applied sciences

MDPI

Article

# Learning, Improving, and Generalizing Motor Skills for the Peg-in-Hole Tasks Based on Imitation Learning and Self-Learning

**Nam Jun Cho [1], Sang Hyoung Lee [2,*], Jong Bok Kim [1] and Il Hong Suh [1,*]**

[1] Department of Electronics and Computer Engineering, Hanyang University, Seoul 04763, Korea; namjun@hanyang.ac.kr (N.J.C.); ggamir99@hanyang.ac.kr (J.B.K.)

[2] Department of Innovative Smart Manufacturing R&D, Korea Institute of Industrial Technology, Cheonan-si 31056, Korea

* Correspondence: zelog@kitech.re.kr (S.H.L.); ihsuh@hanyang.ac.kr (I.H.S.); Tel.: +82-41-589-8283 (S.H.L.); +82-2-2220-0392 (I.H.S.)

Received: 27 February 2020; Accepted: 7 April 2020; Published: 15 April 2020


**Abstract:** We propose a framework based on imitation learning and self-learning to enable robots to learn, improve, and generalize motor skills. The peg-in-hole task is important in manufacturing assembly work. Two motor skills for the peg-in-hole task are targeted: "hole search" and "peg insertion". The robots learn initial motor skills from human demonstrations and then improve and/or generalize them through reinforcement learning (RL). An initial motor skill is represented as a concatenation of the parameters of a hidden Markov model (HMM) and a dynamic movement primitive (DMP) to classify input signals and generate motion trajectories. Reactions are classified as familiar or unfamiliar (i.e., modeled or not modeled), and initial motor skills are improved to solve familiar reactions and generalized to solve unfamiliar reactions. The proposed framework includes processes, algorithms, and reward functions that can be used for various motor skill types. To evaluate our framework, the motor skills were performed using an actual robotic arm and two reward functions for RL. To verify the learning and improving/generalizing processes, we successfully applied our framework to different shapes of pegs and holes. Moreover, the execution time steps and path optimization of RL were evaluated experimentally.

**Keywords:** peg-in-hole task; reinforcement learning; hidden Markov model; dynamic movement primitive; robot; motor skill

## 1. Introduction

In this paper, we propose a framework for learning, improving, and generalizing the motor skills for a peg-in-hole task. Here, we define a tuple of model parameters that can perform both classification of input signals and generation of appropriate motion trajectories as a motor skill. This task plays an important role in assembly work and is frequently encountered in the manufacturing industry [1]. The peg-in-hole task is often performed in conditions where the exact positions/postures of a hole or peg are unknown due to the errors in vision sensors and robot actuators. To solve this problem, robots need to continuously perform the repetition of reaction classification and reaction generation, while the peg and hole maintain contact until their task completion. Therefore, robots need to possess the abilities to classify reaction force/moment types and generate their corresponding reaction motion trajectories in real-time. In this paper, we focus on obtaining these optimal motor skills for the peg-in-hole task.

Numerous researchers have proposed various approaches for obtaining motor skills from human demonstrations [2–4]. This type of approach is an effective way for robots to capture the characteristics

of motor skills. However, for acquiring complete motor skills, it has one evident limitation: it does not ensure that robots acquire motor skills that are optimized for their goals (that is, it generally provides near-optimal solutions) [5]. Furthermore, it is not easy for human performers to provide a demonstration dataset that can cover all situations arising during the execution of a motor skill [6]. Nonetheless, these human demonstrations can be used as a solid starting point for robots to acquire motor skills [7]. To obtain optimal motor skills, robots must be able to improve motor skills through self-learning. However, this self-learning is a time-consuming and expensive process in the absence of references. We attempt to obtain these optimal solutions with fewer trials-and-errors by providing near-optimal solutions learned from human demonstrations. In this paper, robots improve motor skills to optimize them—referred to as improvement—and generalize them so that they are widely applicable—known as generalization—through self-learning.

The peg-in-hole task has also been addressed through several imitation learning studies [8,9]. However, the peg-in-hole task is not easy to learn with this method alone. The main reason is that it is difficult for human performers to provide a complete demonstration dataset to robots, because it is not feasible to prepare all possible reaction situations. In addition, unintended reaction information may be included in the dataset during the demonstration process. These problems may prevent robots from acquiring the complete motor skills. Thus, initial motor skills are learned to classify reaction force/moment signals and generate reaction motion trajectories from human demonstrations, and their parameters are improved and/or generalized during the iterations of the improvement/generalization processes [10]. The motor skills learned from human demonstrations are referred to as initial motor skills. This is because the parameters are initially used to improve and/or generalize them during self-learning. These combination of imitation learning and self-learning allows robots to reduce trial-and-error iterations to obtain optimal solutions for the peg-in-hole task.

The main contribution of this paper is to propose a framework that enables robots to learn, improve, and generalize motor skills for the peg-in-hole task using a mixture of imitation learning and self-learning. In particular, the motor skills required for dealing with a peg and hole with a specific shape are improved so that they can be generalized to pegs and holes of other shapes. To achieve this, the following fundamental approaches are used:

1. A general method is proposed that concatenates the parameters of two different models, one for classification (hidden Markov models (HMMs)) and one for motion generation (dynamic movement primitives (DMPs)), from human demonstrations. Robots are then able to select an appropriate motor skill from a library of motor skills. This method is used to classify various types of reaction force/moment signals and generate their corresponding reaction motion trajectories for the peg-in-hole task.
2. The policy learning by weighting exploration with the returns (PoWER) algorithm is used in the reinforcement learning (RL) process. Using this algorithm, the RL process improves and/or generalizes motor skills. It not only optimizes the parameters to reduce the execution time step and improve path of a DMP, but also re-estimates the parameters of its corresponding HMM to improve the motor skill. Furthermore, the RL process estimates new targets and initial parameters of new motor skills for generalization.

The studies for obtaining motor skills can be broadly divided into three types: (i) predefined strategy; (ii) imitation learning; and (iii) RL methods. In Method (i), motor skills can be used immediately without any training costs. However, it is not easy to manually design motor skills and ensure their optimal solutions. Numerous researchers, such as Xu et al. [11], Park et al. [12], Zhang et al. [13], and Jokesch et al. [14], have proposed methods for solving the peg-in-hole problem, based on this method. These studies are not learning-based approaches; instead, they use predesigned strategies after analyzing the peg-in-hole task. In Method (ii), those can be learned using a few human demonstrations. However, it provides near-optimal solutions, not optimal ones. Calinon et al. [15], Ude et al. [16], and Kyrarini et al. [17] proposed methods to obtain motor skills based on imitation

learning. Their motor skills are modeled from human demonstration dataset. In contrast, motor skills can be optimized based on the RL process, but a large trial-and-error iterations are required to obtain their optimal solutions in Method (iii). Yun [18] and Inoue et al. [19] proposed methods for learning the peg-in-hole task based on RL. Their aim was to enable robots to learn the motor skills for the peg-in-hole task through random explorations without human demonstration.

To reduce training costs and obtain optimal solutions, many researchers have proposed ways to combine both imitation learning and RL process. These methods are able to learn the optimal solutions from initial motor skills learned by imitation learning through several trials-and-errors of the RL process. Various researchers have proposed methods to use imitation learning and RL. Kober et al. obtained DMPs based on imitation learning and improved their parameters using RL [20]. They dealt with the tasks of placing a ball into a cup and paddling a ball. Kormushev et al. used DMP-like dynamic models based on imitation learning and RL to enable a robot to learn and improve a pancake-flipping task [21]. Kroemer et al. used imitation learning and RL to divide a task into multiple phases and enable a robot to learn their transitions and motor skills [22]. The robots performed a bimanual grasping task with contact force. Levine et al. trained a neural network using a policy parameter optimization method—referred to as guided policy searching—to optimize the initial model parameters [23]. They verified their method for various tasks, including the peg-in-hole task, in virtual environments. None of these researchers considered including classification functions in their models for multiple motor skills. They only focused on improving model parameters through imitation learning and RL. This is slightly different from the definition of our motor skills. Moreover, they did not explicitly consider the generalization of motor skills, or both time step and path optimization.

The remainder of this paper is organized as follows. Section 2 presents the details of the proposed framework. We describe how we represent a parameter tuple by concatenating the parameters of two different models for classification and motion generation. We present the slightly modified PoWER algorithms as well as a general reward function to reduce the number of execution time steps, optimize/generalize motion trajectories, and estimate new targets for new motor skills during the improvement and generalization processes. Section 3 presents the experimental results obtained for the peg-in-hole task. Furthermore, two reward functions are specified for the peg-in-hole task from the general reward function in this section. Section 4 discusses the proposed framework and fundamental techniques. In this section, we provide a guideline for applying our RL algorithm to other tasks. Finally, in Section 5, we present the conclusions of this study and directions for future research.

## 2. Learning, Improving, and Generalizing Motor Skills Based on Imitation Learning and Self-Learning

Figure 1 illustrates the overall process of our framework for learning, improving, and generalizing motor skills based on a mixture of imitation learning and self-learning. The framework consists of three processes: learning initial motor skills from human demonstrations, improving (initial) motor skills through RL, and generalizing motor skills (i.e., adding new motor skills) through RL. Here, the RL process is referred to as self-learning process from the viewpoint that a robot itself can determine and perform the process of improving and generalizing motor skills. As mentioned above, a motor skill is represented by concatenating the parameters of a HMM and a DMP. A threshold model (TM) is used to distinguish whether they are familiar (i.e., modeled) or unfamiliar (i.e., not modeled) with respect to existing HMMs [24]. An existing motor skill is optimized using the improvement process when its HMM likelihood is higher than those of both the other HMMs and the TM. When the TM likelihood is higher than the likelihood of all the other HMMs, a new motor skill is created from a similar motor skill belonging to the HMM with the next highest likelihood. The details are presented in Section 2.2.

### 2.1. Learning (Initial) Motor Skills through Imitation Learning

In this process, a human provides a demonstration dataset for learning initial motor skills for classification and motion generation; after this process, a robot can generate appropriate motion

trajectories given its current situation. In this work, classification is also used to select a suitable motor skill from a library. The specific motor skill obtained by the classification result is used to generate motion trajectories.

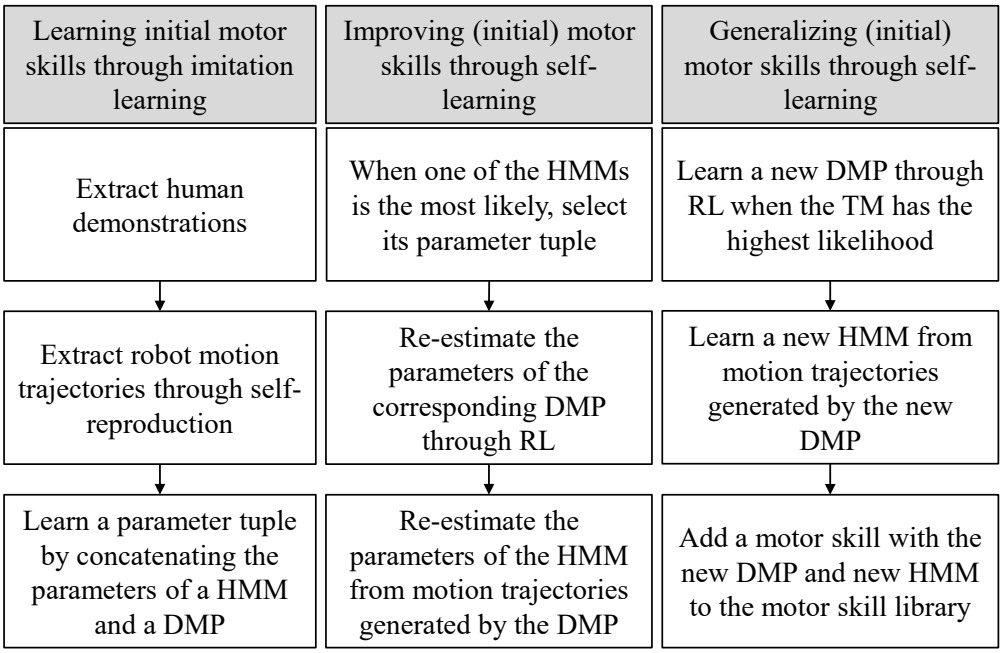


**Figure 1.** The proposed framework consists of three processes: (**left**) learning initial motor skills through imitation learning; (**middle**) improving the (initial) motor skills through RL; and (**right**) generalizing them with RL.

To achieve this, a motor skill is represented as a parameter tuple that concatenates the parameters of a HMM and a DMP. A HMM is a model that is suitable for classifying time-varying signals. The parameters of a HMM are defined as $\lambda = \{\pi_i, a_{ij}, b_i\}_{i,j=1}^{K}$, where $\pi_i$, $a_{ij}$, and $b_i$ denote the initial probability distribution of the $i$th hidden state, the transition probability distribution from the $i$th hidden state to the $j$th hidden state, and the observation probability distribution of $i$th hidden state, respectively. In this case, the number of hidden states $K$ is determined using the Bayesian information criterion [25]. Moreover, parameter $b_i$ is modeled as a Gaussian mixture model to represent continuous non-linear trajectories. The parameters of a HMM are estimated by employing the Baum–Welch algorithm using training data $\{\mathbf{X}_n^m\}$, with $m = 1, 2, ..., M$ and $n = 1, 2, ..., N$, where $M$ and $N$ denote the total number of demonstrations and the number of data points for classification, respectively [26].

A DMP is similar to a linear spring–damper system, which is dependent on the external force term and ensures convergence to the final goal (or target) [27]. The DMP is defined as

$$\dot{\mathbf{V}} = K(\mathbf{X}^g - \mathbf{X}) - D\mathbf{V} + (\mathbf{X}^g - \mathbf{X}^0)\zeta \tag{1}$$

and

$$\tau\dot{\mathbf{X}} = \mathbf{V}, \tag{2}$$

where $\mathbf{X}$, $\mathbf{V}$, $\mathbf{X}^0$, and $\mathbf{X}^g$ represent the position, velocity, initial position, and target position, respectively. All these variables are multidimensional vectors. Moreover, $\tau$, $K$, and $D$ indicate the constants for adjusting the time-scale, spring, and damping terms. The external force term $\zeta$ must be learned from demonstration dataset, and is defined using $s$, which is expressed as

$$\zeta(s) = \frac{\sum_{i=1}^{L} \omega_i \psi_i(s) s}{\sum_{i=1}^{L} \psi_i(s)}, \tag{3}$$

where $\psi_i(s) = exp(-h_i(s-c_i)^2)$ is a Gaussian basis function with center $c_i$ and width $h_i$. Parameter $\omega_i$ is its weighting value. Parameter $L$ represents the number of Gaussian basis functions. Term $\zeta$ is directly dependent on the phase variable $s$, which is monotonically reduced from 1 to 0, independent of time, and is obtained by

$$\tau \dot{s} = -\alpha s, \tag{4}$$

where $\alpha$ is a predefined constant, and this differential equation is referred to as a canonical system. To learn a DMP from demonstrations, the robot learns the average path of several demonstrations. The average path $\mathbf{X}(t)$ is recorded and its derivatives $\mathbf{V}(t)$ and $\dot{\mathbf{V}}(t)$ are computed for each time step $t = 0, ..., T$. Next, the canonical system $s(t)$ is computed for an appropriately adjusted temporal scaling parameter $\tau$. Based on Equations (1) and (2), $\zeta_{\text{target}}(s)$ is computed according to

$$\zeta_{\text{target}}(s) = \frac{-K(\mathbf{X}^g - \mathbf{X}) + \tau D\dot{\mathbf{X}} + \tau\ddot{\mathbf{X}}}{\mathbf{X}^g - \mathbf{X}^0}, \tag{5}$$

where $\mathbf{X}^0$ and $\mathbf{X}^g$ are set to $\mathbf{X}(0)$ and $\mathbf{X}(T)$, respectively. Estimating $\zeta$ is a linear regression problem, which is solved by estimating $\omega_i$ in Equation (3) using the errors as the training data $\{\mathbf{X}_n^m\}$, with $m = 1, 2, ..., M$ and $n = 1, 2, ..., N$, to minimize the error criterion $J = \sum_s (\zeta_{\text{target}}(s) - \zeta(s))^2$ for motion generation.

Finally, using these two types of parameters, the parameter tuple of an initial motor skill is concatenated as

$$\mathbf{\Theta} = \{\boldsymbol{\lambda} = \{\pi_i, a_{ij}, b_i\}_{i,j=1}^K, \mathbf{\Omega} = \{\omega_i\}_{i=1}^L, \mathbf{X}^g, T\}, \tag{6}$$

where $\boldsymbol{\lambda}$ and $\mathbf{\Omega}$ indicate the parameters of a HMM and a DMP, respectively. In this tuple, the target $\mathbf{X}^g$ and total length of policy $T$ are additionally inserted to optimize/generalize the parameters of motor skills in the RL process. As mentioned above, the parameters of the HMM are used to estimate the likelihood under the current situation to select an appropriate DMP from a library of DMPs. Moreover, the parameters of the DMP associated with the appropriate HMM are used to generate motion trajectories. For reference, target $\mathbf{X}^g$ is provided by an external (vision) sensor at execution time, and it tends to be expressed as relative information between a robot and a target. The DMP generates a motion trajectory to reach target $\mathbf{X}^g$ during the length of policy $T$.

### 2.2. Improving and Generalizing Motor Skills through RL

The learned motor skills are improved or generalized through RL using the Algorithm 1. In this algorithm, existing motor skills are optimized in the improvement process and new motor skills are added to the library in the generalization process. To distinguish the improvement and generalization processes, a TM is generated from the HMMs. This TM is used to calculate a threshold based on the likelihood of the input signals [28,29]. As mentioned above, this is used to distinguish whether the input signals are familiar or unfamiliar. We consider the current input signals to be unfamiliar when the likelihood of the TM is higher than the likelihoods of all other HMMs. In contrast, the current input signals are familiar when the likelihood of one HMM is higher than that of the TM. This TM is created by fully connecting all hidden states in all the HMMs. Next, the observation probability distributions of the hidden states are used without any modification, and their transition probability distributions are uniformly assigned for all connections. The details of the TM can be found in [24]. Here, the likelihood of the TM is denoted by $L_{TM}$.

In Algorithm 1, all parameters $\mathbf{\Theta}$ of the motor skills and parameter $\boldsymbol{\lambda}_{TM}$ of the TM are used to identify the improvement and generalization processes. First, the improvement process is performed to optimize the parameters ($\mathbf{\Theta}^*$) of the corresponding motor skill using the iPoWER algorithm (i.e., the PoWER algorithm for improving motor skills; Algorithm 2) when the likelihood of one HMM is higher than those of the HMMs and TM. The iPoWER algorithm optimizes the all parameters for the

existing motor skills except for their targets. In contrast, the generalization process is used to learn the parameters of new motor skills using the gPoWER algorithm (i.e., the PoWER for generalizing motor skills; Algorithm 3) when the likelihood of the TM is higher than those of all the HMMs. The gPoWER algorithm can estimate the parameters ($\boldsymbol{\Theta}_{new}$) of new motor skills including their new targets. Here, new targets are the configurations of robots optimized by their reward functions. These new motor skills are added into the library of motor skills, after which they can be improved as $\boldsymbol{\Theta}^*_{new}$ in the improvement process according to the classification results.

**Algorithm 1** Overall algorithm for improving and/or generalizing motor skills

1: Input: a set of initial parameters $\boldsymbol{\Theta} = \{\boldsymbol{\Theta}_1, \boldsymbol{\Theta}_2, ..., \boldsymbol{\Theta}_N\}$ of all motor skills and the TM $\lambda_{TM}$.
**(Improvement)**
2: **if** $L_{TM} \leq L_{HMMs}$ **then**
3: Using initial parameters $\boldsymbol{\Theta}_i = \{\boldsymbol{\lambda}_i, \boldsymbol{\Omega}_i, \mathbf{X}^{\mathrm{g}}_i, T_i\}$ of a motor skill belonging to the HMM with the maximum likelihood,
4: $\boldsymbol{\Theta}^*_i$ = **iPoWER**$(\boldsymbol{\lambda}_i, \boldsymbol{\Omega}_i, \mathbf{X}^{\mathrm{g}}_i, T_i)$.
**(Generalization)**
5: **else if** $L_{TM} > L_{HMMs}$ **then**
6: Using initial parameters $\boldsymbol{\Theta}_i = \{\boldsymbol{\lambda}_i, \boldsymbol{\Omega}_i, \mathbf{X}^{\mathrm{g}}_i, T_i\}$ of a motor skill belonging to the HMM with the maximum likelihood except for the TM
7: $\boldsymbol{\Theta}_{new}$ = **gPoWER**$(\boldsymbol{\lambda}_i, \boldsymbol{\Omega}_i, \mathbf{X}^{\mathrm{g}}_i, T_i)$.
8: Add the parameters of a new motor skill with $\boldsymbol{\Theta}_{new}$ to the motor skill library.
9: **end if**
10: Output: the parameters $\boldsymbol{\Theta}^*_i or \boldsymbol{\Theta}_{new}$ of the (new) motor skill as well as the parameters $\bar{\boldsymbol{\lambda}}_{TM}$ of updated TM.

**Algorithm 2** iPoWER algorithm for improving the parameters of motor skills considering execution time step and path optimization

1: Input: initial parameters $\boldsymbol{\Theta}_0 = \{\boldsymbol{\lambda}_0, \boldsymbol{\Omega}_0, \mathbf{X}^{\mathrm{g}}, T_0\}$ of a motor skill
2: Set $\boldsymbol{\Theta}_k = \boldsymbol{\Theta}_0$
3: **while** true **do**
4: Sampling: Using $\boldsymbol{\Omega}_k, \mathbf{X}^{\mathrm{g}}$, and $T_k$, generate rollout ($\mathbf{X}$) from $\mathbf{a} = (\boldsymbol{\Omega}_k + \varepsilon_t)^T \boldsymbol{\Psi}(\mathbf{X}, t)$ based on Equation (1)
with exploration $\lfloor \varepsilon_t \rfloor_{ij} \sim N(0, \sigma^2_{ij})$ as a stochastic policy.
5: **if** $\mathbf{X}_t = \mathbf{X}^{\mathrm{g}}$ and $t < T_k$ **then**
6: Set $\tilde{T} = t$ and collect all information $(t, \mathbf{X}_t, \mathbf{a}_t, \mathbf{X}_{t+1}, \varepsilon_t, r_{t+1})$ for $t = \{1, 2, ..., \tilde{T} + 1\}$.
7: **else if** $\mathbf{X}_t = \mathbf{X}^{\mathrm{g}}$ and $t \geq T_k$ **then**
8: Set $\tilde{T} = T_k$ and collect all information $(t, \mathbf{X}_t, \mathbf{a}_t, \mathbf{X}_{t+1}, \varepsilon_t, r_{t+1})$ for $t = \{1, 2, ..., \tilde{T} + 1\}$.
9: **else**
10: Discard all information $(t, \mathbf{X}_t, \mathbf{a}_t, \mathbf{X}_{t+1}, \varepsilon_t, r_{t+1})$ for $t = \{1, 2, ..., \tilde{T} + 1\}$.
11: **end if**
12: Estimating: Use unbiased estimate of the value function $\hat{Q}^{\pi}(\mathbf{X}, \mathbf{a}, t) = \sum_{\tilde{t}}^{\tilde{T}} r(\mathbf{X}_{\tilde{t}}, \mathbf{a}_{\tilde{t}} \mathbf{X}_{\tilde{t}+1}, \tilde{t})$.
13: Reweighting: Reweight rollouts and discard low-reward rollouts.
14: Update the parameters of DMP using
$\boldsymbol{\Omega}_{k+1} = \boldsymbol{\Omega}_k + \left\langle \sum_{t=1}^{\tilde{T}} \varepsilon_t Q^{\pi}(\mathbf{X}, \mathbf{a}, t) / \left\langle \sum_{t=1}^{\tilde{T}} \varepsilon_t Q^{\pi}(\mathbf{X}, \mathbf{a}, t) \right\rangle \right\rangle$.
15: Update the parameters $\boldsymbol{\lambda}_{k+1}$ of the HMM using reaction force/moment recorded during the motion generation $\mathbf{a} = (\boldsymbol{\Omega}_{k+1} + \varepsilon_t)^T \boldsymbol{\Psi}(\mathbf{X}, t)$.
16: Update $T_{k+1} = \tilde{T}$.
17: **if** $\boldsymbol{\Theta}_{k+1} \approx \boldsymbol{\Theta}_k$ **then**
18: **break**
19: **end if**
20: **end while**
21: Output: the optimized parameters $\boldsymbol{\Theta}^* = \{\boldsymbol{\lambda}_{k+1}, \boldsymbol{\Omega}_{k+1}, \mathbf{X}^{\mathrm{g}}, T_{k+1}\}$.

**Algorithm 3** gPoWER algorithm for generalizing motor skills to add new motor skills and new targets

1: Input: initial parameters $\boldsymbol{\Theta}_0 = \{\boldsymbol{\lambda}_0, \boldsymbol{\Omega}_0, \mathbf{X}^{\mathrm{g}}, T_0\}$ of a motor skill
2: Set $\boldsymbol{\Theta}_k = \boldsymbol{\Theta}_0$ and $r_{max} = r(T_0)$.
3: **while** true **do**
4: Sampling: Using $\boldsymbol{\Omega}_k, \mathbf{X}^{\mathrm{g}}$, and $T_k$, generate rollout ($\mathbf{X}$) from $\mathbf{a} = (\boldsymbol{\Omega}_k + \varepsilon_t)^T \boldsymbol{\Psi}(\mathbf{X}, t)$ based on Equation (1)
with exploration $\lfloor \varepsilon_t \rfloor_{ij} \sim N(0, \sigma_{ij}^2)$ as a stochastic policy.
5: Set $\tilde{T} = T_k + \epsilon_{T_k}$ with $\epsilon_{T_k} =$ **randomize**$(1, \frac{T_k}{4})$.
6: **if** $r(t) \geq r_{max}$ **then**
7: Set $\tilde{T} = \mathrm{argmax}_t$ and $r_{max} = r(t)$.
8: Collect all information $(t, \mathbf{X}_t, \mathbf{a}_t, \mathbf{X}_{t+1}, \varepsilon_t, r_{t+1})$ for $t = \{1, 2, ..., \tilde{T} + 1\}$.
9: **else**
10: Discard all information $(t, \mathbf{X}_t, \mathbf{a}_t, \mathbf{X}_{t+1}, \varepsilon_t, r_{t+1})$ for $t = \{1, 2, ..., \tilde{T} + 1\}$.
11: **end if**
12: Estimating: Use unbiased estimate of the value function $\hat{Q}^{\pi}(\mathbf{X}, \mathbf{a}, t) = \sum_{\tilde{t}}^{\tilde{T}} r(\mathbf{X}_{\tilde{t}}, \mathbf{a}_{\tilde{t}} \mathbf{X}_{\tilde{t}+1}, \tilde{t})$.
13: Reweighting: Reweight rollouts and discard low-reward rollouts.
14: Update the parameters of DMP using
$\boldsymbol{\Omega}_{k+1} = \boldsymbol{\Omega}_k + \left\langle \sum_{t=1}^{\tilde{T}} \varepsilon_t Q^{\pi}(\mathbf{X}, \mathbf{a}, t) / \left\langle \sum_{t=1}^{\tilde{T}} \varepsilon_t Q^{\pi}(\mathbf{X}, \mathbf{a}, t) \right\rangle \right\rangle$.
15: Update the parameters $\boldsymbol{\lambda}_{k+1}$ of the HMM using reaction force/moment recorded during the motion generation $\mathbf{a} = (\boldsymbol{\Omega}_{k+1} + \varepsilon_t)^T \boldsymbol{\Psi}(\mathbf{X}, t)$.
16: Update $T_{k+1} = \tilde{T}$ and $\mathbf{X}_{k+1}^{\mathrm{g}} = \mathbf{X}(\tilde{T})$.
17: **if** $\boldsymbol{\Theta}_{k+1} \approx \boldsymbol{\Theta}_k$ **then**
18: **break**
19: **end if**
20: **end while**
21: Output: the optimized parameters $\boldsymbol{\Theta}_{new} = \{\boldsymbol{\lambda}_{k+1}, \boldsymbol{\Omega}_{k+1}, \mathbf{X}_{k+1}^{\mathrm{g}}, T_{k+1}\}$.

The iPoWER and gPoWER algorithms use a deterministic policy $\bar{\mathbf{a}} = \boldsymbol{\Omega}^T \boldsymbol{\Psi}(\mathbf{X}, t)$ with the weighting parameters $\boldsymbol{\Omega}$ and basis functions $\boldsymbol{\Psi}$ of a DMP [20]. When optimizing and generalizing a DMP, this policy is transformed into a stochastic policy using additive exploration $\epsilon(\mathbf{X}, t)$; to perform model-free RL, we always use a policy $\pi(\mathbf{a}_t | \mathbf{X}_t, t)$, which can be modified into the form $\mathbf{a} = \boldsymbol{\Omega}^T \boldsymbol{\Psi}(\mathbf{X}, t) + \epsilon(\Psi(\mathbf{X}, t))$. Here, $\epsilon(\Psi(\mathbf{X}, t)) = \varepsilon_t^T \boldsymbol{\Psi}(\mathbf{X}, t)$ with $\lfloor \varepsilon_t \rfloor_{ij} \sim N(0, \sigma^2)$ is used, where $\sigma_{ij}$ is a meta-parameter of the exploration that can also be optimized in these algorithms. In the iPoWER algorithm, the target $\mathbf{X}^g$ of a motor skill is used without any changes or updates. The length of a corresponding DMP can be reduced by the stop signal $t_{stop}$ when $\mathbf{X}^g = \mathbf{X}_t$ and $t < T_k$. This means that during the RL process, robots can arrive more quickly at the target than the humans who demonstrated the motion. Stop signal $t_{stop}$ is generated when the robot reaches the target within extremely small margins $|\mathbf{X}^g \pm \epsilon_x|$ or when execution time steps reaches a pre-selected length $(T + \epsilon_{T_k})$. The policy parameters of a DMP are optimized, after which the parameters of its corresponding HMM are re-estimated while generating the motion trajectories from the optimized DMP.

In the generalization process, the gPoWER algorithm finds a new target and the policy parameters for achieving it. To do this, parameter $\tilde{T}$ is set to be the time step ($\arg\max_t r(t)$) such that the reward value in all rollouts is higher than $r_{\max}$ and the new target is set to be the robot configuration in time step $\mathbf{X}(\tilde{T})$. Parameters $\mathbf{X}^g$ and $t_{stop}$ are determined by the value of $r_{\max}$, which indicates the highest reward in all rollouts and the time step in the rollout of the value of $r_{\max}$. Parameters $\boldsymbol{\Omega}_{new}$, $\mathbf{X}_{new}^g$, and $T_{new}$ are incrementally updated until the parameters converges: $\boldsymbol{\Theta}_{k+1} = \boldsymbol{\Theta}_k$. Next, the parameters $\boldsymbol{\lambda}_{new}$ of a HMM are estimated during motion generation with $\boldsymbol{\Omega}_{new}$, $\mathbf{X}_{new}^g$, and $T_{new}$. In this algorithm, the length of the new motor skill may be increased from the original one of the initial motor skill. Therefore, the length of motor skill is changed using $\epsilon_{T_k}$ while estimating the new target and the parameters of the new motor skill. The length of the new motor skill can be optimized during the improvement process after the generalization process is complete.

To calculate the expected return values for the improvement and generalization processes, a reward function should be defined. Its general equation is defined as

$$r(t) = \exp\left(-\alpha\left(\mathbf{X}^g - \mathbf{X}(t)\right) - \beta\left(\frac{1}{\mathbf{Y}^s - \mathbf{Y}(t)}\right)\right), \tag{7}$$

where $\mathbf{X}$ and $\mathbf{Y}$ indicate the values that can be measured from the robot and/or sensors. Here, these are only used to represent different variables. The superscripts $g$ and $s$ denote the target and starting values of each variable, which depend on the given task. Here, the term $(\mathbf{X}^g - \mathbf{X}(t))$ is used to obtain a high return value when the robot configuration is close to the target values, and the term $\frac{1}{\mathbf{Y}^s - \mathbf{Y}(t)}$ is used to obtain high return value when it is far from the starting value. Parameters $\alpha$ and $\beta$ are constants to adjust the importance of each term. Equation (7) is designed to take the form of $\exp^{-(x)}$; therefore, a lower value of either term provides a higher return value. In this framework, we can easily obtain the value of $\mathbf{X}^g$ that the robot must finally achieve from human demonstrations. The robot can use the first term in Equation (7) to obtain the optimal path and optimal execution time step while achieving the target $\mathbf{X}^g$. In contrast, the robot may need to be as far from the initial value $\mathbf{Y}^s$ as possible depending on the given tasks. We define this case as the second term in Equation (7). Here, the initial value $\mathbf{Y}^s$ can be easily obtained from a robot or its sensors.

## 3. Experiment

### *3.1. Description of the Peg-in-Hole Task*

To evaluate the proposed method, we applied our framework to the peg-in-hole task. This task consists of the following two motor skills: "Hole search"—a parameter tuple that classifies the direction of a hole and generates the motion trajectories of searching a hole based on reaction force/moment signals in the inaccurate current positions/postures of the peg and hole—and "peg insertion"—the parameter tuple that classifies reactions according to the directions of the hole and generates the motion trajectories of inserting a peg into a hole. Although the "hole search" and "peg insertion" motions can be performed using various strategies [30], humans perform the "hole search" demonstrations by adopting a tilt-search strategy, as illustrated in Figure 2a, and they perform the "peg insertion" demonstrations by employing a two-point contact strategy, as indicated in Figure 2b–e. In the "hole search" motor skill, suitable tilting angles should be learned and then generated: if small tilting motion trajectories are generated, the hole may not be found because reaction force/moment signals cannot be measured; if large tilting motion trajectories are generated, the motion skill of "peg insertion" may not be performed because the peg is caught in the hole even though it has been found by "hole search" motor skill. Moreover, in the "peg insertion" motion, the robot must learn and generate suitable insertion motion trajectories for the peg according to reaction force/moment signals measured depending on the relative positions and directions of the hole and peg. To achieve this, it is important to classify the relative positional and directional relationship between the peg and hole and then to generate their appropriate motion trajectories from the selected motor skills, as shown in Figure 3.

For this purpose, the peg-in-hole task was performed using the experimental setup illustrated in Figure 4. We used the UR3 robotic arm (developed by Universal Robots, Denmark) and the FT300 F/T sensor, 2-finger gripper, and a wrist camera (all developed by Robotiq, Canada). We conducted the experiments using five pegs and five holes with triangle, rectangle, pentagon, hexagon, and star shapes, as indicated in Figure 5a–e. The clearance between the pegs and holes was approximately 200 μm. The vision solution of the robot allows it to recognize the approximate peg and hole locations as well as their exact shapes. However, the peg cannot simply be inserted into the hole due to errors in the vision system of the wrist camera. The errors in the position and posture obtained in the experiments using this vision solution were approximately 5–10 mm and between 2° and 3°, respectively. Therefore, both types of motor skills are needed to complete the peg-in-hole task despite these errors.

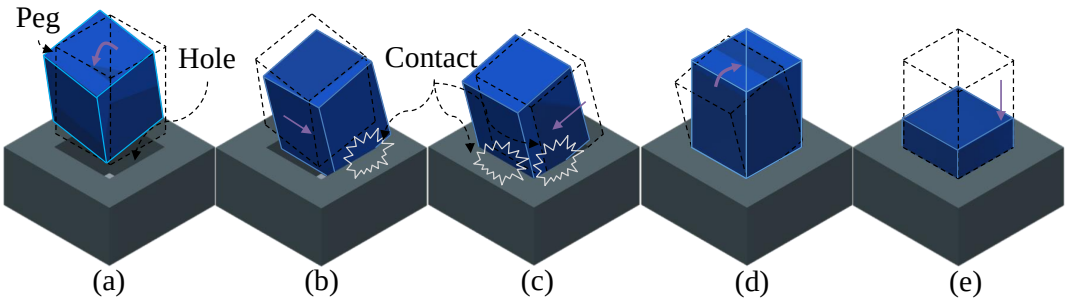


**Figure 2.** "Hole search" and "peg insertion" motions in the peg-in-hole task: (**a**) tilt-search motion; (**b**) single-point contact motion, which pushes on one face of the peg to make contact with the hole; (**c**) two-point contact motion, which pushes two faces of the peg to make contact with the hole; (**d**) tilting motion to align that the peg and hole; and (**e**) pushing motion to push the peg down into the hole.

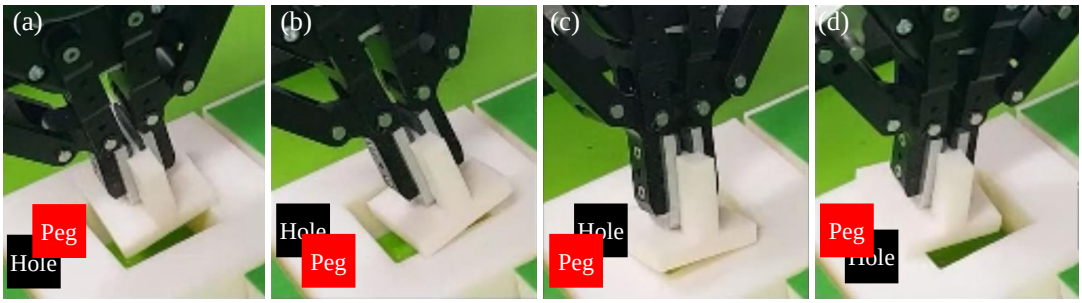


**Figure 3.** Different demonstrations according to relative positions/directions of the peg and hole: pegs and holes at directions of: (**a**) 30°; (**b**) 120°; (**c**) 210°; and (**d**) 300°. Here, these directions represent the relative positions of the peg relative to the hole. The black and red boxes indicate the hole and the peg, respectively. The human needs to provide the corresponding demonstrations according to these configurations, and the robot learns different reaction force/moment signals and reaction motion trajectories from these demonstrations.

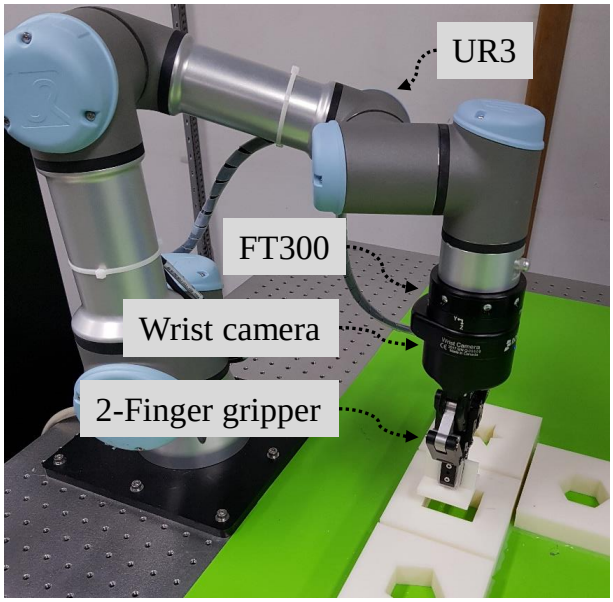


**Figure 4.** Experimental setup for peg-in-hole task.

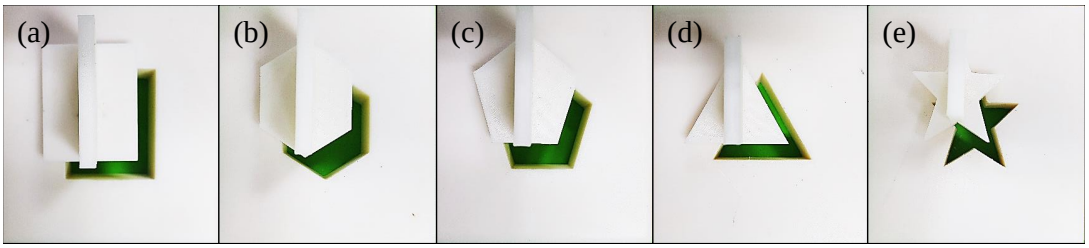


**Figure 5.** Five pegs and five holes with: (**a**) rectangle; (**b**) hexagon; (**c**) pentagon; (**d**) triangle; and (**e**) star shapes.

### *3.2. Reward Functions of Imitation Learning and Self-Learning in the Peg-in-Hole Task*

To calculate the expected return values for the "hole search" and "peg insertion" motor skills, two reward functions (i.e., $r^s$ for "hole search" and $r^i$ for "peg insertion") are, respectively, defined as

$$r^s(t) = \exp\left(-\alpha \mathbf{F}_{x,y,z}(t) - \beta \mathbf{M}_{x,y,z}(t) - \gamma \mathbf{R}_{x,y,z}(t)\right) \tag{8}$$

and

$$r^i(t) = \exp\left(-\alpha \mathbf{F}_{x,y}(t) - \beta \mathbf{M}_{x,y,z}(t) - \gamma \mathbf{P}_z(t)\right), \tag{9}$$

where **F**, **M**, **R**, and **P** indicate the force, moment, rotation, and position measured from the robot, respectively. In particular, the variables **R** and **P** represent the robot configuration measured from the reference axis of the tool coordinate system. In addition, **F** is calculated by $\mathbf{F}_{(x,y,z)}(t) = |F_x^g - F_x(t)| + |F_y^g - F_y(t)| + |F_z^g - F_z(t)|$, and **M** and **P** are calculated using equations with a similar form. In contrast, **R** is calculated using $\mathbf{R}_{(x,y,z)}(t) = \frac{1}{|R_x^s - R_x(t)|} + \frac{1}{|R_y^s - R_y(t)|} + \frac{1}{|R_z^s - R_z(t)|}$. Here, superscripts $g$ and $s$ indicate the target and starting values of each variable depending on the given task, respectively. Subscripts $x$, $y$, and $z$ denote the variables of each measure. Further, parameters $\alpha$, $\beta$, and $\gamma$ are constants, which are used to adjust the weight of each term.

In Equation (8), the motor skills should be able to determine the minimum tilting angle needed to quickly distinguish whether or not a hole exists. This is determined using reward function $r^s$ for the "hole search" motor skill. Here, it increases when all axes of $\mathbf{F}_{(x,y,z)}(t)$ and $\mathbf{M}_{(x,y,z)}(t)$ at every time step are closer to all targets $\mathbf{F}^g$ and $\mathbf{M}^g$. In contrast, the position of the z-axis is incorporated as a reward term for inserting the peg into the hole. The reward increases when the z-axis position $\mathbf{P}_z(t)$ at every time step is closer to target $\mathbf{P}^g$. It is possible for robots to determine the optimal motions. In these two equations, $\mathbf{F}^g$, $\mathbf{M}^g$, $\mathbf{R}^s$, and $\mathbf{P}^g$ are set to zero for the peg-in-hole task.

In the imitation learning process, human demonstrations are provided by employing a kinesthetic teaching method. This is a method for easily and rapidly conveying the motor skills of human performers to robots. However, it is not suitable for the peg-in-hole task, in which reaction classification is important for achieving the goal because unintended reaction force/moment signals may be included in human demonstrations. When reproducing unintended reaction force/moment, the robot does not achieve the goal of the motor skill. Thus, such unintended signals should be eliminated from the human demonstrations through robot self-reproduction. That is, the robot acquires the targets of motor skills as well as the reward functions through this self-reproduction. Despite this self-reproduction, robots may still not be able to obtain an optimal solution. This can be improved through RL during the improvement process.

A demonstration dataset must be modeled to enable reaction classification and motion generation. In this experiment, the robot should generate different motions depending on the relative positions/directions of the peg and the hole, as indicated in Figure 5. The states of both motor skills are defined as $\mathbf{X} = \{\mathbf{F}_{(x,y,z)}, \mathbf{M}_{(x,y,z)}\}$ to enable the HMMs to classify the reaction force/moment. In the DMPs, the state of the "hole search" motor skill is defined as $\mathbf{X} = \{\mathbf{R}_{(x,y,z)}\}$, owing to the fact that a tilt search is performed without changing the peg's position. In contrast, the state of the "peg insertion" motor skill is defined as $\mathbf{X} = \{\mathbf{F}_{(x,y,z)}, \mathbf{R}_{(x,y,z)}\}$ to control the force and rotation. Finally, the state of the

two reward functions for RL are defined as $\mathbf{X} = \{\mathbf{F}_{(x,y,z)}, \mathbf{M}_{(x,y,z)}, \mathbf{R}_{(x,y,z)}\}$ (to calculate Equation (8)) and $\mathbf{X} = \{\mathbf{F}_{(x,y,z)}, \mathbf{M}_{(x,y,z)}, P_z\}$ (to calculate Equation (9)). The states for HMMs, DMPs, and reward functions are summarized, as indicated in Table 1.

**Table 1.** States for HMMs, DMPs, and reward for RL for "hole search" and "peg insertion" motor skills.

| | **Hole Search (Tilt Search)** | **Peg Insertion (Two-Point Contact)** |
|---|---|---|
| **HMMs** | Reaction force/moment $\mathbf{X} = \{f_x, f_y, f_z, m_x, m_y, m_z\}$ | Reaction force/moment $\mathbf{X} = \{f_x, f_y, f_z, m_x, m_y, m_z\}$ |
| **DMPs** | Rotation (No change of position) $\mathbf{X} = \{r_x, r_y, r_z\}$ | Force, rotation $\mathbf{X} = \{f_x, f_y, f_z, m_x, m_y, m_z\}$ |
| **RL** | Reaction force/moment and rotation $\mathbf{X} = \{f_x, f_y, f_z, m_x, m_y, m_z, r_x, r_y, r_z\}$ | Reaction force/moment and z-position $\mathbf{X} = \{f_x, f_y, f_z, m_x, m_y, m_z, p_z\}$ |

### *3.3. Results*

First, we performed human demonstrations on the rectangle shape (Figure 5a) to learn and acquire the initial motor skills based on imitation learning. The different hole and peg shapes (Figure 5b–e) were used to evaluate the generalization of the motor skills. The results of these experiments can be confirmed from supplemental video clip. Human performers provided demonstrations for the "hole search" and "peg insertion" demonstrations, as shown in Figure 6. Figure 7 illustrates an example of the different demonstrations that were performed according to the initial hole and peg positions. Figure 7a shows the initial positions of the pegs with respect to the hole, and Figure 7b shows their clustering results. In this case, the points were clustered using the reaction force/moment measured at the initial positions and the k-means clustering algorithm. The robot acquired four motor skills for the "hole search" and four motor skills for the "peg insertion". The motion trajectories of the robot were extracted at 50 Hz using the kinesthetic teaching method, following which the training data were acquired through self-reproduction. Eight motor skills (i.e., the parameter tuples for classifying reaction signals and generating motion trajectories for four types of "hole search" and four types of "peg insertion") were learned using the training dataset. We configured the reaction classification and motion generation processes to be independent for rational task execution in the RL implementation. In this case, the classification of the reaction force/moment was configured to use five pairs of robot and sensor signals, and the classification time was set to 50 Hz to ensure real-time performance.

Next, the eight initial motor skills were improved using the RL process. For self-improvement of the "hole search" motor skills, the RL rollouts and their rewards were generated according to the following two steps: (i) acquiring rollouts and calculating rewards in a hole-free (blocked) location; and (ii) verifying rollouts and updating policy parameters at the hole position. The robot verified which of the rollouts actually found the hole at its location and changed the rewards of the rollouts that failed to find holes to zero. It then updated the policy parameters using all the rollouts. The verification process in Step (ii) is necessary because the rollouts and their rewards are obtained at a location where no hole exists. As mentioned in Section 2.2, the purpose of improving the "hole search" motor skill is to enable robots to identify whether a hole is present using the tilting angle. The self-improvement of the "peg insertion" motor skill was performed at the hole location. This process did not require a specific verification step for the acquired rollouts because it was performed at the hole location. The robot was able to identify the optimal path for inserting the peg into the hole within a short time.

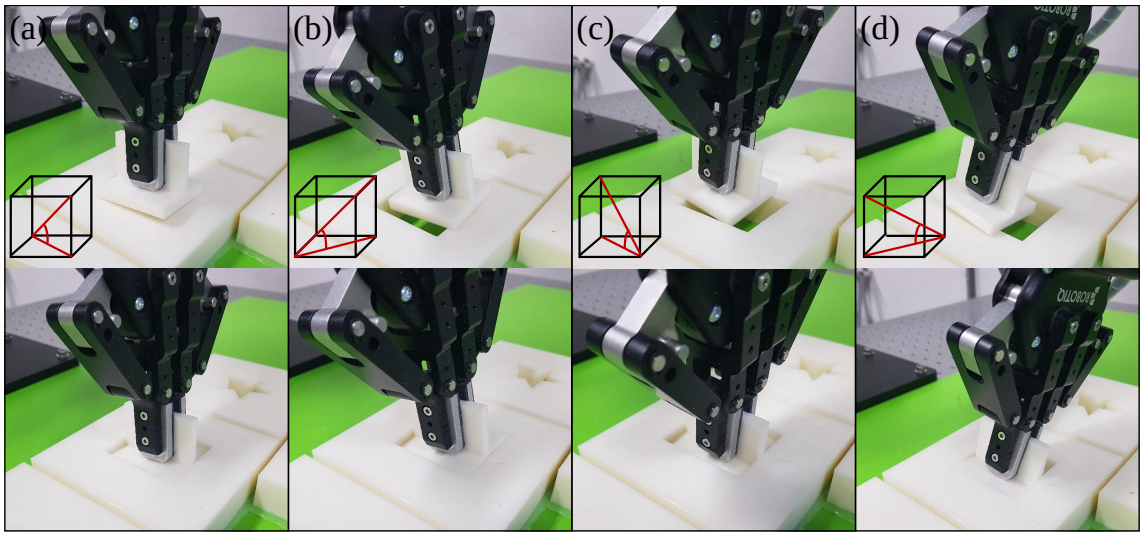


**Figure 6.** Four peg directions of a human demonstration of the "hole search" and "peg insertion" motor skills of the peg-in-hole task: (**a**) 30°; (**b**) 120°; (**c**) 210°; and (**d**) 300°. The upper and lower rows illustrate the motions of "hole search" and "peg insertion", respectively. In the upper low, the black boxes (holes) and red lines (pegs) indicate relative postures.

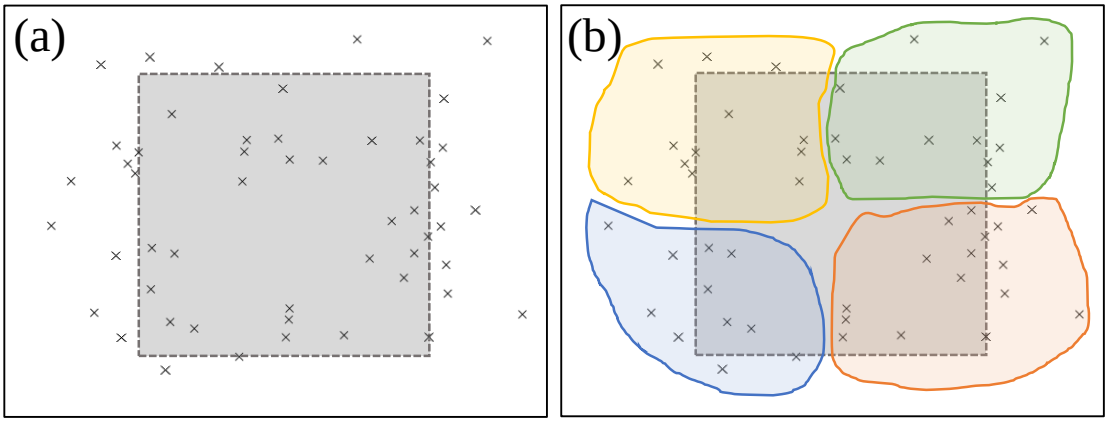


**Figure 7.** Human demonstrations and their clustered results. (**a**) Initial positions of the pegs ("x" symbols) with respect to the hole. The edge of the gray box indicates the boundary between the outside and inside of the hole. (**b**) Clustering results obtained by the k-means clustering algorithm using the reaction force/moment measured at the initial positions. We performed approximately 50 demonstrations to evaluate the clustering results.

The return values increase with the number of iterations, and the number of execution time steps are reduced when using the reward functions and the iPoWER algorithm, as indicated in Table 2. In the "hole search" and "peg insertion" motor skills, the robot obtained the expected return value of 0.7412 during 64 steps of motion in the first iteration while it received the expected return value of 0.9611 during 9 steps of motion after 300 iterations and the expected return value of 0.6233 during 75 steps of motion in the first iteration while it received the expected return value of 0.9249 during 10 steps of motion after 300 iterations, respectively. This table illustrates that the reward functions and the iPoWER algorithm were effectively designed in terms of time and path optimization. Figure 8 illustrates the results of the iPoWER algorithm compared with the original PoWER algorithm. The iPoWER algorithm reduces the number of robot execution time steps. Fewer execution time steps were needed when the iPoWER algorithm were used, as indicated in Figure 8.

**Table 2.** Expected returns and the number of execution time steps with respect to the number of iterations for the peg-in-hole task.

| Hole Search | | | Peg Insertion | | |
|---|---|---|---|---|---|
| **# of Iterations** | **Expected Return** | **# of execution Time Steps** | **# of Iterations** | **Expected Return** | **# of Time Steps** |
| 1 | 0.7412 | 64 | 1 | 0.6233 | 75 |
| 50 | 0.9253 | 17 | 50 | 0.8803 | 20 |
| 150 | 0.9479 | 13 | 150 | 0.9053 | 14 |
| 300 | 0.9611 | 9 | 300 | 0.9249 | 10 |

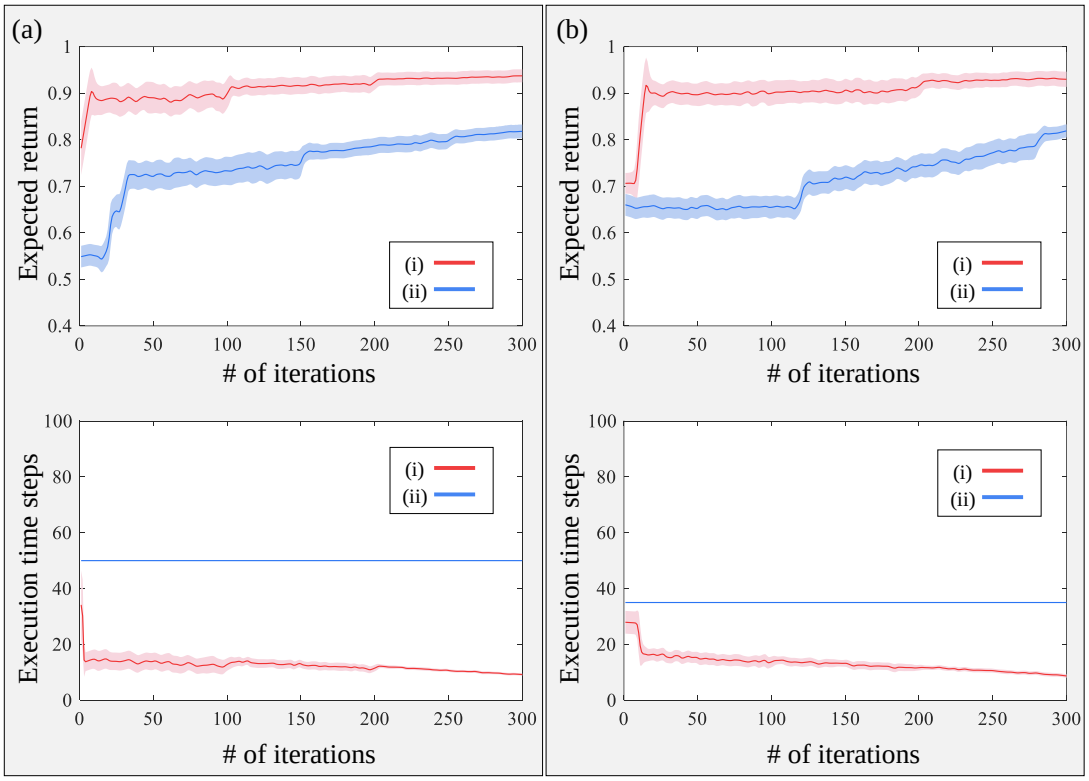


**Figure 8.** Performance with respect to two reward functions and the original PoWER and iPoWER algorithm in the peg-in-hole task for the motor skills: (**a**) "hole search"; and (**b**) "peg insertion". The upper and lower rows illustrate the return values and number of execution time steps, respectively. The red lines indicate the results using the iPoWER algorithm, and the blue lines indicate the results using the original PoWER algorithm.

Figure 9 illustrates the RL performance of our framework for the following cases: (a) the initial motor skill was represented by a HMM only; (b) the initial motor skill was represented by a DMP only; and (c) the initial motor skill was represented by both a HMM and a DMP (our framework). To evaluate Case (a), we performed RL after only learning the initial parameters $\boldsymbol{\lambda}$ for a HMM and target $\mathbf{X}^g$ from human demonstrations. However, the initial weight parameters $\boldsymbol{\Omega}$ for a DMP were randomly assigned without imitation learning (yellow line in Figure 9). This case can classify the reaction forces/moments, but it is impossible to generate their appropriate motion trajectories. The policy parameters of the DMP were improved by the iPoWER algorithm (Algorithm 2). In Case (b), we performed the RL process after only learning the initial weight parameters $\boldsymbol{\Omega}$ and the initial target $\mathbf{X}^g$ for a DMP from human demonstrations. Here, the initial parameters $\boldsymbol{\lambda}$ for a HMM were randomly assigned without imitation learning. In this case, the policy parameters and target of the DMP as well as the parameters of the HMM were generalized through the generalization process. Nevertheless, an unsuitable directions was used to generalize the DMP policy parameters and target, because the parameters of the HMM

were not properly assigned. Both Cases (a) and (b) were started with inappropriate policy parameters and targets. Here, we confirm that the RL process of Case (b) converged more quickly than that of Case (a), because it has the appropriate policy parameters to perform the peg-in-hole task, as illustrated by the blue and yellow lines of Figure 9, respectively. In contrast, in Case (c), our framework converged to the highest reward values in the fewest iterations, because it started with the parameters and target of a suitable DMP and the parameters of a suitable HMM, as illustrated by the red line of Figure 9.

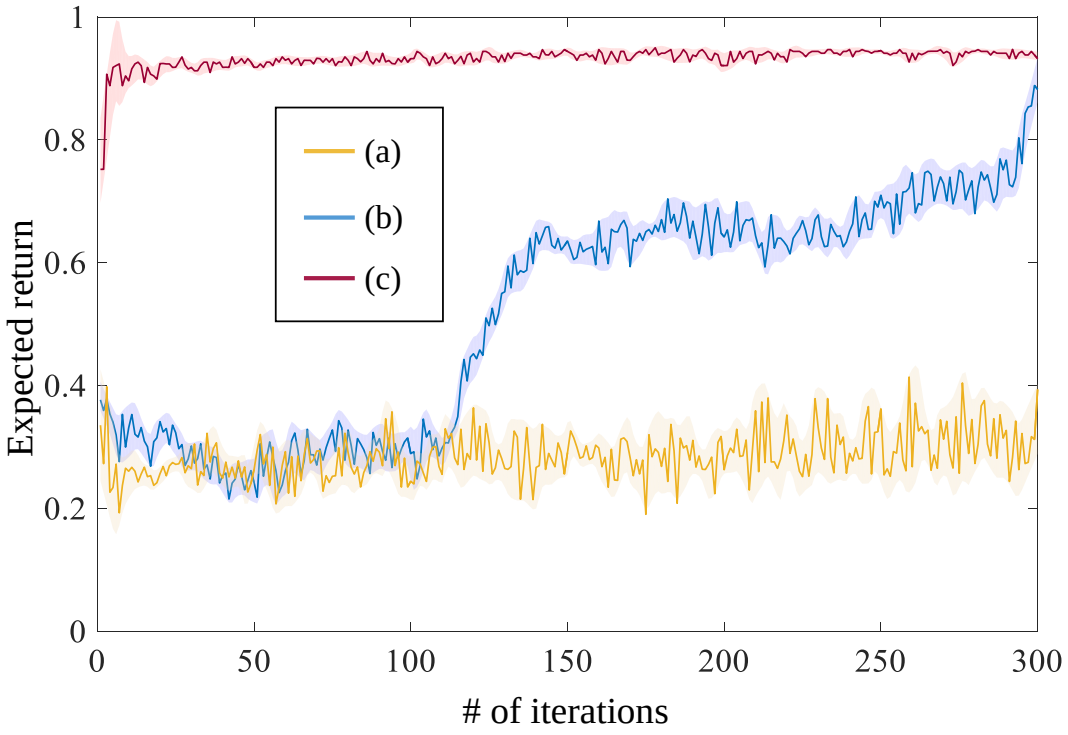


**Figure 9.** Expected returns of policy parameters with respect to number of iterations in the peg-in-hole task: (a) (yellow line) using only a HMM (without a DMP learned from human demonstrations); (b) (blue line) using only a DMP (without a HMM learned from human demonstrations); and (c) (red line) using both a HMM and a DMP. The average and variances of multiple RL trials from four different directions for the "hole search" and the "peg insertion" motor skills are shown.

Figures 10 and 11 illustrate the generalization of the motor skills learned for the rectangle shape to other shapes. Figure 10 presents the successful cases in which the motor skills learned for the rectangle shape could be used for other shapes without any generalization process. In contrast, Figure 11a presents the failure cases, in which the learned motor skills could not be applied to the other shapes. The failures usually occurred when determining the directions of holes in different shapes. In contrast, the "peg insertion" motor skills could be used without any RL process, even for the other shapes. The gPoWER algorithm (Algorithm 3) was used to solve this problem. Figure 11b illustrates the successful results that new motor skills added through the RL-based generalization process was performed.

Table 3 demonstrates the need for both the improvement and generalization processes. The initial motor skills were converged after 35–41 iterations during the improvement process. Generalizations of the improved motor skills converged more quickly than generalizations of the initial motor skills. In these cases, the convergence was approximately two to three times faster. Furthermore, the generalization of the improved motor skills was efficient even in the absence of initial motor skills acquired from human demonstrations. These results confirm that the robot can generalize motor skills even for unfamiliar shapes in the peg-in-hole task.

In these experiments, we generated the stop signal when the robot did not reach its target within $\mathbf{X}^g \pm 0.001$ or when the policy parameter did not converge within 300 iterations. When only the RL process was used without the imitation learning, it did not satisfy the targets of RL process, as shown in Figure 9c. We used several programs by modifying the open codes of DMP, HMM, and PoWER algorithms (these open codes can be downloaded in [31–33]) developed by the python language. In addition, the k-means and BIC algorithms performed using scikit-learn library. This library can be downloaded in [34]. Finally, the TM was manually created based on [24]. All of these experiments were performed on a PC (CPU: Intel i7-6700 3.40GHz, RAM: 32.0GB) with Windows 10 OS and Python 3.6 version.

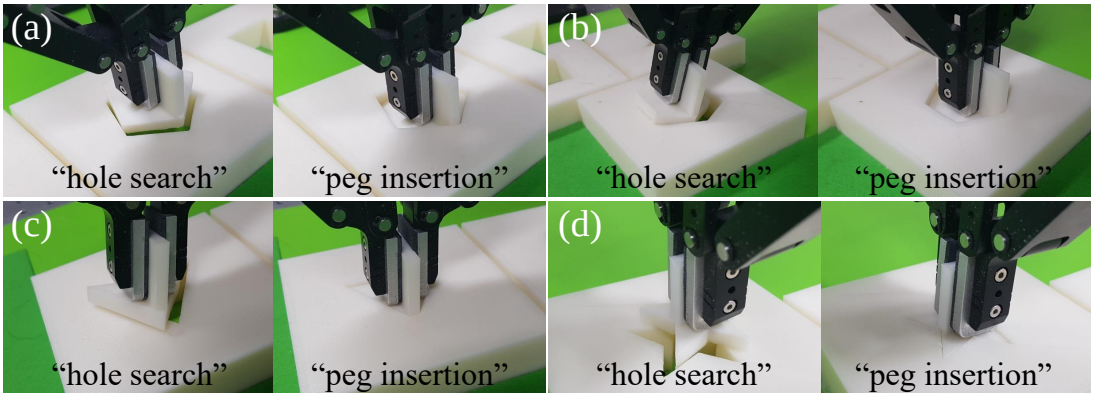


**Figure 10.** Using motor skills learned from the rectangle shape for different shapes: (**a**) pentagon; (**b**) hexagon; (**c**) triangle; and (**d**) star. These figures indicate the successful cases in which the motor skills learned for the rectangle shape can be used for other shapes without any generalization process.

**Table 3.** Comparison of number of iterations used in the improvement and generalization processes.

| Improvement/Generalization | # of Iterations |
|---|---|
| **Improvement** of motor skills for the **rectangle** shape from **initial** motor skills for the **rectangle** shape | 35 |
| **Improvement** of motor skills for the **triangle** shape from **initial** motor skills for the **triangle** shape | 37 |
| **Improvement** of motor skills for the **star** shape from **initial** motor skills for the **star** shape | 41 |
| **Generalization** of motor skills for the **triangle** shape from **initial** motor skills for the **rectangle** shape | 95 |
| **Generalization** of motor skills for the **triangle** shape from **improved** motor skills for the **rectangle** shape | 46 |
| **Generalization** of motor skills for the **star** shape from **initial** motor skills for the **rectangle** shape | 146 |
| **Generalization** of motor skills for the **triangle** shape from **improved** motor skills for the **rectangle** shape | 59 |

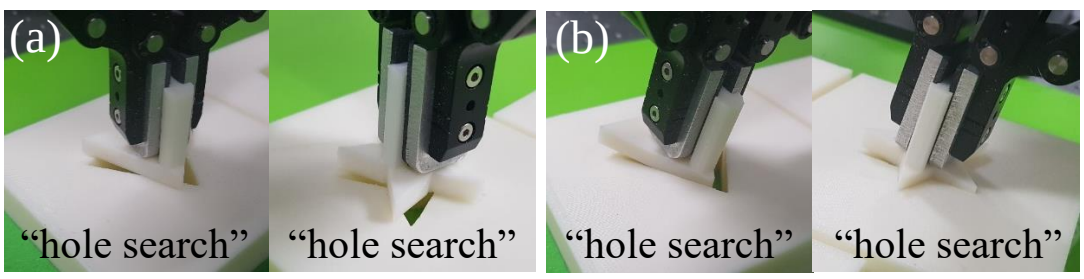


**Figure 11.** Using motor skills learned from rectangle shape for different shapes of triangle and star: (**a**) failure cases; and (**b**) the successful completion of the cases in (**a**).

## 4. Discussion

The proposed framework for the peg-in-hole task uses a mixture of imitation learning and RL. The peg-in-hole task requires the reaction force/moment classification and reaction motion generation because there is error in the sensors and robot actuators. Therefore, a robot should be able to continuously classify reactions and generate appropriate motion trajectories. To achieve this, motor skills are represented by concatenating the model parameters for reaction classification and motion generation. In general, a HMM and a DMP exhibit superior capabilities for time-varying classification and motion generation, respectively, as mentioned in [35]. Therefore, we used both models to consider their advantages in this framework. We refer to these concatenated parameter tuples as motor skills. We can use only HMMs or only DMPs for classification and motion generation, but they perform worse than the combination of HMMs and DMPs (refer Figure 9).

The proposed framework was evaluated for the peg-in-hole task; however, it can be used for various robotic tasks. The algorithms presented in Algorithms 1–3 are task-independent and can be used without modification to improve and generalize the motor skills required for such tasks. In the algorithms, only two elements need to be prepared for various tasks: (i) the reward functions; and (ii) human demonstrations. Designing the reward functions is the most important and difficult process in RL design. The target configurations of robots and/or objects obtained by human demonstrations can be useful in designing reward functions. It is also necessary to acquire the initial parameters of the motor skills (that is, the parameters of the HMMs and DMPs) from the human demonstrations of a target task. The robot can acquire the initial parameters of the motor skill when at least one demonstration has been performed [36]. Thereafter, it can use the proposed algorithms to create various motor skills that are automatically optimized by the improvement and generalization processes. After these two elements have been provided, the robot can obtain motor skills for which the number of execution time step and path have been optimized over several iterations. For example, this framework can also be considered for the use in a variety of industrial applications such as polishing, machine tending, soldering, painting, cutting, grinding, deburring, and inspection. First, humans provide human demonstrations to a robot. Next, the robot learns initial motor skills and collects the parameters of $\mathbf{X}^g$ and $\mathbf{Y}^s$ for reward functions from human demonstrations. Here, their motor skills and the parameters of $\mathbf{X}^g$ and $\mathbf{Y}^s$ can be modeled and extracted using various types of information (e.g., joints, positions, postures, velocities, forces, and/or torques) from human demonstrations depending on the purpose of motor skills. Finally, the robot can optimize or generalize motor skills through the improvement/generalization processes. However, it is necessary to determine the information to be modeled or used through human demonstrations in this process. The automation of this capability is not considered in this paper and it should be done with human.

In general, a robot needs to perform a sequence of some motor skills to perform its task. In other words, motor skills must be selected from a library of multiple ones. However, many researchers have focused on dealing with a single motor skill [37–40]. They suggested the ways to improve the performance of a motor skill. In addition, they did not consider the generalization of reusing the learned motor skills for other similar tasks (e.g., from the "rectangle" to the "triangle" peg-in-hole motor skills). In contrast, our proposed framework was able to handle the library of multiple motor skills based on the concatenated parameters. Furthermore, Algorithms 1–3 provide a way to improve existing motor skills (optimizing paths and reducing execution time steps) as well as add new motor skills.

In the human demonstrations, we adopted a tilt search and two-point contact strategies for the "hole search" and "peg insertion". This is because humans tend to accomplish the peg-in-hole task by tilting a peg into a hole, as analyzed in [30]. The authors of [30] also determined the most appropriate tilting angle for inserting the peg. In this study, a suitable tilting angle was learned for the tilt search through RL. Moreover, our aim was to predict and select appropriate motor skills from current reaction signals using learned motor skills. In contrast, it is difficult to use reaction classification with different peg-in-hole task strategies, such as the spiral path and spray paint strategies, because of

the uncertainties of the initial positions/poses of holes and pegs. These strategies tend to generate motions according to a set of predefined rules without any classification process.

The reward functions include the terms of the force, moment, $z$-axis robot position, $xyz$ robot rotation, and time step. Meta-parameters $\alpha$, $\beta$, and $\gamma$ control the weights of the terms individually because they deal with different information types (see Equations (8) and (9)). We can change the importance of each term in the reward function by adjusting these meta-parameters. We assigned the largest weighting values to the force/moment terms and the z-position term in the "hole search" and the "peg insertion" motor skills, respectively. In other words, it makes sense to assign weighting values depending on terms that have a significant impact on the success of the task.

## 5. Conclusions and Future Work

We propose a framework for learning, improving, and generalizing the "hole search" and "peg insertion" motor skills for the peg-in-hole task. In this framework, motor skills are acquired using a mixture of imitation learning and RL. Reaction classification and motion generation are required in the peg-in-hole task owing to errors in the sensors and actuators. The robot learns the initial motor skills for classifying the reactions and generating the appropriate trajectories from human demonstrations. We designed a motor skill parameter tuple by concatenating the HMM parameters for reaction classification and the DMP parameters for motion generation. The initial motor skills learned using imitation learning are improved and generalized by means of RL. These motor skills are either improved for familiar reaction signals or generalized for unfamiliar reaction signals. We distinguish the improvement and generalization processes as follows: improvement updates the policy parameters using the RL process without changing the target of its DMP, whereas generalization adds new parameter tuples after modifying and updating the policy parameters and target of a DMP. These processes are determined by the HMMs and TM. The generalization process is selected when the likelihood of the TM is higher than those of all other HMMs, and the improvement process is selected when the likelihood of one of the HMMs is higher than that of the TM.

We evaluated these processes by applying them to different peg and hole shapes. The "hole search" and "peg insertion" motor skills learned for the rectangle shape were generalized for triangle, pentagon, hexagon, and star shapes. These algorithm and reward functions improved the paths of the initial motor skills and optimized them to reduce the number of execution time steps.

In the future, we will analyze the manner in which various humans perform peg-in-hole tasks and compare the reward functions with those learned through inverse RL to enable interpretation. We will verify our framework by means of industrial applications and various other robotic tasks. In addition, we will propose a method to determine the information to be noted in the human demonstrations and use them in modeling motor skills and reward functions.

**Supplementary Materials:** The Supplementary Materials are available online at http://www.mdpi.com/2076-3417/10/8/2719/s1.

**Author Contributions:** Conceptualization, S.H.L. and I.H.S.; methodology, S.H.L. and N.J.C.; software, S.H.L. and N.J.C.; validation, N.J.C., J.B.K., and I.H.S.; formal analysis, S.H.L. and N.J.C.; investigation, S.H.L.; resources, N.J.C.; data curation, N.J.C.; writing—original draft preparation, S.H.L.; writing—review and editing, N.J.C. and J.B.K.; visualization, S.H.L. and N.J.C.; supervision, I.H.S.; project administration, I.H.S.; and funding acquisition, S.H.L. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Acknowledgments:** This work was supported by the Technology Innovation Industrial Program funded by the Ministry of Trade, (MI, South Korea) (10073161), Technology Innovation Program. This work also has been conducted with the support of the Korea Institute of Industrial Technology as "Development of holonic manufacturing, system for future industrial environment(KITECH EO-20-0019)". Finally, this work was supported by the Institute for Information&communications Technology Promotion (IITP) grant funded by MSIT (No. 2018-0-00622).

**Conflicts of Interest:** The authors declare no conflict of interest. The funders had no role in the design of the study, the collection, the analyses, the interpretation of data, the writing of the manuscript, and the decision to publish the results.

## References


1. Kronander, K.; Burdet, E.; Billard, A. Task Transfer via Collaborative Manipulation for Insertion Assembly. In Proceedings of the Workshop on Human-Robot Interaction for Industrial Manufacturing, Robotics, Science and Systems, Bielefeld, Germany, 3–6 March 2014; pp. 1–6.
2. Billard, A.; Calinon, S.; Dillmann, R. Learning from Demonstration. In *Springer Handbook of Robotics*; Springer: Berlin, Germany, 2016; pp. 1995–2014.
3. Mollard, Y.; Munzer, T.; Thibaut, B.; Baisero, A.; Toussaint, M.; Manuel, M. Robot Programming from Demonstration, Feedback and Transfer. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–2 October 2015; pp. 1825–1831.
4. Gupta, A.; Eppner, C.; Levine, S.; Abbeel, P. Learning Dexterous Manipulation for a Soft Robotic Hand from Human Demonstrations. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, Korea, 9–14 October 2016; pp. 3786–3793.
5. Kober, J.; Peters, J. Learning New Basic Movements for Robotics. In *Autonome Mobile Systeme*; Springer: Berlin, Germany, 2009; pp. 105–112.
6. Koenig, N.; Mataric, M. Robot Life-long Task Learning from Human Demonstrations: A Bayesian Approach. *Auton. Robot.* **2017**, *41*, 1173–1188. [CrossRef]
7. Zoliner, R.; Pardowitz, M.; Knoop, S.; Dillmann, R. Towards Cognitive Robots: Building Hierarchical Task Representations of Manipulations from Human Demonstrations. In Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA), Barcelona, Spain, 18–22 April 2005; pp. 1535–1540.
8. Abu-Dakka, F.J.; Nemec, B.; Kramberger, A.; Buch, A.; Kruger, N.; Ude, A. Solving Peg-in-Hole Tasks by Human Demonstrations and Exception Strategies. *Ind. Robot Int. J.* **2014**, *41*, 575–584. [CrossRef]
9. Zhao, Y.; Al-Yacoub, A.; Goh, Y.; Justham, L.; Lohse, N.; Jackson, M. Human Skill Capture: A Hidden Markov Model of Force and Torque Data in Peg-in-Hole Assembly Process. In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC), Budapest, Hungary, 9–12 October 2016; pp. 655–660.
10. Cho, N.J.; Lee, S.H.; Suh, I.H.; Kim, H. Relationship Between the Order for Motor Skill Transfer and Motion Complexity in Reinforcement Learning. *IEEE Robot. Autom. Lett.* **2018**, *4*, 293–300. [CrossRef]
11. Xu, Y.; Hu, Y.; Hu, L. Precision Peg-in-Hole Assembly Strategy Using Force-guided Robot. In Proceedings of the 3rd International Conference on Machinery, Materials and Information Technology Applications, Qingdao, China, 28–29 November 2015; pp. 6–11.
12. Park, H.; Park, J.; Lee, D.; Park, J.; Baeg, M.; Bae, J. Compliance-based Robotic Peg-in-Hole Assembly Strategy without Force Feedback. *IEEE Trans. Ind. Electron.* **2017**, *64*, 6299–6309. [CrossRef]
13. Zhang, X.; Zheng, Y.; Ota, J.; Huang, Y. Peg-in-Hole Assembly Based on Two-phase Scheme and F/T Sensor for Dual-arm Robot. *Sensors* **2017**, *17*, 2004. [CrossRef] [PubMed]
14. Jokesch, M.; Suchy, J.; Alexander, W.; Fross, A.; Thomas, U. Generic Algorithm for Peg-in-Hole Assembly Tasks for Pin Alignments with Impedance Controlled Robots. In Proceedings of the Robot 2015: Second Iberian Robotics Conference, Lisbon, Portugal, 19–21 November 2016; pp. 105–117.
15. Calinon, S.; Dhalluin, F.; Sauser, E.; Caldwell, D.; Billard, A. A Probabilistic Approach based on Dynamical Systems to Learn and Reproduce Gestures by Imitation. *IEEE Robot. Autom. Mag.* **2010**, *17*, 44–54. [CrossRef]
16. Ude, A.; Gams, A.; Asfour, T.; Morimoto, J. Task-specific Generalization of Discrete and Periodic Dynamic Movement Primitives. *IEEE Trans. Robot.* **2010**, *26*, 800–815. [CrossRef]
17. Kyrarini, M.; Haseeb, M.A.; Ristic-Durrant, D.A. Graser, Robot Learning of Industrial Assembly Task via Human Demonstrations. *Autom. Robot.* **2018**, *43*, 239–257.
18. Yun, S. Compliant Manipulation for Peg-in-Hole: Is Passive Compliance a Key to Learn Contact Motion? In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Pasadena, CA, USA, 19–23 May 2008; pp. 1647–1652.
19. Inoue, T.; Magistris, G.D.; Munawar, A.; Yokoya, T.; Tachibana, R. Deep Reinforcement Learning for High Precision Assembly Tasks. In Proceedings of the IEEE/RSJ International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 819–825.
20. Kober, J.; Peters, J. Imitation Learning and Reinforcement Learning. *IEEE Robot. Autom. Mag.* **2010**, *17*, 55–62. [CrossRef]

21. Kormushev, P.; Calinon, S.; Caldwell, D. Robot Motor Skill Coordination with EM-based Reinforcement Learning. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Taipei, Taiwan, 18–22 October 2010; pp. 3232–3237.
22. Kroemer, O.; Daniel, C.; Neumann, G.; Hoof, H.V.; Peters, J. Towards Learning Hierarchical Skills for Multi-phase Manipulation Tasks. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 1503–1510.
23. Levine, S.; Abbeel, P. Learning Neural Network Policies with Guided Policy Search Under Unknown Dynamics. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; pp. 1071–1079.
24. Lee, H.; Kim, J. An HMM-based Threshold Model Approach for Gesture Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **1999**, *2*, 961–973.
25. Lee, S.H.; Suh, I.H.; Calinon, S.; Johansson, R. Autonomous Framework for Segmenting Robot Trajectories of Manipulation Task. *Auton. Robot.* **2015**, *38*, 107–141. [CrossRef]
26. Calinon, S.; Dhalluin, F.; Sauser, E.L.; Caldwell, D.G.; Billard, A. Learning and Reproduction of Gestures by Imitation. *IEEE Robot. Autom. Mag.* **2010**, *17*, 44–54. [CrossRef]
27. Paster, P.; Hoffmann, H.; Asfour, T.; Schaal, S. Learning and Generalization of Motor Skills by Learning from Demonstration. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Kobe, Japan, 12–17 May 2009; pp. 763–768.
28. Lee, S.H.; Kim, H.K.; Suh, I.H. Incremental Learning of Primitivie Skills from Demonstration of a Task. In Proceedings of the 6th International Conference on Human-Robot Interaction (HRI), Lausanne, Switzerland, 6–9 March 2011; pp. 185–186.
29. Rautaray, S.S.; Agrawal, A. Vision based Hand Gesture Recognition for Human Computer Interaction: A Survey. *Artif. Intell. Rev.* **2015**, *43*, 1–54. [CrossRef]
30. Savarimuthu, T.; Lijekrans, D.; Ellekilde, L.; Ude, A.; Nemec, B.; Kruger, N. Analysis of Human Peg-in-Hole Executions in a Robotic Embodiment using Uncertain Grasps. In Proceedings of the 9th International Workshop on Robot Motion and Control (RoMoCo), Kuslin, Poland, 3–5 July 2013; pp. 233–239.
31. Dynamic Movement Primitives in Python. Available online: https://github.com/studywolf/pydmps (accessed on 7 April 2020).
32. Hidden Markov Models in Python. Available online: https://github.com/hmmlearn/hmmlearn (accessed on 7 April 2020).
33. Policy Learning by Weighting Exploration with the Returns (PoWER). Available online: http://www.jenskober.de/code.php (accessed on 7 April 2020).
34. Scikit-Learning: Machine Learning in Python. Available online: https://scikit-learn.org/stable/ (accessed on 7 April 2020).
35. Pehlivan, A.; Oztop, E. Dynamic Movement Primitives for Human Movement Recognition. In Proceedings of the Annual Conference of the IEEE Industrial Electronics Society, Yokohama, Japan, 9–12 November 2015; pp. 2178–2183.
36. Suh, I.H.; Lee, S.H.; Cho, N.J.; Kwon, W.Y. Measuring Motion Significance and Motion Complexity. *Inf. Sci.* **2017**, *388*, 84–98. [CrossRef]
37. Colomé, A.; Torras, C. Dimensionality Reduction for Dynamic Movement Primitives and Application to Bimanual Manipulation of Clothes. *IEEE Trans. Robot.* **2018**, *34*, 602–615. [CrossRef]
38. Hazara, M.; Kyrki, V. Model Selection for Incremental Learning of Generalizable Movement Primitives. In Proceedings of the 2017 18th International Conference on Advanced Robotics (ICAR), Singapore, 29 May–3 June 2017; pp. 359–366.
39. Winter, F.; Saveriano, M.; Lee, D. The Role of Coupling Terms in Variable Impedance Policies Learning. In Proceedings of the International Workshop on Human-Friendly Robotics, Genova, Italy, 29–30 September 2016; pp. 1–3.
40. Englert, P.; Toussaint, M. Combined Optimization and Reinforcement Learning for Manipulation Skills. In Proceedings of the Robotics: Science and Systems, Ann Arbor, MI, USA, 18–22 June 2016; pp. 1–9.