Robot Grasping in Clutter: Using a Hierarchy of Supervisors for Learning from Demonstrations

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

***Abstract*— For applications such as Amazon warehouse order fulfillment, robots must grasp a desired object amid clutter: other objects that block direct access. This can be difficult to program explicitly due to uncertainty in friction and push mechanics and the variety of objects that can be encountered. Deep Learning networks combined with Online Learning from Demonstration (LfD) algorithms such as DAgger and SHIV have potential to learn robot control policies for such tasks where the input is a camera image and system dynamics and the cost function are unknown. To explore this idea, we introduce a version of the grasping in clutter problem where a yellow cylinder must be grasped by a planar robot arm amid extruded objects in a variety of shapes and positions. To reduce the burden on human experts to provide demonstrations, we propose using a hierarchy of three levels of supervisors: a fast motion planner that ignores obstacles, crowd-sourced human workers who provide appropriate robot control values remotely via online videos, and a local human expert. Physical experiments suggest that with a fixed budget of 160 expert demonstrations, using the hierarchy of supervisors can increase the probability of a successful grasp (reliability) from 55% to 90%.**

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, poses an interesting challenge to robotic systems. This problem is relevant to industrial applications such as warehouse shipping operations and flexible manufacturing in semi-structured environments. One fundamental approach to grasping in clutter is the analytic model driven approach [6], 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.

Another approach to the grasping in clutter problem is a data-driven manner, where the interaction behavior is learned directly from interactions with the environment and a supervisor, which can be an expert human or an analytical method [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 [34]. 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 [26].

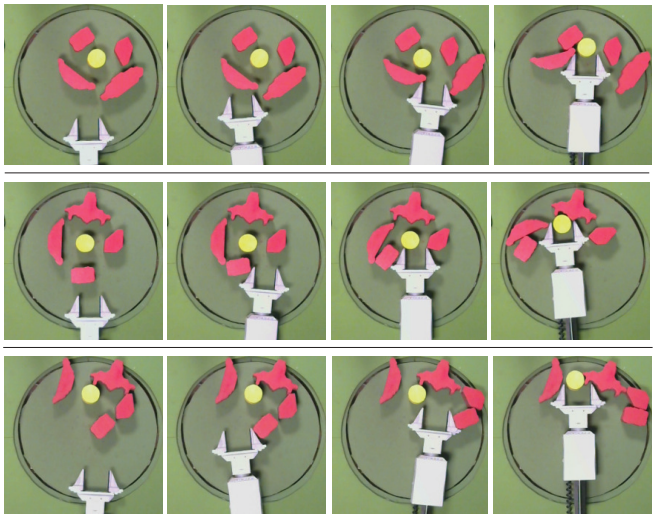


Fig. 1: Three policy roll-outs on a Zymark 3-DOF Robot of a fully trained grasping in clutter policy (one per row, left to right) which was trained using a hierarchy of three supervisors consisting of a motion planning algorithm, crowd-sourcing and a human expert. Red shapes indicate clutter objects and the robot is trained to reach the yellow cylinder. The trained manipulation policy is represented as a deep neural network that receives as input an image of the scene and outputs a change in state position. The resulting policy learns to sweep away clutter objects to reach the goal object.

Our approach is based on Online Learning from Demonstrations where a robot iteratively learns a policy and is provided feedback on the current policy roll out by a supervisor [13], [27], [28]. 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, such as quadrotor flight, natural language and Atari games [14], [12], [29]. Ross et al. showed that DAgger, 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 [28].

One drawback of DAgger is that it imposes a substantial burden on the supervisor, who must label all states that the robot visits during training. Previously, only a single expert supervisor has been considered [27], [21], [28], [29], [12].

We propose to instead utilize a hierarchy of supervisors to incrementally bootstrap the learning process, as opposed to utilizing an expensive expert supervisor from scratch. At the lowest level of the hierarchy, we use a motion planner

[1] Department of Electrical Engineering and Computer Sciences; {mdlaskey, jonathan_lee, calebchuck, dgealy, wesleyhsieh, ftpokorny, anca}@berkeley.edu

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

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

on a simplified version of the grasping in clutter problem that ignores ambient obstacles. For the next supervisor in the hierarchy, we consider crowd-sourced workers on Amazon Mechanical Turk. The robot is finally trained with a PhD student in robotics, who is considered an expert human supervisor. Experiments suggest that with a fixed budget of 160 expert demonstrations leveraging a hierarchy of supervisors can increase reliability from 55% to 90% on a test set of unknown object configurations.

II. Related Work

Below, we summarize related work in Robotic Grasping in Clutter, the Online LfD setting and Curriculum Learning.

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 [16].

Prior work has studied the problem of manipulating objects by performing pushing operations [24]. Cosgun et al. [9] and King et al. [18] consider the problem of planning a series of push operations that move an object to a desired target location. We are interested in reaching a desired goal object, not how to move an object to a specific pose.

Additionally, prior work has studied the use of hierarchical approaches for motion planning amid movable obstacles [32], [35]. The approach is to use sampling-based methods to compute a high-level plan to move objects out of the way and get to a goal location. Dogar and Srinivasa applied this approach to push-grasping in clutter to plan a trajectory towards a grasp by maintaining a set of movable objects [11]. Our approach could be integrated with a high level planner for more advanced problems, where removing objects from the environment is needed.

Dogar et al. proposed a physics-based grasp planning approach that pre-computes and caches interactions with obstacles in the environment, which allows for fast real evaluation for a planner [10]. However, it does not account for inter-object interactions, which can be significant in cluttered environments. A supervisor could potentially impart an intuition for inter-object interaction while training the robot. Kiteav et al. planned a trajectory in a physics simulator using LQG based controllers [19] to reach a goal object surrounded by movable obstacles. They demonstrated that leveraging knowledge of object dynamics can lead to higher success than only using motion planning. However, for unknown objects it can be difficult to know parameters such as mass and friction coefficient, which are important for simulation.

Leeper et al. [22] used 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 was not operated autonomously. We are interested in a data-driven approach that only queries a supervisor at training time and can operate autonomously thereafter.

Prior work has addressed integrated perception and grasping in clutter where the objective is to grasp objects in an unorganized pile [26], [25], but these methods do not specifically aim to grasp a single target object in clutter. We train a robot specifically to move towards a goal object and treat other objects as clutter.

A model-free approach to robotic manipulation is Guided Policy Search [23]. However, this assumes a known low-dimensional state representation to both estimate dynamics and apply an LQG controller. In the grasping in clutter domain, it can be hard to specify such a low dimensional representation due the dependence on the objects' shape.

Online LfD with an Expert Supervisor Successful robotic examples of Online Learning From Demonstration with an expert supervisor include applications of flying a quad-copter through a forest, navigating a wheel chair across a room, teaching a robot to follow verbal instructions and surgical needle insertion [29], [17], [12], [21]. 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 a hierarchy of supervisors with different skill levels and cost to reduce the overall burden on the expert.

Reducing Supervisor Burden in Online LfD Active learning approaches to reducing supervisor burden 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 been utilized for this purpose [8], [15], [13]. Kim et al. demonstrated that due to the non-stationarity of the distribution of states encountered during learning traditional active learning techniques may not be suitable. Thus the use of novelty detection was proposed [17]. Laskey et al. introduced SHIV, which used an active learning approach tailored to high dimensional and non-stationarity state distributions and a modified version of the One Class SVM classifier. This reduced the density estimation problem to a simpler regularized binary classification [21].

However, all of these works focus on reducing the duration of the demonstration an expert needs to provide. We focus on reducing the number of demonstrations the expert needs to provide, not the duration.

Curriculum Learning Our approach is closely related to ideas from curriculum learning, where a neural network is trained incrementally: first on easier examples and then gradually on data of increasing difficulty [5]. Sanger et al. used curriculum learning 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 were shown to the robot first and then gradually increased the difficulty [30]. We avoid the complexity of ranking training data by difficulty and instead propose an iterative scheme where data is presented in an unbiased manner to a hierarchy of supervisors.

III. Robot Grasping in Clutter

We consider a robot grasping an object in a cluttered environment. The robot's goal is to grasp a target, or goal, object in the presence of extruded objects obstructing the path towards the goal object. Kiteav et al. demonstrated that applying only a motion planner to this problem is not as successful as incorporating knowledge of object dynamics [19]. However, since contact dynamics can be difficult to model without information such as friction coefficient and mass [19], [18], we do not assume knowledge of explicit object or dynamics models and instead learn a grasping policy

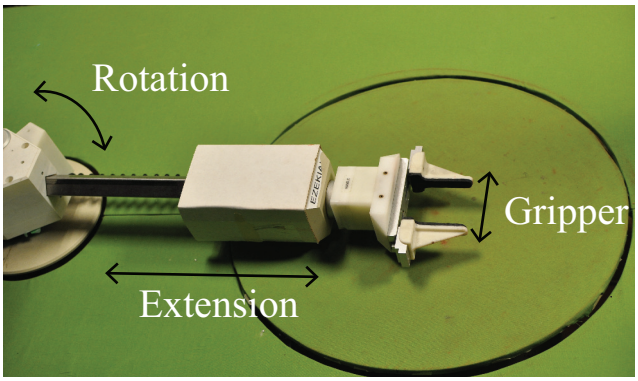


Fig. 2: Shown above is a Zymark robot. The robot consists of a 3DOF arm in a planar workspace. It is able to extend its arm and rotate about a fixed base. The robot can also open and close its gripper.

directly from demonstrations. We consider the following robot system, three supervisors S_1, S_2, S_3 and neural network policy architecture:

Robot System The robot, shown in Fig. 2, has a 3 dimensional internal state of base rotation, arm extension and gripper opening width. The state space of the environment, $\mathcal{X}$, is captured by an overhead Logitech C270 camera, which is positioned to capture the workspace that contains all cluttered objects, the goal object and the robot arm. We use only the current image as state space, which captures positional information. We hypothesize the need to include other information such as velocity, as the speed of the robot is increased. Examples of images from the camera can be seen in Fig. 1. The robot is commanded via position control with a PID controller. Similar to [21], the controls, $\mathcal{U}$, to the robot are bounded changes in the internal state, which allows for us to easily register control signals to the demonstrations provided by supervisors as opposed to torque control. The control signals for each degree of freedom are continuous values with the following ranges: base rotation, $[-15°, 15°]$, arm extension $[-1, 1]$. The units are degrees for base rotation and centimeters for arm extension. The precision of the control signal is 0.001 for each degree of freedom. In supervised learning, or regression, it is common to use a high level of precision for control signals [23], [21], [17].

The robot receives feedback from a supervisor by first executing a trajectory that has a time horizon, T, where $T = 100$. The trajectory, which is composed of 100 states, or images captured by the camera, is then given to a supervisor, who provides a demonstration. A demonstration is defined as the supervisor providing the control they would apply if they were in the robot's current state, for each state in the trajectory. The robot then uses these demonstrations in an approach similar to supervisor learning, described in Sec. IV, to update its current manipulation policy.

MPIO: Motion Planner Ignoring Obstacles Supervisor Supervisor, S_1, is a first order motion planner that computes control signals for the robot to reach the goal object in a straight line, ignoring the presence of obstacles. We refer to this supervisor as Motion Planner Ignoring Obstacles (MPIO). Our method employs template matching to identify the goal object in the image and uses the forward dynamics of the robot to compute the relative change in direction the robot should apply as a control. The template matching is implemented in OpenCV and uses the normalized cross-correlation filter [7]. This supervisor is designed to just provide an initial improvement from a random neural network policy.

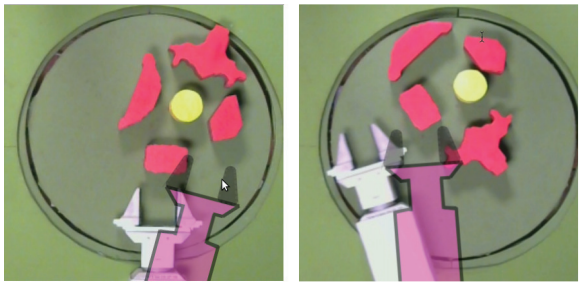


Fig. 3: We display the interface AMT workers see for providing feedback to the robot for the grasping in clutter task. The pink overlay indicates the desire control, which is a bounded change in internal state, the robot should apply to move towards the yellow object. Then the AMT workers can use their intuition for how objects respond to force to provide examples of how the robot should behave. We show two examples of robot state and control label a human supervisor would provide via a Computer Mouse.

Crowd-Sourced Supervisor Supervisor, S_2, uses a crowd source service, called Amazon Mechanical Turk (AMT). The AMT platform makes readily available thousands of human workers, who can provide supervision for \$0.10 per demonstration. The time to provide a demonstration to the robot is 50s, since the robot trial is slowed down to a half second per time-step in the trajectory.

In order to have the AMT workers provide demonstrations, we designed the supervision interface shown in Fig. 3. The interface displays a video of a trajectory in the image state space and allows the human supervisors to command an overlay visualization of the robot arm to demonstrate the correct control the robot should apply. The interface is controlled via a mouse. When the mouse button is pressed a robot visualization appears and can be dragged to the desired control (or change in state) the robot should apply. The desired control is bounded by the range in controls listed above.

We designed a tutorial for the Crowd-Sourced Supervisor. We first describe the learning task to them as a robot that is trying to reach a yellow cylinder and needs their help learning how to get it. In a tutorial, they are then trained to perform designated motions with a virtual robot, to help them understand the robot's kinematics. We additionally provide a video of an expert providing corrections on three robot trajectories. We instruct the Crowd-Sourced Supervisor to provide a control only when they are confident. If a state in the demonstration does not receive a label then that state is not added to the training set.

To help ensure quality, we provide all Crowd-Sourced Supervisors with the same trajectory first and compare their demonstration against a demonstration of a Human Expert Supervisor. If the average Squared-Euclidean distance over control signals between demonstrations is above a threshold, then the Crowd-Sourced Supervisor is not asked to provide additional demonstrations. The threshold is set with the intention to only remove AMT workers, who provide arbitrary demonstrations, no demonstrations or possibly adversarial demonstrations. On average the AMT workers would provide only 8 demonstrations, however for a larger set of demonstrations it is advised to periodically test their quality.

Expert Supervisor Supervisor, S_3 is the Expert Supervisor, who is capable of a better understanding of the problem domain but is a limited resource. An Expert Supervisor in this case is a PhD student in machine learning and robotics. The Expert Supervisor uses the same interface as the Crowd-Sourced supervisors, shown in Fig. 3, to provide examples to the robot.

An Expert Supervisor can utilize a variety of knowledge about the physical limitations of the robot and environment, joint limits and experience in how certain objects might behave under force. Furthermore, the Expert Supervisor might have an intuition for how training examples could lead to better training of the robot.

Neural Network Policy Architecture In Online LfD, a robot is trained by learning a policy, or function mapping state space $\mathcal{X}$ to control space $\mathcal{U}$. Commonly, a policy is a specific function in some function class such as linear regression, kernelized ridge regression, or a neural network. To handle the high dimensional image space of the problem, we are interested in representing the robot's policy as a deep neural network. Deep learning can be advantageous because it has the potential to learn features relevant for manipulation [23], [26]. Furthermore as opposed to kernel based methods, neural networks readily support online updates (i.e. the ability to add more demonstrations without having to retrain on previous demonstrations) [31], which can be important when switching supervisors in the hierarchy.

To facilitate training the neural network, we scaled the controls between 1 and -1 for each dimension independently, which is common for training [1]. To be robust to lighting and to reduce the dimensionality of the problem, we applied a binary mask to each channel of the 250x250 RGB image setting values above 125 to one and 0 otherwise. This results in an image with a black background, red obstacle objects, a yellow goal object and a white robot arm. The policy outputs a 4 dimensional control signals, or delta state positions, for a given input image or state. The network also has the ability to control the turn-table motor controlling the rotation of the disc workspace and the single degree of freedom of the robot gripper. However, we do not use these controls in experiments and in the training data set those controls to 0.

To determine the network architecture we performed a search over 30 different architectures trained after 400 iterations with a batch size of 200. The set of architectures consisted of different network architectures in terms of convolutional layers, filter size, fully connected layers and output dimension of each layer. In determining an architecture, we trained all networks on a dataset of 6K state/control pairs from the MPIO Supervisor.

The found network architecture was as follows. First, to work with image data, it uses a convolutional layer with 5 channels and 11x11 filters. We added a Rectified Linear Unit (i.e. $\max(0, x)$) or ReLu, to add a non-linearity between layers. ReLus have been shown to allow for faster training than other non-linear separators [20]. After the non-linearity, we added a fully connected layer, or a weight matrix, with an output dimension of 128. Then, another ReLu followed by a weight matrix that maps to the 4 dimensional output. We place a final tanh function on the output, which enforces the output to be between -1 and 1.

We trained all networks on a Nvidia Tesla K40 GPU, which is able to train each network in an average of 10 minutes. All networks were trained using Stochastic Gradient Descent with Momentum in TensorFlow [1]. Momentum is a method to control the magnitude of the gradient based on how large previous gradients were. We used a gradient step size, or learning rate, of 0.01 and a momentum term of 0.9. We initialize all weights with zero mean Gaussian noise with a variance of 0.1.

IV. Online Learning From Demonstrations

Given a collection of increasingly skilled supervisors $S_1, \ldots, S_M$, the goal of this work is to learn a policy that closely matches that of the most skilled supervisor S_M while minimizing the overall number of queries to the more skilled supervisors.

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.

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 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 on 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 density of states visited at time t if the robot follows the policy π_θ from time 0 to time $t-1$. Following [28], 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 $\mathbf{x}_{t+1}$ 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.

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 $r : \mathcal{X} \times \mathcal{U} \rightarrow \mathbb{R}$ 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 [23].

Since grasp success is typically considered as a binary reward which is observed only at a delayed final state [19], grasping in clutter poses a challenging problem for traditional reinforcement learning methods. Hence, we instead build upon DAgger which queries a supervisor for appropriate 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 [28], $l : \mathcal{U} \times \mathcal{U} \rightarrow \mathbb{R}$, 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, 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 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 (with respect to cost) 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 a motion planning algorithm that ignores ambient obstacles, S_2 a supervisor consisting of a crowd-sourced Amazon Mechanical Turk laborers and S_3 a human expert supervisor.

V. Approach and Background

A. Details on DAgger: Dataset Aggregation

Since the cumulative expected reward of a policy is difficult to optimize directly, DAgger [28] 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) by executing the policy induced by the current θ, and asking 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, which we approximately solve with a deep neural network. Other approaches such as kernel based regression can also be considered [31].

2) Step 2: The second step at iteration k, DAgger rolls out the current policy, 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 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 $\mathcal{D}_{k+1}$:

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

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

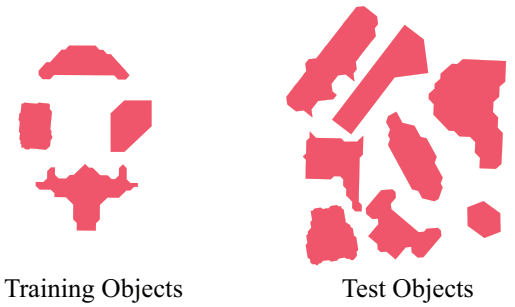
Formulating a framework for supervisor strategy selection poses a challenging problem since no formal model for the effect of supervisor selection on the surrogate loss minimization is available a priori. Additionally, the neural network policy parameters θ are changing at each step of DAgger's iterative training process.

One could consider learning a policy using model-free reinforcement learning algorithms [33], where a selection strategy will query different supervisors and learn over time the best supervisor to select given the current policy. However, this approach requires a substantial number of queries for supervision of the costliest supervisor at each iteration. In light of this, we propose a greedy allocation strategy with respect the cost of the supervisor.

We train a policy with the cheapest supervisor S_1 for a fixed number of K_1 iterations with DAgger and then advance linearly through the hierarchy $S_1, \ldots, S_M$, where the final Neural Network Parameters θ_i trained using S_i are transferred to supervisor S_{i+1} before further training the policy with DAgger. We initialize each DAgger iteration with the parameters from the previous iteration as well. Each supervisor is trained for $K_i, i \in \{1, \ldots, M\}$ iterations in turn.

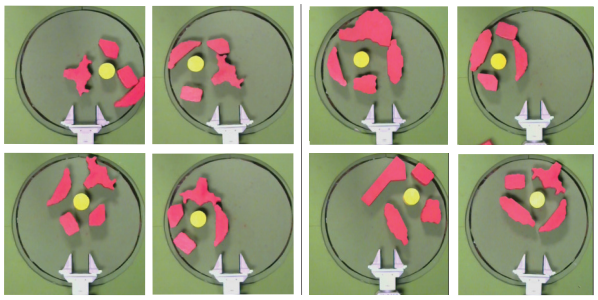
While for large numbers of iterations, convergence tests can be applied to determine a point of diminishing return at which a switch to the next supervisor in the hierarchy can be

[1]In the original DAgger the policy rollout 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 and in practice, it was recommended to set $\beta = 0$ to avoid biasing the sampling [14], [28]



Training Objects Test Objects

Fig. 4: The Training objects on the right are the four objects used in training. The test objects on the left represent objects that were added in our test configurations. Every test configuration contained at least one of the objects on the right.



Training Configurations Test Configurations

Fig. 5: Right: examples of training configurations we presented to the robot. The shapes were arranged around the goal object in different poses and in different parts of the workspace. At test time, we considered similar configurations but added at least 1 unknown object not present in the training set.

initiated, this approach can be challenging when only a limited numbers of roll-outs can be afforded. In practice, we either advanced when the surrogate loss between supervisors was sufficiently low or when the performance on the grasping in clutter task was not improving empirically. For crowdsourcing in particular, we chose the second option since, due to the large variance between demonstrations, we were not able to determine a fixed supervisor switching threshold for the surrogate loss. We hope to significantly scale the number of training steps in future work to investigate various switching criteria in full detail.

VI. Experiments

A. Experimental Setup

For the grasping in clutter task, we consider objects made of Medium Density Fiberboard with an average 4" diameter and 3" in height. The objects used to form clutter are red extruded polygons while the goal object is a yellow cylinder. The objects are light weight and have similar friction coefficients. The robot workspace, which is green, consists of a 30" disk region which is surrounded by an elevated plate constraining the objects to a ring and prohibiting them from leaving the camera's field of view.

To test the performance of each robot policy, we created a test set composed of 20 different configurations each containing objects that were not in the training set. The training objects and test objects introduced during testing are shown in Fig. 4. The test set configurations varied by placing test objects from Fig. 4 with objects we trained on in similar configurations. Examples of training and test configurations are shown in Fig. 5. During training and testing, we manually place the objects in configurations. However, a hardware mechanism that moves the objects around, such as a pre-defined sweeping motion from the robot arm, could be used in future work. We measure success by whether the object was in the gripper at the final timestep. We define reliability or ρ as the percentage of successful configurations.

Per iteration of Hierarchical DAgger, we collect 20 trajectories, each with $T = 100$ timesteps, on different training configurations and then have a supervisor provide 20 demonstrations. We do this to sample a variety of configurations with the current policy before updating the weights. Thus 100 robotic trials correspond to 5 iterations of DAgger.

B. Three Types of Supervisors

We first compare each supervisors' performance. We train the robot with MPIO Supervisor, Crowd-Sourced Supervisor or the Expert Supervisor for a fixed amount of 160 demonstrations on the training configurations. Our results in Fig. 6, show that for our fixed budget of 160 trials the supervisors receive a reliability of $\rho = 30\%$, $\rho = 35\%$ and $\rho = 55\%$, respectively.

When testing the Crowd-Sourced Supervisor, we hired 44 AMT workers. We found only 21 AMT workers were able to pass the quality check and continue providing demonstrations as described in Sec. III. On average a worker would provide 8 demonstrations before choosing to not provide any more. A single demonstration would take 50s to provide.

We are next interested in testing how the supervisors perform in the hierarchy. We divide the original budget of 160 demonstrations to 100 MPIO demonstrations and 60 demonstrations of a higher skill supervisor (i.e. Crowd-Sourced or Expert). The choice of 100 demonstrations was made due to no improvement in reward from the MPIO Supervisor after 100 demonstrations.

Results shown in Fig. 6, suggest that a hierarchy of supervisor with MPIO and Crowd-Sourced achieves a reliability of $\rho = 45\%$ and MPIO and Expert achieves a reliability of $\rho = 60\%$. This suggests that by leveraging a hierachy of supervisors one is able to reduce the burden on a more skilled supervisor in the hierarchy.

It is also interesting that the hierarchy of supervisor achieves higher reliability with the same fixed budget 10% higher for MPIO plus Crowdsourced and 5% for MPIO and Expert. This could suggest that it is easier for a robot to learn core behavior, such as visual servoing in the direction of the goal object, from a more reliable supervisor and infrequent complex behaviors, such as pushing objects in the environment, from a more skilled but noisy human supervisor.

C. Full Hierarchy

Given a fixed budget of 160 Expert Supervisor demonstrations, which achieves 55% reliability, we are interested in increasing relaibility by leveraging the less costly supervisors, MPIO and Crowd-Sourced. We thus had MPIO and Crowd-Sourced supervisor provide demonstrations until we observed the reward achieved was not increasing. This results in the MPIO Supervisors providing 100 demonstrations and the Crowd-Sourced Supervisor providing 120 demonstrations.

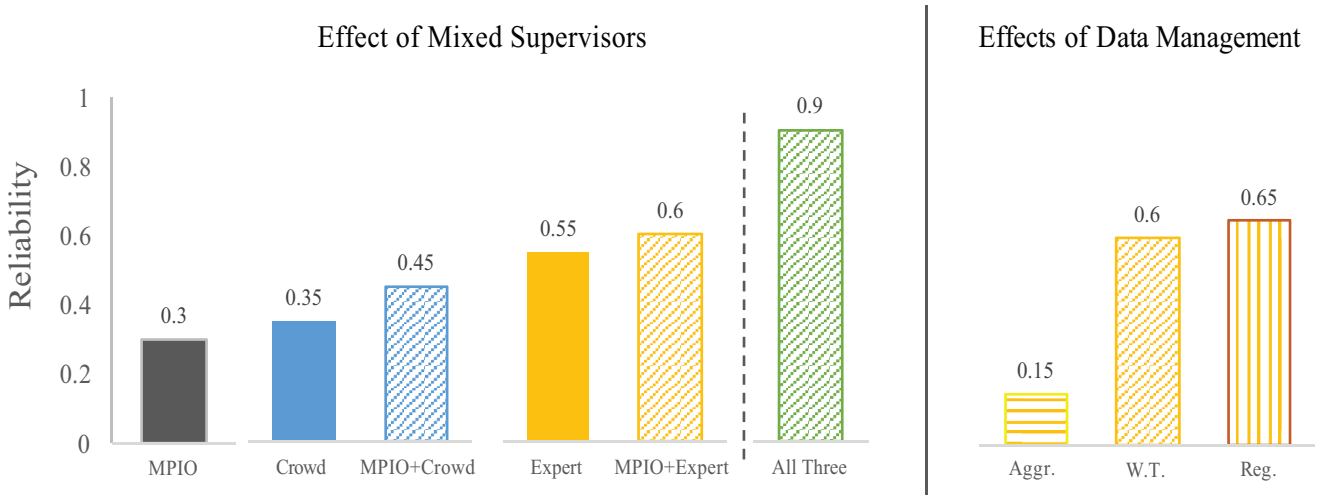


Fig. 6: The reliability or ρ, which is a measure of performance on previously unseen objects, of each policy trained with a supervisor is reported. The bar graphs show the number of successful trials. From Left to Right, we show results for MPIO ($\rho = 30\%$) by itself, Crowd-Sourced ($\rho = 35\%$) and MPIO plus Crowd-Sourced ($\rho = 45\%$), Expert ($\rho = 55\%$) and MPIO plus Expert ($\rho = 60\%$). All policies are trained on 160 demonstrations. The full hierarchy of all three supervisors has a fix budget of 160 Expert demonstrations, but first leverages MPIO and Crowd-Sourced Supervisors until no improvement in reward from their demonstrations occurred and is able to achieve a reliability of $\rho = 90\%$. We also demonstrate three different ways to advance in the hierarchy using 100 demonstrations from MPIO and 60 from Expert. From Left to Right, training on the aggregate dataset of both supervisors ($\rho = 0.15$), weight transfer of the network weights between supervisors ($\rho = 0.6$) and a regularization of the gradient step when training with the Expert ($\rho = 0.65$).

We then had the Expert Supervisor provide a fixed budget of 160 demonstrations. Thus, resulting in a total of 380 demonstrations. The resulting policy had a reliability of 90% as shown in Fig. 6.

Investigating the learned policy, we observed that it appears to find gaps in between the clutter objects. The robot then sweeps left to right until the objects are pushed apart and then moves towards the goal object, as illustrated in Fig. 1. We noticed that the magnitude of the sweeping motion appeared anti-proportional to the size of the gap in between the cluttered objects. For example, two cluttered objects placed close together in front of the goal object would cause a high frequency of sweeping. While, a single object in front of the goal object would often result in a single sweeping motion pushing the obstacle away from the goal object. We also noticed that the robot would in general stop once it reached the goal object. However, it was rare to see the robot go backwards and correct itself, despite those demonstrations being given.

While our policy showed improved performance when trained using a hierarchy, failure modes persisted in our test configurations. Common failure modes are either that the robot sweeps too far and jams the objects against the inscribed circle around the workspace or a smaller obstacle object was caught in the gripper while being pushed.

D. Advancing in the Hierarchy

We lastly test how to advance between supervisors in the hierarchy. We examine the situation where the robot is first trained with the MPIO supervisor for 100 demonstrations and then the Expert Supervisor for 60 demonstrations.

We are interested in testing the following strategies for changing supervisors 1) aggregating the two datasets of the demonstrations collected with both MPIO and Expert Supervisors 2) transferring only the weights of the neural network policy θ to initialize training with the Expert and not retraining on the MPIO Supervisor's demonstrations 3) transferring only the weights of the neural network policy but adding the values output by $\pi_\theta(\mathbf{x})$ instead of $\tilde{\pi}_m(\mathbf{x})$ in the states visited when the Expert Supervisor's demonstrations are similar to the output of the policy as measured by $||\tilde{\pi}_m(\mathbf{x}) - \pi_\theta(\mathbf{x})||_2^2 < 0.01$. This acts as a form of regularization on the optimization since part of the stochastic gradient computed on the mini-batch would be zero for examples where the policy trained with the MPIO Supervisor matches that of the Expert.

Our results reported in Fig.6, show that the aggregation strategy achieves reliability, $\rho = 15\%$, the weight transfer strategy achieves $\rho = 60\%$ and the regularized stochastic gradient update strategy achieves $\rho = 65\%$. Thus, suggesting that techniques to help the policy remain close to the previously trained supervisor are useful for effectively switching between supervisors. In future work, we will look at other ways to better transfer the learned information from supervisors lower in the hierarchy.

VII. Discussions and Future Work

Robot grasping in clutter is a challenging problem for automation in warehouse order assembly and home cleaning due to the uncertainty arising from sensing and control and the inherent complexity of push mechanics. We introduce a version of the problem where a yellow cylinder must be grasped by a planar robot arm amid extruded objects with a variety of shapes and positions. Results suggest that this task is amenable to a Learning from Demonstrations approach such as DAgger or SHIV but can put a burden on human experts.

To reduce the burden, we consider two alternative supervisors: one is a motion planner that ignores obstacles (MPIO) and the second is crowd-sourced workers from Amazon Mechanical Turk who provide demonstrations using a novel overlay interface. Results show, as expected, that the reliability of the policies depend on the skill level of the supervisors. We introduce a hierarchical approach where supervisors with differing skill levels are combined. Results suggest that staging supervisors in terms of increasing skill

level can bootstrap LfD to learn reliable policies; in this case the robot learns a counter-intuitive policy where it sweeps the gripper back and forth to clear obstacles before grasping.

To our knowledge this is the first study of hierarchical supervisors for Learning from Demonstrations. In future work, we will perform more experiments in this context with added noise, different ratios in supervisors, and study the sample complexity needed to learn a policy. We will also study the comparative performance of other motion planning-based supervisors and study how hierarchical approaches can be applied to tasks such as assembly. Further information, videos and code can be found at the following `http://berkeleyautomation.github.io/HierSupCASE/`.

VIII. Acknowledgments

This research was performed in UC Berkeley's AutoLab 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 Steve McKinely and Meng Guo for help in building the hardware components.

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," in *ICRA*. Citeseer, 2000, pp. 348–353.

[7] G. Bradski, *Dr. Dobb's Journal of Software Tools*, 2000.

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

[9] A. Cosgun, T. Hermans, V. Emeli, and M. Stilman, "Push planning for object placement on cluttered table surfaces," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 4627–4632.

[10] M. Dogar, K. Hsiao, M. Ciocarlie, and S. Srinivasa, "Physics-based grasp planning through clutter," *Robotics: Science and systems VIII*, 2012.

[11] M. Dogar and S. Srinivasa, "A framework for push-grasping in clutter," *Robotics: Science and systems VII*, 2011.

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

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

[14] 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.

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

[16] 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.

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

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

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

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[21] M. Laskey, S. Staszak, W. Y.-S. Hsieh, J. Mahler, F. T. Pokorny, A. D. Dragan, and K. Goldberg, "Shiv: Reducing supervisor burden in dagger using support vectors for efficient learning from demonstrations in high dimensional state spaces," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 462–469.

[22] A. E. Leeper, K. Hsiao, M. Ciocarlie, L. Takayama, and D. Gossow, "Strategies for human-in-the-loop robotic grasping," in *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*. ACM, 2012, pp. 1–8.

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

[24] M. T. Mason, "Mechanics and planning of manipulator pushing operations," *The International Journal of Robotics Research*, vol. 5, no. 3, pp. 53–71, 1986.

[25] M. Nieuwenhuisen, D. Droeschel, D. Holz, J. Stuckler, A. Berner, J. Li, R. Klein, and S. Behnke, "Mobile bin picking with an anthropomorphic service robot," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 2327–2334.

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

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

[28] 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.

[29] 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.

[30] T. D. Sanger, "Neural network learning control of robot manipulators using gradually increasing task difficulty," *Robotics and Automation, IEEE Transactions on*, vol. 10, no. 3, pp. 323–333, 1994.

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

[32] M. Stilman, J.-U. Schamburek, J. Kuffner, and T. Asfour, "Manipulation planning among movable obstacles," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 3327–3332.

[33] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 1998.

[34] 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.

[35] J. Van Den Berg, M. Stilman, J. Kuffner, M. Lin, and D. Manocha, "Path planning among movable obstacles: a probabilistically complete approach," in *Algorithmic Foundation of Robotics VIII*. Springer, 2009, pp. 599–614.