IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON ADVANCED COMMUNICATIONS AND NETWORKING TECHNIQUES FOR WIRELESS CONNECTED INTELLIGENT ROBOT SWARMS

Received December 19, 2019, accepted December 31, 2019, date of publication January 13, 2020, date of current version January 31, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2966226

A New Cloud Robots Training Method Using Cooperative Learning

GUANGLONG DU[1], ZHIYAO WANG[1], AND ZHELIN LI[2]
[1]School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China
[2]Guangdong Human-Machine Interaction Design Engineering Research Center, Guangzhou 510000, China

Corresponding author: Zhelin Li (cszlli@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61973126, in part by the Guangdong Natural Science Funds for Distinguished Young Scholar under Grant 2017A030306015, in part by the Pearl River S&T Nova Program of Guangzhou under Grant 201710010059, in part by the Guangdong Special Projects under Grant 2016TQ03X824, in part by the Fundamental Research Funds for the Central Universities under Grant 2019ZD27, in part by the Science and Technology Planning Project of Guangdong Province under Grant 2017B090914002, and in part by the Innovation Team of the Modern Agriculture Industry Technology System in Guangdong Province under Grant 2019KJ139.

ABSTRACT At present, cloud robots tend to be intelligent and cooperative. Based on this, we proposed a teaching method based on Imitation and a learning method that incorporates Incremental Learning and Meta Learning. We use Imitation Learning to teach robots, and more concretely, we propose a natural teaching method based on visual sense by using a depth camera, the robot can learn from the trajectory caught by the camera. Meta Learning helps robots understand the task and split it into some subtasks which enhances the level of generalization. Besides, once the circumstances change the robot can update the cloud database using Incremental Learning. Using proposed method, we make robots capable of learning and cooperating with other robots. It is no longer necessary for robots to learn based on a great number of data which is a shortcoming of traditional robots. The greatest advantage of this method is that we improve the learning efficiency of robots and enhance the level of generalization of the model. Our method was experimentally verified in a laboratory and the results indicated that the method improved the learning efficiency of robots.

INDEX TERMS Imitation learning, cloud robot, incremental learning, meta learning.

I. INTRODUCTION

Nowadays, robot collaborative work has become an urgent need, and the development of Industry 4.0 will accelerate this process. Therefore, optimizing the human-robot cooperation has been a very important matter in many recent works. As a result, a new concept - cloud robots - was put forward. Cloud robots, with the help of cloud platforms, allow robots to learn from each other and share knowledge. The concept of cloud robots was first proposed by Dr. Kuffner of Carnegie Mellon university in 2010 [1]which was then quickly followed by article [2] in which Steve Cousins summed up the concept as ''No robot is an island.'' Cloud robot is the combination of cloud computing and robotics. The robot itself does not need to store all information or have strong computing power and can connect to the relevant server and obtain the required information when needed. Compared with traditional robots, it has stronger learning ability. Cloud robots are developed from traditional robot combining network and cloud computing, which has obvious advantages over traditional robots. With a Shared knowledge base, robots can share information, so that robots around the world can learn from each other through the knowledge base. Nowadays tasks requiring robots to perform are increasingly complex, traditional robots have limited computing power and storage capacity, while cloud robots store dense computing and large storage in the cloud, providing a wider range of applications. In addition, many behaviors and action sequences of cloud robots are encapsulated as modules, and the development based on modularization also brings convenience to the use of program developers.

To this day, many researches have been done on the learning and sharing of knowledge among robots. For example, a multi-robot, multi-tasking learning framework, after a robot has passed the demonstration learning task, lessons learned can be moved to other robots and used to perform another task [3]. A learning framework of adaptive manipulative skills from human to robot to facilitate robot skill generalization is

The associate editor coordinating the review of this manuscript and approving it for publication was Jiankang Zhang.

described in [4]. An adaptive data sharing method in collaborative robots is described in [5]. A robot mutual learning system, each robot in the system is individual but they can also exchange information help each other learn [6]. A language decision tree algorithm for multi-robot path learning problem determines robot's behaviors dynamically [7], and so on.

In previous works, Sumin Cho and Jo [8] proposed an incremental learning method for teaching robots to do tasks through selected kinesthetic teaching trials [8]. A. M. Ghalamzan E. proposed a method for learning a control policy for a task from demonstration [9]. Although [8] requires kinesthetic teaching trials, which is not convenient from a practical standpoint, a better way is to let the robot learn directly from video. The empirical knowledge of [9] also is not self-taught. But in order to improve the adaptability of acquired skills to the environment and the generalization of tasks, it's important to make robots learn by themselves. So, in this paper, a collaborative learning method for cloud robots based on incremental learning is proposed.

The main contribution of this paper lies in the use of human-like thinking to obtain effective methods suitable for robot autonomous learning, including:

(1) A task understanding Meta Learning method based on neural task programming is designed. The dynamic programming idea of this learning method can effectively deal with new tasks.
(2) Based on the essence of learning and the structure of thinking, a human-assisted robot learning method was proposed, and a skill teaching method oriented to human-machine collaboration was constructed. Using this approach, robots can quickly learn new skills with a small amount of demo data.
(3) Combining the robot's learning rules with the knowledge structure characteristics of the robot's brain, a collaborative incremental learning method of openness, sharing, and group cooperation is proposed to obtain a more efficient robot autonomous learning framework.

The remaining sections are as follows. Section II is literature review. Section III is an overview of this paper. The task understanding method is introduced in Section IV. The skill teaching method is introduced in Section V. In Sections VI, we introduce Incremental Learning. The whole collaborative learning frame is introduced in Section VII. VIII describes the experiments and the results in detail. Finally, the conclusion of this paper follows in Sections IX.

II. RELATED WORK

At present, cloud robots system allows robots to learn from each other and share knowledge within the cloud platform. Inspired by biology, John Lones proposed a robotic adaptive approach [10]. Arren J. Glover proposes an incremental learning framework that enables lifelong learning and continuous learning of new things [11]. The above researches verified the feasibility of cloud robots.

In 2011, Eindhoven university et al. and Philips launched the RoboEarth project jointly to build a world wide web for robots [12]. [13] discussed how to make multi-agent work together, while a better solution is to enable multiple robots to learn together.

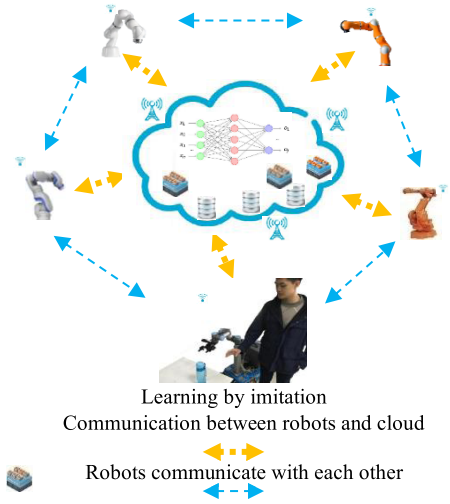


FIGURE 1. The architecture of cloud robot system based on incremental learning.

Sumin Cho and Jo [8] proposed an incremental learning method for teaching robots to do tasks through selected kinesthetic teaching trials [8]. It demonstrates that robot can incrementally refine and reproduce learned behaviors that accurately represent the essential characteristics of the teaching trials through incremental learning method and that it can reject erroneous teaching trials to improve learning performance.

Chelsea Finn *et* al. [14] present a meta-imitation learning method that enables a robot to learn how to learn more efficiently, allowing it to acquire new skills from just a single demonstration [14]. It combines meta-learning with imitation, enabling a robot to reuse past experience and, as a result, learn new skills from a single demonstration. This proves that in the field of robotics, the use of meta-learning can greatly improve the efficiency of robotic learning methods. In [15], based on meta-learning technology, robots can learn from human original video pixels.

In what follows, we will propose a collaborative learning method for cloud robots based on incremental learning.

III. OVERVIEW

Fig. 1 illustrates the architecture of the proposed cloud robot system based on incremental learning. The first part of the system is a collaborative learning platform for cloud robots, which will recycle information from terminal robots to improve the skills learning network and evaluate different states and actions. The second part of the system is the use of terminal robot. The terminal robot first splits the task into different skill combinations during the execution of the task, and at the same time, acquires the skills according to the human's imitative learning for the unknown task, and optimizes the acquired skills by obtaining the execution result during the actual execution and updating the new network, the

new network will be able to better understand the relationship between action and success. In this way, at each regular interval, each robot will obtain a copy of the upgraded learning network from the cloud robot collaborative learning platform, and then begin to use the skills information in these new learning networks to guide the action. Because these loop-updated learning networks will do a better job in assessing real-world actions, the robot itself will achieve better results. This virtuous circle is repeated in the continuous improvement of the task. By that means, the performance of hybrid intelligent networks is improved.

IV. TASK UNDERSTANDING METHOD BASED ON NEURAL TASK PROGRAMMING

Current AI based learning methods, especially deep learning, require large amount of previously labeled data for better training results. However, since robots generally take several seconds or even minutes to complete the learning task, generating a relatively good amount of training data is rather difficult. Furthermore, traditional robot teaching is usually conducted in a structured environment with poor scalability. So, we need a new method to train robots efficiently.

This study adopts a robot task learning and understanding framework of new neural task programming (NTP). It supports a few-shot learning from presentation and neural network program induction [16]. As a novel meta-learning method, NTP is a hierarchical model that recursively splits a large task into simple subtasks that are suitable for robotic demonstration learning like MOVE_LEFT, FOLLOW_THE_CURVE, PICK_UP AND DROP_INTO.

The structure of a task mainly consists of three variations: 1) Repeat Times: the number of times the task needs to be executed; 2) Task Execution Sequence; and 3) Task Content: what the robot need to do and what is the success conditions. Define T as the set of all simple subtasks, note that T can be infinite. S as the environment state space, A as the action space. For each task $t \in T$ the Boolean function g as described in Eq.(1) is the success condition of the task.

$$g : S \times T \rightarrow \{0, 1\} \tag{1}$$

Given the state $s \in S$, if task t is completed in state$s$, then $g(s, t) = 1$, otherwise $g(s, t) = 0$. Then, we use Task Description $\varphi(t) \in \Psi$ to describe each task, where Ψ is a collection of all possible task descriptions. Formally, the task description is treated as a sequence of random variables:

$$\varphi(t) = \{x_1, x_2, \ldots, x_N\} \tag{2}$$

NTP takes the task description $\varphi(t)$ as input to instantiate strategies and is defined as a time series that describes the task process and the ultimate goals. In many real-world tasks, robots cannot access the underlying environment state. It only receives the environmental observation sample $o \in O$ corresponding to the state s, where O is the observation space. Our goal is to learn a "meta-policy" that instantiates feedback strategies from a mission statement as below:

$$\tilde{\pi} : \Psi \rightarrow (O \rightarrow A) \tag{3}$$

During the test, each new task description $\varphi(t)$ is fed into NTP. Then the meta-policy generates a strategy as Eq. (3) to achieve the mission completion state s_T:

$$\pi(a|o; \varphi(t)) : O \rightarrow A \tag{4}$$

where $g(s_T, t) = 1$. For example, NTP splits the task of robot moving objects into subtasks.

NTP has three key components: task description interpreter f_{TDI}, task description encoder f_{TDE}, and core network f_{CN}. The task description is a time series that can describe the entire task step and the final goal, such as the human teaching video or the trajectory of the object. The task description encoder encodes a task description ψ into a vector space v. The core network uses the state s, the program p, and the task description ψ to produce the program key k and an end-of-program probability e. Then we get a program i that gets maximum with the key k in a memory that stores all programs. And when the probability reaches the threshold δ, the program returns. The task description interpreter takes the task description as input and chooses to perform one of the following two operations: (1) When the current program is not the bottom, it predicts the corresponding subtask description for the next subroutine;(2) When the program is the bottom (can be the basic skills of the robot or the API provided by the robot), the task description encoder converts the task description into a vector space.

Define $[M_j^{key}; M_i^{prog}]$ is a learnable key-value memory structure used to generate sub-program, $f_{en}(o)$ is a domain-specific task encoders used to map an observation to a state representation. The core network takes status, program, and task descriptions as input, to generate the next subroutine to be called and the probability of program ending as shown in Algorithm 1.

NTP is a task-independent learning algorithm that can be applied to various tasks with a potentially hierarchical structure, whose key idea is to learn reusable representations that are shared across tasks and domains. NTP explains a task

Algorithm 1 NTP Inference Procedure

Inputs: task description ψ, program id i, and environment observation o
function RUN (i, ψ)
  $e \leftarrow 0, p \leftarrow M_i^{prog}, s \leftarrow f_{en}(o), v \leftarrow f_{TDE}(\psi)$
  while $e \leftarrow \delta$ **do**
   $k, e \leftarrow f_{CN}(v, p, s), \psi' \leftarrow f_{TDI}(\psi, p, s)$
   $i' \leftarrow argmax_{j=1\ldots N}(M_j^{key}, k)$
   if program i' is primitive **then** if i' is an API
    $\boldsymbol{a} \leftarrow f_{TDI}(\psi', i', s)$ decode API *args* $\boldsymbol{a}$
    RUN_API($i', \boldsymbol{a}$) run API i' with a*rgs* $\boldsymbol{a}$
   else
    RUN (i', ψ') RUN program i' with ψ'
   end if
  end while
end function

and instantiates a hierarchical strategy as a neural program, where the underlying program is the basic operation or corresponding basic skills that can be performed in the environment. This hierarchical decomposition facilitates information hiding and modularity because the underlying module can only access the corresponding subtask descriptions related to its function, which prevents the model from learning spurious reliance on training data for better reusability. At the same time, NTP solves the problem of task generalization. As for the basic task at the bottom level, One-Shot learning (One-Shot learning) mode [17] can be used to train new skills and optimize existing ones by incremental learning parameters obtained from human-computer collaborative learning. This section will be described in detail below.

For tasks that are difficult for the robot to understand or tasks that the robot has never met, that is, tasks that cannot be resolved by NTP, the analysis can be assisted by human intervention. For NTP, humans only need to provide task description information in the input of task description interpreter or split and combine the targets so that NTP can split the corresponding skills.

V. SKILL TEACHING METHODS FOR HUMAN-COMPUTER COLLABORATION

In terms of teaching and learning, in many robot-learning methods, a video would be provided to the robot to demonstrate how to do a certain task, which illustrates the trajectory and actions of the task. The combination of the interpretation of the task demonstration by the human through natural interaction and the simulation of the robot can speed up the learning process. Programming by demonstration (PbD) can intuitively teaches robots sophisticated motor skills without the need for an amount of technical knowledge to program the robot [18]–[20]. However, there is a problem with this approach that any minor changes to the task require further demonstrations. Inspired by neurobiology, robot motion is divided into discrete motion states or ''motion primitives,'' which have significant advantages in inferring logical structures and inducing task execution procedures. The common method is to split the trajectory into a series of key points. If enough information locations are selected, interpolation between key point sequences can produce movements close enough to the original demonstration. However, the disadvantage is to ignore important information about changes along the trajectory, or a fixed path between the key points, or add complex point-to-point methods to consider modifications to the path. This paper proposes a new demonstration learning method (Learn from Demonstrations, LfD), which fills these shortcomings by avoiding the definition of state machines or reward functions. The goal of LfD is to learn strategies that can extend beyond the provided examples and be robust to disturbances. Use meta-learning methods to help robots quickly learn new tasks from gradient-based strategy updates [14]. Essentially, it is to learn strategy parameters. A new skill can be directly learned given a single demonstration, allowing the robot to efficiently learn new skills without any other mechanism. The process is performed in a low dimensional subspace where the trajectory of the robot can be effectively controlled. The goal of this project is to learn a strategy that can quickly adapt to new skills. Aiming to eliminate the need for a large amount of task-specific demonstration data, the scheme will reuse demonstration data from other skills to achieve efficient learning of new skills. Through cross-skill adaptation training, meta-learning effectively treats the entire skill as data points. The amount of data available for each individual skill is relatively small. In the context of robotics, this is the universal robot we want to develop—the ability to provide a small amount of supervision for every new skill a robot should perform. Consider a strategy π that maps observations o to predictive actions $\hat{a}$. During Meta Learning, the strategy was trained to accommodate numerous tasks. Formally, we set each simulation skill as:

$$T_i = \left\{\tau = \{o_1, a_1, \ldots, o_T, a_T\} \sim \pi_i^*, L\left(a_{1:T}, \hat{a}_{1:T}\right), T\right\} \quad (5)$$

which is composed of demo data τ generated by the human expert strategy π_i^* and a loss function L for simulation. The feedback is provided by the loss function as below.

$$L\left(a_1, \ldots, a_T, \hat{a}_1, \ldots, \hat{a}_T\right) \rightarrow R \quad (6)$$

We use a mean squared error loss as a function of policy parameters *plc* as follows:

$$L_{T_i}\left(f_{plc}\right) = \sum_{\tau^{(j)} \sim T_i} \sum_{t} ||f_{plc}\left(o_t^{(j)}\right) - a_t^{(j)}||_2^2 \quad (7)$$

We detail the procedure in Algorithm 2.

Due to imperfections in the actions provided in the demonstration video, we further use a neural network to let the robot learn directly from the video frames as shown in Fig. 2., for

Algorithm 2 Meta-Imitation Learning

Require: $p(T)$: distribution over tasks
Require: α, β: step size hyper parameters
randomly initialize ε
while not done do
  Sample batch of tasks $T_i \sim p(T)$
 for all T_i **do**
  Sample demonstration $\tau = \{o_1, a_1, \ldots o_T, a_T\}$ from T_i
  Evaluate $\nabla_\varepsilon L_{T_i}(f_\varepsilon)$ using τ and L_{T_i} in Eq. (2)
  Compute adapted parameters with gradient descent:
  $\varepsilon'_i = \varepsilon - \alpha \nabla_\varepsilon L_{T_i}(f_\varepsilon)$
  Sample demonstration $\tau'_i = \left\{o'_1, a'_1, \ldots o'_T, a'_T\right\}$
  from T_i for the meta-update
 end for
  Update $\varepsilon \leftarrow \varepsilon - \beta \nabla_\varepsilon \sum_{T_i \sim p(T)} L_{T_i}(f_{\varepsilon'_i})$ using
  each τ'_i and L_{T_i} in Equation (7)
end while
return parameters ε that can be quickly adapted to new tasks through imitation.

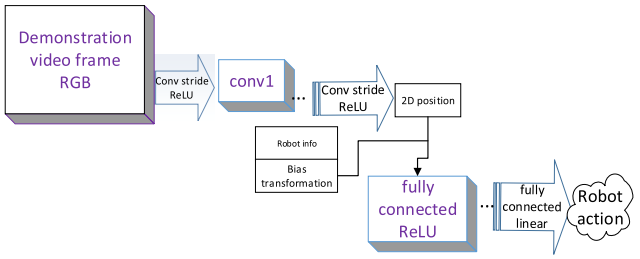


FIGURE 2. Diagrams of meta-imitation learning architecture.

which the loss function is defined as follows:

$$L^*_{T_i}\left(f_{plc}\right) = \sum_{\tau^{(j)}\sim T_i} \sum_{t} ||Wy_t^{(j)} + b - a_t^{(j)}||_2^2 \tag{8}$$

where $y_t^{(j)}$ is the set of post-synaptic activations of the last hidden layer, W and b is the weight matrix and bias of the final layer. With gradient descent and the meta-learned loss function $L^*_{T_i}$, we can get the adapted parameter θ_i' of each task T_i. So, we can get meta-objective as follows:

$$\min_{\theta,W,b} \sum_{T_i\sim p(T)} L_{T_i}(f_{\theta_i'}) = \sum_{T_i\sim p(T)} L_{T_i}\left(f_\theta - \alpha\nabla_\theta L^*_{T_i}(f_\theta)\right) \tag{9}$$

In general, some current image classification networks, such as VGG, can be used to extract 2D position points. The robot information and position information are fused by the full connection layer and the action feature vector is output.

VI. INCREMENTAL LEARNING

With the development of artificial intelligence and machine learning, people have developed many machine learning algorithms. Most of these algorithms are batch learning (Batch Learning) modes. That is, all training samples can be obtained once before training. After learning these samples, the learning process is terminated and no new knowledge is learned. However, in practical applications, the training samples of robots are not usually available all at once, but are gradually obtained with time, and the information reflected by the samples may also change with time. The cloud robot based collaborative learning system is an agent that can continuously learn new knowledge from new samples and can save most of the knowledge that has been learned before. Incremental learning is very similar to human's own learning patterns. Because people learn and receive new things every day while they are growing up, learning is gradually carried out. Moreover, human beings generally cannot forget the knowledge they have learned. Therefore, the incremental learning method is very suitable for the shared collaborative learning model of cloud robots. With the continuous increase in data size, the demand for time and space will increase rapidly, which will eventually lead to the speed of learning not being able to keep up with the speed of data update. If the new sample arrives the robots need to learn all data again, it will consume a lot of time and space, so the batch learning algorithm cannot meet this requirement. Only the incremental learning algorithm can gradually update the knowledge, and can correct and strengthen the previous knowledge, so that the updated knowledge can adapt to the newly arrived data without having to relearn all data. The incremental learning algorithm should simultaneously satisfy the below characteristics: 1) New knowledge can be learned from new data; 2) Data that has been previously processed does not require processing repeatedly; 3) Only one training observation sample is seen and learned at a time;4) Learning new knowledge while preserving most of the previously learned knowledge; 5) Once learning is completed, training observations samples are discarded; 6) The learning system does not have prior knowledge of the entire training sample. The importance of incremental algorithms is reflected in the following aspects: In an actual database, the amount of data tends to increase gradually. Therefore, when dealing with new data, the learning method should be able to make some changes to the trained system to learn the knowledge contained in the new data; The time to modify a trained system is usually lower than the cost of retraining a system. Besides, cloud resources can promote incremental learning [21].

Assume the old skills network have n groups feature mapping nodes and m groups broad enhancement nodes. The new learned skill feature mapping group nodes can be denoted as:

$$Z_{n+1} = \phi(XW_{n+1} + \beta_{n+1}) \tag{10}$$

Its enhancement nodes are as follows:

$$H_m = [\xi(Z_{n+1}W_1 + \beta_1), \ldots, \xi(Z_{n+1}W_m + \beta_m)] \tag{11}$$

where W_i and β_{ex_i} are randomly generated. Assume A is a $n \times m$ pattern matrix of the cloud robots knowledge network and $A^m_{n+1} = [A^m_n|Z_{n+1}|H_m]$ is the upgrade of new mapped features and the corresponding enhancement nodes. The relatively new pseudo-inverse matrix is as follows:

$$(A^m_{n+1})^+ = \begin{bmatrix} (A^m_n)^+ - MB^T \\ B^T \end{bmatrix} \tag{12}$$

$$B^T = \begin{cases} (C)^+ & \text{if } C \neq 0 \\ (1 + M^TM)^{-1}M^T(A^m_n)^+ & \text{if } C = 0 \end{cases} \tag{13}$$

where $M = (A^m_n)^+[Z_{n+1}|H_m]$, $C = [Z_{n+1}|H_m] - A^m_nM$

Then, the upgraded weights are:

$$W^m_{n+1} = \begin{bmatrix} W^m_n - MB^TY \\ B^TY \end{bmatrix} \tag{14}$$

Now we can use $(A^m_{n+1})^+$ and W^m_{n+1} update the network, the above is the steps of incremental learning to add new skill into network.

This paper introduces a mature incremental learning method for multi-machine collaborative learning, such as the network structure of Broad Learning System (BLS) proposed by Prof. Junlong Chen [22]. The BLS can be extended horizontally, using the characteristics of the input mapping as the network's feature node, and then enhanced to randomly generate weights (enhanced nodes), and connect the mapping features and enhancement nodes to the output directly. The

corresponding output coefficients can be obtained by fast *Pseudo* pseudo-inverse. In this way, with the newly added neural nodes, including the newly added feature nodes, BLS does not need to start from scratch. It only needs to adjust the weights associated with the newly added nodes, and it can also perform incremental learning on the newly added inputs. Such a network is very suitable for multi-robot sharing collaborative learning. The above section has already introduced that the robot can learn through human teaching and can also learn through the experience knowledge of other robots. This allow the robot to think like a human being would and learn to execute new tasks by itself without being any specific programming. Incremental learning methods can adapt to constantly learning new information and can use visual, auditory, and tactile as inputs of data. Facing new tasks, the robot will automatically look for previous experiences, and share knowledge and communicate with other robots through the internet.

Assume A is a $n \times m$ pattern matrix of the cloud robots knowledge neural network which has n group of feature mapping nodes and m group of enhancement nodes, W are connecting weights. X is a new knowledge learned by a robot, and $Z_x^n = [\phi\left(XW_{e_1} + \beta_{e_1}\right), \ldots, \phi\left(XW_{e_n} + \beta_{e_n}\right)]$ is the incremental features updated by X. The new node can be described as:

$$A_x = [\phi\left(XW_{e_1} + \beta_{e_1}\right), \ldots, \phi\left(XW_{e_n} + \beta_{e_n}\right) | \xi_x\left(Z_x^n W_{h_1} + \beta_{h_1}\right), \ldots, \xi_x\left(Z_x^n W_{h_m} + \beta_{h_m}\right)] \quad (15)$$

where W_{e_i}, W_{h_j} and β_{e_i}, β_{h_j} are random variation origin from the neural network, and ξ_x is a unique random mappings. Then A_n^m is updated as below:

$${}^x A_n^m = \begin{bmatrix} A_n^m \\ A_x^T \end{bmatrix} \quad (16)$$

The new knowledge updating algorithm is shown in Algorithm 3.

As a result, the pseudo-inverse of ${}^x A_n^m$ can be deduced as follows:

$$({}^x A_n^m)^+ = [(A_n^m)^+ - BD^T | B] \quad (17)$$

where $D^T = A_x^T A_n^{m+}$.

The relatively upgraded pseudoinverse matrix $(A^{m+1})^+$ is deduced as follows:

$$(A^{m+1})^+ = \begin{bmatrix} (A^m)^+ - DB^T \\ B^T \end{bmatrix} \quad (18)$$

Meanwhile, the new weights are:

$${}^x W_n^m = W_n^m + (Y_a^T - A_x^T W_n^m)B \quad (19)$$

where Y_a are the labels of X.

Algorithm 3 Collaborative Incremental Learning

Require: X a new knowledge learned by a robot
$i = 0$;
while $i <= n$ **do**
  Random W_{ei}, β_{ei};
  Calculate $Z_i = [\phi(XW_{ei} + \beta_{ei})]$;
  $i++$;
end while
Set the feature mapping group $Z^n = [Z_1, \ldots, Z_n]$;
for $j = 1; j <= m$ **do**
  Random W_{hj}, β_{hj};
  $H_j = [\xi(Z^n W_{hj} + \beta_{hj})]$;
end
Set the enhancement nodes group $H^m = [H_1, \ldots, H_m]$;
Calculate $A^+ = \lim_{\lambda \to 0}(\lambda I + AA^T)^{-1}A^T$;
while The error of model is not small enough **do**
  if p enhancement nodes are added **then**
    Random $W_{h_{m+1}}$, $\beta_{h_{m+1}}$;
    Calculate $H_{m+1} = [\xi(Z^n W_{h_{m+1}} + \beta_{h_{m+1}})]$;
    Update A_n^{m+1};
    Set $D = (A^m)^+ \xi(Z^n W_{h_{m+1}} + \beta_{h_{m+1}})$;
    update $(A^{m+1})^+$ with Eq. (8)
    Set $C = \xi(Z^n W_{h_{m+1}} + \beta_{h_{m+1}}) - A^m D$;
    if $C \neq 0$ **then**
    $B^T = (C)^+$;
    else
      $B^T = (1 + D^T D)^{-1} B^T (A^m)^+$;
    end
   $m++$;
  else
    Set new knowledge as X;
    update A_x, ${}^x A_n^m$ by Eq. (16),(17);
    update $({}^x A_n^m)^+$ and $({}^x W_n^m)^+$ by Eq. (18),(19);
  end
end while
return W;

VII. COLLABORATIVE LEARNING FRAME FOR CLOUD ROBOTS BASED ON INCREMENTAL LEARNING

Fig. 3 shows the cloud robot collaborative learning method proposed in this paper. This way of accessing information, just like accessing resources on the Internet, has obvious advantage. For tasks that robots do not understand, robots can search for ready-made solutions from the cloud resources (same task solution methods). When a robot starts acting, we will assist in adjusting the actions it chooses. In this way, the results of behaviors will sometimes be better than the execution results of experiences, and sometimes it will be worse because of human's incorrect guidance. This allows every robot to explore different ways of handling a certain task. The actions taken by the robots, their behavior and the records of the result are ultimately sent to the cloud robot. The server collects information about all robots and uses them cyclically (incremental learning) to improve the neural networks used to evaluate different states and actions.

This method mainly applies the current cutting-edge technologies including cloud storage, cloud computing, and big data processing. In addition to the function of sharing knowledge and experience which is analogous to what other most current cloud robot platforms have, another major function

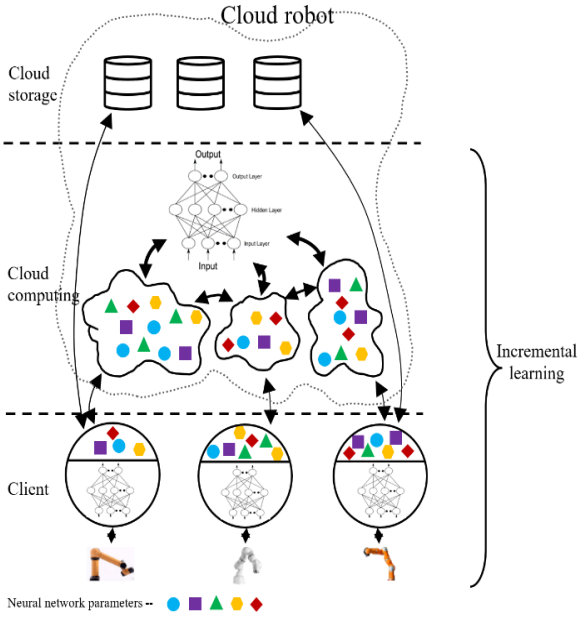


FIGURE 3. Collaborative learning framework based on cloud robots.

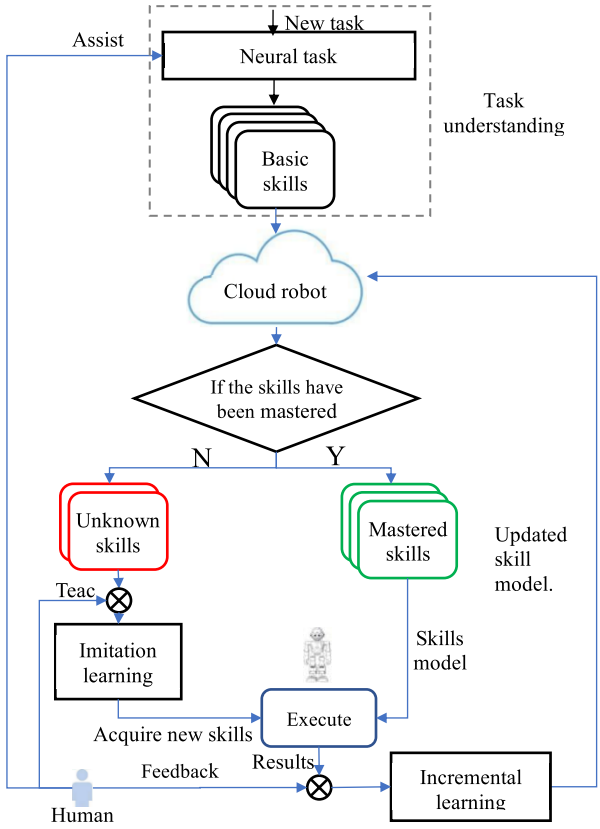


FIGURE 4. Collaborative learning frame for cloud robots based on incremental learning.

of the cloud robot collaborative learning platform is collaborative learning. Fig. 3 shows a diagram of collaborative learning model based on a cloud robot platform. Within this framework, the robot does not need to store any empirical knowledge or skill experiences locally. Instead, it acquires experiences or verified knowledge from the cloud robot. Cloud robots store skill knowledge learned by all robots and are open to all robots. When dealing with tasks, terminal robots only need to obtain relevant knowledge and experience through the Internet platform. When they cannot cope with tasks, they can learn by means of human's collaboration. The new skills and knowledge obtained by the terminal robot learning or the result data obtained by performing tasks and so on can be used to train the copied version neural network obtained from the cloud robot platform. The updated network parameters are used to update the network parameters of the cloud robot platform. Therefore, the newly acquired experience knowledge can be transmitted synchronously to other robots to realize the function of mutual learning and improving together.

We put forward the collaborative learning frame for cloud robots based on incremental learning as shown in Fig. 4. First when the robot is confronted with a new task, the robot splits the task by using the NTP method, which has been introduced in III. This recursively divides large tasks into simple subtasks, which are suitable for robot demonstration learning. We first define a set of subtasks T, which can be combined into all the tasks we need. The image data collected by Kinect2 is used as the current environmental state. For each subtask, we delegate a neural program to perform tasks. The neural program performs end-to-end training with the task decomposition mechanism. We define three key components of NTP: task description interpreter, task description encoder, and core network. The task description encoder is a BP neural network. In the core network, CNN is used to encode the state of the environment, and RNN is used to encode the program into vector features, and these two features are concatenated with the task description vector as the input of the BP neural network.

When a task that cannot be split is encountered, a human expert can intervene, and once the split is done, a series of skill sequence combinations are obtained, and then the central brain, namely the cloud robot, is used to search for related skills. The skills that the cloud robot had mastered, if any, are used to perform the task. For the skills that are dimmed difficult for the robot to master, a human expert input is provided and the robot can then directly learn from imitation and combine with the above-mentioned already acquired skills to complete the task. In this imitation learning setup, a new skill that was extracted from a generated demonstration by sample batch of tasks. During the meta-training, the strategy is to use a demo of the expert about sample batch of tasks to train and then test in a new demo to determine its training and test errors based on the loss. Then, improve the strategy by taking the changed test errors with the new demo of parameters into account. Therefore, the test error of sample demonstrations is a training error in the meta-learning process. At the end of meta-training, new skills are extracted from task set and meta-performance is measured by a demonstrated strategic performance. The result of meta-training is a strategy that can

adapt to new skills through a single demonstration. Therefore, during the meta-test, a new skill will be sampled and a demonstration of the skill will be provided, and the model will be updated to obtain the strategy of the skill. During meta-testing, a new skill may involve new goals or new operations or previously unseen objects.

Because each robot has a different environment and its own individual differences and other reasons, each skill performs differently on each robot. For the execution result after the completion of the task, humans can give a certain evaluation according to the execution situation, and when the robot shows a deviation or error, it can be timely corrected by humans. Humans utilize the execution result data to retrain the skill model. Then when robots in different circumstances upload what they learned, there is always something same and something different. In this case, we use incremental learning to improve the learning efficiency of robots. We only keep the different part and update it to the cloud robot platform to optimize the mastered skills and increased the skill numbers in the cloud database.

VIII. EXPERIMENT

A. ENVIRONMENT OF EXPERIMENT

To verify the feasibility of the proposed method, we used Robot-A (GRB3016), Robot-B (UR3) and Robot-C (KUKA) as our test platforms. Robot-A, Robot-B and Robot-C had all been connected to the Cloud. We use prior knowledge to define a set of valid subtasks T, which include actions such as move, insert, remove, etc. The image data collected by Kinect2 is used as the current environmental state data. In addition, when there is an indivisible task, the experimenters on the side will step in and mark it manually. Before the experiment, we will use the end-to-end method to train the imitation learning network. The input of the imitation learning network is some videos of the same task. In the experiments, PbD trials were first carried out on Robot-A and the learned trajectories were stored for the purpose of teaching the remaining robots. Fig. 5 shows the environments for our experiments. We put a steel plate on the experimental platform that was in front of the robot. And the operator stood beside the experimental platform to demonstrate the correct trajectory to the robot by gestures as shown in Fig. 7 experiment site 1. Our robots have a shaft and a depth camera attached to the end of arm. The camera can catch the image of the steel plate and then tell the robot how deep it should be.

FIGURE 5. Environments for experiment.

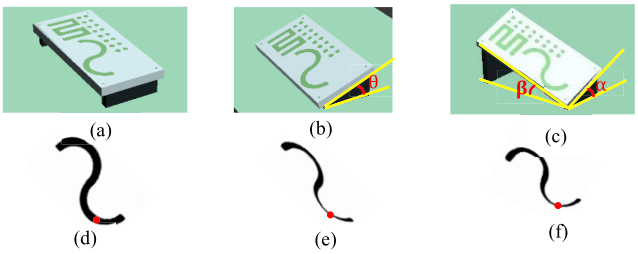


FIGURE 6. Steel plate arrangement and projection. (a) The steel plate parallel to the table. (b) A steel plate with an edge contacting the platform forming an angle of θ degrees. (c) Steel plates with long sides forming an angle of β degrees with the ground and short sides forming an angle of α degrees with the ground. (d) The width of the slot. (e) The width of the slot. (f) The width of the slot.

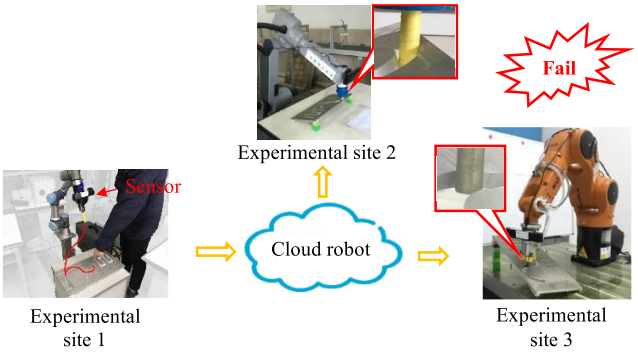


FIGURE 7. Experiment without incremental learning.

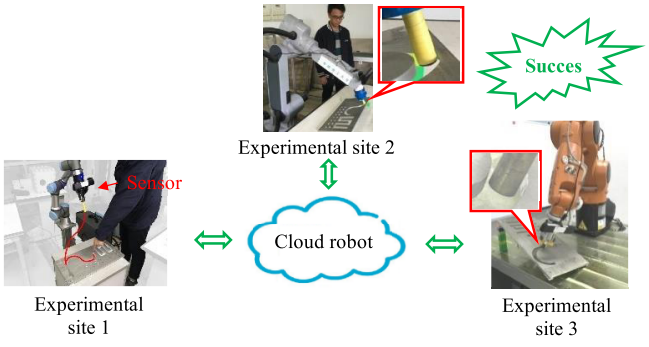


FIGURE 8. Experiment with incremental learning.

Below two experiments are presented. In each experiment, we placed the steel plate parallel to the table and demonstrated the trajectory to Robot-A [see Fig. 6(a)]. However, for Robot-B, we made only one edge of the steel plate contact with the platform to form a θ degree angle. As for Robot-C we made only the corner of the steel plate contact with the platform. In this case, the longer edge of the steel plate and the platform formed a β degree angle while the shorter one and the platform formed a α degree angle as shown in Fig. 6(b)-(c).

As it is known to us all, the projection of an object depends on the placement of the object. Therefore, the projection and the trajectory were the same for Robot-A. The width of the slot was greater than the diameter of the shaft at any position as shown in Fig. 6(d). The black line was the projection of the trajectory and the red point of was the projection of the shaft at the end of arm. For Robot-B, things changed. Because of the

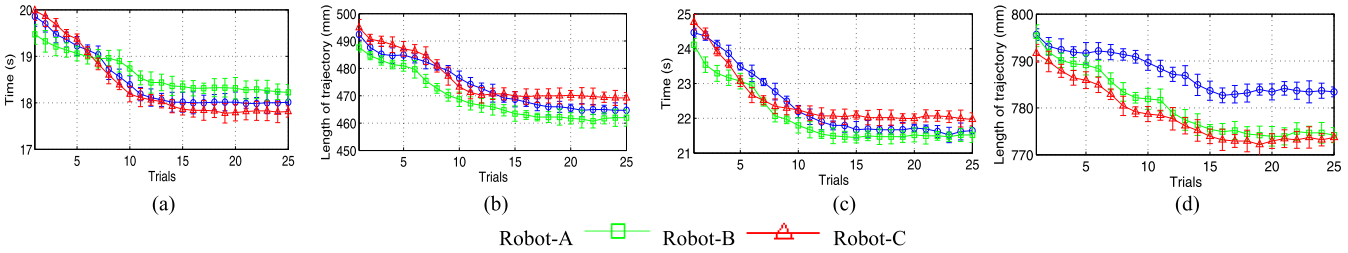


FIGURE 9. Time and Length of trajectory of Peg-into-hole and Trajectory tracking. (a) The trend of the time and the length of trajectory when robots pegged their shaft into the hole. (b) The trend of trajectory tracking. (c) The trend of the time and the length of trajectory when robots pegged their shaft into the hole. (d) The trend of trajectory tracking.

angle, the shape of the projection changed so that the width of the projection was less than that of the trajectory at some positions. Maybe it would become less than the diameter of the shaft as shown in Fig. 6(e). Because the placement of the steel plate became more different for Robot-C, the projection deformed more seriously. It meant that there might be more space where the diameter of the shaft was less than the width of the projection of the trajectory as shown in Fig. 6(f). In the similar circumstance, we let robots peg the shaft into the hole to see results.

B. RESULT ANALYSIS

For the first time, the operator demonstrated the correct trajectory to Robot-A. Robot-A learned the skills by track tracking and then it shared skills with Robot-B and Robot-C through the cloud. And then the shaft succeeded to go along the track without touching the steel plate.

But when Robot-B learned the skill, as we mentioned above we changed the layout of the steel plate and we didn't revise the solution. Thus similarly to Robot-A, Robot-B inserted the shaft vertically into the steel plate. Thus Robot-B just like what Robot-A did inserted the shaft vertically into the steel plate. As a result, the shaft touched the steel plate and Robot-B had to stop working to prevent damage. Robot-C was the same as Robot-B failing to find the solution that is perfect to the problem. Because it didn't learn the correct trajectory from the failure of Robot-B as shown in Fig. 7. But it's not absolute. We found the angle had a critical value. If the angle was greater than the critical value robots is sure to fail because when they inserted the shaft vertically into the steel plate, the width was too small for the shaft to pass. However, if the angle is less than the critical value, because there was enough space for the shaft to pass, robots was able to accomplish the task.

In the second experiment, the operator did something different, showing Robot-B how to find the correct solution when the placement of the steel plate had been changed. Then Robot-B used incremental learning technology and uploaded the skills it learned to the cloud for other robots(Robot-C) to refer. In this case, when Robot-C tried to solve the problem, it didn't insert the shaft vertically into the steel plate directly. Firstly, it found the placement of the steel plate had changed through an image returned by the camera. Then it adjusted the arm making the shaft perpendicular to the steel plate. Finally, Robot-C accomplished the task successfully as shown in Fig. 8.

TABLE 1. The feasibility of trajectory tracking with different methods when the angle changes.

	$\theta < 27°$	$\theta > 27°$	$\alpha < 35°$ $\beta < 24°$	$\alpha < 35°$ $\beta > 24°$	$\alpha > 35°$ $\beta < 24°$	$\alpha > 35°$ $\beta > 24°$
Method [23]	√	×	√	×	×	×
OMWOIL	√	×	√	×	×	×
OMWIL	√	√	√	√	√	√

OMWOIL: our method without incremental learning. OMWIL: our method with incremental learning

TABLE 2. The feasibility of peg-into-hole with different methods when the angle changes.

	$\theta < 23°$	$\theta > 23°$	$\alpha < 31°$ $\beta < 21°$	$\alpha < 31°$ $\beta > 21°$	$\alpha > 31°$ $\beta < 21°$	$\alpha > 31°$ $\beta > 21°$
Method [23]	√	×	√	×	×	×
OMWOIL	√	×	√	×	×	×
OMWIL	√	√	√	√	√	√

The result is shown in TABLE 1. When the angle θ was less than 27 degrees, the three methods performed well when they tried to solve the trajectory tracking problem. And the situation was the same when the angles α and β were less than 35 and 24 degrees, respectively. But when the angle became greater, we found only our method with incremental learning worked out while the other two methods couldn't accomplish the task. This is because, in these cases, the motion planning of the robotic arm can change significantly. Models without incremental learning cannot compare well with previous scenarios and learn such changes, while models using incremental learning can update and optimize neural networks that evaluate different states and actions and upload these changes to the cloud robot.

And unsurprisingly we saw the same case when robots pegged the shaft into the hole in TABLE 2. The first experiment we simulated the traditional batch learning model. Once learned the robots won't continue to learn to revise or optimize the solution. They only repeat what they learned and the learning process is one-way as shown in Fig. 7. So, once the circumstance change, robots will fail to complete the work. And then we will have to make one robot learn how to work in the new circumstance and upload what it learned to the cloud. It's too troublesome.

TABLE 3. The initial study time and the update learning time of robot-a with different methods in two experiments.

	Trajectory tracking		Peg-into-hole	
	Initial study	Update learning	Initial study	Update learning
Our method	138s	22s	64s	18s
Method [23]	146s	163s	68s	74s

We applied incremental learning in the second experiment. The learning process is two-way as shown in Fig. 8. In this case, every robot can upload what they learned in the different circumstances to revise and optimize the solution to a kind of problems instead of only one problem. With the growth of the number of trails, the skills mastered became more and more and were optimized by incremental learning. And as shown in Fig. 8, it led that the time used to learn and the length of trajectory both converged to optimal solution. Fig. 9(a) shows the trend of the time and the length of trajectory when robots pegged their shaft into the hole. Fig. 9(b) shows the trend of trajectory tracking. And as what we can see, they all become less in the process of experiment. End results are just as shown in TABLE 3. Learning from scratch requires much more time than just updating the network. The result proved that the model we proposed was really make robot more intelligent and they learned from each other. Also, the learning and working efficiency of cloud robot with incremental learning is higher than that of traditional batch learning through our experiment. Whenever a new task is processed, it is not necessary to rebuild all the knowledge bases, but only to update the changes caused by the new tasks on the basis of the original knowledge bases. Incremental learning makes full use of historical training results in the current sample training, thereby significantly reducing the time for subsequent training.

We proved that the learning and working efficiency of cloud robot with incremental learning is higher than that of traditional batch learning through our experiment.

IX. CONCLUSION

This paper presents an effective collaborative learning method that incorporates Incremental Learning and Meta Learning to train cloud robots. The knowledge is learned by themselves rather than being added and edited by human. Furthermore, the knowledge learned by a robot can be shared with other robots once uploaded to the cloud.

Our experiment results show that the proposed method using Incremental learning is more efficient than the without, proving that incremental learning and Meta Learning decrease the amount of time necessary to relearn a skill. It makes it not necessary for robots to relearn from the beginning. Also, our results proved that implementing our method is not only feasible but also valid.

There are still some shortcomings in the current research. At present, although we increase the learning and working efficiency of cloud robot by using Incremental Learning and Meta Learning, we only implement some simple functions. Whether our system is intelligent enough to accomplish a lot of complicated tasks remains to be confirmed.

REFERENCES

[1] J. J. Kuffner, "Cloud-enabled robots," in *Proc. IEEE-RAS Int. Conf. Humanoid Robot.*, Nashville, TN, USA, Nov. 2010, pp. 176–181.

[2] E. Guizzo, "Cloud robotics: Connected to the cloud, robots get smarter," IEEE Spectr., Technol., Eng., Sci. News, USA, Tech. Rep., 2011. [Online]. Available: https://spectrum.ieee.org/automaton/robotics/robotics-software/cloud-robotics

[3] K. Pereida, M. K. Helwa, and A. P. Schoellig, "Data-efficient multirobot, multitask transfer learning for trajectory tracking," *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 1260–1267, Apr. 2018.

[4] M. Zambelli and Y. Demiris, "Online multimodal ensemble learning using self-learned sensorimotor representations," *IEEE Trans. Cogn. Develop. Syst.*, vol. 9, no. 2, pp. 113–126, Jun. 2017.

[5] L. Kong, X. Chen, X. Liu, Q. Xiang, Y. Gao, N. B. Baruch, and G. Chen, "AdaSharing: Adaptive data sharing in collaborative robots," *IEEE Trans. Ind. Electron.*, vol. 64, no. 12, pp. 9569–9579, Dec. 2017.

[6] T.-H.-S. Li, C.-Y. Liu, P.-H. Kuo, Y.-H. Chen, C.-H. Hou, H.-Y. Wu, C.-L. Lee, Y.-B. Lin, W.-H. Yen, and C.-Y. Hsieh, "Reciprocal learning for robot peers," *IEEE Access*, vol. 5, pp. 6198–6211, 2016.

[7] H. He, T. M. Mcginnity, S. Coleman, and B. Gardiner, "Linguistic decision making for robot route learning," *IEEE Trans. Neural Netw. Learn.Syst.*, vol. 25, no. 1, pp. 203–215, Jan. 2014.

[8] S. Cho and S. Jo, "Incremental online learning of robot behaviors from selected multiple kinesthetic teaching trials," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 43, no. 3, pp. 730–740, May 2013.

[9] L. Wang, M. Liu, and M. Q.-H. Meng, "Real-time multisensor data retrieval for cloud robotic systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 2, pp. 507–518, Apr. 2015.

[10] J. Lones, M. Lewis, and L. Canamero, "A hormone-driven epigenetic mechanism for adaptation in autonomous robots," *IEEE Trans. Cogn. Devel. Syst.*, vol. 10, no. 2, pp. 445–454, Jun. 2018.

[11] A. J. Glover and G. F. Wyeth, "Toward lifelong affordance learning using a distributed Markov model," *IEEE Trans. Cogn. Devel. Syst.*, vol. 10, no. 1, pp. 44–55, Mar. 2018.

[12] Roboearth.ethz.ch. (2010). *What is RoboEarth?* [Online]. Available: http://roboearth.ethz.ch/

[13] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 38, no. 2, pp. 156–172, Mar. 2008.

[14] C. Finn, T. Yu, and T. Zhang, "One-shot visual imitation learning via meta-learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1087–1098.

[15] T. Yu, C. Finn, S. Dasari, A. Xie, T. Zhang, P. Abbeel, and S. Levine, "One-shot imitation from observing humans via domain-adaptive meta-learning," Feb. 2018, *arXiv:1802.01557*. [Online]. Available: https://arxiv.org/abs/1802.01557

[16] A. M. Ghalamzan E., C. Paxton, G. D. Hager, and L. Bascetta, "An incremental approach to learning generalizable robot tasks from human demonstration," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 5616–5621.

[17] M. Chen and J. Shang, "Recursive spectral meta-learner for online combining different fault classifiers," *IEEE Trans. Autom. Control.*, vol. 63, no. 2, pp. 586–593, Feb. 2018.

[18] F. Stulp and O. Sigaud, "Robot skill learning: From reinforcement learning to evolution strategies," *Paladyn. J. Behav. Robot.*, vol. 4, no. 1, pp. 49–61, 2013.

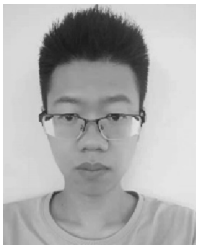
[19] S. Calinon and A. Billard, "What is the Teacher's role in robot programming by demonstration?—Toward benchmarks for improved learning," *Proc. Psychol. Benchmarks Hum.-Robot Interact.*, vol. 8, pp. 441–464, Jan. 2007.

[20] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Springer Handbook of Robotics*. Berlin, Germany: Springer-Verlag, 2008, pp. 1371–1394.

[21] M. Ciocarlie, C. Pantofaru, K. Hsiao, G. Bradski, P. Brook, and E. Dreyfuss, "A side of data with my robot," *IEEE Robot. Automat. Mag.*, vol. 18, no. 2, pp. 44–57, Jun. 2011.

[22] C. L. P. Chen and Z. Liu, "Broad learning system: An effective and efficient incremental learning system without the need for deep architecture," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 1, pp. 10–24, Jan. 2018.
[23] K.-T. Song, S.-H. Song, and H.-S. Liu, "Trajectory modification of a cloud learning robot," in *Proc. Int. Autom. Control Conf. (CACS)*, Nov. 2017, pp. 1–6.

GUANGLONG DU received the Ph.D. degree in computer application technology from the South China University of Technology, Guangzhou, China, in 2013. He is currently an Associate Professor with the Computer Science and Engineering School, South China University of Technology. His research interests include intelligent robotics, human–computer interaction, artificial intelligence, and machine vision.



ZHIYAO WANG received the B.S. degree in software engineering from the Wuhan University of Technology, Wuhan, China, in 2018. He is currently pursuing the master's degree in computer science and technology with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, China. His research interests include physiological electrical signal analysis and human–robot interaction.

ZHELIN LI received the Ph.D. degree in computer application technology from Design, South China University of Technology, Guangzhou Higher Education Mega Centre, Panyu, Guangzhou. He is currently the Vice Director of the Guangdong Human–Machine Interaction Design Engineering Research Center. His research interests concentrate on the human factor and ergonomics, particularly in terms of the man–machine collaboration and virtual digital human behavior.