To appear in *Advanced Robotics*
Vol. XX, No. XX, XX 20XX, 1–17

FULL PAPER

Hierarchical Reinforcement Learning of Multiple Grasping Strategies with Human Instructions

T. Osa[a,b*], Jan Peters[c,d], and G. Neumann[e]

[a] *University of Tokyo, Tokyo, Japan*; [b] *RIKEN, Tokyo, Japan*; [c] *Technische Universität Darmstadt, Darmstadt, Germany*; [d] *Max-Planck Institute, Tübingen, Germany*; [e] *University of Lincoln, Lincoln, UK*

(*Received month 20XX, Accepted month 20XX*)

Grasping is an essential component for robotic manipulation and has been investigated for decades. Prior work on grasping often assumes that a sufficient amount of training data is available for learning and planning robotic grasps. However, constructing such an exhaustive training dataset is very challenging in practice, and it is desirable that a robotic system can autonomously learn and improves its grasping strategy. Although recent work has presented autonomous data collection through trial and error, such methods are often limited to a single grasp type, e.g., vertical pinch grasp. To address these issues, we present a hierarchical policy search approach for learning multiple grasping strategies. To leverage human knowledge, multiple grasping strategies are initialized with human demonstrations. In addition, a database of grasping motions and point clouds of objects is also autonomously built upon a set of grasps given by a user. The problem of selecting the grasp location and grasp policy is formulated as a bandit problem in our framework. We applied our reinforcement learning to grasping both rigid and deformable objects. The experimental results show that our framework autonomously learns and improves its performance through trial and error and can grasp previously unseen objects with a high accuracy.

Keywords: hierarchical reinforcement learning; grasping; point clouds; active learning

1. Introduction

Grasping is an essential motion for robotic manipulation, which has been investigated for decades [1, 2]. Many approaches for robotic grasping are data driven and can leverage a dataset of grasping motions and objects. Prior work on data-driven grasping methods often assumes that the sufficient amount of training data is available for learning and planning robotic grasps. For instance, geometry-based methods such as [3, 4] require thousands of example grasps. However, since constructing such exhaustive training datasets is very challenging in practice, it is desirable that a robotic system autonomously learns and improves its grasping strategy. Recent work has presented autonomous data collection through trial and error [5, 6]. However, such methods are often limited to a single grasp type, e.g., vertical pinch grasps. Since grasping various objects requires multiple grasp types in practice [7], it is necessary to develop a method for autonomously learning multiple grasping policies for different grasp types.

When we learn multiple grasp types, it is essential to learn how to grasp objects with specific grasp types and how to select grasp types according to given objects. In addition, since an object often has multiple potential grasp locations, it is also necessary to learn how to select the grasp locations among multiple options. In this work, we address such autonomous learning of

*Corresponding author. Email: osa@mfg.t.u-tokyo.ac.jp

multiple grasp types. We propose reinforcement learning of a hierarchical policy where lower-level policies learn how to grasp given locations with specific grasp types and an upper-level policy learns how to select the grasp type and location. We address the selection of the grasp type and location using an active learning approach. Our approach learns to grasp objects with multiple grasp types and autonomously improves the grasping performance through trial and error. This paper consolidates our recent work on reinforcement learning of robotic grasping. In our prior study in [8], we presented our preliminary work on reinforcement learning for multiple grasping types. We present a rigorous description of our approach in this paper, including a collision cost for grasping selection, which was not presented in our previous study. While our previous study reported only results of grasping rigid objects based on analytical scores, this paper also presents results of learning to grasp a deformable object from human evaluations. Grasping a deformable object is a challenging task for the template-based approach because the deformation of an object is often hard to model and predict. Since the deformation of the object often leads to slip of the object, grasping deformable objects requires different grasping strategies from that for rigid objects. The experimental results show that our system can adjust the grasping strategy for rigid objects to a deformable object through trial and error on a real robot.

The remainder of this paper is structured as follows. The next section describes related work on grasping and policy search. We present our reinforcement learning method in Section 3. The experimental results are presented in Section 4. After we discuss our results and future work in Section 5, we conclude this paper in Section 6.

2. Related Work

Our work is closely related to two domains: grasping in robotics and policy search in machine learning. We first describe prior work related to robotic grasping and we subsequently discuss prior work related to policy search.

2.1 *Grasping*

Although early studies on data-driven grasping utilize a library of 3D object models [3], recent work demonstrates that grasp planning based on point clouds or RGB-D images can be applied to various objects without solid 3D models [4, 9–13]. However, the performance of the existing methods often significantly depends on the quality of the training dataset of grasping motions and objects. For example, a method in [4] computes Height Accumulated Features (HAF) and detects the grasp locations. This method performs well even in scenarios with multiple objects. However, it requires a training dataset with thousands of grasps. The work by Lenz et al. shows that the potential grasp locations can be detected using a convolutional neural network (CNN) [10]. This method also requires large amount of data to train the CNN. In addition, many methods are merely based on the geometric properties of objects and do not consider the physical interaction with objects. For example, Kopicki et al. proposed to a method for determining where to grasp based on the density of the object point cloud [9]. Although this method does not require a huge amount of the training data, it does not incorporate the quality of the resulting grasp motion.

To address the issue of data collection in grasp learning, reinforcement learning has been employed in recent work. The study in [5] showed the feasibility of autonomously collecting a dataset with thousands of grasps and training a CNN to predict grasp locations as rectangles in a 2D image. Likewise, the study in [6] presented a method for learning the hand-eye coordination for grasping. A vision-based feedback control is learned with a CNN and a dataset of grasping motion and images are constructed autonomously in [6]. Since these methods utilize the results of the executed grasps in order to predict the grasp probability, the physical interaction between objects and a robotic hand is implicitly incorporated. However, the methods in [5, 6] are limited

to specific grasp types, e.g., a vertical precision grasp. In addition, these methods rely on 2D image inputs lacking depth information. However, the use of depth information and learning multiple grasp types are essential to achieve dexterous manipulations.

2.2 *Hierarchical Policy Search*

Policy search is a subset of reinforcement learning [14]. Instead of directly working on the state-action space, policy search explores a solution in a parameterized space. Methods for learning a policy that generates different parameters according to a task context is often referred to as *contextual policy search* [14]. In grasp learning, we learn a policy that generates grasping motion according to a given grasp location, which is considered the context of the task. Thus, selecting a grasp location among several options in grasp learning is equivalent to selecting a context in the contextual policy search setting. Contextual policy search with active context selection is investigated as active contextual policy search in [15]. However, this active contextual policy search has not been extended to learning a hierarchical policy.

Interests on methods for learning a hierarchical policy has been increasing [16–19]. The goal of hierarchical policy search is to learn a hierarchical policy where multiple lower-level policies represent different strategies and an upper-level policy selects the appropriate lower-level policy from the multiple options. For instance, Daniel et al. have developed the hierarchical relative entropy policy search (HiREPS) algorithm for learning a hierarchical policy through reinforcement learning [19]. HiREPS learns a hierarchical policy through the EM-like procedure based on relative entropy policy search (REPS) [20]. Likewise, the option-actor critic proposed in [16] learns a hierarchical policy using an extended version of actor-critic [21]. Prior work on hierarchical reinforcement learning indicates that learning multiple option policies improves the robustness of the policy [19] and that exploitation of the hierarchical policy structure exponentially reduces the search space [22]. However, these methods do not incorporate an active selection of the context as it is needed for grasping. The study by Kroemer et al. proposed to formulate the selection of grasp locations as a bandit problem [23]. This study shows that a grasping policy can be efficiently learned by actively selecting the grasp location in order to deal with the trade off between maximizing the information gain and maximizing the expected grasp quality. However, reinforcement learning of a hierarchical grasping policy is not addressed in [23] and the method in [23] does not generalize its grasping policy to unknown objects.

In this work, we formulate the active selection of both the grasp location and the grasp type as a bandit problem. In contrast with prior work, our approach has the following important properties: 1) learning of multiple grasp types, 2) autonomous construction of a grasp database through trial and error, and 3) planning grasping motions based on point clouds, and 4) an active selection of the grasp type and location. To the best of our knowledge, prior work has not developed a grasp learning method that has all of these properties.

3. Hierarchical Reinforcement Learning for Grasping

We describe our grasp learning framework in this section. After describing the overview of our learning method, we explain the details of each component.

3.1 *Overview of the Proposed Method*

In the scenario of grasp learning using point clouds of objects, the goal is to learn a policy that plans the grasping motion based on a given point cloud. By following the formulation of contextual policy search, this problem can be formulated as

$$\pi(\boldsymbol{\theta}|\mathcal{P}) = \arg\max_{\pi} \mathbb{E}[R(\boldsymbol{\theta}, \mathcal{P})|\pi(\boldsymbol{\theta}|\mathcal{P}), \mathcal{P}], \tag{1}$$

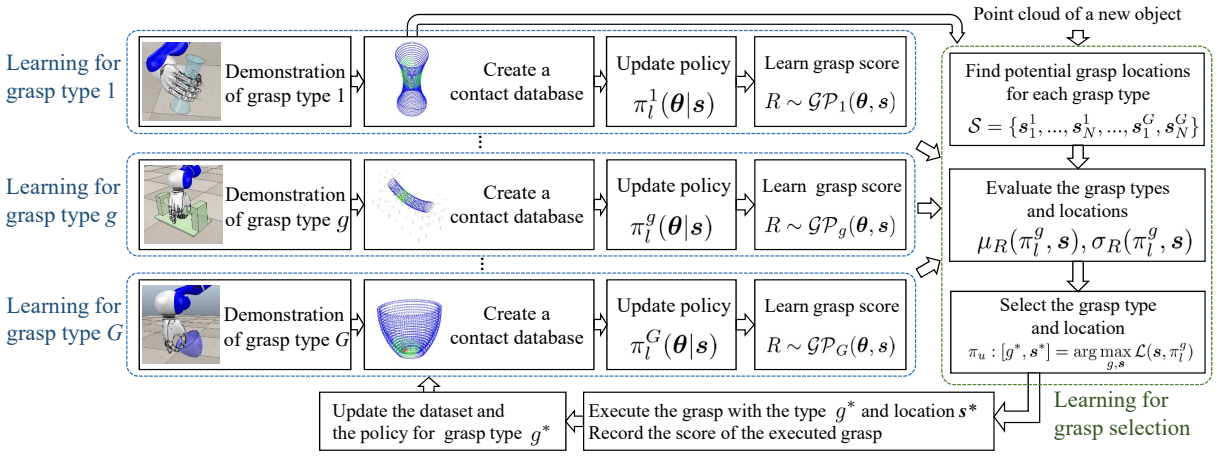


Figure 1. Overview of the algorithm. First, the grasping policy is initialized, and the dataset of contact information is created based on human demonstration. The individual lower-level policy π_l is learned for each grasp type. The grasp quality R is approximated with Gaussian Processes (GPs). GPs are used to evaluate each combination of grasp type and location. When the point cloud of a new object is given, potential grasp locations are estimated using the grasp dataset. Subsequently, the upper-level policy π_u selects the grasp type and location. After every grasp execution, the grasping policy and the dataset are updated. $\pi_l^g = \pi_l(\boldsymbol{\theta}|\boldsymbol{s}, g)$ in this figure.

where $\mathcal{P}$ is a point cloud of the given object, $\pi(\boldsymbol{\theta}|\mathcal{P})$ is a policy that generates a grasping motion parameter $\boldsymbol{\theta}$ and $R(\boldsymbol{\theta}, \mathcal{P})$ is the return, which represents the quality of the grasp. In this work, we assume that a contact between a robotic hand and an object can be represented by a set of contact points. However, directly working on point clouds as contexts is not feasible due to the high dimensionality of point clouds. To make this problem tractable, we extract potential grasp locations from the given point cloud and compute local features $\boldsymbol{s}_i$ for each of these locations. Using these local features, the grasp learning problem can be reduced to

$$\pi(\boldsymbol{\theta}|\mathcal{S}) = \arg\max_{\pi} \mathbb{E}[R(\boldsymbol{\theta}, \mathcal{S})|\pi(\boldsymbol{\theta}|\mathcal{S}), \mathcal{S}], \tag{2}$$

where $\mathcal{S}$ is a finite set of feature vectors $\boldsymbol{s}_i$ of potential grasp locations in the given point cloud. For example, $\boldsymbol{s}$ can be a position of the centroid of the point cloud or a vector normal to the surface. We consider a hierarchical policy where lower-level policies π_l learn how to grasp objects with a specific grasp type and an upper-level policy π_u learns how to select the grasp type and where to grasp. Therefore, our hierarchical grasping policy is given by

$$\pi(\boldsymbol{\theta}|\mathcal{S}) = \sum_{\boldsymbol{s},g} \pi_u(\boldsymbol{s}, g|\mathcal{S}) \pi_l(\boldsymbol{\theta}|\boldsymbol{s}, g), \tag{3}$$

where g is the index of the grasp type. Note that our upper-level policy π_u is deterministic. To learn this hierarchical policy, we divide the problem of grasp learning into four steps:

(1) **Find the potential grasp locations.** Given a point cloud of a target object and the database of the contact information of successful grasps, we find local subsets of the point cloud that represent potential grasping locations, i.e.,

$$\mathcal{P} \mapsto \{\mathcal{P}'_1, ..., \mathcal{P}'_N\}, \tag{4}$$

where $\mathcal{P}'_i$ is the i^{th} subset of the point cloud that represents a potential grasp location.

(2) **Select the grasp type and location.** Given a set of grasping policies $\mathcal{G} = \{\pi_l^1, ..., \pi_l^G\}$ that represents multiple grasp types and subsets of point clouds $\{\mathcal{P}'_1, ..., \mathcal{P}'_N\}$ that represent potential grasping locations, we select a grasp type and a grasp location. For this purpose,

we compute a local feature vector $\boldsymbol{s}_i$ for each local point cloud $\mathcal{P}'$,

$$\mathcal{P}'_i \mapsto \boldsymbol{s}_i \text{ for } i = 0, ..., N. \tag{5}$$

We approximate the return function with a Gaussian Process

$$R \sim \mathcal{GP}(\boldsymbol{\theta}, \boldsymbol{s}) \tag{6}$$

as a function of the motion parameter $\boldsymbol{\theta}$ and the point cloud feature $\boldsymbol{s}$. Using this approximation, we can compute $\mathbb{E}[R|\boldsymbol{s}, g]$ and its variance for each combination of the grasping policy π_l and the feature $\boldsymbol{s}$. The upper-level policy π_u selects the grasping policy and the location based on an objective function U, i.e.,

$$\pi_u : [\boldsymbol{s}^*, g^*] = \arg \max_{\boldsymbol{s} \in \mathcal{S}, g \in \mathcal{G}} U(\boldsymbol{s}, g), \tag{7}$$

where $\mathcal{S}$ is a set of features of possible grasp locations. The objective function U needs to deal with the trade-off between maximizing the expected return and maximizing the information gain, which is often referred to as the "exploration-exploitation trade-off." In this work, we employ upper confidence bounds (UCB) as the objective function.

(3) **Perform the selected grasp.** The grasping motion is planned and performed using the selected grasp type and grasp location by drawing the motion parameter from the selected lower-level policy, i.e.,

$$\boldsymbol{\theta} \sim \pi_l(\boldsymbol{\theta}|\boldsymbol{s}^*, g^*). \tag{8}$$

The motion parameter could be the final configuration for grasping or parameters of a parametric trajectory representation such as Dynamic Movement Primitives [24]. The result of the grasp execution is evaluated and stored in a database $\mathcal{D} = \{(R_i, \boldsymbol{\theta}_i, \boldsymbol{s}_i^*, g_i^*)\}$. The local point cloud $\mathcal{P}'$ that represents the selected grasp location is also stored in the database.

(4) **Update the grasping policy.** Using the data obtained from the new grasp execution, we update the grasping policy and the database of the contact information. This policy update is formulated as a contextual policy search problem, which we describe in Section 3.4.

Our framework is summarized in Fig. 1. By repeating steps (1)-(4), the grasping policy and the database of grasping motions and contact information improve through trial and error. In the following sections, we describe the details of each step.

3.2 *Finding Potential Grasp Locations*

Given a point cloud of a target object, our framework firstly finds potential grasp locations, which are often referred to as "grasp affordances" [11]. For this purpose, our framework constructs a database of the contact information of successful grasps through trial and error. This database of contact information consists of local point clouds of contact surface of each finger for successful grasps. Given a point cloud of a target object, the system searches for local regions that are similar to the contact region stored in the successful grasp database $\mathcal{D}_{\text{contact}}$. In this process, we use the Iterative Closest Points (ICP) algorithm [25], which finds a homogeneous transformation H_{icp} that minimizes the distance between two point clouds.

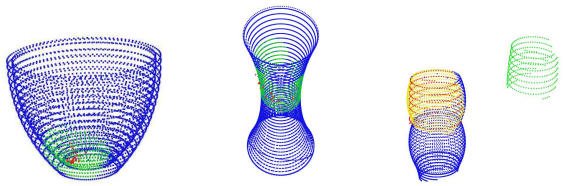
When a point cloud of a target object $\mathcal{P}_{\text{target}}$ is given, the system randomly samples a subset of the point cloud of the target object $\mathcal{P}_i \subset \mathcal{P}_{\text{target}}$. Subsequently, ICP is performed between $\mathcal{P}_i$ and each contact region in our dataset $\mathcal{C}_j \in \mathcal{D}_{\text{contact}}$. ICP returns the residual distance d_{icp} between $\mathcal{P}_i$ and $\mathcal{C}_j$. Using the result of ICP with the smallest residual distance d^*_{icp}, we can find

Algorithm 1 Finding potential grasp locations

Initialization: Store contact regions in M successful grasps $\mathcal{D}_{\text{contact}} = \{\mathcal{C}_1, \ldots, \mathcal{C}_M\}$
Input: number of the grasp candidates N, point cloud of the target object $\mathcal{P}_{\text{target}}$
for $i = 1 : N$ **do**
  Randomly choose a subset of the point cloud of the new object $\mathcal{P}_i \subset \mathcal{P}_{\text{target}}$
  for $j = 1 : M$ **do**
    Perform ICP algorithm between $\mathcal{P}_i$ and $\mathcal{C}_j \in \mathcal{D}_{\text{contact}}$.
    $[d^j_{\text{icp}}, H^j_{\text{icp}}] = \text{ICP}(\mathcal{P}_i, \mathcal{C}_j)$
  end for
  Compute $j^* = \text{argmin}_j \, d^j_{\text{icp}}$
  Find a point cloud $\mathcal{P}^i_{\text{grasp}}$ in the neighborhood of $H^{j^*}_{\text{icp}}\mathcal{C}_{j^*}$ from $\mathcal{P}_{\text{target}}$
  Compute the features of the estimated contact region, $\mathcal{P}^i_{\text{grasp}} \mapsto \boldsymbol{s}_i$
end for

the point cloud subset that is similar to the contact region in the dataset. The point cloud of the potential contact region $\mathcal{P}_{\text{grasp}}$ can be estimated as a point cloud in the neighborhood of $H^{j^*}_{\text{icp}}\mathcal{C}_{j^*}$ from $\mathcal{P}_{\text{target}}$. Fig. 2 shows examples of the contact regions in the dataset and the behavior of ICP. As our experimental result show, this local search enables to find potential grasp locations even if the given point cloud does not exactly fit with the samples in the database. The process of estimating potential grasp locations is summarized in the Algorithm 1. Multiple potential grasp locations can be found by repeating this local search for different subsets of $\mathcal{P}_{\text{target}}$. In our experiments, we performed ICP with 30 point clouds uniformly chosen from the contact database and the 5 subsets of the object point clouds to avoid too heavy computational cost. Therefore, ICP is performed 150 times for one grasp plan, which takes 40-60 seconds.

Separate datasets of successful grasps are maintained for different grasp types, and this process is performed for each grasp type. Our method does not require the entire point cloud of the target object because it searches for local features of the point cloud. This property is useful for grasp planning in real systems in which complete point clouds of objects are not available. The database of the successful grasps is initialized with demonstrated grasps and updated through trial and error in our framework. Therefore, the performance of finding potential grasp locations also improves autonomously.



(a) (b) (c)

Figure 2. (a) and (b):Point cloud of object with contact points. Blue points represent the point cloud of the object $\mathcal{P}$. Red points represent contact points. Green points represent the neighbors of the contact points $\mathcal{C}$. (c)Example of the result of ICP. Blue, green, red, and yellow points represent a partial point cloud $\mathcal{P}_i$ of a given object, the contact part $\mathcal{C}_j$ from the dataset of successful grasps, the result of ICP algorithm $H^j_{\text{icp}}\mathcal{C}_j$, and the estimated grasp part $\mathcal{P}_{\text{grasp}}$, respectively.

In order to obtain a concise description of the local point cloud $\mathcal{P}_{\text{grasp}}$ at the estimated grasp location, we compute the center of the contact points and the normal vector at the center of the contact points for each finger as

$$\boldsymbol{s}^\top = [\boldsymbol{x}^1_{\text{center}}, \boldsymbol{n}^1_{\text{center}}, \ldots, \boldsymbol{x}^F_{\text{center}}, \boldsymbol{n}^F_{\text{center}}], \tag{9}$$

where $\boldsymbol{x}^i_{\text{center}}$ is the center of the contact region of the i^{th} finger, $\boldsymbol{n}^i_{\text{center}}$ is the normal vector at the center of the contact region of the i^{th} finger, and F is the number of fingers of the hand. This local description of contact points $\boldsymbol{s}$ is used as a context for contextual policy search of the lower-level policies π^g_l. Although we used this local description of point clouds, our method is not limited to a specific description. Other descriptions of point clouds such as principal

curvatures [26] can be also used in our framework.

3.3 *Learning to Select the Grasp Type and the Grasp Location*

In our framework, the goal of the upper-level policy is to learn how to select the optimal grasp type and grasp location for a given object. For this purpose, we consider the upper-level policy given by

$$\pi_u : [\boldsymbol{s}^*, g^*] = \arg \max_{\boldsymbol{s} \in \mathcal{S}, \pi_l \in \mathcal{G}} U(\boldsymbol{s}, g),$$

where $U(\boldsymbol{s}, g)$ is the objective function. For grasp selection, we must consider the trade-off between gaining more information and maximizing the expected quality of the grasp. We can interpret the selection of the grasp type and grasp location as a variant of contextual bandit problem in which the context is given by a target object [27]. Therefore, we use an acquisition function based on upper confidence bounds (UCB) [28], which has been shown to perform well in practice. The learner selects the grasp type and location by maximizing the objective function

$$U(\boldsymbol{s}, g) = \mathbb{E}[R^* | \pi_l^g, \boldsymbol{s}] + \beta \sigma_R(\pi_l^g, \boldsymbol{s}^g), \tag{10}$$

where $\pi_l^g = \pi_l(\boldsymbol{\theta}|\boldsymbol{s}, g)$, $\sigma_R(\pi_l^g, \boldsymbol{s}^g)$ is the variance of predicting $\mathbb{E}[R^*|\pi_l^g, \boldsymbol{s}]$ and β is a positive constant that controls the exploration-exploitation trade-off. We heuristically set $\beta = 1.0$ in our experiment. When β is higher, it may require more iteration until convergence but it will encourage the exploration and avoid converging to local optima. The property of β for UCB is discussed in [29].

For evaluating the expected grasp quality $\mathbb{E}[R|\pi_l^g, \boldsymbol{s}]$, we approximate the grasp quality R of the g^{th} grasp type with a GP as a function of the grasp motion parameters $\boldsymbol{\theta}$ and the grasp location features $\boldsymbol{s}$, i.e.,

$$R(\boldsymbol{\theta}, \boldsymbol{s}) \sim \mathcal{GP}_g \left(m\left(\boldsymbol{z}\right), k\left(\boldsymbol{z}, \boldsymbol{z}'\right) \right) \tag{11}$$

where $\boldsymbol{z} = [\boldsymbol{\theta}, \boldsymbol{s}]^\top$. We use a squared exponential covariance function

$$k\left(\boldsymbol{z}_i, \boldsymbol{z}_j\right) = \sigma_f^2 \exp\left(-\frac{\|\boldsymbol{z}_i - \boldsymbol{z}_j\|^2}{2l^2} \right) + \sigma_n^2 \delta_{\boldsymbol{z}_i \boldsymbol{z}_j}, \tag{12}$$

where l is the bandwidth of the kernel, σ_f^2 is the function variance and σ_n^2 is the noise variance. The hyperparameters of GP models $[\sigma_f^2, l, \sigma_n^2]$ are optimized after every rollout by maximizing the marginal log likelihood [30]. Assuming zero mean prior, i.e., $m(\boldsymbol{z}) = 0$, the joint distribution of the quality measure $R_{1:N}$ of the training set and the quality measure of a test data point R^* is Gaussian, i.e.,

$$\begin{bmatrix} \boldsymbol{R}_{1:N}^g \\ R^* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \boldsymbol{K}_g & \boldsymbol{k}_g \\ \boldsymbol{k}_g^\top & k(\boldsymbol{z}^*, \boldsymbol{z}^*) \end{bmatrix} \right) \tag{13}$$

where $\boldsymbol{K}_g$ is the Gram matrix and $\boldsymbol{R}_{1:N}^g$ is a column vector that contains returns of rollouts of the g^{th} grasp type as $\boldsymbol{R}_{1:N}^g = [R_1^g, \cdots, R_N^g]^\top$. In our framework, we employ a stochastic policy for generating the grasp motion parameters, which is represented by a Guassian distribution $\pi_l^g(\boldsymbol{\theta}|\boldsymbol{s}) \sim \mathcal{N}(\boldsymbol{\mu}^g(\boldsymbol{s}), \boldsymbol{\Sigma}^g(\boldsymbol{s}))$. In order to estimate $\mathbb{E}[R|\pi_l^g, \boldsymbol{s}]$ using GPs, we can assume that a

Algorithm 2 Learning Multiple Grasp Types

Input: point cloud of the object $\mathcal{P}_{\text{target}}$, number of grasp location candidates N, number of grasp types G
Initialization: Initialize policies and GP models based on demonstrations
repeat
  for $g = 1 : G$ **do**
    Find grasp location candidates $\mathcal{P}_{\text{target}} \mapsto \{\mathcal{P}^1_{\text{grasp}}, \ldots, \mathcal{P}^N_{\text{grasp}}\}$
    Compute feature vectors of grasp location candidates $\mathcal{S}^g = \{\boldsymbol{s}^g_1, \ldots, \boldsymbol{s}^g_N\}$
    for every grasp location candidate $\boldsymbol{s}^g \in \mathcal{S}^g$ **do**
      Compute $\mathbb{E}[R^*|\pi^g_l, \boldsymbol{s}^g]$ and $\sigma(\pi^g_l, \boldsymbol{s}^g)$ using (16) and (17)
    end for
  end for
  Select the grasp type and location using the acquisition function in (10)
  Execute grasp with the grasp parameter $\boldsymbol{\theta} \sim \pi_l(\boldsymbol{\theta}|\boldsymbol{s}^*, g^*)$
  Update the GP model for the $g^{*\text{th}}$ grasp type
  Update the policy for the $g^{*\text{th}}$ grasp type with a policy search method
until grasping learned

query data point $\boldsymbol{z}^*$ for the GP is drawn from a Gaussian distribution

$$\boldsymbol{z}^* \sim \mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{z}^*}, \boldsymbol{\Sigma}_{\boldsymbol{z}^*}\right) \text{ where } \boldsymbol{\mu}_{\boldsymbol{z}} = \begin{bmatrix} \boldsymbol{\mu}^g(\boldsymbol{s}) \\ \boldsymbol{s} \end{bmatrix}, \boldsymbol{\Sigma}_{\boldsymbol{z}} = \begin{bmatrix} \boldsymbol{\Sigma}^g(\boldsymbol{s}) & 0 \\ 0 & 0 \end{bmatrix}. \tag{14}$$

Strictly speaking, this Gaussian distribution is singular. To estimate the expected return for grasp type g when given a context $\boldsymbol{s}$, we need to compute the integral over $\boldsymbol{z}^*$, i.e.,

$$p(R^*|\boldsymbol{\mu}_{\boldsymbol{z}^*}, \boldsymbol{\Sigma}_{\boldsymbol{z}^*}) = \int p(R^*|\boldsymbol{z}, \mathcal{D}) p(\boldsymbol{z}^*) d\boldsymbol{z}^*, \tag{15}$$

where $\mathcal{D}$ represents the dataset of motion parameters, contexts, and resulting returns. The studies in [31–33] show that this marginal distribution can be approximated by a Gaussian distribution with mean μ_R and variance σ_R

$$\mu_R(\pi^g_l, \boldsymbol{s}^g) = \mathbb{E}[R^*|\pi^g_l, \boldsymbol{s}^g] = \boldsymbol{q}^\top \boldsymbol{\beta} \tag{16}$$

$$\sigma_R(\pi^g_l, \boldsymbol{s}^g) = k_g\left(\boldsymbol{z}^*, \boldsymbol{z}^*\right) - \boldsymbol{k}_g^\top \boldsymbol{K}_g^{-1} \boldsymbol{k}_g + \text{Tr}\left[\boldsymbol{K}_g^{-1}(\boldsymbol{k}_g \boldsymbol{k}_g^\top - \boldsymbol{Q})\right] + \text{Tr}\left[\boldsymbol{\beta}\boldsymbol{\beta}^\top(\boldsymbol{Q} - \boldsymbol{q}\boldsymbol{q}^\top)\right] \tag{17}$$

where $\boldsymbol{\beta} = \boldsymbol{K}_g^{-1} \boldsymbol{R}^g$, and the vector $\boldsymbol{q}$ and the matrix $\boldsymbol{Q}$ are given by

$$q_j = \frac{\exp\left(-\frac{1}{2}(\boldsymbol{\mu}_{\boldsymbol{z}^*} - \boldsymbol{z}_j)^\top (\boldsymbol{\Lambda} + \boldsymbol{\Sigma}_{\boldsymbol{z}^*})^{-1} (\boldsymbol{\mu}_{\boldsymbol{z}^*} - \boldsymbol{z}_j)\right)}{|\boldsymbol{\Sigma}_{\boldsymbol{z}^*} \boldsymbol{\Lambda}^{-1} + I|^{1/2}}, \tag{18}$$

$$Q_{ij} = \frac{\exp\left(-\frac{1}{2}\left[(\boldsymbol{\mu}_{\boldsymbol{z}^*} - \boldsymbol{z}_d)^\top (\frac{\boldsymbol{\Lambda}}{2} + \boldsymbol{\Sigma}_{\boldsymbol{z}^*})^{-1} (\boldsymbol{\mu}_{\boldsymbol{z}^*} - \boldsymbol{z}_d) + (\boldsymbol{z}_i - \boldsymbol{z}_j)^\top (2\boldsymbol{\Lambda})(\boldsymbol{z}_i - \boldsymbol{z}_j)\right]\right)}{|2\boldsymbol{\Sigma}_{\boldsymbol{z}^*} \boldsymbol{\Lambda}^{-1} + I|^{1/2}}, \tag{19}$$

where $\boldsymbol{z}_d = \frac{1}{2}(\boldsymbol{z}_i + \boldsymbol{z}_j)$, and $\boldsymbol{\Lambda}$ is a diagonal matrix with $\boldsymbol{\Lambda} = l^2 \boldsymbol{I}$. We learn this GP model for each grasp type. The learned GP models are used to evaluate the use of the lower-level policy π_l with the grasp location described by $\boldsymbol{s}$.

Algorithm 2 summarizes the procedure for learning to select the grasp type and location. As with many reinforcement learning algorithms, the iteration can be stopped when the improvement of the performance is converged or when the sufficiently good performance is achieved.

3.4 *Learning the Policy for the Desired Grasp Type*

In order to learn the lower-level policies $\pi_l(\boldsymbol{\theta}|\boldsymbol{s}, g)$ that generate the grasping motion parameter vector $\boldsymbol{\theta}$ for a given context, we use the episodic version of the contextual relative entropy policy search (REPS) algorithm [14, 20]. It is worth noting that our framework is not limited to specific policy search methods, although we implement our grasp learning system with REPS. In policy search, a policy is updated in order to maximize the expected return. However, if the "difference" between the old and updated policies in the policy update step is too large, the exploration in the policy space can be unstable. To address this issue, REPS constrains the differences between the updated and old policies, and it is guaranteed that REPS achieves the local update. The theoretical property is discussed in [14, 19]. Policy search without this property might end up with a policy very different from the initial policy, which is not preferable because the resulting behavior is not predictable. In our framework, each lower-level policy is initialized by human demonstrations, and REPS finds a locally optimal policy that is associated with the grasp type indicated by human demonstrations.

REPS employs the KL divergence to quantify the difference between the updated and old policies in its policy update. The policy update using contextual REPS is formulated as a constraint optimization problem:

$$\max_{\pi} \int \mu_{\boldsymbol{s}}(\boldsymbol{s}) \int \pi(\boldsymbol{\theta}|\boldsymbol{s}) R(\boldsymbol{\theta}, \boldsymbol{s}) d\boldsymbol{\theta} d\boldsymbol{s} \tag{20}$$

$$\text{s.t. } \epsilon \geq \int \mu_{\boldsymbol{s}}(\boldsymbol{s}) \mathrm{KL}\left(\pi(\boldsymbol{\theta}|\boldsymbol{s}) || q(\boldsymbol{\theta}|\boldsymbol{s})\right) d\boldsymbol{s}, \quad \int \pi(\boldsymbol{\theta}|\boldsymbol{s}) d\boldsymbol{s} = 1. \tag{21}$$

For details, please refer to the original study and its extensions [20, 34]. Contextual REPS models a policy as a Gaussian distribution

$$\pi(\boldsymbol{\theta}|\boldsymbol{s}) = \mathcal{N}(\boldsymbol{\phi}(\boldsymbol{s})^\top \boldsymbol{w}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}})$$

with a mean vector $\boldsymbol{\mu}_{\boldsymbol{\theta}} = \boldsymbol{\phi}(\boldsymbol{s})^\top \boldsymbol{w}$ that is linear in the context features $\boldsymbol{\phi}(\boldsymbol{s})$. Since robotic grasping is a complex task, we require a policy that is non-linear in the original context $\boldsymbol{s}$. Therefore, we use a squared exponential feature, which is defined by M context samples $\boldsymbol{s}^f$ randomly selected from our dataset, i.e., the i^{th} dimension of $\boldsymbol{\phi}$ is given by

$$\phi_i(\boldsymbol{s}) = \exp\left(-\frac{1}{2}\left(\boldsymbol{s}_i^f - \boldsymbol{s}\right)^\top \boldsymbol{\Lambda}_\phi \left(\boldsymbol{s}_i^f - \boldsymbol{s}\right)\right), \tag{22}$$

where $\boldsymbol{\Lambda}_\phi$ is a diagonal matrix that defines the bandwidth for each element of the context vector $\boldsymbol{s}$. In our grasping framework, $\boldsymbol{s}$ is the local features of a point cloud, including the centroid of the point cloud and a vector normal to the surface approximated from the point cloud. We heuristically set the number of RBF basis function as 20, the bandwidth as 0.1, and $\epsilon = 0.1$. We chose these values based on prior work such as [19]. This nonlinear feature function $\boldsymbol{\phi}(\boldsymbol{s})$ allows us to learn a policy that is nonlinear to the original context.

3.5 *Extension with Additional Metrics for Grasp Selection*

In practice, we can often analytically evaluate a metric of the motion quality before the actual execution. For example, methods for analytically computing the cost of colliding with objects have been developed in the field of trajectory optimization [35, 36]. Although we can learn and estimate the expected return for selecting the grasp type and grasp location using the method described in Section 3.3, we can reduce the learning costs by incorporating such an analytically computable metric. In our framework, we can incorporate additional metrics for the

grasp selection by extending the objective function in (10) as

$$U_{\text{col}}(\boldsymbol{s}, g) = \mathbb{E}[R^*|\pi_l^g, \boldsymbol{s}] + \beta\sigma_R(\pi_l^g, \boldsymbol{s}^g) - c_{\text{add}}(\boldsymbol{\tau}), \tag{23}$$

where $c_{\text{add}}(\boldsymbol{\tau})$ is the additional cost which we would like to incorporate, and $\boldsymbol{\tau}$ is the trajectory to reach the grasp position.

In this work, we employed the cost of colliding with objects $c_{\text{obs}}(\boldsymbol{\tau})$ proposed in [36] as $c_{\text{add}}(\boldsymbol{\tau}) = c_{\text{obs}}(\boldsymbol{\tau})$. We denote by u the index of the body point and by $\boldsymbol{x}_u(t)$ the position of the body point u in task space at time t. The obstacle cost $c_{\text{obs}}(\boldsymbol{\tau})$ can be computed as

$$c_{\text{obs}}(\boldsymbol{\tau}) = \frac{1}{2}\int_0^1 \int_{\mathcal{B}} c\left(\boldsymbol{x}_u(t)\right) \left\| \frac{d}{dt}\boldsymbol{x}_u(t) \right\| du\ dt, \tag{24}$$

where $\mathcal{B}$ is a set of body points which comprise the robot body, and the local collision cost function $c(\boldsymbol{x}_u)$ is defined as

$$c(\boldsymbol{x}_u) = \begin{cases} 0, & \text{if } d(\boldsymbol{x}_u) > \epsilon, \\ \frac{1}{2\epsilon}(d(\boldsymbol{x}_u) - \epsilon)^2, & \text{if } 0 < d(\boldsymbol{x}_u) < \epsilon, \\ d(\boldsymbol{x}_u) + \frac{1}{2}\epsilon, & \text{if } d(\boldsymbol{x}_u) < 0, \end{cases} \tag{25}$$

where ϵ is a constant that scales the margin, and $d(\boldsymbol{x}_u)$ represents a signed distance between the body point u and the nearest obstacle. $d(\boldsymbol{x}_u)$ is negative when the body point is inside obstacles, and zero at the boundary. By considering the collision cost, we can select a grasp location which has less risk of colliding with objects in a given environment. Although we employed the collision cost as the additional metric in our experiments, other metric can be used when available.

4. Experimental Results

We evaluated the performance of our grasp learning framework in simulation and on a real robot platform using rigid and deformable objects.

4.1 *Simulations*

We first evaluate the learning performance of our framework in simulation. The system learned three grasp types with a five-finger hand: precision grasp, power grasp, and medium wrap [7]. In order to initialize the grasping policy, a human operator specified the control parameters to demonstrate each grasp type. For each grasp type, 12 demonstrations were used to initialize the policy. After initializing the grasping policy,the system learned to generalize the control parameters for given objects in different positions. During the learning phase, point clouds of objects were provided to the system, and the system autonomously chose the grasp type and location and executed the grasp using the motion parameters $\boldsymbol{\theta}$. The grasping policy was updated after every grasp execution. We used the model of KUKA Light Weight Robot and DLR/HIT II Hand as a robotic manipulator in the simulation.

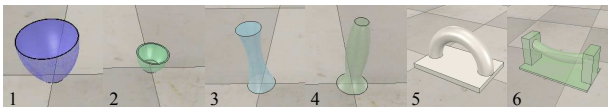
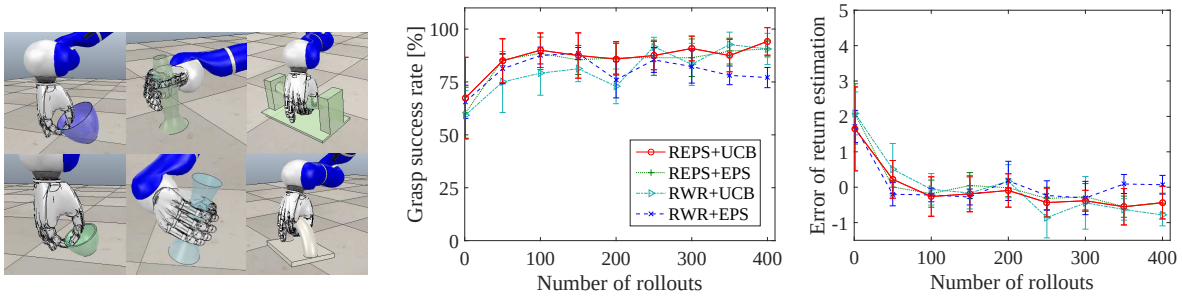


Figure 3. Objects used to learn multiple grasp types: objects 1 and 2 were used for precision grasps, objects 3 and 4 were used for power grasps, and objects 5 and 6 were used for medium-wrap grasps.

The motion parameter $\boldsymbol{\theta}$ of the lower-level policies was defined as

$$\boldsymbol{\theta} = [\boldsymbol{x}_{\text{grasp}}, \boldsymbol{x}_{\text{via}}, \boldsymbol{q}_{\text{grasp}}], \tag{26}$$



(a) (b) (c)

Figure 4. Performance in simulation. (a) Grasps performed in simulation. (b) Improvement in grasp success rate. (c) Improvement in grasp quality estimation.

where $\boldsymbol{x}_{\text{grasp}}$ is the grasp position of the end-effector in task space, $\boldsymbol{x}_{\text{via}}$ is the via point of the end-effector in task space, and $\boldsymbol{q}_{\text{grasp}}$ is a quaternion that represents the orientation of the end-effector in the grasp position. The detailed motions were planned by interpolating the initial robot configuration, pre-grasp configuration, and the grasp configuration with the spline interpolation. The finger configuration was initialized using the human demonstration, and was not included in the motion parameter for the learning phase. We used the contact information of the thumb and index finger of the hand as a context $\boldsymbol{s}$. Therefore, the context vector $\boldsymbol{s}$ had 12 dimensions.

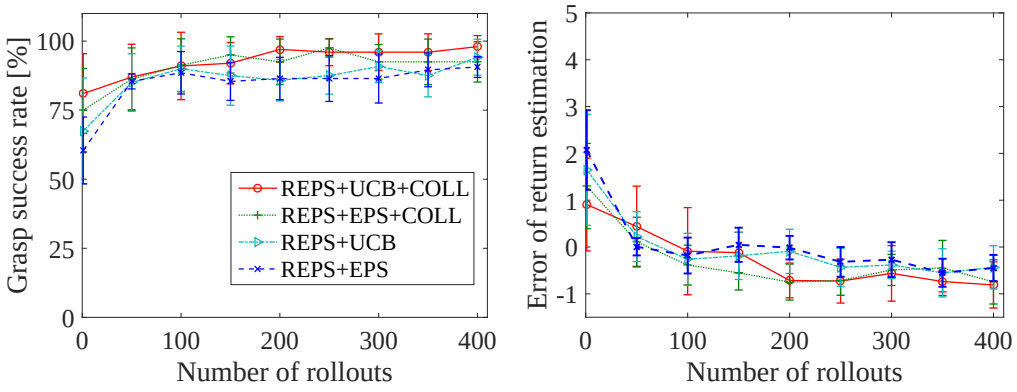
The grasp quality R for each rollout is computed based on the force-closure condition and L^1 grasp quality measure [37–39] as

$$R = c_1 Q + c_2 \delta_{\text{FC}}, \tag{27}$$

where Q is the L^1 grasp quality measure, and δ_{FC} is equal to 1 when the grasp is force-closure and is equal to zero otherwise. The variables c_1 and c_2 are positive constants.

We compared Reward-Weighted-Regression (RWR) algorithm [40] with REPS in learning lower-level policies in the proposed framework. RWR is a policy search method that performs well for real robot tasks, however, it does not constrain the KL divergence in the policy update. Therefore, a comparison between REPS and RWR indicates how the KL bound in the policy update influences the proposed framework. With regard to the upper-level policy, we compared ϵ-greedy policy with UCB [21]. We set $\epsilon = 0.05$ for the ϵ-greedy policy. We used six objects shown in Fig. 3. Objects 1 and 2 were used to demonstrate precision grasps, objects 3 and 4 were used to demonstrate power grasps, and objects 5 and 6 were used to demonstrate medium-wrap grasps. In the learning phase, test objects were randomly selected from these objects. These objects were designed such that the system can grasp them with only one of the three grasp types. Therefore, it was expected that grasping fails when the system chose the wrong grasp type.

Simulation results show that our method is able to learn and improve policies for multiple grasp types through trial and error. Fig. 4(a) shows grasps performed in simulation. In the simulation, a grasping is considered successful when the force closure condition is satisfied. The collision cost is not used in the upper-level policy in the results shown in Fig. 4. The grasp success rate improved through trials from 67.5% at the beginning to 94.1% after 400 trials of grasping (Fig. 4(b)). In addition, the estimation of the grasp quality with GPs improved through trials as shown in Fig. 4(c). The comparison between REPS+EPS and RWR+EPS shows that the KL bound in the policy update leads to better solutions. The differences between REPS+UCB and RWR+UCB and between REPS+EPS and RWR+EPS are statistically significant at the 5% level. The comparison between UCB and the ϵ-greedy implied that UCB can deal with the exploration-exploitation trade-off better than the ϵ-greedy policy in the proposed framework, although the differences between RWR+UCB and RWR+EPS and between REPS+UCB and REPS+EPS were not statistically significant.



(a) (b)

Figure 5. The effect of the collision cost. The performance of the system using the collision cost is shown by "REPS+UCB+COLL" and "REPS+EPS+COLL". (a) Improvement in grasp success rate. (b) Improvement in grasp quality estimation. By using the collision cost in the upper-level policy, a higher success rate can be achieved.

Figure 7. Objects used in the experiment. We used a set of objects such that multiple grasp types were necessary to grasp them.

The effect of the collision cost embedded in the upper-level policy is shown in Fig. 5. By using the collision cost in the upper-level policy, the system achieved a success rate of 98.0% while a success rate was 94.1% without the collision cost. If the collision cost is not explicitly used in the upper-level policy, the system needs to implicitly learn the effect of collisions, which results in slower learning.

4.2 *Transfer to a Real Robot with Unseen Objects*

We tested whether our learned model can be transferred to a real robotic system. The grasping policy was learned through 400 grasp executions in the simulation described in Section 4.1. We used 10 different objects as shown in Fig. 7 for the real robot evaluation. For each object, grasps were tested five times by changing the object position and orientation. The positions of a test object were roughly placed at four corners and a center of a square with 20 cm on a side. We considered that a gasping was successful when the object was lifted up and held for 2 seconds. KUKA Light Weight Robot and DLR/HIT II Hand were used for this experiment. The arm and fingers of the robot were controlled using impedance control.

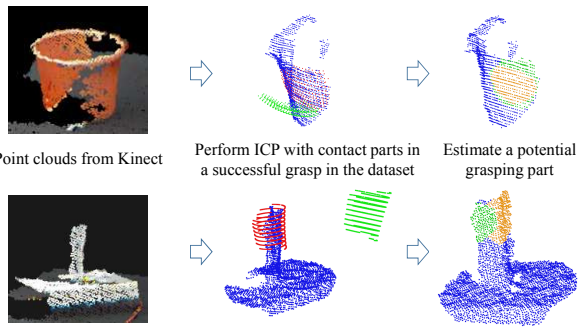


Figure 6. Examples of the process of finding local features that are similar to stored successful grasps. In the middle figures, the green dots represent the contact part in the dataset of successful grasps, and the red dots represent the result of ICP. In the right figures, the green dots represent the estimated contact region of the thumb, and the orange dots represent the estimated contact region of the index finger.

The results of the experiment are summarized in Table 1. The success rate was 90%, and the performed grasps are shown in Figure 8. Figure 6 shows the process for finding potential grasping regions. As shown, the potential grasp locations were estimated with partial point clouds of given

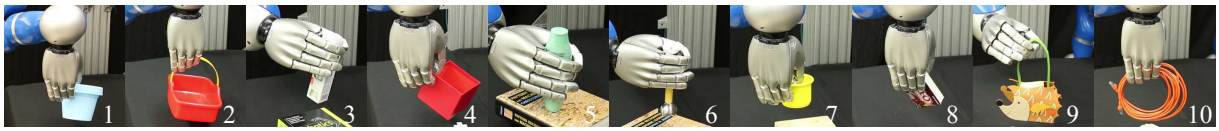


Figure 8. Grasps performed in the experiment. The robot chose the appropriate grasp types and successfully executed the grasps using the given point clouds.

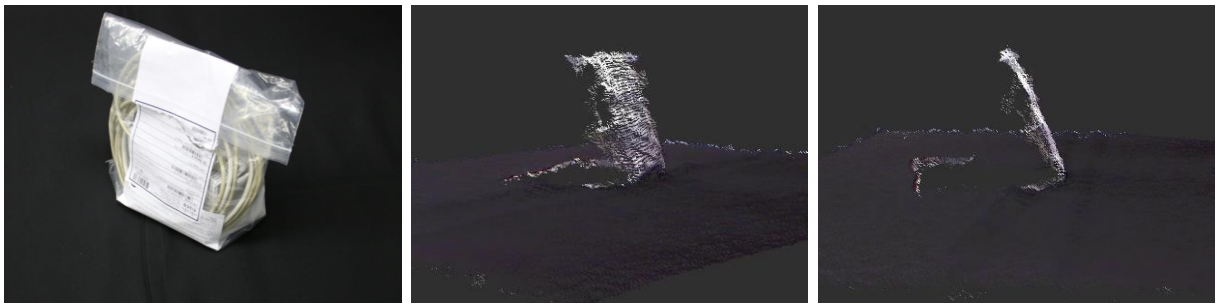


Figure 9. The deformable object used in the experiment. Left: A picture of the slippery and deformable plastic bag of cables used in the experiment in Section 4.3. Middle and right: A point cloud of the object from two different points of view. The point clouds were captured from a single fixed point of view using a Kinect sensor. As shown in the right figure, the backside of the object is not visible.

objects, and the system estimated the contact location for each finger. In addition, as shown in Figures 3 and 7, the shapes of the objects are different from ones used in simulations. Hence, these results show that the our approach can work well with partial point clouds of unseen objects in a real robotic system.

4.3 *Grasp Refinement for Deformable Objects using Human Evaluation*

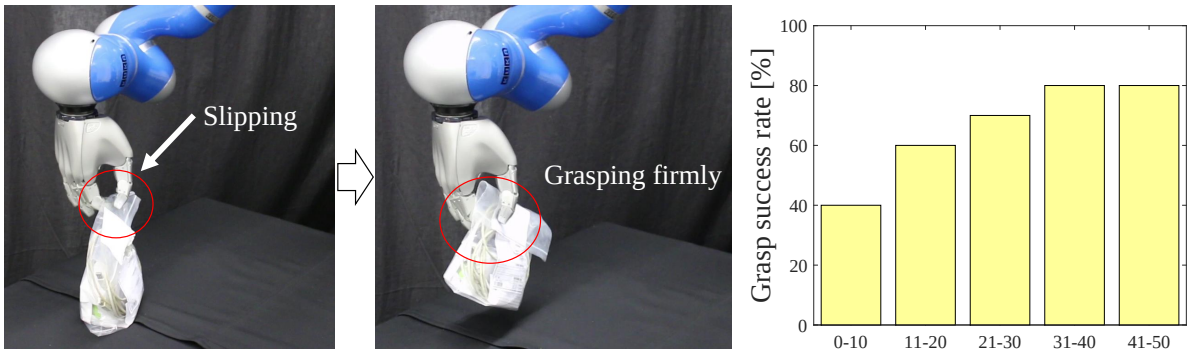
When grasping a slippery and deformable object, collision between the hand and the object is often negligible, while collision is problematic when grasping a rigid object. On the other hand, the object needs to be grasped firmly such that the object is held stably in the hand. Therefore, grasping deformable objects often requires a grasping strategy different from the one for rigid objects.

In this experiment, we trained a grasping policy in simulation using rigid objects and subsequently fine-tuned using a real robot and a deformable object. We used a slippery and deformable plastic bag of cables, which is shown in Figure 9, as a target object. In the additional training using a real robot, we used a binary return based on the evaluation from a human operator instead of analytically computing a grasping quality. The force closure condition is hard to compute in a real robotic system since it is challenging to measure the contact position and the contact force. When the human operator considered a given grasp successful, a constant positive value was given as a return to the system. Otherwise, the return given to the system was zero. if we close the fingers tightly enough, we can avoid the slipping and the grasping rate would be much higher. However, since we were interested in the reduction of the slipping in this experiment, we did not close the finger so tightly.

The experimental results show that our grasping system can learn a policy for grasping a deformable object through trial and error using human evaluations. Figure 10 illustrates our results. In the beginning of learning, the system planned to grasp the top of the object, which

Table 1. Performance of grasping rigid objects with a real robotic system. The object numbers correspond to the numbers in Fig. 8. The grasping policy learned in simulation was successfully transfered to a real robotic system.

Obj. No.	1	2	3	4	5	6	7	8	9	10	Avg.
Success rate	5/5	4/5	5/5	5/5	5/5	5/5	4/5	5/5	4/5	3/5	90.0 %



(a) (b)

Figure 10. Results of grasping a deformable object. The left figure in (a) shows the result of the fourth trial, and the right figure in (a) shows the result of the 30th trial. After learning with a real robot, the deformable object was grasped firmly.

would be a good solution for a rigid object. As a result, the object often slipped from the robot hand. After additional training with a real robot, the system learned to firmly grasp the center of the object as shown in Figure 10(a). Although the robot hand occasionally collided with the object when approaching, the deformable object could still be grasped due to the compliance of the object. The grasp success rate improved from 40% in the first 10 trials to 80% in after 30 trials as shown in Figure 10(b). Although the resulting grasp success rate is not high, we did not close the fingers tightly to observe the slipping of the object. The grasp rate would be higher if we use a standard force control for closing fingers. Our experimental results show that the grasping behavior can be adapted through trial and error with human evaluation in a real robot.

5. Discussion

In hierarchical policy search, learning the upper-level and lower-level policies simultaneously is not trivial in many cases since the behaviors of policies in different layers often influence each other. For instance, the method in [15] learns the state value function $V(\boldsymbol{s})$ for selecting the context. Since the true value of $V(\boldsymbol{s})$ is dependent on $\pi_l(\boldsymbol{\theta}|\boldsymbol{s})$, the estimate of $V(\boldsymbol{s})$ needs to be updated when the lower-level policy $\pi_l(\boldsymbol{\theta}|\boldsymbol{s})$ is updated. On the contrary, we approximate the return function $R(\boldsymbol{\theta}, \boldsymbol{s})$, which is independent of the lower-level policies $\pi_l(\boldsymbol{\theta}|\boldsymbol{s})$. The estimation of the return function using GPs steadily improves as the system increases the number of data samples. Using GPs, we can analytically approximate the expected return for each lower-level policy and potential grasp locations. When the upper-level policy has learned to select a grasp type for a given context, the rest of the process is a standard policy search problem since each lower-level policy is learned independently in our framework. Thus, the policy updates of the lower-level policies are expected to be stable. Although the independence of lower-level policies simplifies the problem, such learning does not exploit the data obtained from different grasp types. Transferring the policies between different grasp types may lead to a more efficient learning method in future work. In addition, the number of the grasp types needs to be specified in our framework. Recent studies on hierarchical reinforcement learning [19, 41] have proposed methods for automatically optimizing the number of lower-level policies. Automatic identification of the necessary grasp types is also interesting research direction.

With regard to finding grasp affordances, although we used the method described in Section 3.2 to find potential grasp parts in point clouds, our hierarchical learning algorithm is not limited to specific methods for finding grasp affordances. Therefore, the alternative methods such as [11, 13] can also be used to find grasp affordances.

Experimental results show that our framework can learn multiple grasp types and a policy to select them according to the given objects. However, it is often necessary to select the grasp type

and location on the basis of additional factors, such as human preferences and the tasks planned to be performed after grasping. Although we computed the return function based on the grasp stability in this study, our framework can be easily extended to other grasp quality metrics. For example, learning the return function from human relative feedback using a GP in [42–44] can also be used. In future work, such additional factors for the selection of grasp types and locations should be considered. Likewise, although we employed the simple parameterization of the grasping motion in this work, we can employ more sophisticated trajectory parameterization such as DMP. For details, please refer to a survey on imitation learning such as [45].

6. Conclusions

We presented an hierarchical reinforcement learning algorithm for learning multiple grasping strategies. In our framework, the lower-level policies learn to plan grasping motions corresponding to individual grasp types, and the upper-level policy learns to select the grasp type and grasp location. Our system autonomously constructs a database of grasping motions and corresponding objects and learns the grasping policy through trial and error. The developed system was tested with both simulation and a real robotic system. The experimental results show that our approach can autonomously learn multiple grasp types and that the grasping policy learned in simulation can be successfully transferred to a real robotic system. In addition, the grasping policy trained for rigid objects in simulation can be refined to a deformable object through trial and error using a real robot. In future work, we will extend our framework by combining the approach of learning from human preference.

Acknowledgement

This work has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement #645582 (RoMaNS). T. Osa was supported by JSPS KAKENHI 17H00757.

References

[1] Bicchi A, Kumar V. Robotic grasping and contact: a review. In: IEEE international conference on robotics and automation (ICRA). 2000. p. 348–353.
[2] Bohg J, Morales A, Asfour T, Kragic D. Data-driven grasp synthesis- a survey. IEEE Transactions on Robotics. 2014 April;30(2):289–309.
[3] Goldfeder C, Allen PK. Data-driven grasping. Autonomous Robots. 2011;31:1–20.
[4] Fischinger D, Weiss A, Vincze M. Learning grasps with topographic features. International Journal of Robotics Research. 2015;.
[5] Pinto L, Gupta A. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In: Proceedings of the IEEE international conference on robotics and automation (ICRA). 2016.
[6] Levine S, Pastor P, Krizhevsky A, Quillen D. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. In: Proceedings of international symposium on experimental robotics (ISER). 2016.
[7] Cutkosky MR, Howe RD. Human grasp choice and robotic grasp analysis. In: Venkataraman ST, Iberall T, editors. Dextrous robot hands. Springer-Verlag New York, Inc.. 1990. p. 5–31.
[8] Osa T, Peters J, Neumann G. Experiments with hierarchical reinforcement learning of multiple grasping policies. In: Proceedings of the international symposium on experimental robotics (ISER). 2016.
[9] Kopicki M, Detry R, Adjigble M, Stolkin R, Leonardis A, Wyatt JL. One-shot learning and generation of dexterous grasps for novel objects. International Journal of Robotics Research. 2015;35(8):959 – 976.

[10] Lenz I, Lee H, Saxena A. Deep learning for detecting robotic grasps. International Journal of Robotics Research. 2015;34(4-5):705 – 724.
[11] ten Pas A, Platt R. Localizing handle-like grasp affordances in 3d point clouds. In: Proceedings of the international symposium on experimental robotics (ISER). 2014.
[12] Gualtieri M, Ten Pas A, Saenko K, Platt R. Using geometry to detect grasp poses in 3d point clouds. In: Proceedings of the international symposium on robotics research (ISRR). 2015.
[13] ten Pas A, Gualtieri M, Saenko K, Platt R. Grasp pose detection in point clouds. The International Journal of Robotics Research. 2017;36(13-14):1455 – 1473.
[14] Deisenroth MP, Neumann G, Peters J. A survey on policy search for robotics. Foundations and Trends in Robotics. 2013;2(1-2):1–142.
[15] Fabisch A, Metzen JH. Active contextual policy search. Journal of Machine Learning Research. 2014; 15:3371–3399.
[16] Bacon PL, Harb J, Precup D. The option-critic architecture. In: Proceedings of the AAAI conference on artificial intelligence (AAAI). 2017.
[17] Florensa C, Duan Y, Abbeel P. Stochastic neural networks for hierarchical reinforcement learning. In: Proceedings of the international conference on learning representations (ICLR). 2017.
[18] Vezhnevets AS, Osindero S, Schaul T, Heess N, Jaderberg M, Silver D, Kavukcuoglu K. FeUdal networks for hierarchical reinforcement learning. In: Proceedings of the international conference on machine learning (ICML). 2017.
[19] Daniel C, Neumann G, Kroemer O, Peters J. Hierarchical relative entropy policy search. Journal of Machine Learning Research. 2016;17:1–50.
[20] Peters J, Muelling K, Altun Y. Relative entropy policy search. In: Proceedings of the AAAI conference on artificial intelligence (AAAI). 2010.
[21] Sutton R, Barto A. Reinforcement learning: An introduction. The MIT Press. 1998.
[22] Dietterich TG. Hierarchical reinforcement learning with the MAXQ value function decomposition. Journal of Artificial Intelligence Research. 2000;13:227–303.
[23] Kroemer O, Detry R, Piater J, Peters J. Combining active learning and reactive control for robot grasping. Robotics and Autonomous Systems. 2010;(9):1105–1116.
[24] Ijspeert AJ, Nakanishi J, Schaal S. Learning attractor landscapes for learning motor primitives. In: Advances in neural information processing systems (nips). 2002.
[25] Besl PJ, McKay ND. A method for registration of 3-d shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence. 1992 Feb;14(2):239–256.
[26] Spivak M. A comprehensive introduction to differential geometry, volume 1. Brandeis University. 1970.
[27] Auer P, Cesa-Bianchi N, Fischer P. Finite-time analysis of the multiarmed bandit problem. Machine Learning. 2002;47(2):235–256.
[28] Auer P. Using confidence bounds for exploitation-exploration trade-offs. Journal Machine Learning Research. 2003 Mar;3:397–422.
[29] Srinivas N, Krause A, Kakade SM, Seeger M. Gaussian process optimization in the bandit setting: no regret and experimental design. In: Proceedings of the international conference on international conference on machine learning (ICML). 2010.
[30] Rasmussen CE, Williams CKI. Gaussian processes for machine learning (adaptive computation and machine learning). The MIT Press. 2005.
[31] Girard A, Rasmussen CE, Candela JQ, Murray-Smith R. Gaussian process priors with uncertain inputs – application to multiple-step ahead time series forecasting. In: Advances in neural information processing systems (NIPS). 2002.
[32] Candela JQ, Girard A. Prediction at an uncertain input for gaussian processes and relevance vector machines – application to multiple-step ahead time-series forecasting. Danish Technical University. 2002. Tech Rep.
[33] Deisenroth MP. Efficient reinforcement learning using gaussian processes. Phd thesis. 2010.
[34] Kupcsik A, Deisenroth MP, Peters J, Loh AP, Vadakkepat P, Neumann G. Model-based contextual policy search for data-efficient generalization of robot skills. Artificial Intelligence. 2014;.
[35] Kalakrishnan M, Chitta S, Theodorou E, Pastor P, Schaal S. STOMP: Stochastic trajectory optimization for motion planning. In: Proceedings of the IEEE international conference on robotics and automation (ICRA). 2011 May. p. 4569–4574.
[36] Zucker M, Ratliff N, Dragan A, Pivtoraiko M, Klingensmith M, Dellin C, Bagnell JA, Srinivasa S. CHOMP: Covariant hamiltonian optimization for motion planning. The International Journal of

Robotics Research. 2013;32:1164–1193.
[37] Murray RM, Sastry SS, Zexiang L. A mathematical introduction to robotic manipulation. 1st ed. Boca Raton, FL, USA: CRC Press, Inc.. 1994.
[38] Ferrari C, Canny J. Planning optimal grasps. In: Proceedings of the IEEE international conference on robotics and automation (ICRA). 1992 May. p. 2290–2295 vol.3.
[39] Pokorny F, Kragic D. Classical grasp quality evaluation: New algorithms and theory. In: Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS). 2013 Nov. p. 3493–3500.
[40] Peters J, Schaal S. Reinforcement learning by reward-weighted regression for operational space control. In: Proceedings of the international conference on machine learning (ICML). 2007.
[41] Osa T, Sugiyama M. Hierarchical policy search via return-weighted density estimation. In: Proceedings of the AAAI conference on artificial intelligence (AAAI). 2018.
[42] Daniel C, Kroemer O, Viering M, Metz J, Peters J. Active reward learning with a novel acquisition function. Autonomous Robots. 2015;39(3):389–405.
[43] Kupcsik A, Hsu D, Lee W. Learning dynamic robot-to-human object handover from human feedback. In: Proceedings of international symposium on robotics research (ISRR). 2016.
[44] Pinsler R, Akrour R, Osa T, Peters J, Neumann G. Sample and feedback efficient hierarchical reinforcement learning from human preferences. In: Proceedings of the IEEE international conference on robotics and automation (ICRA). 2018.
[45] Osa T, Pajarinen J, Neumann G, Bagnell JA, Abbeel P, Peters J. An algorithmic perspective on imitation learning. Foundations and Trends in Robotics. 2018;7(1-2):1–179.