ROBOMECH Journal

RESEARCH ARTICLE Open Access

Rapid prototyping for series of tasks in atypical environment: robotic system with reliable program-based and flexible learning-based approaches

Hiroshi Ito[1*] and Satoshi Nakamura[2]

Abstract
We propose a novel robotic system that combines both a reliable programming-based approach and a highly generalizable learning-based approach. How to design and implement a series of tasks in an atypical environment is a challenging issue. If all tasks are implemented using a programming-based approach, the development costs will be huge. However, if a learning-based approach is used, reliability is an issue. In this paper, we propose novel design guidelines that focus on the respective advantages of programming-based and learning-based approaches and select them so that they complement each other. We use a program-based approach for motions that is rough behavior and a learning-based approach for motion that is required complex interaction between robot and object of robot tasks and are difficult to achieve with a program. Our learning approach can easily and rapidly accomplish a series of tasks consisting of various motions because it does not require a computational model of an object to be designed in advance. We demonstrate a series of tasks in which randomly arranged parts are assembled using an actual robot.

Keywords: Autonomous robot, Assembly, Motion generation, Deep predictive learning

Introduction

Robotic automation has traditionally been used mainly for repetitive tasks in known environments, such as automotive assembly. However, in recent years, there has been a growing need to automate atypical tasks in unknown or undeveloped environments that are difficult to perform with conventional robotic technology. For example, shortages of workers are becoming more serious at construction and maintenance sites where dangerous and heavy work is required. For robots to perform on-site work, the emphasis is not on the ability to repeat simple tasks at high speed and with high precision as in the past but on the ability to perform tasks reliably even in unexpected situations. In factory automation, it is also desirable for robots to work autonomously in a new work environment without humans having to teach them operations in detail since a quick response to frequent process changes is required for small-volume production. However, there are several issues that need to be addressed in order to design a robot system that can perform atypical tasks.

Big issues for design a robot system for perform series of task in atypical tasks is the implementation cost. At atypical work sites, robots need to be adaptive to the conditions of work objects and the positioning of parts. To execute a series of tasks autonomously, three actions must be done: (1) recognizing the work situation and planning a procedure, (2) recognizing the position, shape, and posture of parts, tools, etc., and (3) controlling the motion of the robot. However, if we were to implement these functions for each task, the development

*Correspondence: hiroshi.ito.ws@hitachi.com
[1] Center for Technology Innovation - Controls and Robotics, Research & Development Group, Hitachi, Ltd., Ibaraki 312-0025, Japan
Full list of author information is available at the end of the article

costs would be huge. In addition, a high level of expertise is required to developtasks that involve contact with an object (screw fastening and fitting) or are difficult to describe in a program (handling flexible objects). Therefore, learning-based approaches have been proposed to accomplish complex tasks that are difficult to accomplish with conventional programming-based approaches [1–3].

However, learning-based approaches have safety and reliability issues. Unlike image recognition and natural language processing, robots perform tasks that involve physical contact in the real world. This can lead to collisions and other dangers. Therefore, it is necessary to have both "safety" to prevent failure and damage as a result of physical contact and collision and "reliability" to ensure that behavioral rules are improved through learning. For the former, safety has been ensured by detecting abnormalities and loads of robots by analyzing signals such as current values, acoustic sensors, and acceleration sensors [4–7]. However, the learning approach is a black box because it acquires behavioral rules on the basis of data, and there are still problems with reliability. Some methods have been proposed to use a model-based controller as prior knowledge for deep reinforcement learning [8, 9] or to monitor the state of a system in real time during operation generation and execute a predetermined recovery action when an anomaly is detected [10–12]. However, none of these methods focus on the reliability of behavior.

In this paper, we aim to accomplish complex tasks in an atypical environment in a simple and rapid way that is both reliable and flexible. How to design and implement a set of tasks in an atypical environment is a challenging issue. If we try to implement all tasks using a programming-based approach, the implementation costs will be huge. However, if a learning-based approach is used, reliability becomes an issue. In this paper, we propose a novel robotic system that focuses on the respective advantages of reliable program-based and flexible learning-based motion generation, which complement each other. To verify the effectiveness of the proposed method, as a first step for an atypical environment task, an assembly task with randomly placed tools and parts is performed with a real robot. This task is very challenging compared with general assembly tasks because it requires trajectory planning based on the positions of randomly placed parts and tools, as well as their positioning and screw fastening. The contributions of this paper are as follows.

1. Categorization of robotic tasks based on the therblig: A method is proposed for designing a robotic system using therbligs that visualizes and improves the efficiency of human tasks. The robot's various tasks (e.g., grasping, carrying, fitting) were categorized, and actions that can be realized with relatively simple definitions were defined as program-based ones, while complex actions were defined as learning-based ones. For example, a program-based approach is used for operations such as reaching, where the robot moves dynamically, and safety and reliability are required, and a learning approach is used for operations such as screw fastening and fitting, where the robot's work area is limited, and the operations in this area are difficult to accomplish with a program. The system designer can easily and rapidly realize a robotic system in which programming and learning approaches coexist by implementing various functions on the basis of the design guidelines presented in this paper.
2. Realizing a series of tasks in an atypical environment: The robot can flexibly generate motions even in an atypical environment where the placement of parts, work areas, and tools change, and a series of assembly tasks can be accomplished with an 80% success rate. In the experiment, the alignment of each part is important because the assembly work is performed using parts placed in random positions. Our learning-based approach allows the robot to acquire the desired operation by being taught the motion using teleoperation and learning models. Therefore, there is no need to design a computational model of an object in advance. During motion execution, the robot can perform screw fastening and object alignment on the basis of its sensor information in real time.

Design concept

How to design and implement a series of tasks in an atypical environment is a challenging issue. If all the tasks are implemented using a programming approach, the costs of development will be huge, and if a learning-based approach is used, the data collection cost, learning cost, and reliability will become issues. In this section, we focus on the advantages of both reliable programming-based and flexible learning-based methods, and we describe how to design robotic systems that complement each other's shortcomings.

Task categorization

The issue is whether a programming-based or learning-based approach is more suitable for implementing each function of a robot and how best to combine them. As a solution to this problem, in this paper, we define a classification of robot functions and the execution order of each function with reference to the therblig. Therbligs are 18 kinds of elemental motions that a worker needs in order to perform a task [13, 14]. In the field of industrial engineering, therbligs are used to visualize the content and series of human work and to propose and more efficient work methods and improve them. Here, we describe how to design a robotic system on the basis of the therblig.

First, we extracted 12 kinds of elemental motions for robotic system design while excluding six elemental motions that should be unnecessary for design the robot tasks, these are stopped motion elements that is no longer needed for work (i.e. "hold", "unavoidable delay", "avoidable delay", "rest"), "pre-position" element that become unnecessary by optimizing in advance and "find" element that is almost same as "search" and seldom use. Each elemental motion is explained using the screw handling in assembly tasks as an example.

- Search: Searching the location of an object with the eyes or hands. The five senses other than the eyes, such as when searching for holes with the fingertips, are also equivalent. Find out where the screws are.
- Inspect: Determining an object by a defined standard (quality or characteristics) using visual, auditory, tactile, etc. Check how tightly screws have been tightened between parts.
- Select: Choosing one out of several. Select an appropriate screw from several screws.
- Grasp: Grasping an object with hands or fingers. Grab the screw.
- Release load: Releasing control of an object. Release the screw.
- Transport empty: Reaching an object with an empty hand. Reach out to where the screw is.
- Transport load: Moving an object in hand from one place to another. Carry the screw.
- Position: Adjusting the position and orientation of objects. Adjust the position to insert the screw into the hole.
- Use: Manipulating tools, instruments, and equipment for a purpose. Tighten the screws.
- Assemble: Joining two parts together. Use the screws to fix the two parts together.
- Disassemble: Separating multiple components that are joined. Loosen the screws and disassemble the parts.
- Plan: Planning to do this next or later. Think about the procedure of the task.

Implementing all these 12 kinds of elemental motions with a programming-based approach would lead to huge development costs. Therefore, we propose a design method (guidelines) for a robot system with reliability and versatility by classifying the 12 motions into three categories on the basis of the characteristics of each function and then implementing them as modules. Table 1(a) shows three categorized elemental motions, and (b) shows the corresponding robot functions. (1)–(3) Show the three categories, and their details are as follows. (1) "Perception" is a motion to judge or measure without actual robot action based on information such as vision, and corresponds to functions such as image recognition, object detection, and position estimation based on the robot's visual information. It is also used for checking the task status (progress) and motion planning based on visual information. Since deep learning has made recognition technologies more diverse and accurate, it is possible to choose a method of implementation in accordance with the task. Template matching [15], which is conventionally used, is effective when the number of target objects is limited or patterned, while learning-based object recognition algorithms [16–18] are effective for various types and complex shapes of objects. (2) "Motion" is relatively simple motion that have not complex interactions between robot and object of robot tasks, and it also requires a high level of reliability and safety because the robot moves widely in the real world. Many of the motions are somewhat patterned, such as "take object A and place it at position B." Teaching playback, point-to-point (PtP), and trajectory planning algorithms such as rapidly-exploring random tree (RRT) [19, 20] are examples of implementation methods. (3) "Sensorimotor" is a motion that is necessary to have complex interactions between robot and object of robot task, and relies on the five senses and adjusts the position and force

Table 1 Categorization of robot functions based on therbligs

	(1) Perception	**(2) Motion**	**(3) Sensorimotor**
(a) Basic motion	Search Inspect Select Plan	Grasp Release load Transport empty Transport load	Position Use Assemble Disassemble
(b) Key function of robot control	Image recognition Object detection Position estimation	Point-to-point motion Interference avoidance	Learning-based motion generation

in accordance with the situation. For tasks that are difficult to describe in a program, such as screw tightening and fitting, a learning-based motion generation method would be effective. The diversification and development of learning-based approaches makes it possible to perform complex tasks [21–24]. Deep reinforcement learning, imitation learning, and other implementation methods are examples of implementation methods.

We have described the categorization of robot functions and examples of their implementation, referring to the therblig. In the next section, we describe the execution procedure and how to combine the three functions.

System architecture

When a robot is performing a task, the timing at which tasks are switched and the order in which the motion and sensorimotor are executed are issues. Figure 1 shows the execution procedure (system configuration) for the series of tasks proposed in this paper. The squares in the figure represent each of the functions defined above (hereafter referred to as modules). The modules consist of a "perception module" for recognizing situations, a "process manager" for switching tasks, and a "motion module" and a "sensorimotor module" for executing subtasks. On a simple production line, each task is executed in a feed-forward sequence. However, with a pre-planned work schedule, it is not possible to respond to unexpected events. In particular, since this paper assumes an atypical environment where the positions of objects change, process control in a feed-forward manner is difficult. To manage work processes in an atypical environment, it is necessary to recognize where objects are and what state they are in on the basis of vision information. In this paper, we use the perception module of Fig. 1 (1) to detect the position and state labels of objects. (2) The process manager switches tasks using real-time process planning methods [25, 26] or pre-designed if-then rules.

Next, we describe the series in which tasks are executed as shown in Fig. 1c. In this paper, a task consists of up to three kinds of modules. A typical robotic system consists of "perception" and "motion planning and control." It recognizes object names and locations on the basis of vision and executes trajectories that satisfy geometric, mechanistic, and dynamic constraints on the basis of the recognition results. If we apply this to the categorization in Table 1, the task will transition in the order of "perception module" and "motion module." However, not all tasks can be realized with a program-based motion module. Some tasks define the "motion module" to be followed by the "sensorimotor module." Basically, a programming-based approach is used for rough motions that are defined as "Motion" in Table 1, and a learning-based approach is used for complex motions that are defined as "Sensorimotor" in Table 1. Thus, the number of modules (subtasks) varies depending on the task. For example, Task 1 in Fig. 1 consists of subtask 1 to pick up the object and Subtask 2 to place it at the target position. Subtask 1 generates a grasping motion with the motion module on the basis of the position information of the target object recognized by the perception module. Similarly, subtask 2 generates a motion to place the object at the goal position with the motion module on the basis of the goal position information recognized by the perception module. However, if positioning accuracy is required, such as in assembly work, it is impossible to complete a series of tasks by simply placing the object at the goal position and it is required to adjust position according to state of each parts that would be assembled, such as adjustment

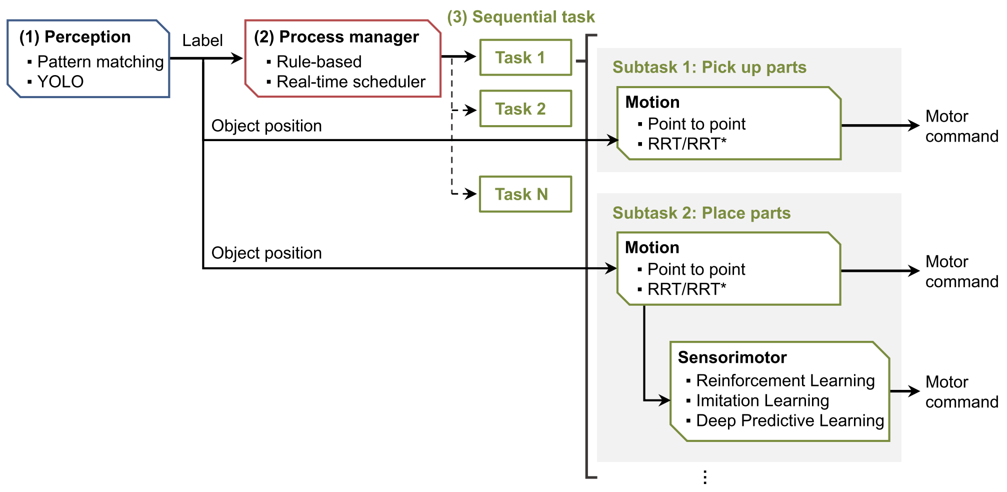


Fig. 1 Software architecture for performing series of tasks

motion with vision sensing or searching motion with force feedback. In addition, implementing a high-precision positioning algorithm using a programming-based approach requires complex programming and real-world tuning. Therefore, we use the sensorimotor module to adjust the position after the motion module. This makes it possible to perform complex tasks robustly with lower development costs.

As described above, by defining (implementing) various functions and execution procedures for the robot on the basis of the task categories and design policies proposed in this paper, a robot system can be constructed in which a highly reliable program base and a highly robust learning base coexist.

Set up for verification

To verify the effectiveness of the proposed method, we performed an assembly task in an atypical environment. Here, we describe the experimental setup and the experimental task.

Robotic hardware

Figure 2 shows the experimental setup. As shown in (a), two robot arms (KUKA LBR iiwa 14 R820) are used, and a robotic hand (Robotiq 2F-85 Adaptive Gripper) and an RGB camera (Buffalo BSW500M Series) are attached to the end of the arms. The other equipment includes a bird's-eye-view camera (Intel RealSenes D435i) for taking full view of work place, and an electric screwdriver connected to a tool balancer. The workbench, cover, base, and screws are the parts needed for assembly. (b) Shows an enlarged view near the robotic hand when hand grasping the screw driver. As shown in (b), hand camera installed near base of robotic hand and it can take an image of near the finger of robotic hand. The robot arms can be remotely controlled using a joystick to teach the desired motion. Impedance control can also be performed by using a torque sensor at each joint. For example, even if the axis of the driver and the screw hole is misaligned when tightening a screw and the hand is overloaded, it can prevent the screw and screw hole from being overloaded by passively moving the hand. However, since there is an inverse relationship between arm strength and positioning accuracy with impedance control, and the arm does not move accurately in relation to the hand position command, the control mode is switched depending on the task. In this paper, we used position control mode when executing the motion module and impedance control mode when executing the sensorimotor module, which involves contact with the object.

Assembly task for verification

Recently, robots are required to have the ability to handle high-mix low-volume production, rather than the conventional ability to repeat simple tasks at high speed and with high precision. These robots must have the ability to work autonomously in atypical environments. As mentioned above, in this paper, we focus on assembly tasks in the atypical environment shown in Fig. 3. To evaluate such tasks in this environment, the parts and workbench are not fixed but are randomly placed within the robot's operating range before work is started. The assembly task consists of five steps: (1) pick and place a randomly placed base, (2) pick and place a randomly placed cover, (3) grasp the electric screwdriver, (4) pick up a screw (attach it to the screw bit with the magnetic bit on the tip of the electric driver), and (5) screw the four corners of the cover to the base. To avoid contact between the robot and floor, the assembly is performed on a workbench. We

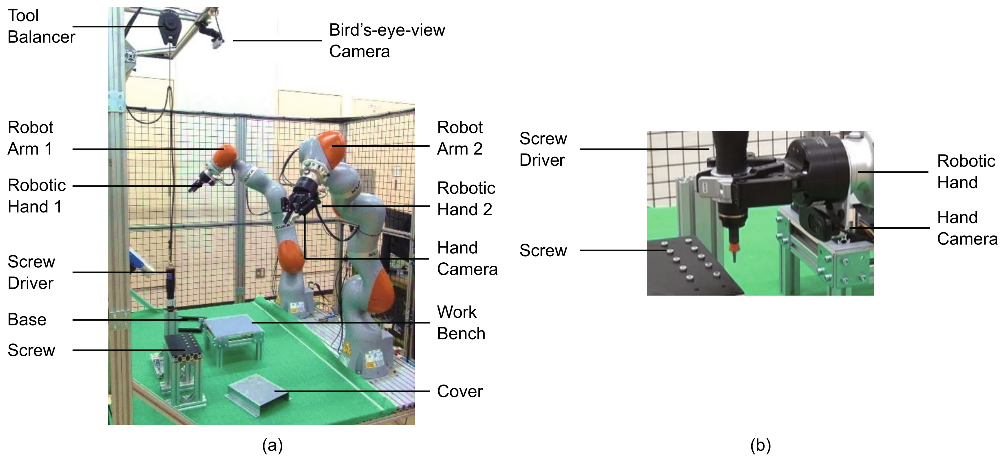


Fig. 2 Experimental environment: sensors and equipment

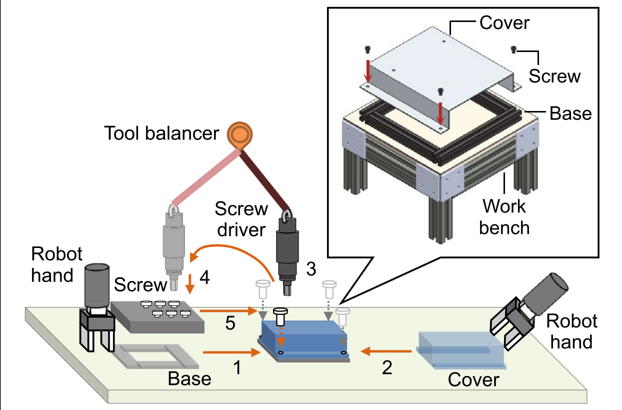


Fig. 3 Assembly tasks in atypical environment. Each part and tool are randomly placed

evaluated the success rate of the proposed method by performing a series of assembly tasks 10 times. In these tasks, some complex behavior that needs interaction between robot and object for assembly task is required, for example, align two parts while considering position and orientation of each parts so that the two parts overlap exactly, or adjust position and force of the hand to fit the screw to the tip of the screwdriver.

System configuration

In this section, we describe the specific method to configure the robot system with classification of the experimental tasks and the specific implementation method. Table 2 shows the tasks and their execution procedures. The assembly task consisted of five tasks, which were executed in order from 1 to 5. The numbers in each task correspond to the numbers in Fig. 3. The subtasks were executed in order from A to C. The implementation method for each function is described below.

Process manager

We used the Single Shot Multibox Detector (SSD) [17] as an object recognition algorithm based on deep learning to recognize the work process. To reduce the implementation costs, we used the TensorFlow Object Detection API [27]. The robot performed rule-based task switching on the basis of object labels recognized by SSD. For example, when two parts are to be assembled, the name of each part is output as an object label before assembly, and the states are combined after assembly. If a part name is output, the corresponding action (motion or/and sensorimotor module) will be performed; if a state is output, the next task will be performed. Thus, a series of tasks can be accomplished by combining object recognition and rule-based process planning. General object recognition algorithms, such as template matching, have low robustness to illumination changes. In particular, it is difficult to accurately recognize metals such as assembly parts because the way an object looks depends on how the light hits it (reflection angle). In comparison, SSD can perform object detection in real time and robustly against illumination and background changes. Furthermore, it can simultaneously predict location information in addition to object label information. We can share the modules and reduce development costs by passing the label information predicted by SSD to the process manager and the location information to the motion module.

Perception module

Since SSD detects the positional information of objects in the two-dimensional plane of a camera image, it cannot be used directly for robot control. For the robot to perform tasks on the basis of location information from SSD, the information is converted into 3D coordinates of the real environment. The camera used for perception in this paper is bird's-eye-view camera that is shown in

Table 2 Task categorization for assembly work

Task	Sub-task	(A) Perception	(B) Motion	(C) Sensorimotor
(1) Base operation	Picking	Base	Pick up base	–
	Placing	Workbench	Place base on workbench	–
(2) Cover operation	Picking	Cover	Pick up cover	–
	Moving	Workbench	Layer cover over base	Alignment adjustment
	Placing	Workbench	Place cover on base	–
(3) Take screwdriver	Grasping	Screwdriver	Reach for screwdriver	Position adjustment
	Moving	Screw holder	Move to screw holder	–
(4) Pick up screw	Moving	Screw	–	Approach screw
	Grasping	Screw	–	Pick up screw
(5) Screw Fasten	Moving	Screw hole	Move to screw hole	–
	Fasten	Screw hole	Fasten screw	–

Fig. 2, and it has a one-to-one correspondence between each pixel of an RGB image and depth image. Therefore, the center coordinates of the object recognized by SSD (f_x,f_y) are converted to 3D positional information (x, y, z) using a depth image. In addition, to enable the process manager to check the progress of the work, we trained the module to change the recognition result in accordance with the state of the object.

Motion module

In the motion module, the robot generates a trajectory on the basis of object position information (x, y, z). Point-to-Point (PtP) control is used to generate the robot's motion. PtP control is implemented in common industrial robots and can be implemented at a small cost. The robot automatically generates motions toward the goal position on the basis of the position information of the object recognized by the perception module. For example, in Task (2) of Table 2, the robot grasps the cover and places it at the goal position. However, since both the cover and the base stand are placed in random positions and postures, it is difficult to accurately align both positions. Therefore, in the subtask "Moving," the sensorimotor module is used to adjust the position of the cover to overlap the base platform perfectly.

Sensorimotor module

For the sensorimotor module, we use "Deep Predictive Learning (DPL)," which is able to acquire desired behaviors with low data-collection and learning costs [28–30]. This method learns a time series of sensory-motor information when the robot operates in the real world, enabling it to perform complex tasks that are difficult to realize with programs. Specifically, it consists of three steps: (1) collect sensory-motor information (e.g., camera image, joint angle, and torque) with the robot as learning data when a human teleoperates the robot or performs direct teaching, (2) input the sensor information x_t at time t into the model, output the sensor information $\hat{y}_{t+1}$ at the next time $t+1$, and update the weights of the model to minimize the error between the predicted value $\hat{y}_{t+1}$ and the true value x_{t+1}, and (3) at execution time, the robot is made to generate sequential motions by inputting the robot's sensor information x_t and inputting the predicted value (motion command value) to the robot for the next time. This method can be used to perform various tasks, such as flexible object handling, which is difficult to do with the conventional method [31, 32].

Figure 4 shows the details of the sensorimotor module. Using a raw visual image of a robot increases the calculation cost, making it difficult to generate motion in real time. When a visual image is simply resized, the important areas for the task are also compressed into a small size, making it difficult to recognize detailed tasks and states. In a typical environment where the position of an object does not change, it is sufficient to crop a specific area of the camera image. However, in an atypical environment where the position of an object randomly changes, simple cropping cannot be used. In this paper, we extract and resize images of the surroundings on the basis of the location information of the object recognized by the perception module, as shown in Fig. 4a. The object position is extracted only at the initial time $t=0$, and an image of the same region is used continuously after that to ensure real-time performance. Figure 4b shows the motion generation model (DPL module) used in this paper. The model consists of a convolutional layer [33] that extracts image features from the robot's visual information, a long short-term memory (LSTM) [34] that learns image features and the robot's body information in a time series, and a transposed convolution layer that reconstructs images from image features. The LSTM predicts the next-time image (situation) and motor command from the current sensor information. By learning visual and physical information simultaneously, the convolutional layer extracts the appropriate image features for motion generation

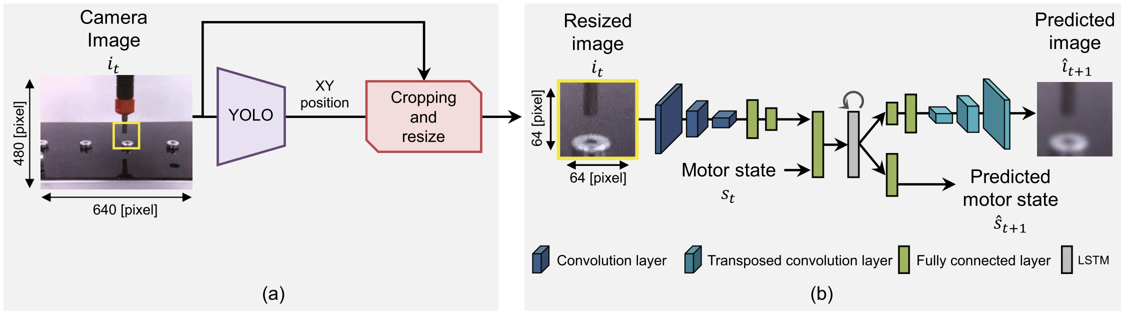


Fig. 4 Network architecture of sensorimotor module

[35]. The robot generates real-time motions after the motor commands predicted by the LSTM are inputted.

As shown in Table 2, the sensorimotor module is used for three tasks: (1) adjusting the position and orientation of the cover and base, (2) grasping the electric driver, and (3) adjusting the position of the screwdriver bit and the top of the screw (hexagonal hole) and inserting it. We designed the these tasks with sensorimotor modules since these have characteristics that is required interaction with robot and object of tasks, and also significant impact on the later stages of the process. If the positions of the cover and the base are misaligned, the cover may fall off the base during the task, or the screw-hole positions may not match, and the task may fail. In addition, the grasping position and direction of the electric driver will change the way in which screws are picked up and tightened. To accomplish a series of tasks with a high success rate, the adjustment operation of the sensorimotor module is indispensable. Note that we used program based approach for sub-task of fastening the screw since it was easy to achieve by using impedance control so that following hand position to screw hole when pre-examination, though it should be classified as sensorimotor motion based on the proposal.

Datasets for deep-learning method

Training data of perception module

In this paper, we used a pre-trained SSD model to train the recognition and grasping position of assembly parts. A total of 693 images of nine objects were taken using the bird's-eye-view camera at the top of the experimental apparatus: the workbench, base, grasping position of the base, cover, grasping position of the cover, the base on the workbench, the cover on the workbench, the screw storage area, and the electric screwdriver. The images were taken when the position and orientation of the object were randomly changed. We prepared a set of images of assembly parts and their labels (object name and location information) as training data. To increase the amount of training data, we performed image positioning, rotation, and flipping for data augmentation. In addition, by randomly varying the brightness and contrast of the images, we obtained an object recognition model that was robust to changes in illumination.

Training data for sensorimotor module

In deep predictive learning, the model learns sensor information as a robot operates in the real world. In this paper, we taught the robot the three motions shown in Table 2(C). Since the method for teaching an operation is different for each task, the details are given below.

First, we describe the teaching method for adjusting the position of the cover. In an atypical environment, the position and orientation of a cover change randomly each time. It is not easy to grasp a randomly placed cover and align its position and orientation so that it overlaps the base perfectly. Even if a joystick is used to remotely control a robot to teach movements, there is the problem of learning not proceeding well because human movements are inconsistent. In particular, it is difficult to achieve high-precision operation due to minute misalignment of the end state. Here, we used the motion teaching method shown in Fig. 5 to ensure that the training data contains consistency. (a) Shows the sample of collected image data by bird's-eye-view camera, and it is found that the center coordinates of the robot hand (yellow dotted line) and the cover (red dotted line) were not always on a straight line. Desired operation is that align the orientation of cover and then align the position of covers, like flow from (d) to (b). For achieving this motion, we collect the data that move position randomly from the aligned state, then rotate orientation randomly. When the model training, the time series of the collected data is inverted (played backwards). It is based on the ease of generating motions that shift from an aligned state to a random direction and position, and this made it possible to collect data that were consistent with the end state. Therefore, it is expected that the robot can move an object in a random initial position to the same end state each time like as shown in from (d) to (b) during motion generation. We acquired the image data of bird's-eye-view camera as input data and command data of hand positioning and rotating as output data of DPL. 504 training data where the orientation and position of the cover were changed, furthermore changing the grasping finger position and orientation of cover were acquired. Each piece of data was acquired for 10 s at a sampling rate of 10 Hz per piece of data.

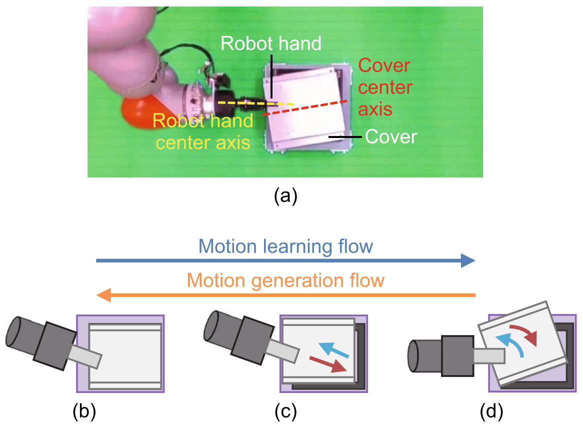


Fig. 5 Overview of motion teaching: position adjustment of cover and base

Next, we describe the teaching method for screw handling. Here, screw handling consists of the two operations shown in Fig. 6, the screw approach motion and the screw pickup motion. (a) Is a motion for bringing the electric screwdriver close to the screw (screw approach motion). Two motions are taught to the robot for the screw approach action: (1) leftward (positive y) movement and (2) forward (positive x) movement. As a recovery operation in the case of operation failure, we taught several operations to move the screwdriver directly above the screw from different directions as shown by the thin arrows in Fig. 6a. This allowed the robot to go back and continue the positioning operation even if the amount of movement of the hand position was too large and thus passed the target position. We acquired the image data of hand camera as input data and command data of hand positioning as output data of DPL. 441 training data where the approach motion when the position of the screw storage area was changed were acquired. From the bird's-eye-view camera, it was possible to recognize the screw storage area and generate motion toward its vicinity, but the resolution was not high enough to recognize the exact location of the screw. Here, the robot's hand camera was used to generate an accurate approach motion to the screw. (b) Is the action of fitting the bit of the electric screwdriver into the hexagonal hole at the top of the screw (screw pickup motion). In the screw approach operation, it is difficult for the bit of the electric screwdriver to stop exactly above the screw. Therefore, the pickup work was performed by moving (searching) back and forth and left and right while pressing the bit against the screw. Impedance control was used to prevent overloading between the robot and the screw. The robot was taught two motions as screw pickup motions: (1) perform adjustment in the left-right (y-axis) direction and (2) in the front-back (x-axis) direction. The six horizontal arrows shown in (b) are teaching positions, and 378 training data that is same contents as (a) were acquired. The training data were collected by a person using a joystick to remotely control the robot.

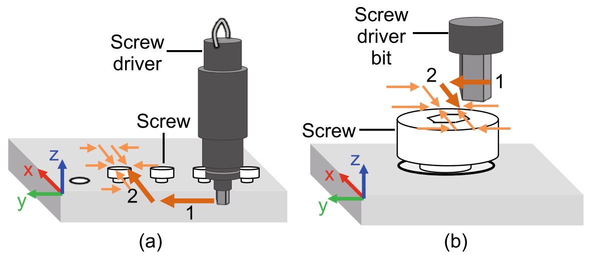


Fig. 6 Overview of motion teaching: screw approach and pickup

Finally, we describe the teaching method for electric screwdriver grasping. The robot grasped the electric screwdriver, which was placed on a holder. The height of the electric screwdriver was the same each time, but the screwdriver was placed at random positions. Therefore, the robot was taught the motion in the same way as the screw approach motion described above. We acquired 86 training data that is same contents as motion of picking up screw by randomizing the initial position of the driver each time. The robot was taught to move from the screwdriver grasping position to a random position. The time series was reversed (played backwards) and used as training data. This made it possible to execute position adjustment to align the robot hand with the screwdriver in a random position.

Results and discussion

In this section, we verify whether the perception module and sensorimotor module implemented using the learning approach worked properly. In addition, we describe the results of 10 assembly operations performed in an atypical environment as a validation of the effectiveness of the proposed method.

Object recognition

Figure 7 shows the results of object recognition using SSD. (a) Is the initial state of assembly, (b) is after placing the base, and (c) is the end-of-task state. The parts and tools required for assembly were recognized. The model

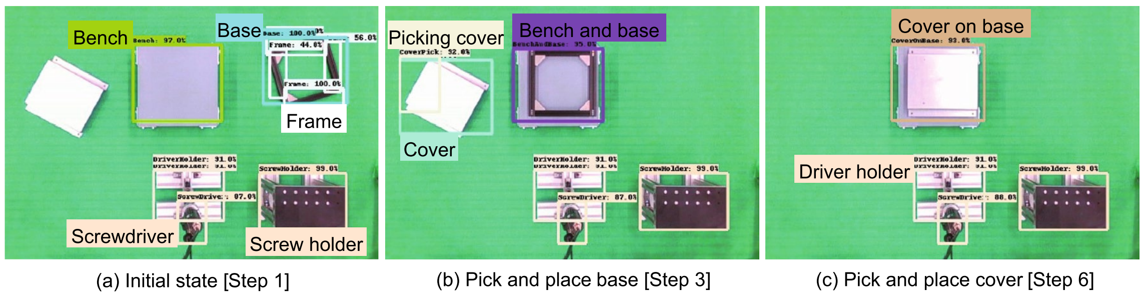


Fig. 7 Task execution result: object labels and states recognized by SSD

recognized the object name as a label when each part was on the floor and the state name as a label when they were combined. For example, as shown in Fig. 7c, the model output "Cover on Base" with the base and cover placed on top of the bench. By switching the recognition label in accordance with the status of the part, the process manager could check the progress status and switch tasks.

Cover adjustment motion

Figure 8 shows the results of adjusting the position of the cover. The positions of the workbench and the base were set up randomly. The center position of the image (intersection of the white dotted lines) is indicated to make it easier to understand the position change. (a) Shows that the robot learned to adjust the position of the cover. The robot adjusted the orientation to make a parallel line between the silver cover and the black base stand (step 2). Then, it adjusted the position of the cover and the base stand (step 3). Since the size of the cover and the base stand were the same, the robot adjusted the position to make the black base invisible. Steps 2 and 3 correspond to Fig. 5d, c, respectively, indicating that the robot generated a sequence of the motions. Thus, even if the base and cover were placed randomly and furthermore the grasping position were different each time, the robot could adjust its position accurately.

Next, we discuss the reusability of the cover adjustment module. We verified whether the robot, which had not learned the cover positioning operation, could adjust the position of the cover. In an atypical environment, the robot arm is expected to move appropriately in accordance with the position of the manipulated object. However, reusability (commonality and diversion) of tasks is expected because implementing each behavior would incur development costs. In this section, we will learn a behavior only with robot arm 1, shown in Fig. 2, and verify whether the same behavior can be executed with robot arm 2, which has not yet been trained. Each robot was installed symmetrically around the bird's-eye-view camera. Therefore, the image from the camera appeared differently to each robot working on the task. In Fig. 8a, the robot arm appears to be on the left side, while in (b), it is on the right side. Therefore, to transfer the model learned with robot arm 1 to robot arm 2, the input images and command values to the DLP were inverted. Specifically, the input images were flipped left and right, and the signs of the command values in the left and right (x) directions were reversed for position and orientation adjustment. (b) Is the execution result for robot arm 2. The direction and position of the cover could be adjusted even though the position of the robot arm was reversed from that of the learning process. By simply learning the motion of one robot arm, it was possible for the other robot to perform the same motion. This allows us to reduce the cost of motion learning. However, this is limited to cases where there is symmetry in the sensor information of the robot and the work object.

Screw pickup motion

Figure 9 shows a hand camera image of the robot during the execution of the screw handling operation. (a–d) Are the operation of approaching the screw, and (e–h) are the screw pickup operation. The robot was holding the electric screwdriver and stopped in front of the screw storage area. There were several screws, and the robot went to grab the leftmost one. At point (c), it recognized the end position of the screw and generated a return motion. When the tip of the screwdriver bit and the screw were aligned, it moved in the forward direction. Then, it switched to the screw pickup operation and executed the

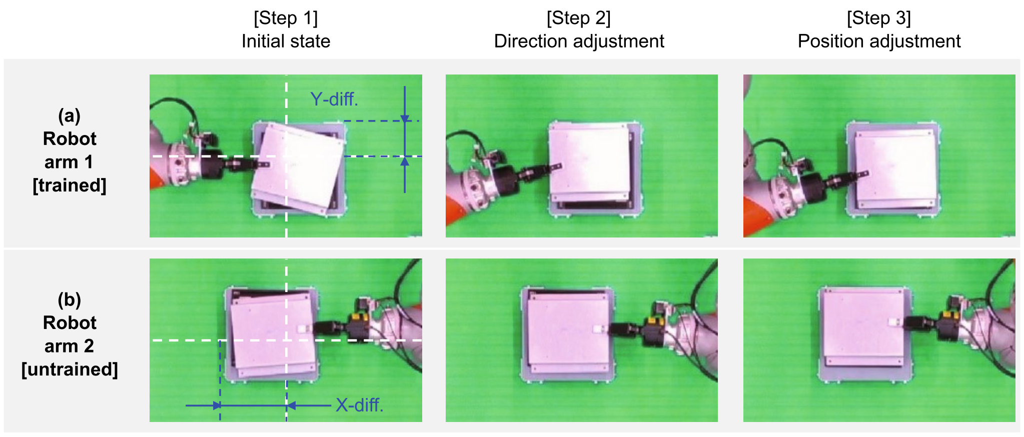
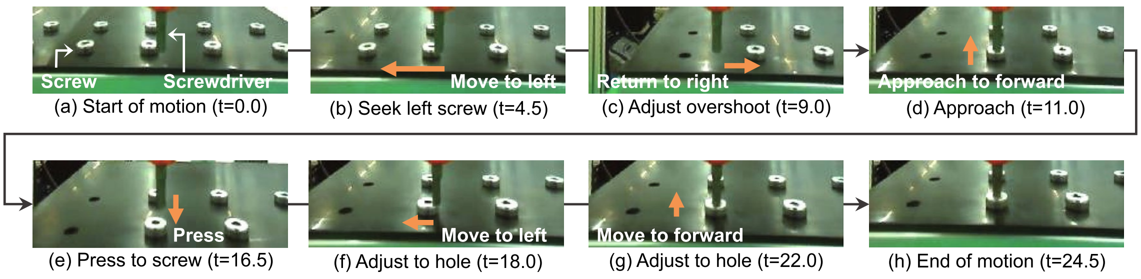


Fig. 8 Task execution result: positioning cover on randomly placed base


Screw
Screwdriver
Move to left
Return to right
Approach to forward
Press
Move to left
Move to forward
(a) Start of motion (t=0.0)
(b) Seek left screw (t=4.5)
(c) Adjust overshoot (t=9.0)
(d) Approach (t=11.0)
(e) Press to screw (t=16.5)
(f) Adjust to hole (t=18.0)
(g) Adjust to hole (t=22.0)
(h) End of motion (t=24.5)
Fig. 9 Task execution result: approaching top of screw and picking it up

search operation to fit the tip of the screwdriver into the hexagonal hole of the screw. The robot could perform the tasks shown in Fig. 6 by generating (adjusting) the motions in real time.

Verification of series of assembly tasks

To verify the effectiveness of the proposed method, assembly work was performed with random positions of parts, work tables, etc. Figure 10 shows the initial status of the parts and workbench before the start of assembly for each session. The white dashed line is shown in the center of the image to make it easier to understand the situation of misalignment. In (a–f), the work was started with the base and cover placed on both sides of the working table respectively, and in (g–j), the base and cover were placed on only one side of the table. Furthermore, in addition to the positions and orientations of the base and cover, the work was started with the positions of the workbench, screw storage area, and electric driver changed.

Figure 11 shows the assembly process with different initial positions for the objects. The numbers in the figure correspond to the task numbers in Table 2. In this section, we explain the software architecture of Fig. 1 in comparison with the actual operation. First, the perception module was used to check the current work status and detect the position of the object. Next, the process manager selected a task on the basis of the recognition results. The process manager was implemented with if-then rules. Task 1 was executed since the base was placed on the green desk. Task 1 consisted of two subtasks: pick up motion toward the recognized base and place motion toward the workbench. After Task 1 was completed, the perception module was used to check the work status. The base was now placed on top of the workbench, which changed the state name from workbench to “Bench and base” as in Fig. 7b. This change in the state name caused the process manager to select Task 2. Task 2 involved the motion module and sensorimotor module. Task 2-1 shows the grasped cover stacked on top of the base table, and it can be seen that a small misalignment occurred. If the cover were placed on the base in this state, it would fall off. Therefore, in Task 2-2, the cover adjustment module was used to align the position and orientation of both covers. Program-based execution of simple operations was followed by fine-tuning using a learning-based approach to generate reliable and generalizable motions. At the end of Task 2, the state name changed from “Bench and base” to “Cover on base” as in Fig. 7c. A series of tasks can be performed by executing

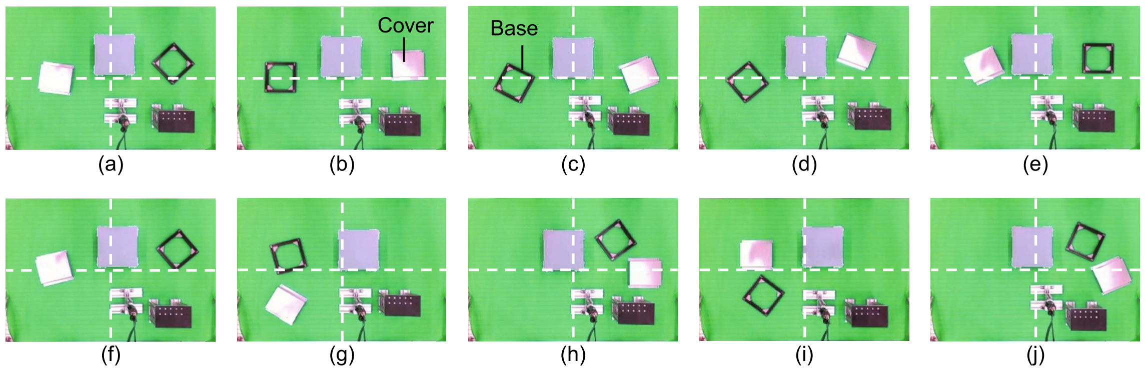


Fig. 10 Atypical environment used to verify sequence of tasks

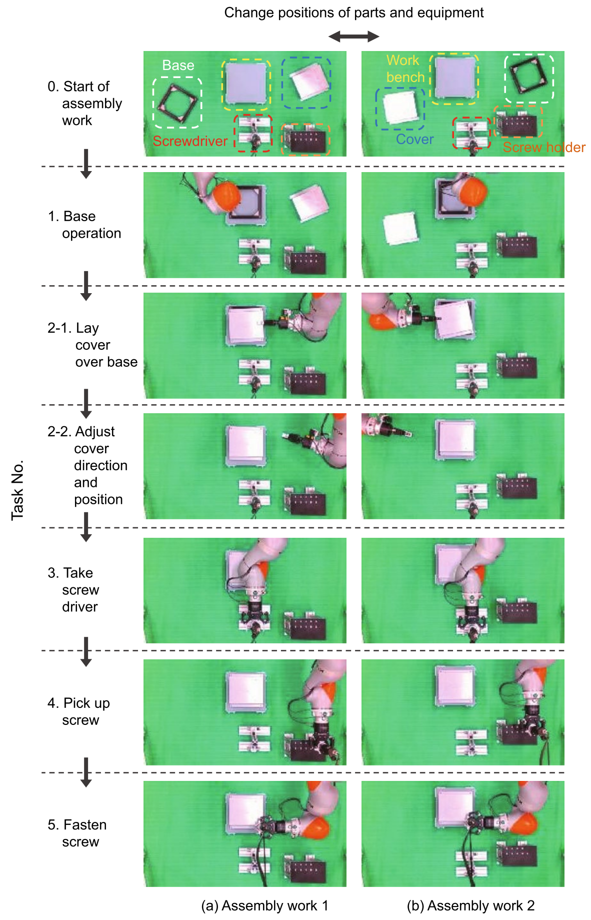


Fig. 11 Task execution result: parts and tools are in different positions

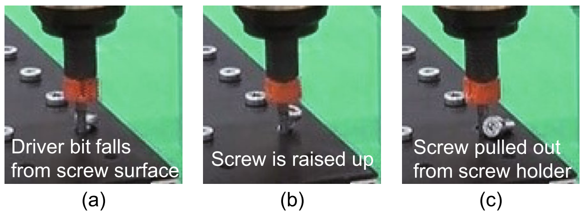

Driver bit falls from screw surface
Screw is raised up
Screw pulled out from screw holder
(a)
(b)
(c)

Fig. 12 Task execution result: task aborted due to failed screw pickup

Task 3 through Task 5 in the same manner as described above.

We confirmed that the robot could flexibly generate motions and realize a series of assembly operations even in an atypical environment where the positions of parts, workplaces, and tools changed. Ten assembly operations were performed, and the work was completed eight times. By implementing the various functions of a robot on the basis of the therblig, a robot system can be realized in which a reliable program-based approach and a flexible learning-based approach coexist.

Failure case

Figure 12 shows the case of a task failure. (a) Shows the screw pickup operation being executed. Normally, the screwdriver bit would be inserted into the screw hole, but the screwdriver bit came off the top of the screw. In this experiment, the screwdriver bit used had a built-in magnet for picking up screws. Therefore, in (b), the driver bit stuck to the side of the screw when it fell. In (c), the screw popped out of the screw holder, making it impossible to continue working.

We consider there to be two reasons for this failure. The first is that the accuracy of stopping the screw approach motion was low. The screwdriver bit may have come off the top of the screw during the next process because the position was slightly off when it should have stopped directly above the screw. The second is that the direction of the screw pickup motion was not appropriate. The screw approach/pickup motion uses a learning approach, which enables the operation to be performed with generalized performance. However, it is difficult to elucidate the causes of failures and take countermeasures because this approach is data-driven. One possible solution is to improve the accuracy of the operation by increasing the number of its learning patterns. Future tasks include real-time failure determination and estimation of recovery operations [11, 12, 25] and improvements to stability through hybridization with conventional control [36].

Conclusion

In this paper, we developed a novel robotic system that uses both a reliable programming-based approach and a robust learning-based approach. The program-based approach is used for four elemental motion that have not complex interactions between robot and object of robot tasks and is rough behavior, and the learning-based approach is used for four elemental motion that is required complex interaction between robot and object of robot tasks and that are difficult to describe in a program. Our learning approach does not require the prior design of a computational model of an object. The robot's visual-image and joint-angle information can be used to fasten screws and adjust the positions of objects. To verify the effectiveness of the proposed method, we created an assembly task with randomly placed tools and parts using a real robot. This task was challenging compared with a typical assembly task because it required trajectory planning based on the positions of the randomly placed parts and tools, and it also required that their positions be adjusted and that screws be fastened. As a future works, upgrade of process manager will be required. Current process manager is configured by simple if-then rules, however autonomous function to configur the motion modules from the stored modules will be required for adapting more atypical task without high implementation cost. And considering the best perception method will also be issues so that process manager determines the suitable task and motion by recognizing the situation of task and environment and automatically configure motions. Furthermore, we will update the system to be able to perform recovery operations based on work failure decisions in order to improve reliability.

Acknowledgements
Not applicable.

Authors' contributions
All authors contributed to the concept and overall development of this study. HI developed the software for the learning-based approach. NA implemented the rest of the software and performed the experiments. All authors worked together to wrote the manuscript. All authors read and approved the final manuscript.

Funding
Not applicable.

Availability of data and materials
Not applicable.

Declarations

Competing interests
The authors declare that they have no competing interests.

Author details
[1]Center for Technology Innovation - Controls and Robotics, Research & Development Group, Hitachi, Ltd., Ibaraki 312-0025, Japan. [2]Present Address: Center for Technology Innovation - Controls and Robotics, Research & Development Group, Hitachi, Ltd., Ibaraki 312-0025, Japan.

Received: 5 November 2021 Accepted: 17 February 2022
Published online: 04 March 2022

References

1. Arnold S, Yamazaki K (2019) Fast and flexible multi-step cloth manipulation planning using an encode-manipulate-decode network (em* d net). Front Neurorobot 13:22
2. Sasagawa A, Fujimoto K, Sakaino S, Tsuji T (2020) Imitation learning based on bilateral control for human-robot cooperation. IEEE Robot Autom Lett 5(4):6169–6176
3. Zeng A, Song S, Lee J, Rodriguez A, Funkhouser T (2020) Tossingbot: learning to throw arbitrary objects with residual physics. IEEE Trans Robot 36(4):1307–1319
4. Olsson E, Funk P, Bengtsson M (2004) Fault diagnosis of industrial robots using acoustic signals and case-based reasoning. In: European conference on case-based reasoning, Springer, pp 686–701
5. Hornung R, Urbanek H, Klodmann J, Osendorfer C, Van Der Smagt P (2014) Model-free robot anomaly detection. In: 2014 IEEE/RSJ international conference on intelligent robots and systems, IEEE, pp 3676–3683.
6. Jaber AA, Bicker R (2014) The optimum selection of wavelet transform parameters for the purpose of fault detection in an industrial robot. In: 2014 IEEE international conference on control system, computing and engineering (ICCSCE 2014), IEEE, pp 304–309
7. Cheng F, Raghavan A, Jung D, Sasaki Y, Tajika Y (2019) High-accuracy unsupervised fault detection of industrial robots using current signal analysis. In: 2019 IEEE international conference on prognostics and health management (ICPHM), IEEE, pp 1–8
8. Johannink T, Bahl S, Nair A, Luo J, Kumar A, Loskyll M, Ojea JA, Solowjow E, Levine S (2019) Residual reinforcement learning for robot control. In: 2019 international conference on robotics and automation (ICRA), IEEE, pp 6023–6029
9. Okawa Y, Sasaki T, Iwane H (2019) Control approach combining reinforcement learning and model-based control. In: 2019 12th Asian control conference (ASCC), IEEE, pp 1419–1424
10. Suzuki K, Mori H, Ogata T (2018) Motion switching with sensory and instruction signals by designing dynamical systems using deep neural network. IEEE Robot Autom Lett 3(4):3481–3488
11. Chen T, Liu X, Xia B, Wang W, Lai Y (2020) Unsupervised anomaly detection of industrial robots using sliding-window convolutional variational autoencoder. IEEE Access 8:47072–47081
12. Hung C-M, Sun L, Wu Y, Havoutis I, Posner I (2021) Introspective visuomotor control: exploiting uncertainty in deep visuomotor control for failure recovery. arXiv preprint arXiv:2103.11881
13. Price B (1989) Frank and lillian gilbreth and the manufacture and marketing of motion study, 1908-1924. Business and economic history, pp 88–98
14. Gilbreth FB (1909) Bricklaying system. MC Clark Publishing Company, New York
15. Brunelli R (2009) Template matching techniques in computer vision: theory and practice. Wiley, Hoboken
16. Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 779–788
17. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu C-Y, Berg AC (2016) Ssd: Single shot multibox detector. In: European conference on computer vision, Springer, pp 21–37
18. Mahler J, Liang J, Niyaz S, Laskey M, Doan R, Liu X, Ojea JA, Goldberg K (2017) Dex-net 2.0: deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. arXiv preprint arXiv:1703.09312
19. LaValle SM et al (1998) Rapidly-exploring random trees: a new tool for path planning
20. LaValle SM, Kuffner JJ Jr (2001) Randomized kinodynamic planning. Int J Robot Res 20(5):378–400
21. Duan Y, Andrychowicz M, Stadie BC, Ho J, Schneider J, Sutskever I, Abbeel P, Zaremba W (2017) One-shot imitation learning. arXiv preprint arXiv:1703.07326
22. Tobin J, Fong R, Ray A, Schneider J, Zaremba W, Abbeel P (2017) Domain randomization for transferring deep neural networks from simulation to the real world. In: 2017 IEEE/RSJ international conference on intelligent robots and systems (IROS), IEEE, pp 23–30
23. Levine S, Pastor P, Krizhevsky A, Ibarz J, Quillen D (2018) Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. Int J Robot Res 37(4–5):421–436
24. Yu T, Finn C, Xie A, Dasari S, Zhang T, Abbeel P, Levine S (2018) One-shot imitation from observing humans via domain-adaptive meta-learning. arXiv preprint arXiv:1802.01557
25. Tokuda S, Katayama M, Yamakita M, Oyama H (2020) Generating new lower abstract task operator using grid-tli. In: 2020 IEEE/RSJ international conference on intelligent robots and systems (IROS), IEEE, pp 6578–6584
26. Matsuoka S, Sawaragi T, Horiguchi Y, Nakanishi H (2016) Hierarchical planning for error recovery in automated industrial robotic systems. In: 2016 IEEE international conference on systems, man, and cybernetics (SMC), IEEE, pp 001406–001410
27. Huang J, Rathod V, Sun C, Zhu M, Korattikara A, Fathi A, Fischer I, Wojna Z, Song Y, Guadarrama S *et al.* (2017) Speed/accuracy trade-offs for modern convolutional object detectors. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 7310–7311
28. Noda K, Arie H, Suga Y, Ogata T (2014) Multimodal integration learning of robot behavior using deep neural networks. Robot Auton Syst 62(6):721–736
29. Kase K, Suzuki K, Yang P-C, Mori H, Ogata T (2018) Put-in-box task generated from multiple discrete tasks by ahumanoid robot using deep learning. In: 2018 IEEE international conference on robotics and automation (ICRA), IEEE, pp 6447–6452

30. Ichiwara H, Ito H, Yamamoto K, Mori H, Ogata T (2021) Spatial attention point network for deep-learning-based robust autonomous robot motion generation. arXiv preprint arXiv:2103.01598
31. Yang P-C, Sasaki K, Suzuki K, Kase K, Sugano S, Ogata T (2016) Repeatable folding task by humanoid robot worker using deep learning. IEEE Robot Autom Lett 2(2):397–403
32. Suzuki K, Kanamura M, Suga Y, Mori H, Ogata T (2021) In-air knotting of rope using dual-arm robot based on deep learning. arXiv preprint arXiv:2103.09402
33. Fukushima K, Miyake S, Ito T (1983) Neocognitron: a neural network model for a mechanism of visual pattern recognition. IEEE Trans Syst Man Cybern 5:826–834
34. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
35. Ito H, Yamamoto K, Mori H, Ogata T (2020) Evaluation of generalization performance of visuo-motor learning by analyzing internal state structured from robot motion. New Gener Comput 38:7–22
36. Suzuki K, Mori H, Ogata T (2021) Compensation for undefined behaviors during robot task execution by switching controllers depending on embedded dynamics in rnn. IEEE Robot Autom Lett 6(2):3475–3482

Publisher's Note