \{0, \ldots, T\}$. The states and labeled controls

are then aggregated into the next data set of demonstrations $\mathcal{D}_{k+1}$:

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

FP: can we just have a neat algorithm box here?

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

B. DAgger with Supervisor Hierarchy

FP: Michael will rewrite this to give a simple explanation of the greedy approach here. We will argue about the choices in the experimental settings using sparsity While the hierarchical supervisor problem can be framed as a contextual multi-armed bandit – an extension of a traditional multi-armed bandit except the sampled reward depends on an additional state term [17]. At each iteration a supervisor is selected from M supervisors and the state is the current policy parameters or θ_k. Then a sampled reward is received which is how well the trained policy performed on the sample surrogate loss measured against the $\tilde{\pi}_M$ or the supervisor at the top of the hierarchy. Note we only measure against the supervisor at the top of the hierarchy because that is our primary objective (i.e. Eq. 1).

While a large amount of literature exists to solve contextual multi-armed bandit problems, a common assumption is that the sampled reward can be observed [17]. However, in our situation evaluation of the sampled reward requires querying the costliest supervisor every iteration. Thus, defeating the purpose of cost reduction. In light of this we propose a greedy allocation strategy where we train a policy with the cheapest supervisor first until convergence and then work up the hierarchy training with each supervisor.

Our algorithm is as follows: first iterate through Step 1 and 2 of DAgger for a given supervisor, however after iteration K or when the current policy, $\pi_{\theta_{m,k}}$ is able to achieve a loss ϵ on the sampled J states at that current iteration (i.e):

$$\frac{1}{J}\sum_{j=1}^{J} ||\pi_{\theta_{m,k}}(\mathbf{x}_j) - \tilde{\pi}_m(\mathbf{x}_j)||_2^2 \leq \epsilon$$

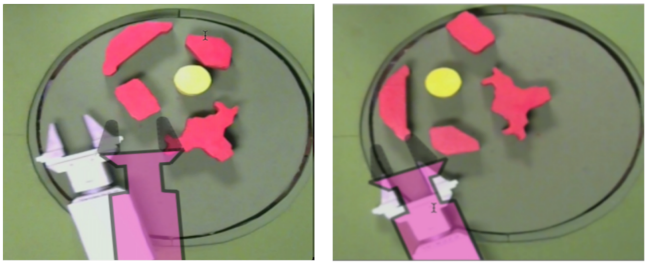
Then iterate to the next supervisor in the hierarchy or $m = m+1$. The hyperparameter ϵ can be set as a measure of how different the controls applied between the robot's policy and the current supervisors a sequential process can tolerate. For example surgical sequential processes could warrant a very low threshold, but less precise sequential processes, like grasping in clutter, can tolerate a higher value.

An issue arises though when performing this iterations because now the current dataset $\mathcal{D}_K$ has examples from a different supervisor that receives a smaller expected cumulative reward. This could cause a learning algorithm to try and fit to contradictory examples and be worse than either supervisor trained with [23].

We are thus interested in not storing the dataset collected from the previous supervisor and only remembering the current $\theta_{m,K}$ weight parameters. Thus, we propose only performing DAgger for each individual supervisor in the hierarchy and using the resulting weight vector to bootstrap the learning of the next supervisor. We found empirically in Sec. VI that it also beneficial to add to the new dataset the states and controls applied by the current policy $\pi_{\theta_{m,k}}$ that the current supervisor agreed with by some euclidean distance $\tilde{\epsilon}$.Thus, acting as a regularization on the optimization since stochastic gradient on a section of the min-batch update would be zero for examples where the next supervisor agrees with the previous.

ML: we can formally prove the following statements or point to the web similar to SHIV... it will be alot easier to write these out though if we have space With respect to the theoretical analysis of DAgger's convergence rate, one can see that in our algorithm you are solving M individual DAgger's [21]. Thus, in the worst case you will converge to the supervisor at the top of the hierarchy $\tilde{\pi}_M$ in $\tilde{O}(MT)$ iterations instead of DAgger's original $\tilde{O}(T)$.

It should be noted though that as you iterate through the hierarchy of supervisors, each iteration starts out with an error bounded by $\int_{\mathcal{X}_{m,m+1}} ||\pi_{\theta_{0,m+1}}(\mathbf{x}) - \tilde{\pi}_{m+1}(\mathbf{x})||_2^2 p(\mathbf{x}|\theta_{0,m+1}) d\mathbf{x} + \epsilon$. Thus, if the supervisors are able to agree on a large part of the state space, there should be less iterations with the expert supervisor than $\tilde{O}(T)$, which we experimentally show in our grasping in clutter task.



(a) Rotation (b) Extension

Fig. 2: The interface AMT workers see for providing feedback to the robot for the grasping in clutter task. The pink overlay indicates the desire change in the robot position with respect to the current robot state. Then AMT workers can use their intuition for how objects respond to force to provide examples of how the robot should behave. Each window shows a correction in a degree of freedom a) is the rotation of the robots base b) is extending the robot's arm.

V. Robot Grasping in Clutter

We now describe the grasping in clutter problem, the supervisor hierarchy that we used to solve it and our deep learning policy architecture.

A. Grasping in Clutter

Consider a robot in an Amazon warehouse. The robot has an intended goal object for it to grasp on a shelf, however other objects could obstruct the path towards the goal object. These obstructions of movable objects prevent leveraging common motion planning techniques, because they do not posses a dynamics model of how the objects behave under different contact forces. [15], [14].

Thus, the robot must reason about how its interactions with the environment will affect the intended outcome. This can be hard for two reasons 1) it becomes hard to find a

[1]In the original DAgger the policy rolled out was stochastically mixed with the supervisor, thus with probability β it would either take the supervisor's action or the robots. The use of this stochastically mix policy was for theoretical analysis. In practice, it is recommended to set $\beta = 0$ to avoid biasing the sampling [10], [21]

low dimensional state representation for the task because the environment dynamics are affected by the global shape of each object 2) the dynamics of the environment involve modeling how K objects interact with each other under an arbitrary force [15].

In order to achieve real time grasping in clutter, which is important for a robot in a shipping warehouse, we leverage visuo-motor deep learning to learn a control policy for the task over a large number of different configurations working with a hierarchy of supervisors. Deep learning has the potential to take high dimensional image data of the scene and learn task specific features relevant for grasping in clutter. Additionally a trained neural network policy is can be computational inexpensive to evaluate and can result in real time performance [16].

B. Supervisor Hierarchy

In this section, we list each supervisor in the order they appear in our hierarchy. The first supervisor is deemed the least expensive.

Motion Planning Supervisor Our first supervisor is a an analytical method that computes the trajectory the robot should move in to reach the goal object, or yellow circle, with a relax dynamic model of the environment. Our method employs template matching to identify the goal object in the image and the using the forward dynamics of the robot to computes the relative change in direction the robot should apply as an control. The template matching is implemented in OpenCV and uses the normalized cross-correlation filter [**?**].

In this stage, the motion planning supervisors only tries to teach the robot to move towards the goal. Note that the motion planning supervisor's advice does not have any knowledge about how the cluttered objects will respond to the forces applied via the robot arm, which results in sub-optimal polices. However, our motion planning supervisor is both computationally and monetarily inexpensive to run which allows us to provide a large number of examples.

CrowdSourced Supervisor Our next level of supervisor relies on a crowd source service, called Amazon Mechanical Turk (AMT). Chung et al. demonstrated the reliable data learning from demonstration data could be obtained from an AMT platform. The AMT platform makes readily available thousands of human workers that can perform remedial task for a price range of $0.1 per robot trajectory[**?**]. Thus potentially providing a higher quality supervisor who has an intuition for how the cluttered objects interact with the world, but at a slightly higher cost.

In order to get examples from a CrowdSourced Supervisor, we designed an interface shown in Fig. 2. The interface draws a transparent overlay that shows how the robot would respond provided the current control this allows for the AMT worker to see some of the effect of their control. We rely on their natural intuition for how objects to infer how the change in robot position will change the objects.

We also design a tutorial for the work that first has them perform designated motions on a virtual robot and introduces them to a problem. We additionally provide a video of an expert providing corrections on three robot trajectories.

Expert Supervisor Finally after training with the Motion Planning and Crowdsourced Supervisor we leverage the Expert supervisor, who is capable of achieving high cumulative reward but is a limited resource. An expert supervisor in this case is a Phd student in machine learning and robotics, which can cost on average ten times more than an AMT worker to employ [].

An expert supervisor can be used in a variety of scenarios.They first would have a better intuition of the physical limitations of the robot and environment, such as joint limits or how certain objects might behave under force. Furthermore, they would also understand how the examples given could lead to better training of the parameterized policy. For example, understanding the feature space the policy lies in can lead to the expert not providing contradictory examples.

It can still be hard for an expert to provide the correct feedback without knowing the magnitude of the controls suggested, so they use the same interface as the CrowdSourced Supervisor.

C. Neural Network Policy Architecture

Our policy is represented as deep neural network, which was trained using TensorFlow [1]. Our network architecture consists of 1 convolutional layer with 5 channels and filters with size 11x11, a fully connected layer with an output of 128 dimensions and a final layer that maps to a four dimensional control signal. We used ReLus to separate the different layers and a final tanh on the output to scale the output between -1 and 1.

The control examples was scaled between 1 and -1 for each dimension independently. To be robust to lighting and reduce the dimensionality of the problem, we applied a binary mask to each channel of the 250x250 RGB image, the mask would set to 1 values above 125 and 0 other wise. We then validated that all information (i.e. location of the gripper, cluttered shapes and goal object) where still visible in the masked image.

To determine our architecture we preformed a grid search ofver different architectures trained after 400 iterations with a batch size of 200. The set of architectures consist of different network architectures as well as different momentum terms, and weight initialization schemes. We trained all networks on a dataset of 6K images labeled with the Analytical Supervisor on a Nvidia Tesla K40 GPU, which is able to train each network in an average of 10 minutes.

VI. Experiments

For an experiment in the grasping in clutter domain, we are interested in training a Zymark robot to perform a grasping in clutter task on image data taken from a Logitech C270 camera. Examples of images from the camera can be seen in Fig. 1.

The objects in clutter are made Medium Density Fiberboard and each one is on average 4" in diameter. The objects deemed cluttered are painted Red and the goal object is painted yellow. There is an inscribed circle around the work space to keep the robot from pushing the objects outside of the work space. The inscribed circle can make the task more challenging because the robot cannot simply "sweep" the cluttered objects off the table.

The robot, shown in Fig. **??**, has a 3 dimensional internal state of base rotation, arm extension and gripper. The robot is

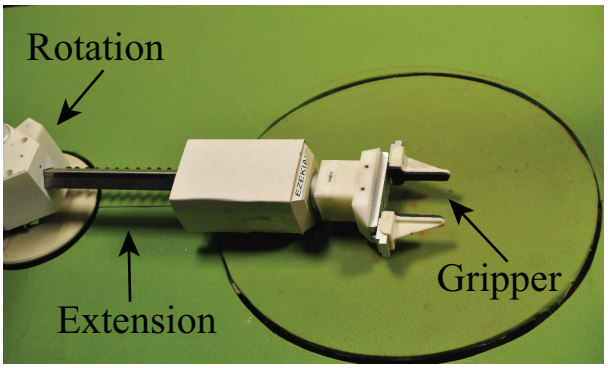
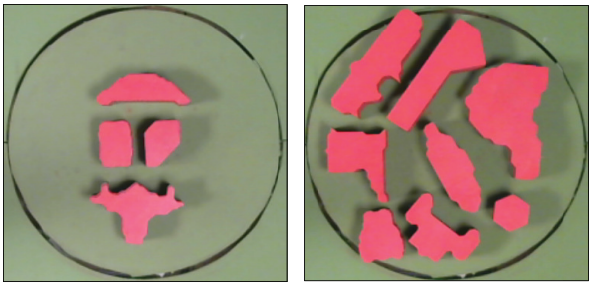


Fig. 3: Shown above is a Zymark robot. The robot consists of an 3DOF arm that lies in a planar workspace and the ability to rotate the turn table. The inscribed circle in the work space prevents the robot from learning the trivial policy of just pushing the objects off the table.



Training Objects Test Objects

Fig. 4: Examples of different states a supervisor is ask to provide labels for. a) Represents states that are easy for the supervisor to provide labels for. b) Represents states that are difficult for a human to provide examples and can lead to contradictory examples in similar states

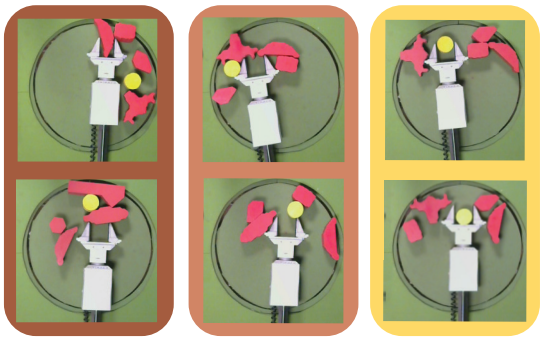
commanded via state position requests, which are tracked by a tuned PID controller. The policy π_θ outputs delta positions that are bounded by $15\,\text{deg}$ for the gripper and turntable, 1 cm for the arm extension and 0.5 cm for the gripper at each time step. There is $T = 100$ time steps in a trajectory.

To test the performance of a policy, we created a test set composed of 20 different configurations each containing objects that were not trained on. The test set configurations varied in number of objects on the table, size of objects and relative pose of each object. We measure success as defined by three situations not successful, near successful, successful. We report total success as a score out of 2: 0 for not successful, 1 for near successful, 2 for successful.

The not successful situation corresponds to the case when the robot's gripper is not touching the goal object at the end of a rollout. Examples of when this happens is when the robot fails to push the objects away or does not head in the direction of the yellow circle. The near successful situation is defined as the robot touches the goal object with its gripper, but it is not enclosed in the gripper. Examples of when this happens is when the robot gets a small obstacle object trapped in the gripper first, or when the root slightly misses the yellow circle on its approach. Successful is defined as getting the goal object inside the gripper at the end of a trajectory.

A. Hierarchical Supervisors

We first test that using a hierarchy of supervisors can reduce total cost needed to train a policy, but still maintain



Not Successful Near Successful Successful

Fig. 5: Examples of different states a supervisor is ask to provide labels for. a) Represents states that are easy for the supervisor to provide labels for. b) Represents states that are difficult for a human to provide examples and can lead to contradictory examples in similar states

similar performance . We compare having an analytical only supervisor for a fixed amount of data of 160 demonstrations. The non-hierarchical policies trained with the expert supervisor only for 160 demonstrations or only the MPIO supervisor for 160 demonstrations. The policy trained with a hierarchical supervisor first receives 100 demonstrations from the MPIO supervisor and 60 demonstrations from the expert supervisor.

Our results, shown in Fig. 6, are the policy trained with just the MPIO supervisor achieves a score of 0.325%, the policy trained with just an expert achieves 0.54% and the policy trained with a hierarchy of supervisor achieves 0.55%. Thus, the hierarchical supervisor and expert supervisor achieve approximately the same performance. However,training with a hierarchical supervisor yields a 40% reduction in cost incurred compared to the policy trained with only an expert supervisor.

Thus demonstrating that by using a hierarchy of supervisors, we can reduce the cost of training a policy and achieve similar performance to the

B. Quality Crowdsourced Supervisor

We next evaluate the potential of a crowdsourced supervisor for a being part of the hierarchy. Thus, we perform an experiment using the AMT platform described in Sec. V-B. We test how well a crowdosurce supervisors performs as part of a hierarchy of supervisors, we had a policy train with 100 demonstrations from the MPIO supervisor , then had 60 demonstrations provided by AMT workers. We also compare how well the crowdsource supervisors performs just as well as a single supervisor, by having them provide all 160 demonstrations.

Our results, shown in Fig. 6, are the policy trained with just the crowdsourced supervisor achieves a score of $?\%$. The policy trained with the hierarchical crowdsourced and MPIO supervisor receives 0.43% on our test set compared to 0.55% with the motion planning and expert supervisors hierarchy. The total cost received with the only crowdsource supervisor was 15, while the hierarchical supervisor was 7.

ML: talk about crowdosuce statistics

C. Advancing in the Hierarchy

We further test how to advance between supervisors in the hierarchy.We examine three different strategies for dataset management when changing supervisors 1) aggregating the dataset of the data collected with both supervisors 2) transferring only the weights of the nerural network policy

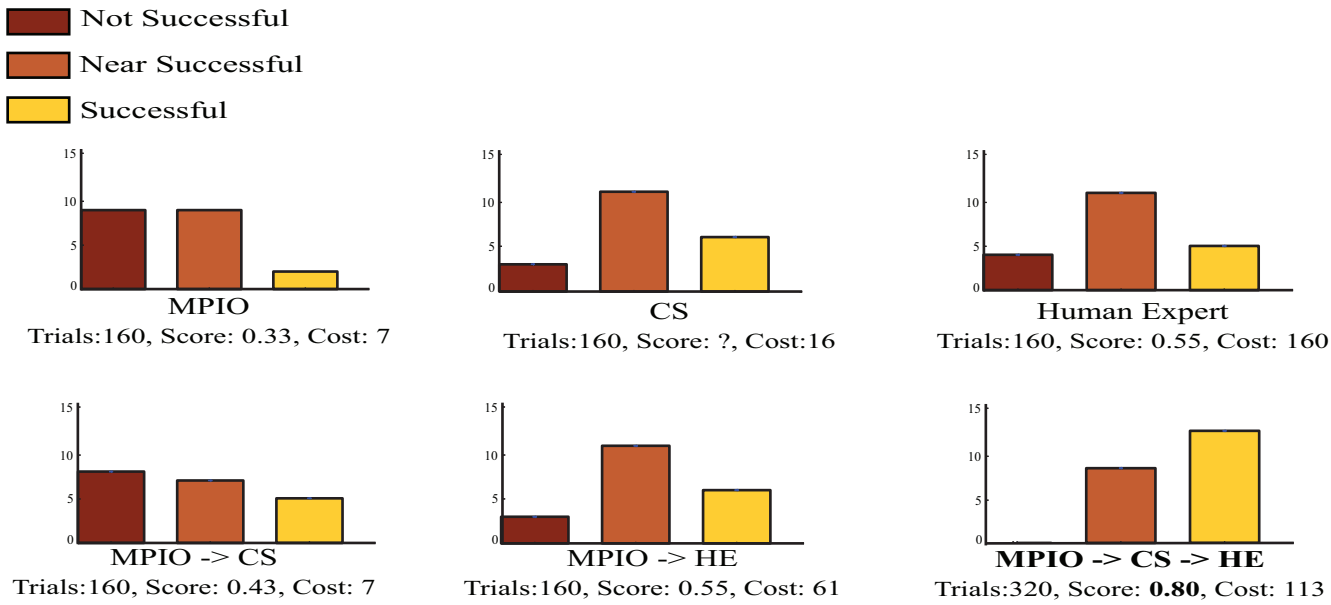


Fig. 6: Examples of different states a supervisor is ask to provide labels for. a) Represents states that are easy for the supervisor to provide labels for. b) Represents states that are difficult for a human to provide examples and can lead to contradictory examples in similar states

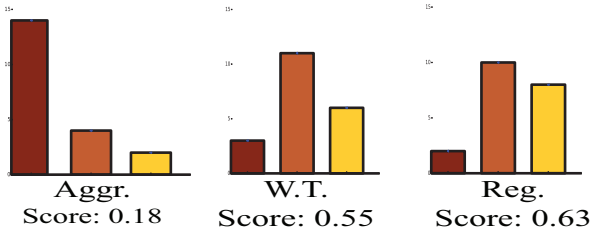


Fig. 7: Examples of different states a supervisor is ask to provide labels for. a) Represents states that are easy for the supervisor to provide labels for. b) Represents states that are difficult for a human to provide examples and can lead to contradictory examples in similar states

or θ vector between the supervisors and removing the first's supervisors dataset completely 3) transferring only the weights of the neural network policy but adding the values output by $\pi_\theta(\mathbf{x})$ instead of $\tilde{\pi}_m(\mathbf{x})$at parts of the states visited that the current supervisor is close to agreement with as measured by the following $||\tilde{\pi}_m(\mathbf{x}) - \pi_\theta(\mathbf{x})||_2^2 < 0.01$ **ML: need to get this number off the local code, will update tomorrow**. Thus, acting as a regularization on the optimization since stochastic gradient on a section of the min-batch update would be zero for examples where the next supervisor agrees with the previous.

We compare each strategy on a hierarchy of a MPIO supervisor trained with 100 iterations and then advancing to a human expert supervisor trained with 60 iterations. Our results reported in Fig.7 , show that aggregation strategy achieves 0.18%, the weight transfer strategy achieves 0.55% and the regularized stochastic gradient update strategy achieves 0.63%. Thus, suggesting that techniques to help the policy remain close to the previously trained supervisor are useful for effectively switching between supervisors.

D. Scaling the Hierarchy

The final experiment we ran is to run our algorithm on a hierarchy with 3 supervisors: MPIO, crowdsourced and human expert. We ran each policy until we observed the pay out in terms of reward achieved was not increasing. This results in MPIO for 100 demonstrations, crowdsourced for 120 demonstrations and human expert for 100 demonstrations. Thus resulting in 320 trials on the robot and 320,000 annotated images of what control the robot should apply.

The results of our trained policy, as shown in Fig. 6, demonstrate that by leveraging a hierarchy we are able to achieve a score of 80% with a cost of only 113. We note that policy trained with the human expert supervisor, which incurred a cost of 160, was only able to achieve a score of 0.66%.

While our policy was able to do substantially better, to near successful mode a reasonable portion of the time. These failure modes where do to the robot not being able to accurately manipulate smaller objects, where slight perturbations could result them being pushed into the gripper, and accurately controlling the goal object to land in the circle (i.e. sometimes pushing against the side of the circle instead). We attribute this problem to a limit in the detail of precision our supervisors can currently give, future work will look at more precise analytical methods and fine tuning the policies via self-learning.

VII. Discussions and Future Work

VIII. Acknowledgments

This research 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. This work is supported in part by the U.S. National Science Foundation under Award IIS-1227536, NSF-Graduate Research Fellowship, by the Knut and Alice Wallenberg Foundation and the National Defense Science and Engineering Graduate Fellowship Program. We thank Pieter Abbeel, Sergey Levine and Sanjay Krishnan for insightful feedback.

REFERENCES

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

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

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

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

[5] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*. ACM, 2009, pp. 41–48.

[6] A. Bicchi and V. Kumar, "Robotic grasping and contact: A review." Citeseer.

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

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

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

[10] X. Guo, S. Singh, H. Lee, R. L. Lewis, and X. Wang, "Deep learning for real-time atari game play using offline monte-carlo tree search planning," in *NIPS*, 2014, pp. 3338–3346.

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

[12] D. Katz, J. Kenney, and O. Brock, "How can robots succeed in unstructured environments," in *In Workshop on Robot Manipulation: Intelligence in Human Environments at Robotics: Science and Systems*. Citeseer, 2008.

[13] B. Kim and J. Pineau, "Maximum mean discrepancy imitation learning." in *Robotics Science and Systems*, 2013.

[14] J. E. King, J. A. Haustein, S. S. Srinivasa, and T. Asfour, "Nonprehensile whole arm rearrangement planning on physics manifolds," *ICRA 2015 IEEE*.

[15] N. Kitaev, I. Mordatch, S. Patil, and P. Abbeel, "Physics-based trajectory optimization for grasping in cluttered environments," pp. 3102–3109, 2015.

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

[17] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 661–670.

[18] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," *arXiv preprint arXiv:1509.06825*, 2015.

[19] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," Carnegie-Mellon University, Tech. Rep., 1989.

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

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

[22] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert, "Learning monocular reactive uav control in cluttered natural environments," in *ICRA, 2013 IEEE*. IEEE.

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

[24] J. Van Den Berg, S. Miller, D. Duckworth, H. Hu, A. Wan, X.-Y. Fu, K. Goldberg, and P. Abbeel, "Superhuman performance of surgical tasks by robots using iterative learning from human-guided demonstrations," in *ICRA, 2010 IEEE*. IEEE, 2010, pp. 2074–2081.