Experiments with Hierarchical Reinforcement Learning of Multiple Grasping Policies

Takayuki Osa, Jan Peters, and Gerhard Neumann

Technische Universität Darmstadt,
Hochschulstr. 10, 64289 Darmstadt, Germany
{osa,peters,neumann}@ias.tu-darmstadt.de

Abstract. Robotic grasping has attracted considerable interest, but it still remains a challenging task. The data-driven approach is a promising solution to the robotic grasping problem; this approach leverages a grasp dataset and generalizes grasps for various objects. However, these methods often depend on the quality of the given datasets, which are not trivial to obtain with sufficient quality. Although reinforcement learning approaches have been recently used to achieve autonomous collection of grasp datasets, the existing algorithms are often limited to specific grasp types. In this paper, we present a framework for hierarchical reinforcement learning of grasping policies. In our framework, the lower-level hierarchy learns multiple grasp types, and the upper-level hierarchy learns a policy to select from the learned grasp types according to a point cloud of a new object. Through experiments, we validate that our approach learns grasping by constructing the grasp dataset autonomously. The experimental results show that our approach learns multiple grasping policies and generalizes the learned grasps by using local point cloud information.

Keywords: Grasping, Reinforcement learning, Point clouds

1 Introduction

Grasping is a crucial aspect in robot manipulation, and many approaches to achieve robust and adaptive grasping have been proposed in literature [1, 2]. Despite these efforts, robotic grasping has not achieved human-level performance. Th data-driven approach yields a promising class of grasping methods [2, 3]. These methods leverage grasp datasets and transfer grasping motions to new target objects based on geometric information. Recent data-driven methods generalize grasps to various objects based on point clouds or RGB-D image data [4–8]. However, the performance of these data-driven methods is highly dependent on the quality of the grasp dataset, which is not easy to obtain with sufficient quality. The manual collection of such a grasp dataset can be avoided if a robotic system autonomously collects the dataset by trial and error.

One solution to this problem is to use reinforcement learning [9]. A few methods for autonomous data collection of grasping have been proposed recently

[10, 11]. However, these methods are based on convolutaional neural network (CNN) and limited to 2D image inputs lacking depth information. Consequently, these methods are limited to simple grasping motions, e.g., vertical pinch grasps. However, studies in the area of human grasping indicate that multiple grasp types are necessary in order to achieve dexterous and human-like manipulation [12,13].

In this paper, we present a hierarchical reinforcement learning approach for learning to plan grasping motions based on point clouds. In our approach, the lower-level hierarchy learns multiple grasp types, and the upper-level hierarchy learns a policy to select from the learned grasp types according to the point cloud of the given object. We empirically validate that our approach learns grasping by constructing the grasp dataset autonomously. The experimental results demonstrate that the grasping performance of our approach improves iteratively by updating the grasping policy and the grasp dataset. In addition, from our experiments, we verify that the learned grasping policies can be generalized for various objects by leveraging local features of the given point cloud of the object.

2 Related Work

Data-driven methods have been very popular in the field of robotic grasping. Recent studies demonstrated that grasp planning based on point clouds or RGB-D images can be generalized to various objects without solid 3D models [4–8]. However, the performance of these methods depends significantly on the quality of the training dataset of grasping motions and objects. In addition, these methods using RGB-D data are often limited to simple two-finger grippers and specific approach directions. 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.

Reinforcement learning is a promising approach for autonomous data generation. The study by Kroemer et al. showed that grasping can be learned and improved autonomously using such a reinforcement learning approach [14]. However, the authors did not completely address the problem of generalizing grasps for new scenes. Recent studies have investigated methods for autonomous large-scale data collection for grasp learning [10, 11]. The method presented in [10] showed the feasibility of autonomously collecting a dataset with thousands of grasps and training a CNN to predict grasp locations. Levine et al. proposed the learning of hand-eye coordination for grasping by using CNN [11]. However, the methods in [10,11] are limited to a specific grasp type and 2D image input lacking depth information, although use of depth information and learning multiple grasp types are essential to achieve dexterous manipulations.

In contrast with previous studies, our approach has three important features: 1) learning multiple grasp types, 2) autonomously constructing the grasp dataset through trial and error, and 3) planning the grasping motion based on point

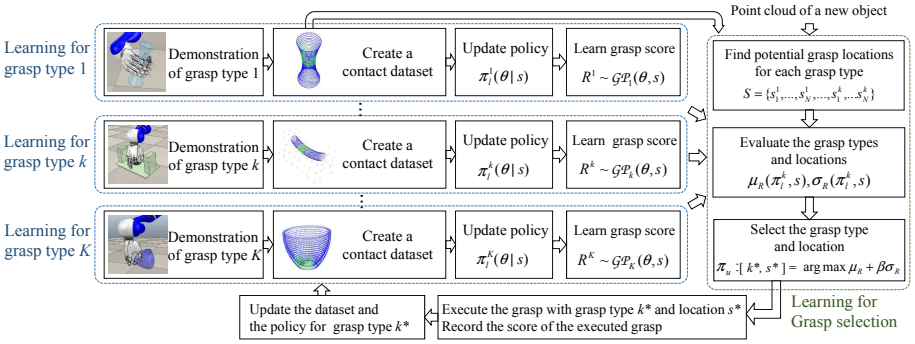


Fig. 1. Overview of the algorithm. First, the grasping policy is initialized, and the dataset of contact information is created based on human demonstration. The individual lower-level policy π_l is learned for each grasp type. The grasp quality R is approximated with Gaussian Processes (GPs). GPs are 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.

clouds. To the best of our knowledge, previous studies have not proposed a learning method that has includes all these features.

3 Learning Multiple Grasping Policies

In order to make our problem tractable, we devide the problem of learning to grasp into four steps: 1) find the potential grasp locations, 2) select the grasp type and location, 3) perform the selected grasp, and 4) update the grasping policy. Our framework is summarized in Fig. 1. The system learns a policy consisting of two layers: the upper-level policy π_u selects the appropriate grasp types and grasp locations, and the lower-level policy π_l maps the desired grasp location to the grasping motion with a specific grasp type.

The grasping policy is initialized using human demonstrations. For each demonstrated grasp, the system stores the contact information of the successful grasp. Simultaneously, the grasping policy, which consists of π_u and π_l, is initialized based on the demonstrated motions. When a new point cloud of the object is given, the system finds multiple local parts similar to the contact parts in the dataset of the successful grasps. Each grasp candidate is provided to the upper-level policy π_u, which selects one candidate for execution.

Grasping motions for different grasp types are learned as independent policies π_l^k by the contextual relative entropy policy search (REPS) algorithm [15–17]. The learned policy π_l^k generates the motion parameter $\boldsymbol{\theta}$ using the local features of the estimated grasping part $\boldsymbol{s}$.

The grasp quality R is approximated using Gaussian Processes (GPs) as a function of motion parameter $\boldsymbol{\theta}$ and the feature of the potential grasping part

$\boldsymbol{s}$. Based on the evaluation with the learned GP models, the upper-level policy π_u selects the appropriate set of the grasp type and location. We use the upper confidence bound (UCB) objective, which is a well-known acquisition function from Bayesian Optimization (BO) [18–20].

After every grasp execution, the GPs are updated, and the estimation of the grasp quality improves iteratively. Simultaneously, the executed lower-level policy π_l is updated using REPS. When the grasping is successful, the contact information is stored in the dataset for the corresponding grasp. Consequently, the dataset containing the contact information of the grasp is constructed autonomously.

3.1 Learning to Select the Grasp Type and the Grasp Location

In order to select the grasp type and the grasp location from the given candidates, we need to evaluate the expected grasp quality $\mathbb{E}[R|\pi_l^k, \boldsymbol{s}]$. We approximate the grasp quality of the kth grasp type with a GP as a function of the movement parameters $\boldsymbol{\theta}$ and the grasp location features $\boldsymbol{s}$, i.e.,

$$R^k(\boldsymbol{\theta}, \boldsymbol{s}) \sim \mathcal{GP}\left(m\left(\boldsymbol{z}\right), g\left(\boldsymbol{z}, \boldsymbol{z}'\right)\right) \tag{1}$$

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

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

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 updated after every rollout by maximizing the marginal log likelihood [21]. We assume zero prior mean, i.e., $m(z) = 0$; therefore, joint distribution of the quality measure $R_{1:N}$ of the training set and the quality measure of a query data point R^* is Gaussian, i.e.,

$$\begin{bmatrix} \boldsymbol{R}_{1:N}^k \\ R^* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \boldsymbol{G}_k & \boldsymbol{g}_k \\ \boldsymbol{g}_k^T & g(\boldsymbol{z}^*, \boldsymbol{z}^*) \end{bmatrix}\right) \tag{3}$$

where $\boldsymbol{G}$ is the Gram matrix and $\boldsymbol{R}_{1:N}^k$ is a column vector that contains rewards of rollouts with the kth grasp type as $\boldsymbol{R}_{1:N}^k = [R_1^k, \cdots, R_N^k]^T$. In this framework, we employ a stochastic policy $\pi_l^k(\boldsymbol{\theta}|\boldsymbol{s}) \sim \mathcal{N}(\boldsymbol{\mu}^k(\boldsymbol{s}), \boldsymbol{\Sigma}^k(\boldsymbol{s}))$. In order to estimate $\mathbb{E}[R|\pi_l^k, \boldsymbol{s}]$ using GPs, we can consider that the inputs of GPs are drawn from the distribution

$$\boldsymbol{z}^* \sim \mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{z}^*}, \boldsymbol{\Sigma}_{\boldsymbol{z}^*}\right) \text{ where } \boldsymbol{\mu}_{\boldsymbol{z}} = \begin{bmatrix} \boldsymbol{\mu}^k(\boldsymbol{s}) \\ \boldsymbol{s} \end{bmatrix}, \boldsymbol{\Sigma}_{\boldsymbol{z}} = \begin{bmatrix} \boldsymbol{\Sigma}^k(\boldsymbol{s}) & 0 \\ 0 & 0 \end{bmatrix}. \tag{4}$$

To estimate the expected reward for grasp type k when given a context $\boldsymbol{s}$, we need to compute the integral

$$p(R^*|\boldsymbol{\mu}_{\boldsymbol{z}^*}, \boldsymbol{\Sigma}_{\boldsymbol{z}^*}) = \int p(R^*|\boldsymbol{z}, D) p(\boldsymbol{z}^*) d\boldsymbol{z}^* \tag{5}$$

Algorithm 1 Learning Multiple Grasp Types

Input: point cloud of the object $\boldsymbol{P}_{\text{target}}$, the number of locations candidates N, the number of grasp types K
Initialization: Initialize policies and GP models based on demonstrations
repeat
  for $k = 1 : K$ **do**
    Find grasp location candidates $\boldsymbol{P}_{\text{target}} \mapsto \{\boldsymbol{p}^1_{\text{grasp}}, \ldots, \boldsymbol{p}^N_{\text{grasp}}\}$
    Compute the features of grasp location candidates $\boldsymbol{S}^k = \{\boldsymbol{s}^k_1, \ldots, \boldsymbol{s}^k_N\}$
    for every grasp location candidate $\boldsymbol{s}^k \in \boldsymbol{S}^k$ **do**
      Compute $\mathbb{E}[R^*|\pi^k_l, \boldsymbol{s}^k]$ and $\sigma(\pi^k_l, \boldsymbol{s}^k)$ using (6) and (7)
    end for
  end for
  Select the grasp type and location using the UCB objective in (10)
  Execute grasp with the grasp parameter $\boldsymbol{\theta} \sim \pi^{k^*}(\boldsymbol{\theta}|\boldsymbol{s}^*)$
  Update the GP model for the k^*th grasp type
  Update the policy for the k^*th grasp type with a policy search method
until grasping learned

where D represents the dataset of motion parameters, contexts, and resulting rewards. The studies in [22,23] showed that the mean and the covariance of this distribution are given by

$$\mu_R = \mathbb{E}[R^*|\pi^k_l, \boldsymbol{s}^k] = \boldsymbol{q}^T\boldsymbol{\beta} \tag{6}$$

$$\sigma_R = g_k\left(\boldsymbol{z}^*, \boldsymbol{z}^*\right) - \boldsymbol{g}_k^T\boldsymbol{G}_k^{-1}\boldsymbol{g}_k + \text{Tr}\left[\boldsymbol{G}_k^{-1}(\boldsymbol{g}_k\boldsymbol{g}_k^T - Q)\right] + \text{Tr}\left[\boldsymbol{\beta}\boldsymbol{\beta}^T(Q - \boldsymbol{q}\boldsymbol{q}^T)\right] \tag{7}$$

where $\boldsymbol{\beta} = \boldsymbol{G}_k^{-1}\boldsymbol{R}^k$, and the vector $\boldsymbol{q}$ and the matrix $\boldsymbol{Q}$ are given by

$$q_j = \frac{\exp\left(-\frac{1}{2}(\boldsymbol{\mu}_{\boldsymbol{z}^*} - \boldsymbol{z}_j)^T(\boldsymbol{\Lambda} + \boldsymbol{\Sigma}_{\boldsymbol{z}^*})^{-1}(\boldsymbol{\mu}_{\boldsymbol{z}^*} - \boldsymbol{z}_j)\right)}{|2\boldsymbol{\Lambda}\boldsymbol{\Sigma}_{\boldsymbol{z}^*} + I|^{1/2}}, \tag{8}$$

$$Q_{ij} = \frac{\exp\left(-\frac{1}{2}\left[(\boldsymbol{\mu}_{\boldsymbol{z}^*} - \boldsymbol{z}_d)^T(\frac{\boldsymbol{\Lambda}}{2} + \boldsymbol{\Sigma}_{\boldsymbol{z}^*})^{-1}(\boldsymbol{\mu}_{\boldsymbol{z}^*} - \boldsymbol{z}_d) + (\boldsymbol{z}_i - \boldsymbol{z}_j)^T(2\boldsymbol{\Lambda})(\boldsymbol{z}_i - \boldsymbol{z}_j)\right]\right)}{|2\boldsymbol{\Lambda}\boldsymbol{\Sigma}_{\boldsymbol{z}^*} + I|^{1/2}}, \tag{9}$$

where $\boldsymbol{z}_d = \frac{1}{2}(\boldsymbol{z}_i + \boldsymbol{z}_j)$, and $\boldsymbol{\Lambda}$ is a diagonal matrix with $\boldsymbol{\Lambda} = l^2\boldsymbol{I}$.

These GP models are used to evaluate the grasp locations found for each grasp type. For grasp selection, we must consider the exploration-exploitation trade-off between gaining more information and maximizing the expected quality of the grasp. Such an exploration-exploitation trade-off is considered by many acquisition functions used in BO. We use the UCB [18] acquisition function, which has been shown to perform well in practice. The learner selects the grasp type and location by maximizing the acquisition function

$$u(\pi^k_l, \boldsymbol{s}^k) = \mathbb{E}[R^*|\pi^k_l, \boldsymbol{s}^k] + \beta\sigma_R(\pi^k_l, \boldsymbol{s}^k), \tag{10}$$

where β is a positive constant that controls the exploration-exploitation trade-off. The algorithm to select the grasp types and locations is summarized in Algorithm 1.

3.2 Finding Potential Grasp Locations

In order to estimate the potential grasp location from a given point cloud, the system searches for local parts that are similar to the point clouds of the contact parts in the library of the successful grasps $\mathcal{D}_{\text{contact}}$. In this process, we use the Iterative Closest Points (ICP) algorithm [24], which finds a homogeneous transformation H_{icp} that minimizes the distance between two point clouds.

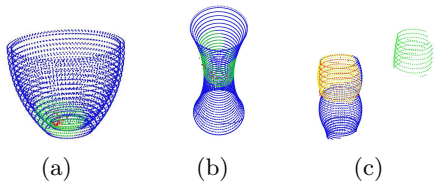


Fig. 2. (a) and (b):Point cloud of object with contact points. Blue points represent the point cloud of the object $\boldsymbol{P}$. Red points represent contact points. Green points represent the neighbors of the contact points $\boldsymbol{C}$. (c)Example of the result of ICP. Blue, green, red, and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

When the point cloud of the target object $\boldsymbol{P}_{\text{target}}$ is given, the system randomly samples a subset of the point cloud of the new object $\boldsymbol{p}_i \subset \boldsymbol{P}_{\text{target}}$. Subsequently, ICP is performed between $\boldsymbol{p}_i$ and each contact parts in our dataset $\boldsymbol{C}_j \in \mathcal{D}_{\text{contact}}$. ICP returns the residual distance d_{icp} between $\boldsymbol{p}_i$ and $\boldsymbol{C}_j$; therefore, we can determine the successful grasp that is the most similar to the sampled part $\boldsymbol{p}_i$ from the dataset. Using the result of ICP with the smallest residual distance d^*_{icp}, we can find the point cloud part that is similar to the grasp part in the dataset. The potential grasp part $\boldsymbol{p}_{\text{grasp}}$ can be estimated as a point cloud in the neighborhood of $H^{j^*}_{\text{icp}}\boldsymbol{C}_{j^*}$ from $\boldsymbol{P}_{\text{target}}$. Fig. 2 shows examples of the contact parts in the dataset and the behavior of ICP. By repeating this local search for different subsets of $\boldsymbol{P}_{\text{target}}$, we can find multiple potential grasp locations. The process to obtain the grasp locations is summarized in the Algorithm 2. Separate datasets of successful grasps are maintained for different grasp types, and this process is performed for each grasp type.

This method does not require the entire point cloud of the target object because it searches for local features of the point cloud. This feature is useful in the planning of grasps in real systems in which complete point clouds of objects are not available.

In order to obtain a concise description of the local point cloud $\boldsymbol{p}_{\text{grasp}}$ 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

$$\boldsymbol{s} = [\boldsymbol{x}^1_{\text{center}}, \boldsymbol{n}^1_{\text{center}}, \ldots, \boldsymbol{x}^f_{\text{center}}, \boldsymbol{n}^f_{\text{center}}], \tag{11}$$

where $\boldsymbol{x}^i_{\text{center}}$ is the center of the contact part of the ith finger, $\boldsymbol{n}^i_{\text{center}}$ is the normal vector at the center of the contact part of the ith finger, and f is the number of fingers of the hand. This local description of contact points $\boldsymbol{s}$ is used as a context in the contextual policy search of the lower-level policies π^k_l.

Algorithm 2 Finding potential grasp locations

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

3.3 Learning the Policy for the Desired Grasp Type

We use the contextual REPS algorithm [15, 17] to learn the lower-level policies $\pi^k_l(\boldsymbol{\theta}|\boldsymbol{s})$ that estimate the parameters $\boldsymbol{\theta}$ of the grasping motions with the given contexts for each grasp type. In policy search, the policy must be updated in order to maximize the expected reward. For stable exploration, the "difference" between the old and new policies is bounded in the policy update of REPS. Therefore, the resulting policy will remain close to the initial policy even if the reward function is multi-modal. 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 uses the KL divergence between the sample distribution $q(\boldsymbol{s}, \boldsymbol{\theta})$ and the updated distribution $\pi(\boldsymbol{\theta}|\boldsymbol{s})\mu_{\boldsymbol{s}}(\boldsymbol{s})$ as a similarity measure in the policy update. $\mu_{\boldsymbol{s}}$ is the distribution of the context. The policy update using contextual REPS is formulated as a constraint optimization problem,

$$\max_{\pi} \int \mu_{\boldsymbol{s}}(\boldsymbol{s}) \int \pi(\boldsymbol{\theta}|\boldsymbol{s}) R(\boldsymbol{\theta}, \boldsymbol{s}) d\boldsymbol{\theta} d\boldsymbol{s} \tag{12}$$

$$\text{s.t. } \epsilon \geq \int \mu_{\boldsymbol{s}}(\boldsymbol{s}) \text{KL}\left(\pi(\boldsymbol{\theta}|\boldsymbol{s}) || q(\boldsymbol{\theta}|\boldsymbol{s})\right) d\boldsymbol{s}, \quad 1 = \int \pi(\boldsymbol{\theta}|\boldsymbol{s}) d\boldsymbol{s} \tag{13}$$

For details, please refer to the original study and its extensions [15, 16]. Contextual REPS models the policy as a Gaussian policy

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

with a mean vector $\boldsymbol{\mu}_{\boldsymbol{\theta}} = \boldsymbol{\phi}(\boldsymbol{s})^T \boldsymbol{w}$ that is linear in the context features $\boldsymbol{\phi}(\boldsymbol{s})$. We require a policy that is non-linear in the original context $\boldsymbol{s}$ because grasping is a complex task. Therefore, we use a squared exponential feature where we select M random samples $\boldsymbol{s}$ from our dataset, i.e., the ith dimension of $\boldsymbol{\phi}$ is given by

$$\phi_i(\boldsymbol{s}) = \exp\left(-\frac{1}{2}\left(\boldsymbol{s}_i - \boldsymbol{s}\right)^T \boldsymbol{\Lambda}_{\phi} \left(\boldsymbol{s}_i - \boldsymbol{s}\right)\right), \tag{14}$$

where $\boldsymbol{\Lambda}_{\phi}$ is a diagonal matrix that defines the bandwidth for each element of the context vector $\boldsymbol{s}$.

4 Experimental Results

4.1 Simulations

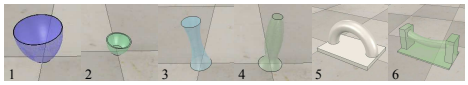


Fig. 3. Objects used to learn multiple grasp types: objects 1 and 2 were used for precision grasp, objects 3 and 4 were used for power grasp, and objects 5 and 6 were used for medium-wrap grasp.

In the simulation experiments, the system learned three grasp types: precision grasp, power grasp, and medium wrap [13]. In order to initialize the grasping policy, a human operator specified the control parameters to demonstrate each type of grasping. For each grasp type, 12 demonstrations were given to the system. Then, the grasping policy was initialized, and 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.

In the simulation, the motion parameter $\boldsymbol{\theta}$ of the lower-level policies was given as

$$\boldsymbol{\theta} = [\boldsymbol{x}_{\text{grasp}}, \boldsymbol{x}_{\text{via}}, \boldsymbol{q}_{\text{grasp}}], \tag{15}$$

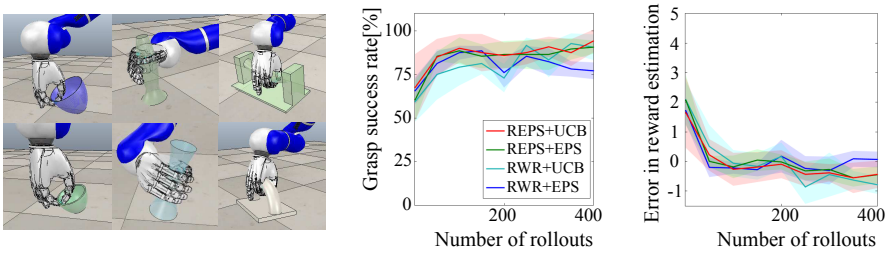
where $\boldsymbol{x}_{\text{grasp}}$ is the grasp position of the end-effector in Cartesian space, $\boldsymbol{x}_{\text{via}}$ is the via point of the end-effector in Cartesian space, and $\boldsymbol{q}_{\text{grasp}}$ is a quaternion that represents the orientation of the end-effector in the grasp position. 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 [25–27] as

$$R = c_1 Q + c_2 \delta_{\text{FC}}, \tag{16}$$

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 two policy search methods in the proposed framework. Although we use REPS for the lower policies to learn multiple grasp types in this study, other policy search methods can be used in the proposed framework. We compared Reward-Weighted-Regression (RWR) algorithm with REPS [28]. 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 the manner in which the KL bound in the policy update influences the proposed framework. With regard



(a) (b) (c)

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

to the upper-level policy, we compared ϵ-greedy policy with UCB [9]. In this simulation, We set $\epsilon = 0.05$.

As shown in Fig. 3, we used six objects. 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 chosen from these objects.

Grasps performed in the simulation are shown in Fig. 4(a). 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 and RWR shows that the KL bound in the policy update enables efficient exploration in the search space. 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.

4.2 Experiments with a Real Robot

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 objects as shown in Fig. 6. For each object, grasps were tested five times by changing the object position and orientation. KUKA Light Weight Robot and DLR/HIT II Hand were used for this experiment. The arm and fingers of the robot were controlled by using impedance control.

The results of the experiment are summarized in Table 1. The success rate was 90%, and the performed grasps are shown in Fig. 7. Fig. 5 shows the steps in the ICP process to find potential grasping parts. Our approach using ICP performed well with partial point clouds obtained from real scenes using the Kinect.

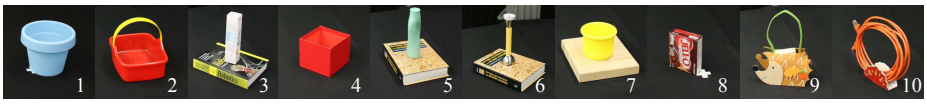


Fig. 6. Objects used in the experiment.

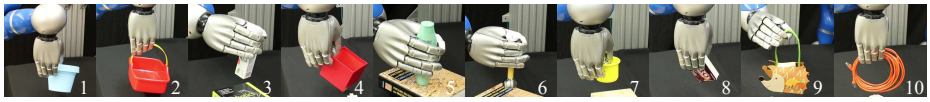


Fig. 7. Grasps performed in the experiment. The robot chose the appropriate grasp types and successfully executed the grasps using the given point clouds.

The shapes of the objects are different from the shapes of the object models used in simulations. Hence, these results show that the learned policy can be used for objects with unseen shapes. In addition, the results demonstrate that the learned policy can successfully plan the grasping motion by using only the partial point clouds of objects.

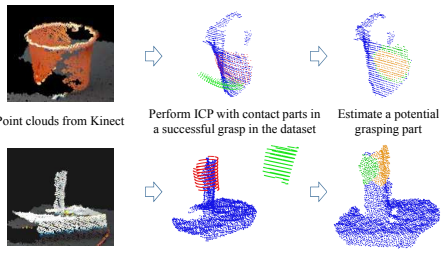


Fig. 5. Examples of the process using ICP to find local features that are similar to stored successful grasps. In the middle figures, the green dots represent the contact part in the dataset of successful grasps, and the red dots represent the result of ICP. In the right figures, the green dots represent the estimated contact part of the thumb, and the orange dots represent the estimated contact part of the index finger.

5 Discussion

In hierarchical policy search, typically, the process of learning the upper-level and lower-level policies simultaneously is not trivial because the behaviors of policies in different layers often influence each other. However, we did not observe undesired behavior in the system. In our framework, $R(\boldsymbol{\theta}, \boldsymbol{s})$ is stationary. Therefore, the estimation using GPs improves as the system performs more rollouts and increases the number of data samples. After the upper-level policy selects the grasp type for the given context, the rest of the process is a standard policy search problem because each lower-level policy is learned independently in our framework. Thus, the behaviors of the lower-level policies are expected to be stable. Although the independence of lower-level policies simplifies the problem, in future work, transferring the policies between different grasp types may lead to a more efficient learning method.

Although we used the method described in Section 3.2 to find potential grasp parts in point clouds, our framework is not limited to specific methods for finding grasp affordances. Therefore, the existing methods such as [7] can be also used to find grasp affordances in our framework.

Table 1. Grasp performance in a real robotic system. The object numbers correspond to the numbers in Fig. 7.

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 %

Experimental results shows that our framework learns multiple grasp types and a policy to select them according to the given objects. However, the grasp types and locations should be selected on the basis on additional factors, such as human preferences and the tasks planned to be performed after grasping. Our framework selects grasps based on only the grasp stability. Therefore, in future work, the selection of grasp types and locations should consider the additional factors.

6 Conclusion

In this paper, we presented a framework for hierarchical reinforcement learning of grasping policies. Our approach autonomously constructs the dataset of grasping motions and point clouds of objects through trial and error. The proposed framework learns multiple grasp types and a policy to select from the learned grasp types for the given objects. In contrast with previous studies, our approach is not limited to specific grasp types and leverages local features of point clouds of objects instead of 2D images. We performed experiments with simulations and with a real robot to test the performance of our approach in learning to grasp with a five-finger hand. The experimental results indicate that our approach learns appropriate grasps by autonomously updating the grasping policy and the grasp dataset. In future work, the selection of grasp candidates based on human preferences and other factors must be investigated.

References

1. Bicchi, A., Kumar, V.: Robotic grasping and contact: a review. In: IEEE International Conference on Robotics and Automation (ICRA). (2000) 348–353
2. Bohg, J., Morales, A., Asfour, T., Kragic, D.: Data-driven grasp synthesis- a survey. IEEE Transactions on Robotics **30**(2) (April 2014) 289–309
3. Goldfeder, C., Allen, P.K.: Data-driven grasping. Autonomous Robots **31** (2011) 1–20
4. Fischinger, D., Weiss, A., Vincze, M.: Learning grasps with topographic features. International Journal of Robotics Research (2015)
5. Kopicki, M., Detry, R., Adjigble, M., Stolkin, R., Leonardis, A., Wyatt, J.L.: One-shot learning and generation of dexterous grasps for novel objects. International Journal of Robotics Research (2015)
6. Lenz, I., Lee, H., Saxena, A.: Deep learning for detecting robotic grasps. International Journal of Robotics Research (2015)
7. Ten Pas, A., Platt, R.: Localizing handle-like grasp affordances in 3d point clouds. In: Int'l Symposium on Experimental Robotics (ISER). (2014)
8. Gualtieri, M., Ten Pas, A., Saenko, K., Platt, R.: Using geometry to detect grasp poses in 3d point clouds. In: Int'l Symposium on Robotics Research (ISRR). (2015)

9. Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. The MIT Press (1998)
10. Pinto, L., Gupta, A.: Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In: IEEE International Conference on Robotics and Automation (ICRA). (2016)
11. Levine, S., Pastor, P., Krizhevsky, A., Quillen, D.: Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. CoRR **abs/1603.02199** (2016)
12. Napier, J.R.: The prehensile movements of the human hand. Journal of Bone and Joint Surgery **38-B**(4) (1956) 902–13
13. Cutkosky, M.R., Howe, R.D.: Human grasp choice and robotic grasp analysis. In Venkataraman, S.T., Iberall, T., eds.: Dextrous Robot Hands. Springer-Verlag New York, Inc. (1990) 5–31
14. Kroemer, O., Detry, R., Piater, J., Peters, J.: Combining active learning and reactive control for robot grasping. Robotics and Autonomous Systems (9) (2010) 1105–1116
15. Peters, J., Muelling, K., Altun, Y.: Relative entropy policy search. In: AAAI Conference on Artificial Intelligence (AAAI). (2010)
16. Kupcsik, A., Deisenroth, M.P., Peters, J., Loh, A.P., Vadakkepat, P., Neumann, G.: Model-based contextual policy search for data-efficient generalization of robot skills. Artificial Intelligence (2014)
17. Deisenroth, M.P., Neumann, G., Peters, J.: A survey on policy search for robotics. Foundations and Trends in Robotics (2013) 388–403
18. Auer, P.: Using confidence bounds for exploitation-exploration trade-offs. J. Mach. Learn. Res. **3** (March 2003) 397–422
19. Srinivas, N., Krause, A., Kakade, S., Seeger, M.: Information-theoretic regret bounds for gaussian process optimization in the bandit setting. IEEE Transactions on Information Theory **58**(5) (May 2012) 3250–3265
20. Calandra, R., Seyfarth, A., Peters, J., Deisenroth, M.P.: Bayesian optimization for learning gaits under uncertainty. Annals of Mathematics and Artificial Intelligence **76**(1) (2016) 5–23
21. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning). The MIT Press (2005)
22. Girard, A., Rasmussen, C.E., Candela, J.Q., Murray-Smith, R.: Gaussian process priors with uncertain inputs – application to multiple-step ahead time series forecasting. In: Advances in Neural Information Processing Systems. (2002)
23. Candela, J.Q., Girard, A.: Prediction at an uncertain input for gaussian processes and relevance vector machines – application to multiple-step ahead time-series forecast- ing. Technical report, Danish Technical University (2002)
24. Besl, P.J., McKay, N.D.: A method for registration of 3-d shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence **14**(2) (Feb 1992) 239–256
25. Murray, R.M., Sastry, S.S., Zexiang, L.: A Mathematical Introduction to Robotic Manipulation. 1st edn. CRC Press, Inc., Boca Raton, FL, USA (1994)
26. Ferrari, C., Canny, J.: Planning optimal grasps. In: IEEE International Conference on Robotics and Automation (ICRA). (May 1992) 2290–2295 vol.3
27. Pokorny, F., Kragic, D.: Classical grasp quality evaluation: New algorithms and theory. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). (Nov 2013) 3493–3500
28. Peters, J., Schaal, S.: Reinforcement learning by reward-weighted regression for operational space control. In: International Conference on Machine Learning (ICML). (2007)