Learning Task-Oriented Grasping for Tool Manipulation from Simulated Self-Supervision

Kuan Fang Yuke Zhu Animesh Garg Andrey Kurenkov Viraj Mehta Li Fei-Fei Silvio Savarese
Stanford University, Stanford, CA 94305 USA

***Abstract*—Tool manipulation is vital for facilitating robots to complete challenging task goals. It requires reasoning about the desired effect of the task and thus properly grasping and manipulating the tool to achieve the task. Task-agnostic grasping optimizes for grasp robustness while ignoring crucial task-specific constraints. In this paper, we propose the Task-Oriented Grasping Network (TOG-Net) to jointly optimize both task-oriented grasping of a tool and the manipulation policy for that tool. The training process of the model is based on large-scale simulated self-supervision with procedurally generated tool objects. We perform both simulated and real-world experiments on two tool-based manipulation tasks: sweeping and hammering. Our model achieves overall 71.1% task success rate for sweeping and 80.0% task success rate for hammering. Supplementary material is available at: `bit.ly/task-oriented-grasp`.**

I. Introduction

Tool manipulation can be defined as the employment of a manipulable object, i.e. a tool, to fulfill a task goal. For this purpose, the agent needs to effectively orient and then manipulate the tool so as to achieve the desired effect. According to Brown et al. [4], there are four key aspects to learning task-oriented tool usage: (a) understanding the desired effect, (b) identifying properties of an object that make it a suitable tool, (c) determining the correct orientation of the tool prior to usage, and (d) manipulating the tool. A task-oriented grasp is therefore a grasp that makes it possible to correctly orient the tool and then manipulate it to complete the task.

Consider a hammer object as shown in Figure 1. The best grasp predicted by a *task-agnostic* task-agnostic grasp prediction model, such as Dex-Net [35], is likely to reside close to the center of mass to optimize for robustness. However, the hammering task can be best achieved by holding the hammer at the far end of the handle, thus to generate a high moment at the point of impact on the head. And yet when the same object is used for the sweeping task, it should be grasped by the head since that spares the largest contact surface area with the target objects. In order to optimize for the task success, both grasping robustness and suitability for the manipulation should be considered.

The problem of understanding and using tools has been studied in robotics, computer vision, and psychology [1, 14, 16, 20, 41, 55]. Various studies in robotics have primarily focused on reasoning about the geometric properties of tools [7, 47]. They often assume prior knowledge of object geometry and require predefined affordance labels and semantic constraints, which has constrained their usefulness in realistic environments with sensory and control uncertainty. Some pioneering works have also grounded the tool grasping problem in an interactive environment [37, 38]. Nonetheless, their approaches relied on hand-engineered feature representations and simplified action spaces, which are limited to the tasks they are tuned to work for and allow only limited generalization to novel objects in complex manipulation tasks.

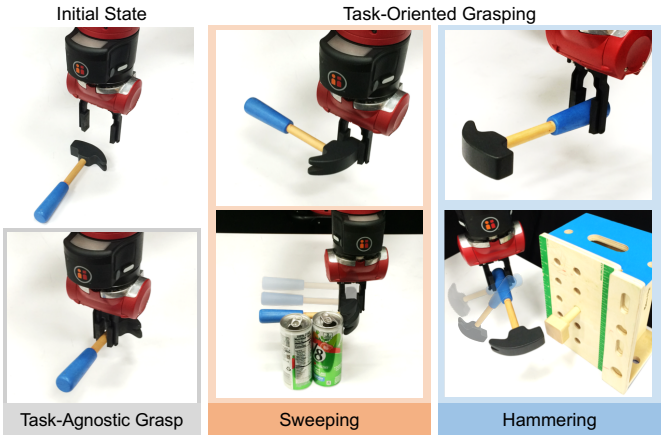

Figure 1: The same object can be grasped by the robot in different ways from the initial state on the table. A task-agnostic grasp can lift up the hammer but it might not be suitable for specific manipulation tasks such as sweeping or hammering. We aim to directly optimize for task success in each episode, by jointly choosing a task-oriented grasp and the subsequent manipulation actions.

In this work, we focus on tool manipulation tasks that consist of two stages. First, the robot picks up a tool resting on the tabletop. Second, it manipulates this tool to complete a task. We propose the Task-Oriented Grasping Network (TOG-Net), a learning-based model for jointly predicting task-oriented grasps and subsequent manipulation actions given the visual inputs. To accommodate the need for large amounts of training data for deep learning, we embrace the self-supervised learning paradigm [31, 38, 42], where the robot performs grasping and manipulation attempts and the training labels automatically generated thereby. To scale up our self-supervision data collection, we leverage a real-time physics simulator [6] that allows a simulated robot to perform task executions with diverse procedurally generated 3D objects. We evaluate our method on a hammering task and a sweeping task. Our model is proved to learn robust policies that generalize well to novel objects both in simulation and the real world. In the real-world experiments, our model achieves 71.1% task success rate for sweeping and 80.0% task success rate for hammering using 9 unseen objects as tools.

Our primary contributions are three-fold: 1) We propose a learning-based model for jointly learning task-oriented grasping and tool manipulation that directly optimizes task success; 2) To train this model, we develop a mechanism for generating large-scale simulated self-supervision with a large repository of procedurally generated 3D objects; 3) We demonstrate that our model can generalize to using novel objects as tools in both simulation and the real world.

II. Related Work

Task-Agnostic Grasping: Robotic grasping is a long-standing challenge that involves perception and control. Classical methods approach the grasping problem from a purely geometric perspective. They optimize grasp quality measures based on analytic models of geometric constraints, such as force closure and form closure [12, 44, 52]. While grasp quality measures offer a principled mathematical framework of grasp analysis, their practical usage is limited by the large space of possible grasps. Several techniques have been proposed to restrict the search space of grasp candidates. This includes representing objects with shape primitives [40], simplifying the search of grasps in a subspace of reduced dimensionality [5], and leveraging a dataset of objects with known grasps to speed up grasp selection for novel objects [15, 34]. Another limitation of these approaches is that they require the full 3D geometry of the object, which restricts their usage in unstructured real-world environments. The recent development of machine learning techniques, especially deep learning, has enabled a surge of research that applies data-driven methods to robotic grasping [2, 25]. These learning methods largely fall into two families depending on the source of supervision: 1) supervised learning approaches [30, 35, 46], where the models are trained with a dataset of objects with ground-truth grasp annotations, and 2) self-supervision approaches [23, 31, 42, 3, 11], where grasp labels are automatically generated by a robot's trial and error on large numbers of real-world or simulated grasp attempts. To address the data-hungry nature of deep neural networks, several works [35, 51] relied on depth sensors to train their models in simulation and transfer to the real robot.

Task-Oriented Grasping: A major portion of research in grasping aims at holding the object in the robot gripper so as to not drop it despite external wrenches. In practice, however, the end goal of grasping is often to manipulate an object to fulfill a goal-directed task once it has been grasped. When the grasping problem is contextualized in manipulation tasks, a grasp planner that solely satisfies the stability constraints is no longer sufficient to satisfy the task-specific requirements. In classical grasping literature, researchers have developed task-oriented grasp quality measures using task wrench space [18, 32, 43].

Data-driven approaches have also been used to learn task-related constraints for grasp planning [7, 47]. These studies incorporate semantic constraints, which specify which object regions to hold or avoid, based on a small dataset of grasp examples. However, these grasping methods cannot entail the success of the downstream manipulation tasks, and the hand-labeled semantic constraints cannot generalize across a large variety of objects. On the contrary, our work jointly learns the task-aware grasping model and the manipulation policy given a grasp. Thus, our grasping model is directly optimized to fulfill its downstream manipulation tasks. Furthermore, we employ deep neural networks to train our task-aware grasping models on a large repository of 3D objects, enabling it to generalize from this repository of objects to unseen objects as well as from simulation to the real world.

Affordance Learning: Another line of related work centers around understanding the affordances of objects [8, 28, 55, 56]. The notion of affordances introduced by Gibson [14] characterizes the functional properties of objects and has been widely used in the robotics community as a framework of reasoning about objects [26, 50]. Prior art has developed methods to learn different forms of object affordance such as semantic labels [56], spatial maps [24], and motion trajectories [55]. Our work follows a progression of previous work on behavior-grounded affordance learning [13, 22, 37, 38, 48], where the robot learns object affordance by observing the effects of actions performed on the objects. Nonetheless, we do not explicitly supervise our model to learn and represent goal-directed object affordances. Instead, we demonstrate that our model's understanding of object affordance naturally emerges from training grasping and manipulation simultaneously. Recent work by Mar et al. [38] has the closest resemblance to our problem setup; however, their action space consists of a small set of discrete actions, while we employ a multi-dimensional continuous action space. Aside from the problem, their method of self-organizing maps uses hand-designed tool pose and affordance descriptors, while we eschew feature engineering in favor of end-to-end deep learning.

III. Problem Statement

Our goal is to control a robot arm to perform tool-based manipulation tasks using novel objects. Each task is a two-stage process. In the first stage, the robot grasps an object as a tool for a task. In the second stage, the robot manipulates the tool to interact with the environment to complete the goal of the task. The visual appearance of the tool is provided for the robot to accomplish this.

Notations of Grasping: The robot operates in a workspace based on camera observations, where $\mathcal{O}$ denotes the observation space. We consider the grasping problem in the 3D space, where $\mathcal{G}$ denotes the space of possible grasps. Given a pair of observation $\mathbf{o} \in \mathcal{O}$ and grasp $\mathbf{g} \in \mathcal{G}$, let $S_G(\mathbf{o},\mathbf{g}) \in \{0,1\}$ denote a binary-valued grasp success metric, where $S_G = 1$ indicates that the grasp is successful according to the predefined metric. In practice, the underlying sensing and motor noise introduce uncertainty to the execution of a grasp. We measure the robustness of a grasp $Q_G(\mathbf{o},\mathbf{g})$ by the probability of grasp success under uncertainty, where $Q_G(\mathbf{o},\mathbf{g}) = \Pr(S_G = 1|\mathbf{o},\mathbf{g})$. This grasp metric S_G is *task-agnostic*, which evaluates the quality of a grasp without grounding to a specific task. As we noted in Section II, data-driven grasping methods [35, 51] have focused on optimizing *task-agnostic* grasps.

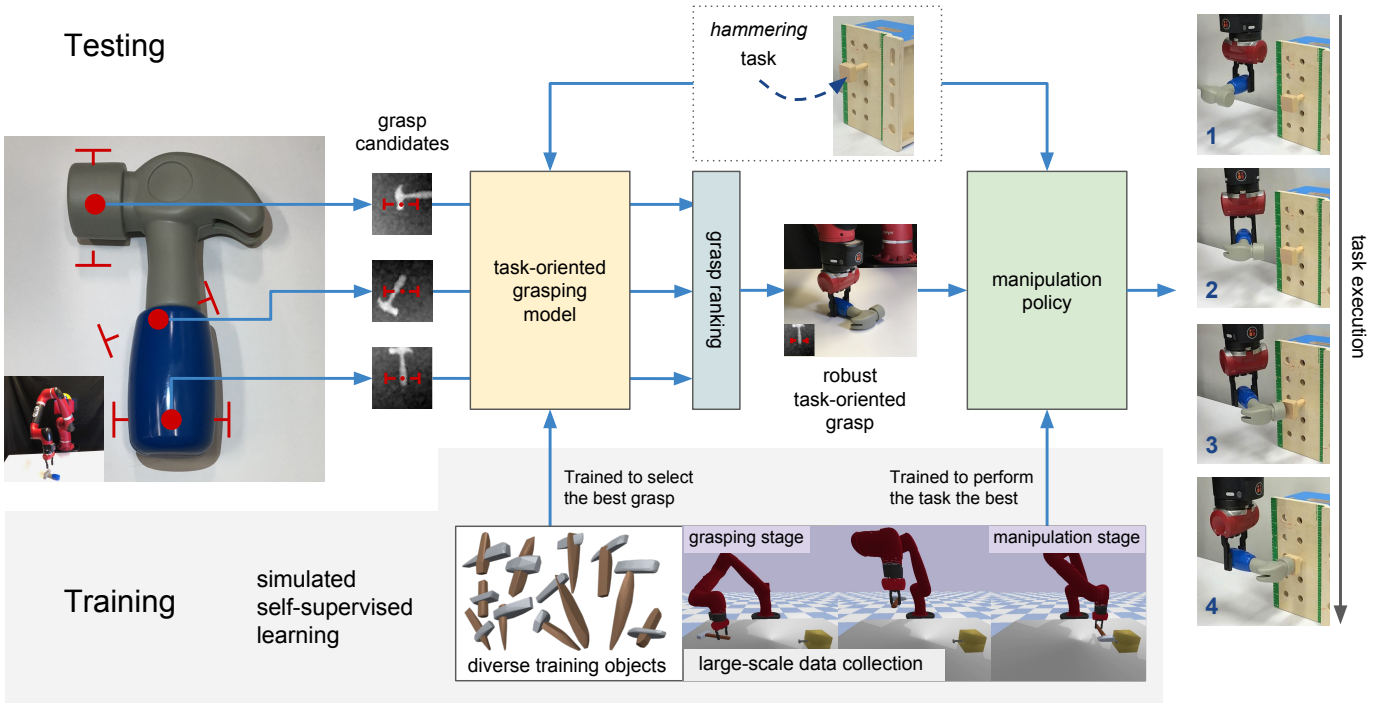

Figure 2: **Model Overview.** Our model consists of a task-oriented grasping model and a manipulation policy. Given the visual inputs of the object, we sample multiple grasp candidates. The task-oriented grasping model computes a grasp quality score for each candidate based on the planned task, and chooses the grasp with the highest score. Given the observation of the scene, the manipulation policy outputs actions conditioned on the selected grasp. The two modules are trained jointly using simulated self-supervision.

Problem Setup: By contrast, we contextualize the grasping problem in tool manipulation tasks. In our setup, the grasping stage is followed by a manipulation stage, where a policy π produces actions to interact with the environment once the object is grasped. Intuitively, both the choice of grasps and the manipulation policy play an integral role in the success rate of a task.

Let $S_T(\mathbf{o},\mathbf{g}) \in \{0,1\}$ denote a binary-valued task-specific success metric, where $S_T = 1$ indicates that the task T is successfully done based on the goal specification. Clearly the grasp success is the premise of the task success, i.e., $S_T = 1$ entails $S_G = 1$. Given a manipulation policy π for the task, we measure the robustness Q_T^π of a *task-oriented* grasp by the probability of task success under policy π, where $Q_T^\pi(\mathbf{o},\mathbf{g}) = \Pr(S_T = 1|\mathbf{o},\mathbf{g})$. Thereafter, the overall learning objective is to train both policies simultaneously such that:

$$\mathbf{g}^*, \pi^* = \arg\max_{\mathbf{g},\pi} Q_T^\pi(\mathbf{o},\mathbf{g}). \quad (1)$$

We aim at selecting the optimal grasp $\mathbf{g}^*$ that is most likely to lead to to the completion of the task, and at the same time finding the best policy π^* to perform the task conditioned on a grasp. In practice, we implement both the grasping policy and the manipulation policy using deep neural networks. We detail the design of the neural network models and their training procedures in Sec. IV.

Assumptions: We consider the problem of task-oriented grasp planning with a parallel-jaw gripper based on point clouds from a depth camera. The training of our model uses simulated data generated from a real-time physics simulator [6]. Our design decision is inspired by the effective use of depth cameras in transferring the grasping model and manipulation policy trained in simulation to reality [35, 51]. Further, to reduce the search space of grasping candidates, we restrict the pose of the gripper to be perpendicular to the table plane. In this case, each grasp $\mathbf{g} = (g_x, g_y, g_z, g_\phi)$ has 4 degrees of freedom, where $(g_x, g_y, g_z) \in \mathbb{R}^3$ denotes the position of the gripper center, and $g_\phi \in [0, 2\pi)$ denotes the rotation of the gripper in the table plane. Each observation $\mathbf{o} \in \mathbb{R}_+^{H\times W}$ is represented as a depth image from a fixed overhead RGB-D camera with known intrinsics.

IV. Task-Oriented Grasping for Tool Manipulation

As shown in Fig 2, our task-oriented grasping network (TOG-Net) consists of a task-oriented grasping model and a manipulation policy. The two modules are coupled and learn to achieve task success together. In this section, we first present the design and implementation of the two modules, and then describe how they are jointly trained using simulated self-supervision.

A. Learning Task-Oriented Grasp Prediction

High-quality task-oriented grasps should simultaneously satisfy two types of constraints. First, the tool must be stably held in the robot gripper, which is the goal of task-agnostic grasping. Second, the grasp must satisfy a set of physical and semantic constraints that are specific to each task.

For a given observation $\mathbf{o} \in \mathcal{O}$, a small subset of grasps $\mathcal{G}_\alpha \subseteq \mathcal{G}$ can robustly lift up the object with a grasp quality higher than α, i.e. $Q_G(\mathbf{o},\mathbf{g}) \geq \alpha$. Hence, the task agnostic grasp

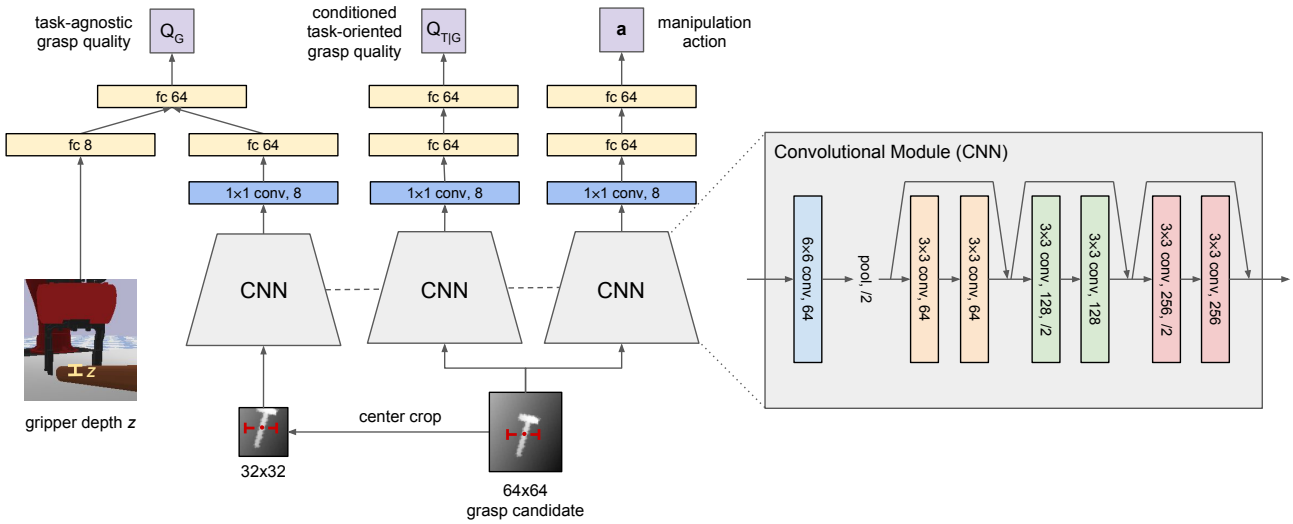

Figure 3: **Task-Oriented Grasping Network (TOG-Net).** The inputs to the network are two depth image crops and the sampled gripper depth z. The network predicts task-agnostic grasp quality, conditioned task-oriented grasp quality, and manipulation actions. The CNN modules share parameters as denoted by the dashed lines. Residual network layers and batch normalization are used in the CNN modules.

prediction problem involves finding the corresponding grasp $\mathbf{g}$ that maximizes the grasp quality $Q_G(\mathbf{o},\mathbf{g}) = \Pr(S_G = 1|\mathbf{o},\mathbf{g})$. This prediction problem can be solved with a variety of methods, and we build upon the approach of Mahler et al. [35] that uses quality function approximation with algorithmic supervision via analytic models.

As noted in Section III, the overall objective is maximizing the probability of task success $Q_T^\pi(\mathbf{o},\mathbf{g})$ under a policy π, grasp $\mathbf{g} \in \mathcal{G}$. However, directly solving this problem results in a discrete space search over a large space of grasps, and then a subsequent optimization problem to solve for a manipulation policy given each grasp. Furthermore, we note that tool manipulation task execution only succeeds if the grasp has succeeded. However, not all grasps $\mathbf{g} \in \mathcal{G}_\alpha$ result in a successful task. Specifically, we can define a conditional robustness metric $Q_{T|G}$ that measures the probability of task success (under policy π) conditioned on a successful grasp, where $Q_{T|G}(\mathbf{o},\mathbf{g}) = \Pr_\pi(S_T = 1|S_G = 1,\mathbf{o},\mathbf{g})$. Then, task-oriented grasps form a subset of task-agnostic grasps: $\mathcal{G}_{\alpha,\delta} \subseteq \mathcal{G}_\alpha$, i.e. the grasps which lead to task success with a task quality conditioned on the grasp $Q_{T|G}(\mathbf{o},\mathbf{g}) \geq \delta$ where δ is a chosen threshold.

This formulation lets us to effectively decouple the two problems as: 1) finding robust task-agnostic grasps; and 2) finding a robust task-oriented grasp among robust grasps and the corresponding manipulation policy. The key observation is that the task quality metric $Q_T^\pi(\mathbf{o},\mathbf{g})$ can be factorized into independently computable terms: $Q_{T|G}(\mathbf{o},\mathbf{g})$ and $Q_G(\mathbf{o},\mathbf{g})$. Formally, the task robustness $Q_T(\mathbf{o},\mathbf{g})$ can be decomposed as follow:

$$\begin{aligned} Q_T^\pi(\mathbf{o},\mathbf{g}) &= \Pr_\pi(S_T = 1|\mathbf{o},\mathbf{g}) \\ &= \Pr_\pi(S_T = 1, S_G = 1|\mathbf{o},\mathbf{g}) \\ &= \Pr_\pi(S_T = 1|S_G = 1,\mathbf{o},\mathbf{g}) \cdot \Pr(S_G = 1|\mathbf{o},\mathbf{g}) \\ &= Q_{T|G}(\mathbf{o},\mathbf{g}) \cdot Q_G(\mathbf{o},\mathbf{g}). \end{aligned}$$

Our model learns to approximate the values of grasp quality $Q_G(\mathbf{o},\mathbf{g})$ and task quality conditioned on grasp $Q_{T|G}(\mathbf{o},\mathbf{g})$ using deep neural networks given object $\mathbf{o}$ and grasp $\mathbf{g}$ as inputs. We denote the predicted values as $\hat{Q}_G(\mathbf{o},\mathbf{g};\theta_1)$ and $\hat{Q}_{T|G}(\mathbf{o},\mathbf{g};\theta_2)$, where θ_1 and θ_2 represent the neural network parameters.

The pipeline during testing is shown in Figure 2. We first sample 200 antipodal grasp candidates based on depth gradients [35]. Then $\hat{Q}_T(\mathbf{o},\mathbf{g};\theta_1,\theta_2) = \hat{Q}_{T|G}(\mathbf{o},\mathbf{g};\theta_2) \cdot \hat{Q}_G(\mathbf{o},\mathbf{g};\theta_1)$ is computed for each grasp candidate. We run the cross-entropy method [45] for 3 iterations as in [35], in order to rank and choose the task-oriented grasp corresponds to the highest $\hat{Q}_T$.

B. Learning the Manipulation Policy

To complete the task with different tools and different grasps, the manipulation policy needs to be conditioned on $\mathbf{o}$ and $\mathbf{g}$. The manipulation policy can be either an external motion planner or a learned policy. While our pipeline is not limited to a specific action space, here we choose to use parameterized motion primitives parallel to the planar table surface to control the robot in an open-loop manner. After the motion primitive is chosen based on the task environment, our manipulation policy predicts the continuous manipulation actions $\mathbf{a} = (a_x, a_y, a_z, a_\phi) \in \mathbb{R}^3$, where (a_x, a_y, a_z) and a_ϕ are the translation and rotation of the motion primitive. We use a Gaussian policy $\pi(\mathbf{a}|\mathbf{o},\mathbf{g};\theta_3) = \mathcal{N}(f(\mathbf{o},\mathbf{g};\theta_3),\Sigma)$, where $f(\mathbf{o},\mathbf{g};\theta_3)$ is a neural network for predicting the mean with parameters θ_3 and the covariance matrix Σ is a constant diagonal matrix.

C. Neural Network Architecture

In Figure 3 we propose a three-stream neural network architecture for jointly predicting $\hat{Q}_G$ and $\hat{Q}_{T|G}$ and $\mathbf{a}$. Following the practice of [35], we convert $\mathbf{o}$ and $\mathbf{g}$ into gripper depth z and image crops as inputs to the neural network. The gripper depth is defined as the distance from the center of the two fingertips to the object surface. The image crops are centered at the grasp center (g_x, g_y, g_z) and aligned with the grasp axis orientation ϕ. [35] uses image crops of size 32×32 to focus on the contact between the gripper and the tool. To achieve

the task success, our model is supposed to reason about the interactions between the tool and the task environment which requires a holistic understanding of the shape of the tool. Thus our model predicts $\hat{Q}_{T|G}$ and $\mathbf{a}$ using larger image crops of size 64×64 which covers most of training and testing objects. Meanwhile the center crop of 32×32 is used to predict $\hat{Q}_G$. Our neural network is composed of three streams which share parameters in their low-level convolutional layers, extracting identical image features, denoted by dotted lines. Building atop the GQCNN in [35], we use residual network layers [19] and batch normalization [21] to facilitate the learning process. On top of the convolutional layers with shared weights, we apply bottleneck layers of 1×1 convolutional filters for each stream to reduce the size of the network.

D. Learning Objectives and Optimization

We jointly train the task-oriented grasping model and the manipulation policy with simulated robot experiments of grasping and manipulation. Each simulated episode in our training dataset contains sampled grasp $\mathbf{g}$, action $\mathbf{a}$ and the resultant grasp success label S_G, task success label S_T. We use cross-entropy loss $\mathcal{L}$ for training the grasp prediction functions $\hat{Q}_G$ and $\hat{Q}_{T|G}$. For training the policy π, we use the policy gradient algorithm with gradients $\nabla \log \pi(\mathbf{a}|\mathbf{o},\mathbf{g};\theta_3)$. We use the task success label as the reward of the manipulation policy. Since we are using a Gaussian policy as described in Sec. IV-B, this is equivalent to minimizing $\frac{1}{2}||f(\mathbf{o},\mathbf{g};\theta_3)-\mathbf{a}||^2_\Sigma \cdot S_G$ with respect to θ_3. Let the parameters of the neural network to be denoted as $\theta = \{\theta_1,\theta_2,\theta_3\}$, we jointly train our model by solving the following optimization problem:

$$\begin{aligned}\theta^* = \arg\min_{\theta} \sum_{i=1}^{N} & \mathcal{L}(S_G, \hat{Q}_G(\mathbf{o},\mathbf{g};\theta_1)) \\ & + \mathbb{1}[S_G = 1] \cdot \mathcal{L}(S_T, \hat{Q}_{T|G}(\mathbf{o},\mathbf{g};\theta_2)) \\ & + \mathbb{1}[S_T = 1] \cdot \frac{1}{2}||f(\mathbf{o},\mathbf{g};\theta_3)-\mathbf{a}||^2_\Sigma.\end{aligned} \tag{2}$$

V. Self-Supervision for Grasping and Manipulation

A. Procedural Generation of Tool Objects

We train our model in simulation with a large repository of 3D models so as to generalize to unseen objects. However, existing 3D model datasets do not contain enough objects suitable for tool manipulation while exhibiting rich variations in terms of their geometric and physical properties. As shown in Figure 4, we leverage a common strategy of procedural generation [3, 49] to produce a large set of diverse and realistic objects that can be used as tools for tasks we are interested in.

While the generation process can be arbitrarily complex, we choose to generate objects composed of two convex parts. The two parts are connected by a fixed joint. We define three type of composed shapes: T-shapes, L-shapes, and X-shapes. For each object, two convex meshes are first sampled. Then the meshes are randomly scaled along the x, y, and z axes. Depending on the type of object shape, the parts are shifted and rotated with respect to each other. We randomly sample physical dynamic properties such as density and friction coefficients.

We use two sets of meshes to generate two different set of objects: primitive and complex. We generate primitive meshes that are composed by a set of parameterized shape primitives including cuboids, cylinders, and polytopes. The dimensions and textures are randomly chosen from predefined ranges. The primitive meshes are generated by OpenScad [53]. We also obtain convex object parts from a variety of realistic 3D object models as the complex meshes. This done by running convex decomposition [36] on each object from [2].

B. Data Generation with Simulated Self-Supervision

In order to collect large-scale datasets for training and evaluating our model, we develop a self-supervision framework to automatically generate training data. We leverage an open-source real-time physics simulator, Bullet [6], which allows a simulated robot to perform trial and error in millions of trails. We record grasp and task success labels in each trial and use them to train our models described in Section IV. For training each task we collect the data in three rounds. After each round, we train the grasping model and the manipulation policy using the collected data to obtain an updated model. In each round we run the simulation for 500K trials.

In the first round, we perform a random policy using a GQCNN model trained on Dex-Net 2.0 Dataset [35]. The original GQCNN model uses a cross entropy method (CEM) [45] to sample robust grasps corresponds to the highest task-agnostic grasp quality scores. But the trained GQCNN usually lead to a collapsed mode of grasps which is most robust according to the ranking of the predicted scores. Ideally we want to collect data of diverse sampled grasps and evaluate how well they can be used in each task. An alternative is to sample uniformly sample grasps with grasp quality scores higher than a threshold. In practice we found such sampling usually clusters on the long edge of a tool object since there are more antipodal grasps possible there. To encourage diverse exploration, we instead use non-maximum suppression (NMS) [17] which is widely used in object detection algorithms. The NMS algorithm goes through the ranked antipodal grasps, and removes grasps which have short Euclidean distances with previous grasps with higher grasp quality scores. This guarantees all remaining grasps are separate from each other and usually produces 10 to 30 distinguished modes. With these sampled grasp, the random policy uniformly samples manipulation action from the action space for each task.

In the following rounds, we use the ε-greedy strategy with the updated grasping model. The grasping model uses the CEM method described in Section IV with probability $1-\varepsilon_1$, and uses the NMS method with GQCNN predictions as described above with ε_1 probability. The manipulation policy predicts and manipulation action parameters in Section IV with probability $1-\varepsilon_2$, and use random actions with probability ε_2. We set 0.2 for both ε_1 and ε_2.

VI. Experiments

The goal of our experimental evaluation to answer following questions: (1) Does our method improve task performance as

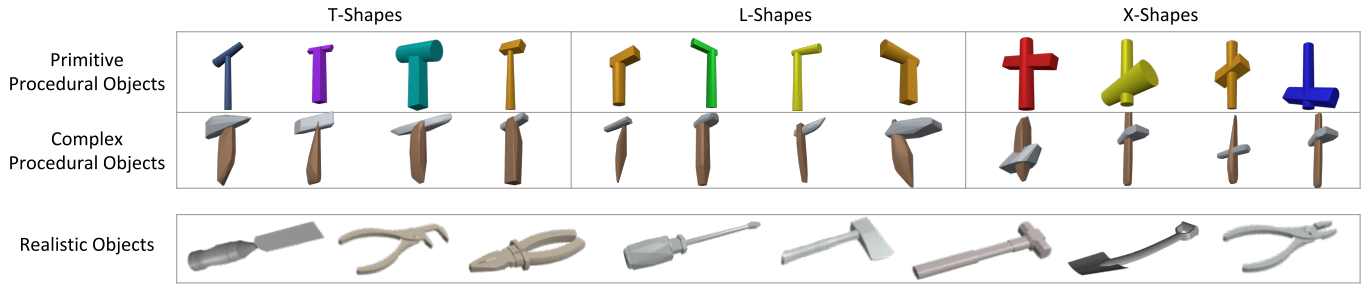

Figure 4: **Example of objects in simulation.** The first two rows show the procedurally generated objects based on shape primitives as well as complex meshes. These objects are generated using three predefined composing rules to result in T-shapes, L-shapes, and X-shapes. The last row shows the realistic shapes from existing 3D model datasets.

compared to baseline methods? (2) Does the joint training qualitatively change the mode of grasping? (3) Can the model be trained with simulated self-supervision work in the real world?

We evaluate our method on two tabletop manipulation tasks: *sweeping* and *hammering*. We define our hammering task as a motion primitive that achieves fitting a peg in a hole with tight tolerance, which is prevalent in assembly tasks as shown in [54, 27]. Sweeping, on the other hand, is a primitive in autonomous manipulation, such as in part positioning and reorientation [33], grasping in clutter [9], object manipulation without lifting [39]. Sweeping with tools has been studied in the context of singulation and part retrieval [10, 29]. Each of these tasks requires grasping objects in specific modes which can often be different from the best stable grasp available, thereby resulting in competing objectives.

We evaluate our model in both simulation and the real world. The basic setup of both tasks includes a 7-DoF Rethink Robotics Sawyer Arm with a parallel jaw gripper, a $48'' \times 30''$ table surface, and an overhead Kinect2 camera. In simulation, the robot and camera are placed according to the real-world camera calibration results, in order to obtain consistent performance. For both experiments, we use models solely trained using simulated data as described in Section IV.

A. Task Design

The table surface is split into two halves: a grasping region and a manipulation region. Before each episode starts, an object is sampled and randomly dropped onto the grasping region to be used as the tool. A depth image is taken from the overhead Kinect2 camera as the input of the model. The model then predicts the 4-DOF grasp and the parameters of the motion primitive. The robot grasps the object from the grasping region and performs the task in the manipulation region. In our task design, the motion primitive is a predefined single step action. Our model predicts the starting gripper pose relative to a reference point.

Sweeping: Target objects are randomly placed in the manipulation region as the target objects. In the real world we use two soda cans as the target objects, and in simulation we randomly place one or two 3D models of cans. The task goal is to sweep all target objects off the table using the tool. The motion primitive of sweeping is a straight line trajectory of the gripper parallel to the table surface. The gripper trajectory starts from the pose (a_x, a_y, a_z, a_ϕ) and moves 40cm along y-axis of the world frame. (a_x, a_y, a_z, a_ϕ) is predicted relative to the mean position of the target objects. The task success is achieved when all target objects contact the ground. For robust sweeping, the tool ideally need to have a large flat surface in contact with the target object.

Hammering: A peg and a slot are randomly placed in the manipulation region, where the peg is horizontally half-way inserted into the slot. The task goal is to use the tool to hammer the peg fully into the slot. The motion primitive of hammering is a rotation of the gripper along the z-axis. The trajectory starts with the gripper pose (a_x, a_y, a_z, a_ϕ) and ends after the last arm joint rotates by 90 degree counterclockwise at full speed. (a_x, a_y, a_z, a_ϕ) is predicted relative to the position of the peg. The task success is achieved when the whole peg is inside the slot. This task requires a sufficient contact force between the tool and the peg to overcome the resistance. Meanwhile the tool should avoid collisions with the peg before the hammering.

B. Experiment Setup

Training uses 18,000 procedurally generated objects including 9,000 PG-Primitive objects and 9,000 PG-Complex objects. In addition to randomizing physical properties, we randomly sample the camera pose and intrinsics by adding disturbances to the values obtained from the real-world setup.

During testing, we use 3000 instances of each type of procedurally generated object. We also test on 55 realistic objects selected from Dex-Net 1.0 [34] and MPI Grasping dataset [2]. These objects contain both tool-like and non-tool like objects as shown in Figure 4. None of these test objects are seen during training.

We compare our method to 4 baselines:

1) *Antipodal+Random*: Use a sampled antipodal grasp with a random action uniformly sampled with x, y, z positions in $[-5, 5]$ in terms of centimeters and θ in $[-\frac{\pi}{20}, \frac{\pi}{20}]$.
2) *Task_Agn+Random*: A task-agnostic grasp from Dex-Net 2.0 [35] with a random action.
3) *Task_Agn+Trained*: Same as above but with a manipulation policy trained with our method. This is akin to the current best solution.
4) *Task_Ori+Random*: An ablative version of our model with task-oriented grasps executed with a randomized action.

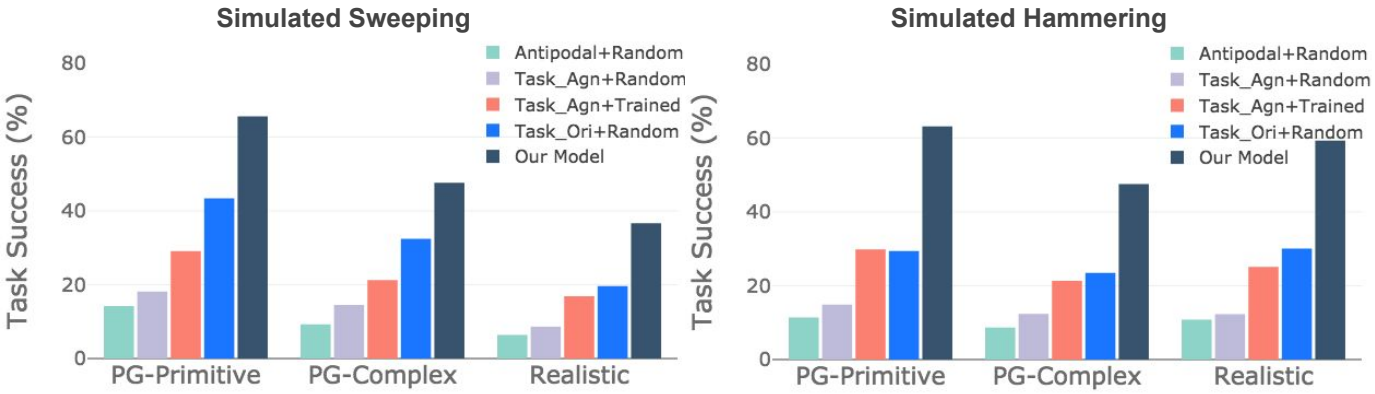

Figure 5: **Performance of simulated experiments.** We perform an evaluation of our model for sweeping and hammering in simulation. We compare performance separately on three object categories as shown in Figure 4: procedurally generated objects with primitive meshes (PG-Primitive), procedurally generated objects with complex meshes (PG-Complex), and 3D objects from existing datasets (Realistic). Our model outperforms all baselines using the three object categories in both tasks.

C. Simulated Experiments

We evaluate our method on both tasks using the simulated setup described above. For each algorithmic method, we run 100,000 episodes in simulation and report the task success rate. The task success analysis for each of the baselines and our method is presented in Figure 5.

Our model outperforms the four baselines in both tasks for all object categories. The contrast is more significant for hammering than sweeping. This is because hammering requires well-trained manipulation policy to direct the tool to hit the peg. A small deviance from the optimal hammering trajectory can let the tool miss the peg or collide with the slot. While for the sweeping task, when the robot uses a long edge of the tool to sweep, there is a high tolerance of manipulation action errors. Even random actions can often succeed. Among the three object categories, PG-Primitive is usually the easiest to manipulate with. Complex meshes cause more grasping failures and are harder for their geometric properties are harder to reason about. Realistic objects are not usually good for sweeping since they are more roundish and very few have long edges. While the hammering performance with realistic objects are much better, because many of these objects are cylinder objects with a bulky head and even actual hammers.

D. Real-World Experiments

For our real-world experiments, we use 9 unseen objects consisting of three categories as shown in Figure 6. T-shape and L-shape objects have similar geometric properties with our procedurally generated objects during training, whereas the miscellaneous objects have structures and curvatures totally unseen during training.

In the real world, we compare our model with two baseline methods: antipodal grasping with trained manipulation policy (antipodal + trained) and task-agnostic grasping with trained manipulation policy (task-agnostic + trained). We perform each task with each object for 5 robot trials for a total of 270 trials. The per-category and overall task success rates are shown in Table I. For all object categories, our model achieved better performance compared to the baselines.

For sweeping, our model can successfully grasp the head of T-shapes or the short edge of L-shapes, and sweep with the longer part. For more complex miscellaneous objects, it is less obvious for the model to figure out which part should be grasped. But for most trials, the grasp predicted by our model is intuitive to humans and leads to successful sweeping. For T-shapes, the difference between task-agnostic and task-oriented grasping is larger since the object usually only has one long handle. In contrast for some L-shapes, the two edges are both long enough for the task, grasping either edge does not make a significant difference. For miscellaneous objects, the model can have problems reasoning about novel object parts. For instance, it sometimes chooses to grasp the handle of the pan and sweep with the round part, which is unstable for sweeping roundish target objects.

For hammering, our model performs equally well for T-shapes and L-shapes. And the failures are usually caused by occasional deviations during the execution of grasping or manipulation. As a comparison, baseline models often choose to grasp the head and hammer with the handle, which is sub-optimal. Compared to T-shapes and L-shapes, there might not be an obvious way to use miscellaneous objects as hammers. Among miscellaneous objects, the pan can be used as a hammer very well. The model tends to grasp the handle and hammer with the bulky roundish part.

E. Qualitative Analysis

In Figure 7, we demonstrate the same object can be grasped for different tasks by our trained model. Here we show the modes of task-agnostic grasping and task-oriented grasping for 4 example objects, 2 in simulation and 2 in the real world.

For the sweeping task, it is challenging to sweep all target objects off the table in one shot. It requires the tool to have a flat contact surface to facilitate the manipulation of roundish objects and to sweep across large enough area to catch both of them. Our model learns to grasp the end of the tool object and spare as much surface area as possible for sweeping. This enables the robot to robustly sweep the cans most of the time.

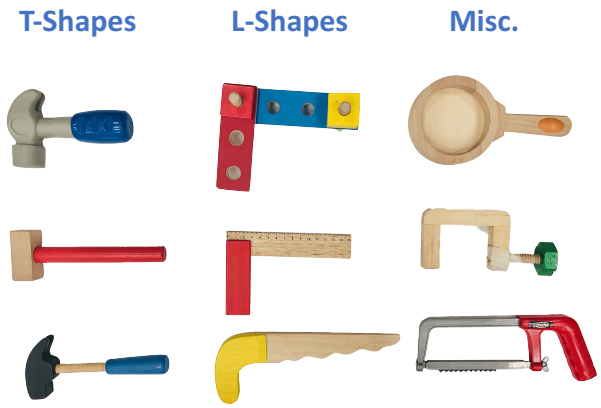

Figure 6: **Real-world test objects.** We used 9 unseen objects for our real-world experiments. These objects are grouped into three categories: T-shapes, L-shapes and miscellaneous objects.

Real-World Sweeping	Grasping Model		
	Antipodal + Trained	Task-Agnostic + Trained	Our Model
T-Shapes	13.3	20.0	**73.3**
L-Shapes	23.7	46.7	**80.0**
Misc	33.3	13.3	**60.0**
Overall	24.4	23.6	**71.1**

Real-World Hammering	Grasping Model		
	Antipodal + Trained	Task-Agnostic + Trained	Our Model
T-Shapes	46.7	60.0	**86.7**
L-Shapes	13.3	33.3	**86.7**
Misc	40.0	53.3	**66.7**
Overall	33.3	44.4	**80.0**

Table I: **Performance of real-world experiments.** We compare our model with other grasping methods in terms of task success rates. We use 9 real-world objects grouped into 3 categories. We perform 5 trials with each object for each method, for a total of 270 robot trials. The per-task and overall task success rates are reported in each cell.

For the hammering task, the main concerns are overcoming the resistance of the peg while avoiding collisions during hammering. We expect the robot to grasp the far end of the handle and hit the peg with the bulky part as the hammer head. Ideally, this could generate the largest torque on the hammer head when hitting the peg. In practice, we found the trained model tends to grasp a little closer to the hammer head on the handle. This is because we use a parallel jaw gripper and it is hard to balance the tool when the fingers are far away from the center of mass.

For robust task-agnostic grasping, the GQCNN model usually chooses the thin part near the center of mass. Although this sometimes still overlaps with the set of task-oriented grasps, it is not guaranteed if the selected grasp from the task-agnostic GQCNN is suitable for the task. Sometimes the gripper fingers are on the right position, but the orientation of the grasp is the opposite to the optimal task-oriented grasp. We often observe the task-agnostic grasping model chooses to hammer with the handle or sweep with a short edge. In Figure 7 we show example task-agnostic grasps which are different from the task-oriented grasps mentioned above.

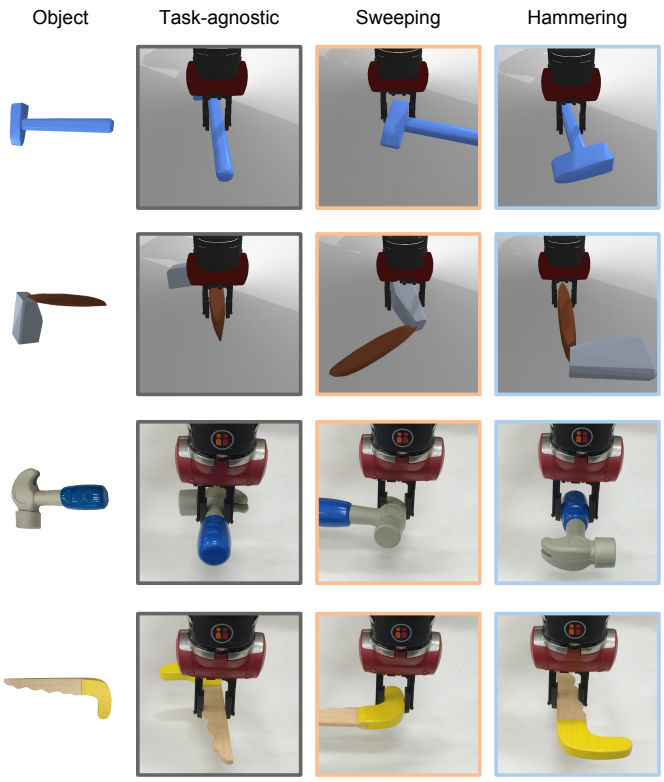

Figure 7: **Qualitative results.** Column 1 shows RGB images of the tool objects. Column 2 shows example task-agnostic grasps. And Columns 3 and 4 show task-oriented grasps chosen by our model for the sweeping and hammering tasks. Our model favors wide flat surfaces for sweeping and long moment arms for hammering.

VII. Conclusion

We develop a learning-based approach for task-oriented grasping for tool-based manipulation trained using simulated self-supervision. It jointly optimizes a task-oriented grasping model and its accompanying manipulation policy to maximize the task success rate. We leverage a physics simulator that allows a robot to autonomously perform millions of grasping and manipulation trials. The trial and error of the robot provides training data to supervise the deep neural network models. Our experimental results demonstrate that the task-oriented grasps selected by our model are more suitable for downstream manipulation tasks than the task-agnostic grasps.

In the future, our goal is to further improve the effectiveness and robustness of our model by training on a large dataset of realistic 3D models. Additionally, we plan to scale up our model to complex manipulation tasks with end-to-end trained closed-loop manipulation policies. Supplementary material is available at: **`bit.ly/task-oriented-grasp`**

Acknowledgement

We acknowledge the support of Toyota (1186781-31-UDARO). We thank Ozan Sener for constructive discussions about the problem formulation. We thank Alex Fu, Julian Gao and Danfei Xu for helping with the real-world infrastructure. We thank Erwin Coumans for helping with the Bullet simulator.

References

[1] Christopher Baber. *Cognition and tool use: Forms of engagement in human and animal use of tools*. CRC Press, 2003.

[2] Jeannette Bohg, Antonio Morales, Tamim Asfour, and Danica Kragic. Data-driven grasp synthesis – a survey. *IEEE Transactions on Robotics*, 30(2):289–309, 2014.

[3] Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, Kurt Konolige, et al. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. *arXiv preprint arXiv:1709.07857*, 2017.

[4] Solly Brown and Claude Sammut. Tool use and learning in robots. In *Encyclopedia of the Sciences of Learning*, pages 3327–3330. Springer, 2012.

[5] Matei T Ciocarlie and Peter K Allen. Hand posture subspaces for dexterous robotic grasping. *IJRR*, 28(7): 851–867, 2009.

[6] Erwin Coumans and Yunfei Bai. pybullet, a python module for physics simulation, games, robotics and machine learning. http://pybullet.org/, 2016–2017.

[7] Hao Dang and Peter K Allen. Semantic grasping: Planning robotic grasps functionally suitable for an object manipulation task. In *IROS*, pages 1311–1317, 2012.

[8] Thanh-Toan Do, Anh Nguyen, Ian Reid, Darwin G Caldwell, and Nikos G Tsagarakis. Affordancenet: An end-to-end deep learning approach for object affordance detection. *arXiv preprint arXiv:1709.07326*, 2017.

[9] Mehmet Dogar and Siddhartha Srinivasa. A framework for push-grasping in clutter. *Robotics: Science and systems*, 1, 2011.

[10] Andreas Eitel, Nico Hauff, and Wolfram Burgard. Learning to singulate objects using a push proposal network. *arXiv preprint arXiv:1707.08101*, 2017.

[11] Kuan Fang, Yunfei Bai, Stefan Hinterstoisser, Sivlvio Savarese, and Mrinal Kalakrishnan. Multi-task domain adaptation for deep learning of instance grasping from simulation. In *ICRA*, 2018.

[12] Carlo Ferrari and John Canny. Planning optimal grasps. In *ICRA*, pages 2290–2295, 1992.

[13] Paul Fitzpatrick, Giorgio Metta, Lorenzo Natale, Sajit Rao, and Giulio Sandini. Learning about objects through action-initial steps towards artificial cognition. In *ICRA*, volume 3, pages 3140–3145, 2003.

[14] James J. Gibson. *The Ecological Approach to Visual Perception*. 1979.

[15] Corey Goldfeder, Matei Ciocarlie, Hao Dang, and Peter K Allen. The columbia grasp database. In *ICRA*, pages 1710–1716, 2009.

[16] Frank Guerin, Norbert Kruger, and Dirk Kraft. A survey of the ontogeny of tool use: from sensorimotor experience to planning. *IEEE Transactions on Autonomous Mental Development*, 5(1):18–45, 2013.

[17] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[18] Robert Haschke, Jochen J Steil, Ingo Steuwer, and Helge Ritter. Task-oriented quality measures for dextrous grasping. In *Computational Intelligence in Robotics and Automation, 2005. CIRA 2005. Proceedings. 2005 IEEE International Symposium on*, pages 689–694, 2005.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[20] Ana Huaman Quispe, Heni Ben Amor, Henrik Christensen, and Mike Stilman. Grasping for a purpose: Using task goals for efficient manipulation planning. *arXiv preprint arXiv:1603.04338*, 2016.

[21] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456, 2015.

[22] Raghvendra Jain and Tetsunari Inamura. Learning of tool affordances for autonomous tool manipulation. In *System Integration (SII), 2011 IEEE/SICE International Symposium on*, pages 814–819, 2011.

[23] Eric Jang, Sudheendra Vijaynarasimhan, Peter Pastor, Julian Ibarz, and Sergey Levine. End-to-end learning of semantic grasping. *arXiv preprint arXiv:1707.01932*, 2017.

[24] Yun Jiang, Marcus Lim, and Ashutosh Saxena. Learning object arrangements in 3d scenes using human context. *arXiv preprint arXiv:1206.6462*, 2012.

[25] Daniel Kappler, Jeannette Bohg, and Stefan Schaal. Leveraging big data for grasp planning. In *ICRA*, pages 4304–4311. IEEE, 2015.

[26] Dov Katz, Arun Venkatraman, Moslem Kazemi, J Andrew Bagnell, and Anthony Stentz. Perceiving, learning, and exploiting object affordances for autonomous pile manipulation. *Autonomous Robots*, 37(4):369–382, 2014.

[27] Shunsuke Komizunai, Fumiya Nishii' Teppei Tsujita, Yuki Nomura, and Takuya Owa. Experiments on hammering a nail by a humanoid robot hrp-2. In *Proceedings of the 17th CISM-IFToMM Symposium on Robot Design, Dynamics, and Control*, 2008.

[28] Hema Swetha Koppula, Rudhir Gupta, and Ashutosh Saxena. Learning human activities and object affordances from RGB-D videos. *The International Journal of Robotics Research*, 32(8):951–970, 2013.

[29] Michael Laskey, Caleb Chuck, Jonathan Lee, Jeffrey Mahler, Sanjay Krishnan, Kevin Jamieson, Anca Dragan, and Ken Goldberg. Comparing human-centric and robot-centric sampling for robot deep learning from demonstrations. In *ICRA*, pages 358–365, 2017.

[30] Ian Lenz, Honglak Lee, and Ashutosh Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5):705–724, 2015.

[31] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination

for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 2016.

[32] Zexiang Li and S Shankar Sastry. Task-oriented optimal grasping by multifingered robot hands. *IEEE Journal on Robotics and Automation*, 4(1):32–44, 1988.

[33] Kevin M Lynch and Matthew T Mason. Stable pushing: Mechanics, controllability, and planning. *The International Journal of Robotics Research*, 15(6):533–556, 1996.

[34] Jeffrey Mahler, Florian T Pokorny, Brian Hou, Melrose Roderick, Michael Laskey, Mathieu Aubry, Kai Kohlhoff, Torsten Kröger, James Kuffner, and Ken Goldberg. Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards. In *ICRA*, pages 1957–1964, 2016.

[35] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*, 2017.

[36] Khaled Mamou and Faouzi Ghorbel. A simple and efficient approach for 3d mesh approximate convex decomposition. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 3501–3504. IEEE, 2009.

[37] Tanis Mar, Vadim Tikhanoff, Giorgio Metta, and Lorenzo Natale. Self-supervised learning of grasp dependent tool affordances on the icub humanoid robot. In *ICRA*, pages 3200–3206, 2015.

[38] Tanis Mar, Vadim Tikhanoff, Giorgio Metta, and Lorenzo Natale. Self-supervised learning of tool affordances from 3d tool representation through parallel som mapping. In *ICRA*, pages 894–901. IEEE, 2017.

[39] Tekin Meriçli, Manuela Veloso, and H Levent Akın. Push-manipulation of complex passive mobile objects using experimentally acquired motion models. *Autonomous Robots*, 38(3):317–329, 2015.

[40] Andrew T Miller, Steffen Knoop, Henrik I Christensen, and Peter K Allen. Automatic grasp planning using shape primitives. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 2, pages 1824–1829, 2003.

[41] François Osiurak, Christophe Jarry, and Didier Le Gall. Grasping the affordances, understanding the reasoning: toward a dialectical theory of human tool use. *Psychological review*, 117(2):517, 2010.

[42] Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *ICRA*, pages 3406–3413, 2016.

[45] Reuven Y Rubinstein and Dirk P Kroese. The cross-

[43] Mario Prats, Pedro J Sanz, and Angel P Del Pobil. Task-oriented grasping using hand preshapes and task frames. In *ICRA*, pages 1794–1799. IEEE, 2007.

[44] Alberto Rodriguez, Matthew T Mason, and Steve Ferry. From caging to grasping. *The International Journal of Robotics Research*, 31(7):886–900, 2012.
entropy method: A unified approach to monte carlo simulation, randomized optimization and machine learning. *Information Science & Statistics, Springer Verlag, NY*, 2004.

[46] Ashutosh Saxena, Justin Driemeyer, and Andrew Y Ng. Robotic grasping of novel objects using vision. *The International Journal of Robotics Research*, 27(2):157–173, 2008.

[47] Dan Song, Kai Huebner, Ville Kyrki, and Danica Kragic. Learning task constraints for robot grasping using graphical models. In *IROS*, pages 1579–1585, 2010.

[48] Alexander Stoytchev. Behavior-grounded representation of tool affordances. In *ICRA*, pages 3060–3065, 2005.

[49] Joshua Tobin, Wojciech Zaremba, and Pieter Abbeel. Domain randomization and generative models for robotic grasping. *CoRR*, abs/1710.06425, 2017.

[50] Karthik Mahesh Varadarajan and Markus Vincze. Afrob: The affordance network ontology for robots. In *IROS*, pages 1343–1350, 2012.

[51] Ulrich Viereck, Andreas ten Pas, Kate Saenko, and Robert Platt. Learning a visuomotor controller for real world robotic grasping using easily simulated depth images. *arXiv preprint arXiv:1706.04652*, 2017.

[52] Jonathan Weisz and Peter K Allen. Pose error robust grasping from contact wrench space metrics. In *ICRA*, pages 557–562, 2012.

[53] Wikibooks. Openscad user manual — wikibooks, the free textbook project, 2018. URL https://en.wikibooks.org/w/index.php?title=OpenSCAD_User_Manual&oldid=3423143. [Online; accessed 26-May-2018].

[54] Matthew Murray Williamson. *Robot arm control exploiting natural dynamics*. PhD thesis, Massachusetts Institute of Technology, 1999.

[55] Yixin Zhu, Yibiao Zhao, and Song Chun Zhu. Understanding tools: Task-oriented object modeling, learning and recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2855–2864, 2015.

[56] Yuke Zhu, Alireza Fathi, and Li Fei-Fei. Reasoning about object affordances in a knowledge base representation. In *ECCV*, 2014.