CORRECTED 3 MAY 2022; SEE FULL TEXT

SCIENCE ROBOTICS | RESEARCH ARTICLE

ARTIFICIAL INTELLIGENCE

Efficient multitask learning with an embodied predictive model for door opening and entry with whole-body control

Hiroshi Ito[1,2]*, Kenjiro Yamamoto[1], Hiroki Mori[2], Tetsuya Ogata[2]*

Robots need robust models to effectively perform tasks that humans do on a daily basis. These models often require substantial developmental costs to maintain because they need to be adjusted and adapted over time. Deep reinforcement learning is a powerful approach for acquiring complex real-world models because there is no need for a human to design the model manually. Furthermore, a robot can establish new motions and optimal trajectories that may not have been considered by a human. However, the cost of learning is an issue because it requires a huge amount of trial and error in the real world. Here, we report a method for realizing complicated tasks in the real world with low design and teaching costs based on the principle of prediction error minimization. We devised a module integration method by introducing a mechanism that switches modules based on the prediction error of multiple modules. The robot generates appropriate motions according to the door's position, color, and pattern with a low teaching cost. We also show that by calculating the prediction error of each module in real time, it is possible to execute a sequence of tasks (opening door outward and passing through) by linking multiple modules and responding to sudden changes in the situation and operating procedures. The experimental results show that the method is effective at enabling a robot to operate autonomously in the real world in response to changes in the environment.

INTRODUCTION

The activities of robots are extending from conventional well-defined environments, such as factories and warehouses, to society. In particular, it is expected that production efficiency will improve and that human work will be supported or replaced in various industries beyond manufacturing. Robots working in such fields are required to have the ability to work in various environments rather than the conventional ability to repeat simple tasks with high speed and high accuracy. However, there are several difficult issues to deal with when considering such applications.

The first is the cost of manually designing the models. In a limited environment such as a factory, it is possible to prepare the environment for a robot. However, because cities and homes are more complicated than a well-defined factory, a robot must work autonomously in response to changes in the environment. In general, it is necessary to give the robot an environment, a target object, and a physical model of itself. The functions of "perception" (*1*), which accurately recognizes the situation, and "motion planning and control" (*2*, *3*), which generates a trajectory, require these models. In recent years, with the advent of deep learning, each of these capabilities has been greatly improved. For example, perception has made it possible to recognize complex objects such as a wide variety of products, indefinite postures, and mixed products (*4*, *5*). In addition, for motion planning and control, methods have also been proposed for learning complicated assembly tasks and precision tasks (*6*, *7*). However, the design of an environmental model itself is indispensable, and this requires a high degree of expertise and substantial costs.

Learning-based approaches are expected to reduce the design costs of the model. By learning the relationship between a robot's sensor information and motion, recognition and motion generation models can be acquired simultaneously without precisely constructing an environment model. However, this approach also has the essential problem of a "learning data collection cost." For example, "deep reinforcement learning" can generate robust motions by learning to predict an appropriate joint angle from a visual image when the robot is conducting trial and error (*8*–*10*). In addition, because trial and error is conducted to maximize the expected reward, the emergence of new motions and optimal trajectories that humans have not thought of may be sought. However, collecting learning data involves a lot of trial and error in the real world; for example, one study reported 2 months of grasping an object a total of 800,000 times by using 14 robots (*11*). Unlike image recognition and natural language processing, robots perform tasks that involve physical contact, so collisions, wear, and tear are inevitable. In the end, human intervention is required while the robot is learning, and the teaching cost is greater than the conventional method. To reduce the trial-and-error process in the real world, research on "simulation-to-real transfer," which transfers learning results in a simulation environment, is conducted (*12*–*14*). However, it is difficult to reproduce nonlinear physical phenomena such as tactile sensation, deformation, and friction, so the robot cannot fill the gap with the real world. In recent years, to reduce the cost of collecting learning data, methods using imitation learning (*15*, *16*) or inverse reinforcement learning (*17*) have been proposed. A person demonstrates a model behavior, and the robot learns from it. By observing and learning a person's model behavior, it is possible to acquire a motion even when the arrangement of objects and the background are different. However, humans and robots have essentially different body structures, so there is a limit to mapping their motions. As described previously, the learning-based approach is powerful and important for acquiring complex real-world models. However, many methods aim to acquire the "optimal model" and require substantial learning costs. It is also necessary to design (assume) a reward for guaranteeing the optimality of the model.

[1]Research and Development Group, Hitachi Ltd., Ibaraki, Japan. [2]Faculty of Science and Engineering, Waseda University, Tokyo, Japan.
*Corresponding author. Email: hiroshi.ito.ws@hitachi.com (H.I.); ogata@waseda.jp (T.O.)

Here, instead of having a perfect behavioral model that can be adapted to all situations in advance through learning, an approach that adapts a model's state and behavior in real time while acknowledging the model's imperfections can be considered. For example, Friston *et al.* (*18*) proposed a free energy principle that unifies the various functions of the brain. According to this principle, the brain always predicts the next state of sensation and movement. It behaves to minimize the error (prediction error ≃ uncertainty) between the prediction and reality. The essence of this is described as adjusting cognitive models (perceptual inference) and behavior to the outside world (active inference) in a short term or in real time rather than long-term model learning (*19*). In a related manner, Ahmadi and Tani (*20*) have been researching intelligent systems driven by the prediction error minimization principle through a series of robotics research using recurrent neural network (RNN) models.

Influenced by this philosophy, we previously proposed deep predictive learning (DPL) that performs real-time prediction and motion generation (*21*–*23*). In this approach, the robot learns a time series of sensory and motor information experienced in the real world. Specifically, an RNN model is trained to minimize the prediction error of sensory-motor information between the current time and the next time by using the sensor information of the robot when a person teleoperates the robot multiple times as training data. As a result, the learning behavior is represented as an attractor in this RNN. At runtime, the robot predicts near-future sensory and motor information in real time on the basis of visual and behavioral time series information. The retractor function of the trained RNN model performs perceptual inference (the fusion of the predicted sensory and current sensory inputs) and active inference (behavioral adjustment), resulting in an attractor transition that reduces its prediction error. As a result, multiple tasks—such as handling flexible objects (*24*, *25*), liquids, and powder—are realized in the real world with a multi–degree-of-freedom robot. We also realized flexible contact-based object manipulation that was difficult to control with vision alone (*26*). Movie S1 shows the robot application of DPL, which was demonstrated at an exhibition. The former is a task to weigh a liquid of an arbitrary target value. The robot weighed 174.47 g of liquid against the target value of 175. If a human beginner were to perform this task, the discrepancy would be much more drastic. By learning to weigh liquids with multiple viscosities, it is possible to handle unfamiliar liquids and quantities. The latter is a task to weigh a powder, and salt was used as an example. The robot can adapt to powder position and shape changes. If there is too much powder, the robot performs the recovery operation. All robots have different degrees of freedom and mechanisms but are able to acquire the desired motion by DPL.

However, even in this DPL, if there is a large prediction error that the robot cannot handle in real time, the state transition using RNN retraction alone cannot handle it. Here, we have developed a method to execute appropriate actions in more complex tasks by introducing multiple DPL modules and a mechanism for switching between them by the prediction error. The complex task here is not a single manipulation with a high degree of difficulty, such as a multiproduct/indefinite posture, but for a robot to perform a whole-body movement in which the upper limbs (manipulation) and lower limbs (movement) work together or to perform tasks that require complex operating procedures.

A similar approach is the hierarchical module mechanism (*27*). In this method, behavioral modules that directly link "perception and action" are individually designed, and the modules are selected in accordance with dynamic changes in the environment. Mixture of experts is a method that adds a learning mechanism to the hierarchical module mechanism (*28*, *29*). In addition, modular selection and identification for control (*30*) is a method in which multiple modules with predictors and controllers as units exist in parallel, and the output is switched in accordance with the prediction error of each module. However, only low-dimensional sensorimotor data are handled, and there are very few application examples using multi–degree-of-freedom robots. Furthermore, low-dimensional signals make it difficult to process predictions, which makes it difficult to design module switching. Many engineering approaches have been proposed, but module switching, scalability, and retraining are issues. The planning system (*31*–*33*) generates complex robot behaviors by selecting the appropriate actions to achieve the goal. Because humans can design robot action choices, it is possible to understand the system's behavior. However, the system becomes more complex as a result because it requires a variety of behaviors, including recovery, to respond robustly to environmental changes (*34*). One method for integrating multiple combinations of motor primitives using a recurrent graph neural network has been proposed to solve this problem (*35*). However, the scalability of the primitives is an issue because the recurrent graph network needs to be retrained every time a primitive is added. The symbolic planners' methods solve complex planning problems by planning, reusing, and connecting multiple modules (*36*–*38*). Because these methods focus on planning, robots need to be operated in a completely known environment, and robustness to environmental changes and disturbances has not been deeply discussed. In addition, many state-of-the-art methods are performed in a simulator, and robustness and disturbance in a real environment are issues.

Here, we propose a method for realizing complex and varied motions with a real robot that is easily scalable and that can combine multiple modules appropriately depending on the situation. In the proposed method, multiple DPL models (hereafter, "modules") compete and solve the problem autonomously and in a decentralized way. Each module calculates the prediction error (certainty) of high-dimensional data for the current situation in real time, and the module with the lowest error is automatically executed. Specifically, the prediction error (confidence level) between the predictions of each module and the robot's visual image is calculated, and the module with the minimum error is selected. Because the robot always executes the motion with the minimum prediction error, it is possible to execute behavior adaptively. Because the designer has to design only the competing part of each module, it is easy to delete or add a function (modules). In addition, there is no need to define the order in which actions are executed.

Figure 1 shows a schematic diagram of the modular motion generation model proposed here. Figure 1A shows a system architecture, which consists of multiple modules and one operation selector. Figure 1B shows an overview of a module. This module predicts near-future situations, i.e., predicted images and motion commands, from the robot's sensory and motor information. The operation selector calculates the image prediction error (certainty) on the basis of the predicted image of each module and actual camera images and selects the module with the highest certainty as the robot's motion command.

For demonstrating the effectiveness of the proposed method, we developed an actual robot to perform a door-opening and

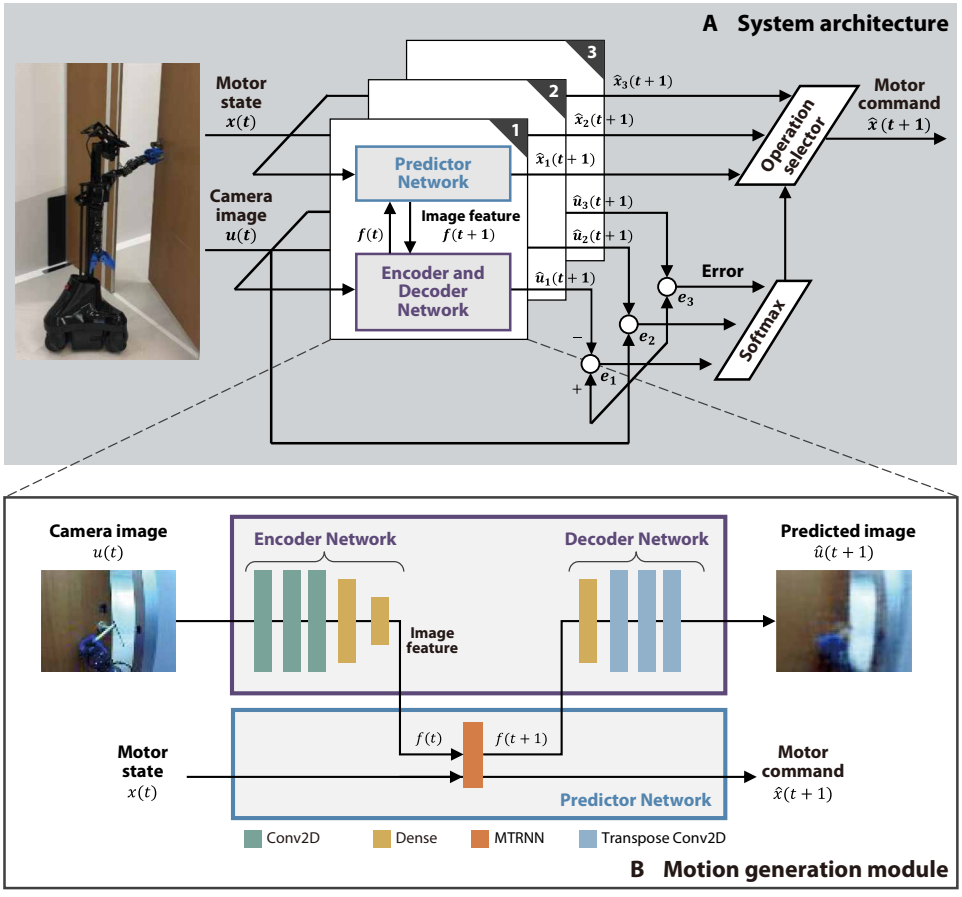
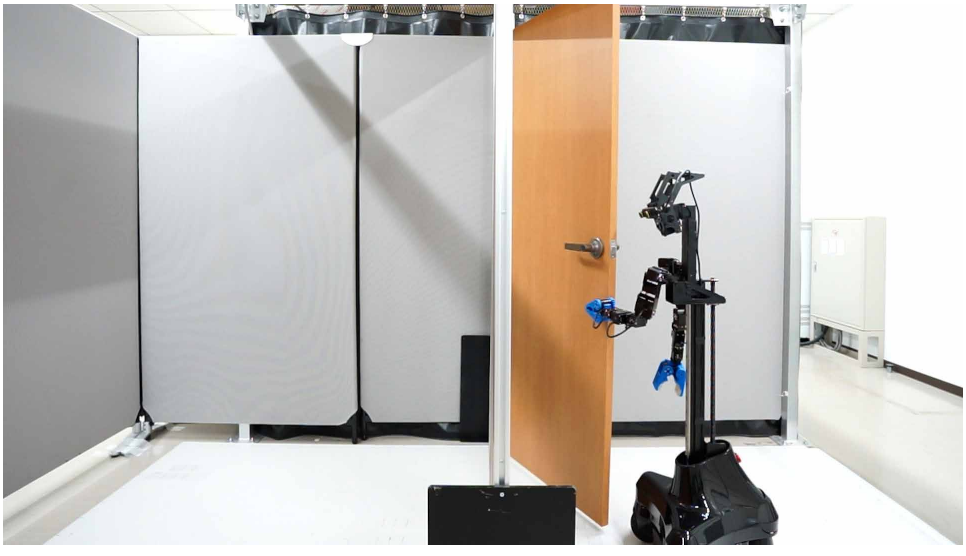


Fig. 1. System overview of modular motion generation model. Each module predicts predicted images and motion commands from the robot's sensory and motor information. The operation selector calculates the image prediction error of each module and actual camera images and selects the module with the highest certainty as the robot's motion command.



Movie 1. Summary of study. By calculating the prediction error of each module in real time, it is possible to execute a sequence of tasks (opening door outward and passing through) by linking multiple modules and responding to sudden changes in the situation and operating procedures.

passing-through motion, which is a daily task for people. There are a lot of motion patterns in the door-opening and passing-through task that depend on complex factors such as door type (inward, outward, and sliding), door construction (open right or left), and door handle type/position. Therefore, door opening is a difficult task that requires the extraction and programming of all possible situations and their behavioral patterns for a particular situation (*39*–*43*). In an experiment, we train three motions using DPL for outward-opening doors (approaching, opening, and passing through a door) and confirm whether a sequence of motions (opening outward door and passing through) can be generated on the basis of the certainty of each module. The contributions of this article are as follows.

By teaching a motion 108 times using teleoperation and learning modules, the following three conclusions are drawn: First, by interpolating, the robot can open doors in unfamiliar positions with an average success rate of 96.8%. Second, the results of an internal state analysis of the module using principal components analysis (PCA) show that motion was structured according to the position of the door handle. Third, the result of visualizing the basis of the module's motion decisions using the gradient method shows that the robot generated motion by focusing only on the door handle.

By automatically switching between multiple modules on the basis of certainty, the following three conclusions are drawn: First, we confirmed that the robot could generate a series of motions by sequentially calculating the certainty of each module and switching the motions. Second, in a general robot system, it is necessary to preprogram exception handling depending on the situation. In comparison, the proposed method can adaptively select modules depending on the situation. Even if a disturbance occurs during the motion generation, the robot can immediately switch the module to a more appropriate one. Last, by adding a new inward-door-opening module and a pivot turn, a robot performed door-opening and passing-through motion for an inward/outward door for about 30 min (15 round trips).

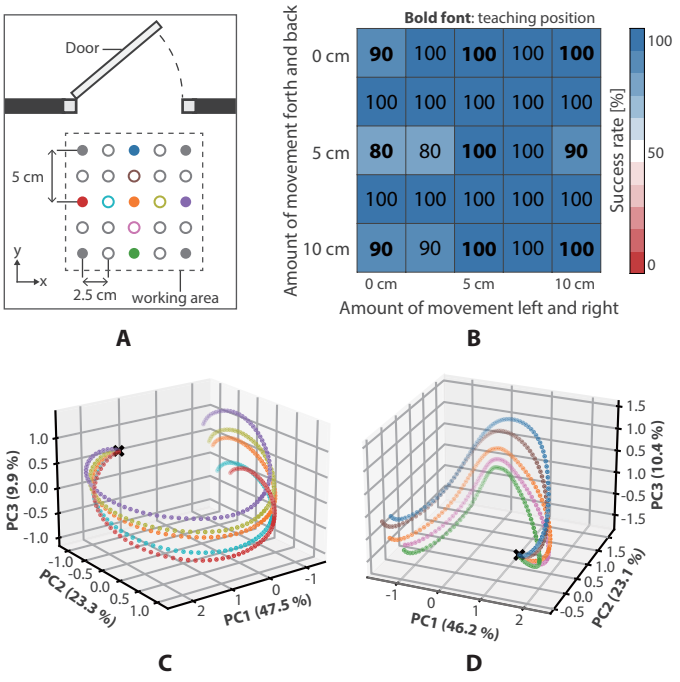


Fig. 2. Task success rate and internal state analysis of door-opening module. (**A**) Top-down view of the door-opening task. The closed circle positions are used during training, others during evaluation. (**B**) Success rates of the door-opening motion. (**C**) The internal state of the module when the robot was moved to five places in the horizontal direction against the door. (**D**) The vertical direction.

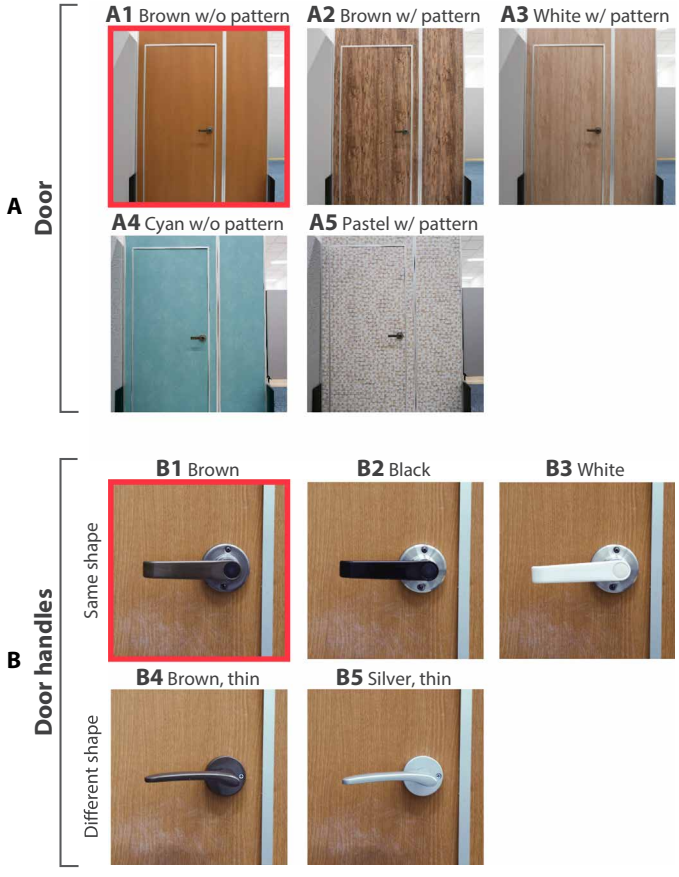


Fig. 3. Variation in door panels and door handles used in experiments. (**A1**) and (**B1**) were used during training, and others were used for evaluation.

RESULTS

Movie 1 and movie S2 summarize the results and methods of this research. The following subsections give details.

Positional generalization of module

Figure 2A shows a top-down view of the door-opening motion for the experiment. The closed circles in the figure are the positions where the robot was taught to open the door, and the open circles are the positions where it was not taught to do so. Figure 2B shows the success rate of the door-opening motion at the taught and untaught positions when the robot was shifted back and forth, left and right, at 2.5-cm intervals. The bold numbers in the success rate table indicate the positions where the robot was taught the motion, and the fine numbers indicate the positions where it was not taught the motion. The position of each cell corresponds to the positions in Fig. 2A. We tried the door-opening motion 10 times at each position, a total of 250 times, and it was confirmed that the motion was generated with a success rate of 80% or higher (average 96.8%) at all positions. "Success" in this test was defined as the robot grasping the door handle and pushing the door open.

To investigate why the robot could generate the door-opening motion with a high success rate, we performed an internal state analysis of the module. An MTRNN in Fig. 1B consists of two context layers with different time constants. The layer with the faster time constant (fast context) learns motor primitives, and the layer with the slower time constant (Cs: slow context) learns combinations (sequences) of motion primitives (*44*). This article describes the relationship between the positional generalization performance of motions and PC of the Cs layer. The method for analyzing RNN by using PCA (*45*) and the details of the MTRNN are shown in the Supplementary Materials.

Figure 2C shows the PCA result for when the robot was moved to five places in the horizontal direction (*x*-axis direction) against the door. The values of the Cs layer when the robot generated the door-opening motion were stored as time series data and compressed into three-dimensional data by using PCA for visualization. The contribution rates of the first three PCs were 47.5, 23.3, and 9.9%, respectively, which comprised 80.7% of the original data, so it is possible to get a rough trend of the internal state from this figure. The × mark in the figure is the operation start point, and each color curve is called an attractor. The color of an attractor corresponds to the color of the position in Fig. 2A. The operation start points were integrated into one point because the initial value of the Cs layer was set to zero. However, the attractors diverged immediately after the start of operation because the robot reached toward the door in different ways depending on the position. Comparing the color of the attractors with the positions in Fig. 2A, it can be seen that they are arranged in the same order. The light blue attractor is plotted between the two attractors (red and orange) at which the door-opening motion was taught. The yellow attractor is similarly plotted between purple and orange.

Figure S3 is the result for the door-approach module, and fig. S5 is that for the door-passage module. From fig. S3C, the door-

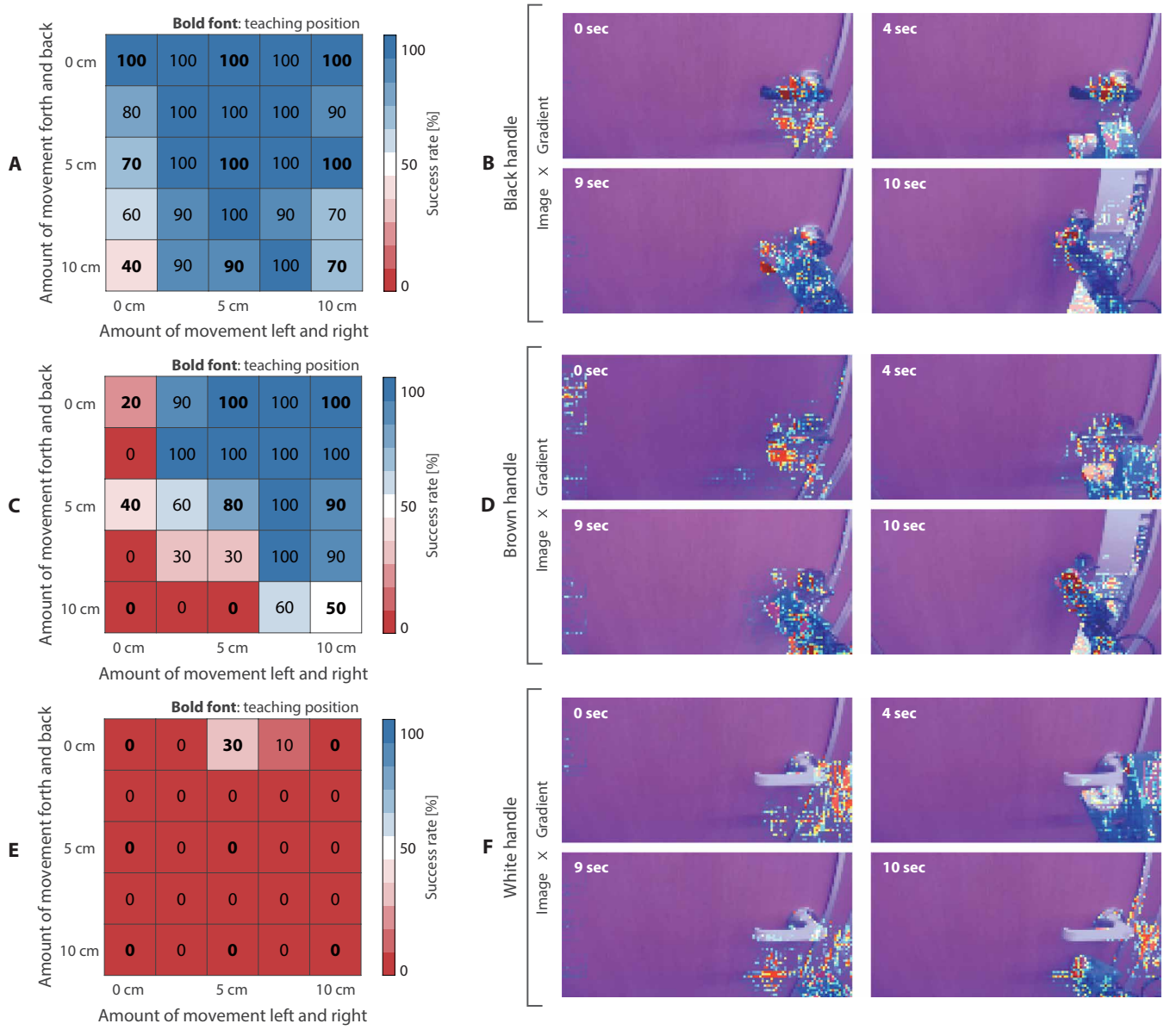


Fig. 4. Results of evaluating robustness of door-opening module to environmental changes. Task success rates and visualized images of rationale for operation decision using gradient method are shown. (**A** and **B**) Results for door handles used for training. (**C** to **F**) Door handles not trained with different colors and shapes as evaluation of generalization performance of trained modules.

approach module, which taught the motion of approaching the door from multiple start positions, also showed a trend for the attractors to diverge depending on the start position. However, the once separated attractors finally converged to one point because the states at the end of the task (stop in front of the door) were similar. Figure S5B shows that, for the door-passing module, which taught the robot to move in a straight line, the attractors converged to one point from start to end. It is thought that the reason is that there was little change in the visual image of the robot and the speed/steering angle during task execution. The details of these experimental results are shown in the Supplementary Materials.

From the above, we have shown that by using DPL to learn motions, attractors can self-organize (diverge or converge) on the basis of sensory-motor information, especially the position of the door and the amount of movement of chassis and arms. Furthermore, we suppose that it is possible to manipulate an object at an untaught position by interpolating multiple attractors acquired by learning.

Module robustness to environmental changes

To evaluate the robustness of the door-opening module to environmental changes, experiments were conducted using the door panels and door handles shown in Fig. 3. The results for the door-approach and door-passage modules are shown in the Supplementary Materials.

Success cases of door-opening motion

As a result of the robustness evaluation, the task was completed with a high success rate, although the door handle had the same shape as and similar color to that during training, as shown in Fig. 3B2, and

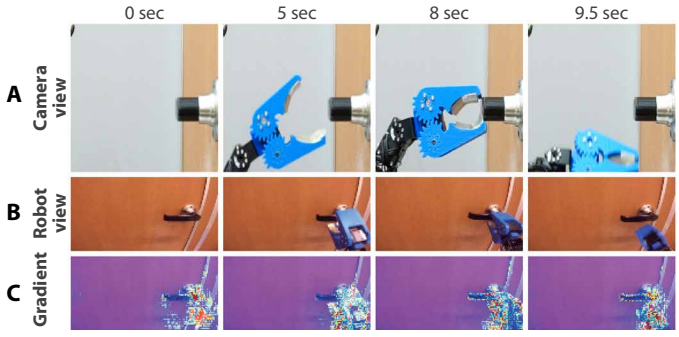


Fig. 5. Task failure case of door-opening module. (**A**) Camera image of the robot from the side. (**B** and **C**) The robot focused on the door handle but failed to predict the arm reach distance.

the door panel had different colors and patterns, as shown in Fig. 3 (A2 to A5). Figure 4A is the success rate of door opening when using a black door handle with the same shape as during training, and Fig. 4B is an image that visualizes the rationale for making its decision using the gradient method. The same door panel was used as that during training. The gradient image (sensitivity map) is overlaid on the visual image of the robot; the areas (pixels) that are strongly focused on during motion generation are shown in red, and the low areas are shown in blue. The visualization method of the sensitivity map is given in the Supplementary Materials. Immediately after the start of operation (0 s), the robot focused on the door handle and its surroundings. When reaching for the handle (4 s), it focused on its own hand and the handle. In other words, immediately after the start of operation, the robot arm is expected to predict (focus on) the position where it should reach in addition to the door handle in advance. After grasping the door handle (9 and 11 s), the robot focuses on its arm, especially near the door handle.

Figure S2 shows the results when using the door panels with different colors. Although there was some noise compared with the gradient image of the learned door panel (Fig. 4B), it can be seen that the robot was focused on the door handle and hand as in the other experiments.

Rare failure cases of door-opening motion

The robot rarely failed to open the door if the door handle was the same shape and color as that during training. Figure 5A shows a camera image of the robot from the side. Figure 5B shows a visual image of the robot, and Fig. 5C shows a sensitivity map. From the sensitivity map, the robot generated motions while focusing on the door handle and hand, showing a similar trend as the image (Fig. 4B) when the door was successfully opened. However, the camera image at 8.0 s shows that the robot was grasping the door handle with its fingertips. As a result, at 9.5 s, the finger slipped when twisting the door handle, and the task failed.

Furthermore, even if the door handle has a similar color to that used during training as in Fig. 3B4, the robot tended to fail if the shape was different. This door handle was thinner, from 23 to 9 mm. Figure 4C shows the success rate of door opening, and Fig. 4D shows a sensitivity map. From the sensitivity map, the door-opening motion was executed while the door handle was focused on, which was the same trend as the successful case described above. However, the success rate gradually decreased as the robot moved further away from the door. In many of these cases, the robot reached its arm near the door handle but not far enough to grasp it.

In this research, we used monocular images for robot vision. Therefore, the robot needs to predict the position to reach for and the distance of the arm on the basis of the position and size of the door handle. The larger the door handle appears in a visual image, the closer the distance is between the robot and the door. In comparison, the smaller it is, the farther away the robot is. From Fig. 5C, the robot was able to recognize the position of the door handle because it is focused on the door handle. However, the size of the door handle, i.e., the percentage of the image occupied by the door handle, could not be correctly estimated, which may have caused a failure in predicting the arm reach distance. As shown in this experiment, the success rate tends to be lower as the door handle becomes thinner because the handle occupies a smaller percentage of the image. Furthermore, the farther the distance between the robot and the door handle, the more difficult it is to recognize a change in the size of the handle. From the above, we suppose that failing to recognize the size of the handle accurately was the cause of the failure to estimate the arm reach distance.

Failure cases of door-opening motion

The robot tended to fail if the color of the door handle conflicted with the door handle used for training. The results when using a white door handle are shown in Fig. 4 (E and F). From Fig. 4E, it was possible to open the door near its front, although the success rate was low. However, in other places, the door-opening motion was not possible at all. From Fig. 4F, immediately after the start of motion (0 s), the robot did not focus on the handle but rather on the rightmost edge. Just before grasping the handle (4 s), the robot arm was reaching the place on which it was focused at 0 s. This suggests that the arm failed to reach the handle because it misrecognized the door handle position. Similarly, door handles of different colors and shapes also failed with the same trend, and the results are shown in fig. S1.

There were other failure cases caused by disturbances. For example, when we removed the gripper holding the doorknob as a disturbance for the door-opening module, the robot could not continue the door-opening operation. The reason for this is that the robot's behavior depends on the context of the RNN. Therefore, a recovery operation such as grasping the doorknob again is a future task (*46*).

Series of motions by module integration

To verify the effectiveness of integrating multiple modules as proposed in this article, we integrated three such modules: one that learns a door-approach motion, one that learns a door-opening motion, and one that learns a door-passage motion. Each module's certainty and motor command are predicted in real time, and the module with the highest certainty is selected to generate a suitable operation for the situation.

The robot generated a series of motions by integrating modules with the proposed method, and they are shown in Fig. 6 (A to E): (A) motion of opening outward-opening door and passing through it, (B) images predicted by each module, (C) certainty of each module, (D) speed of the robot's moving chassis, and (E) joint angles of the robot arms. Immediately after the start of the robot's motion (0 s), the difference between the camera images of the robot (Fig. 6B1) and the images predicted by the approach module (Fig. 6B2) was very small. On the contrary, as shown in Fig. 6 (B3 and B4), the door-opening and door-passage modules had not learned the door-approach motion; thus, the respective initial states (i.e., the state

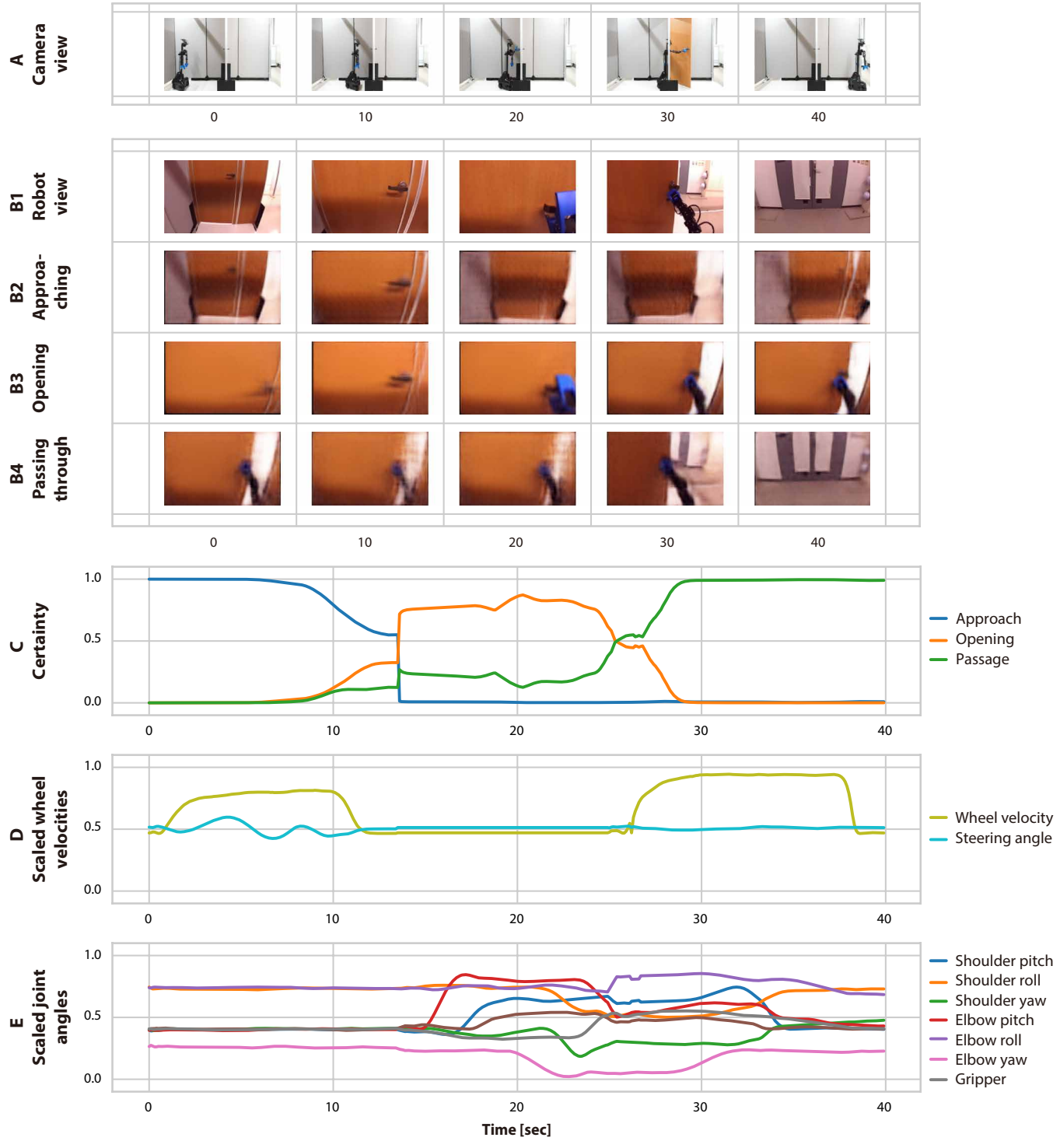


Fig. 6. Module integration task. (**A**) Robot-generated series of motions using the three modules. (**B** to **E**) Each module's certainty and motor command are predicted in real time, and the module with the highest certainty is selected to generate a suitable operation for the situation.

when the robot was in front of the door and the state in which the door was open) were generated as predicted images. Figure 6C shows that the certainty factor was weighted in accordance with the prediction error of each module. It is clear from Fig. 6 (C and D) that because the certainty factor of the door-approach module was the highest immediately after the operation started, the robot approached the door by sending an approach-motion command to the moving chassis. However, the prediction image of the door-approach module in Fig. 6B2 became distorted as the robot approached the door; in contrast, the difference between the predicted

images (Fig. 6B3) of the door-opening module and the camera images (Fig. 6B1) gradually decreased. In this method, the execution time is restricted so that a specific module does not occupy the process. At 12 s in Fig. 6C, the execution time of the door-approach module was exceeded, so the module was reset. Resetting the module here means setting the values of the hidden layers of the MTRNN to the initial value. Therefore, the door-approach module after 12 s generated the initial state (state away from the door) as a predicted image. As a result of recalculating the certainty, the door-opening module (orange line) exceeded that of the door-approach module (blue line), indicating that the modules switched. The operation command to the moving chassis was stopped upon the switching of the modules, and the door-opening motion was generated by sending a door-opening command to the robot arm. Similarly, the certainty factor of the door-passage module exceeded that of the door-opening module at around 25 s, when the door-opening operation was completed, indicating that the modules switched again. Here, the door-opening module was not reset because the module was switched before the maximum execution time. Therefore, the door-opening module after 25 s generated a predictive image of the state during the door-opening process. From the above results, it can be said that by calculating the certainty in real time of the three modules on the basis of the difference between the actual camera images of the robot and the images predicted by each module, it is possible to perform a series of operations, namely, the door-opening and passing-through operation generated by linking multiple modules.

Module scalability

Here, we will discuss the ease of expansion by adding a new module for inward-opening doors. The door-opening module used in the experiments is specialized for outward-opening doors that are pushed open, so it does not correspond to inward-opening doors that are pulled open. In addition, the stop position of the robot and its movement when passing through differ between inward- and outward-opening doors. Therefore, we teleoperated the robot to perform three motions (door approach, opening, and passage) for an inward-opening door, as shown in the lower row of fig. S10. We added this as a new module to verify whether the robot could perform the door-opening and passing-through motion for inward- and outward-opening doors.

Success case of module switching

Figure S6 shows the experimental results. Figure S6A shows the operation of the inward-opening door, which needs to be opened with both arms and the moving chassis because of the size of the robot. First, the right arm of the robot is used to open the door slightly. Because of the torque limit of the motor, the door cannot be opened by the force of the arm alone. Therefore, the chassis moves backward while opening the door to allow enough space so that the motion for passing through can be generated. This motion is similar to that of a small child opening a door with his or her whole body. Then, after making sure that the door is sufficiently open, the chassis gets into the gap while pushing the door with the right arm, generating the door-opening and passing-through motions. In this way, complex movements combined with manipulation and movement are also possible to generate. Note that the appearance of opening the outward-opening door is the same as in Fig. 6A. Figure S6B shows the certainty of each module, and by adding the three new modules, there is a total of six certainties. It can be seen that each module was properly switched out depending on the situation. Furthermore, by adding a pivot turn after the door-passage motion was executed (green and brown lines), the robot could generate a repeated door-opening motion. The pivot turn is a motion programmed to make the robot rotate in place for a few seconds and is not included in the module. Movie 1 shows that adding a pivot turn and the module for inward-opening doors made it possible to generate continuous motions for opening and passing through the inward- and outward-opening door for about 30 min (15 round trips). In this case, the difference between the two doors was decided on the basis of certainty.

Failure case of module switching

In the case of similar modules, there was a tendency for the wrong module to be selected depending on the lighting conditions. Here, we show the case of a module switching failure that occurred with an inward- and outward-door-opening module. Figure 7 shows the scene when the outward-opening door was in front, from left to right: robot view, predicted image of the outward-door-opening module, and predicted image of the inward-door-opening module. The blue frame in the figure shows the module selected by the operation selector. The white letters show the mean squared error (MSE) and standardization error (score) between the robot view and the predicted image. The score was used to calculate the certainty, and it was defined that the closer to zero, the more likely the module is to be the correct module.

The upper row of Fig. 7 shows the experimental results under the same lighting conditions as during training. Both the MSE and score of the outward-door-opening module were closest to zero, indicating that the appropriate module was selected. Comparing the robot view with the predicted images of the outward-door-opening module shows that similar images were generated. In particular, the outward-door-opening module was trained with a shadow under the door handle, so the shadow was displayed. The inward-door-opening module shows the door handle in a similar position, but the direction of the handle was different. In addition, a shadow was not displayed because the motion was trained in a situation without a shadow.

The lower row of Fig. 7 shows the experimental results under different lighting

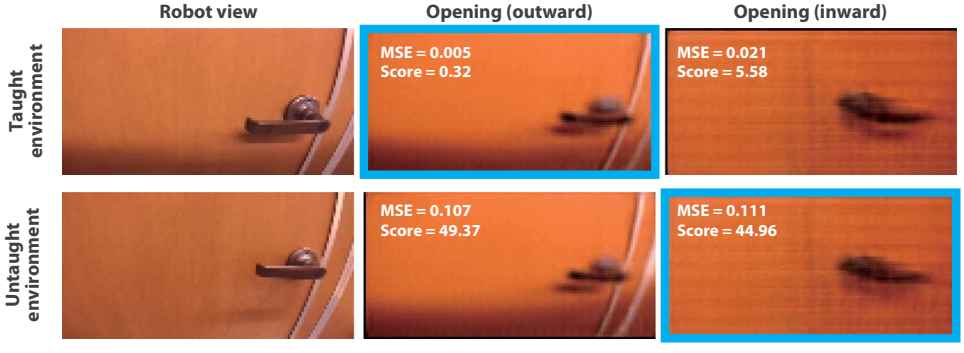


Fig. 7. Failure cases of door-opening module switching. The blue frame in the figure shows the module selected by the operation selector. The possibility of failing to select a module increases depending on the lighting conditions.

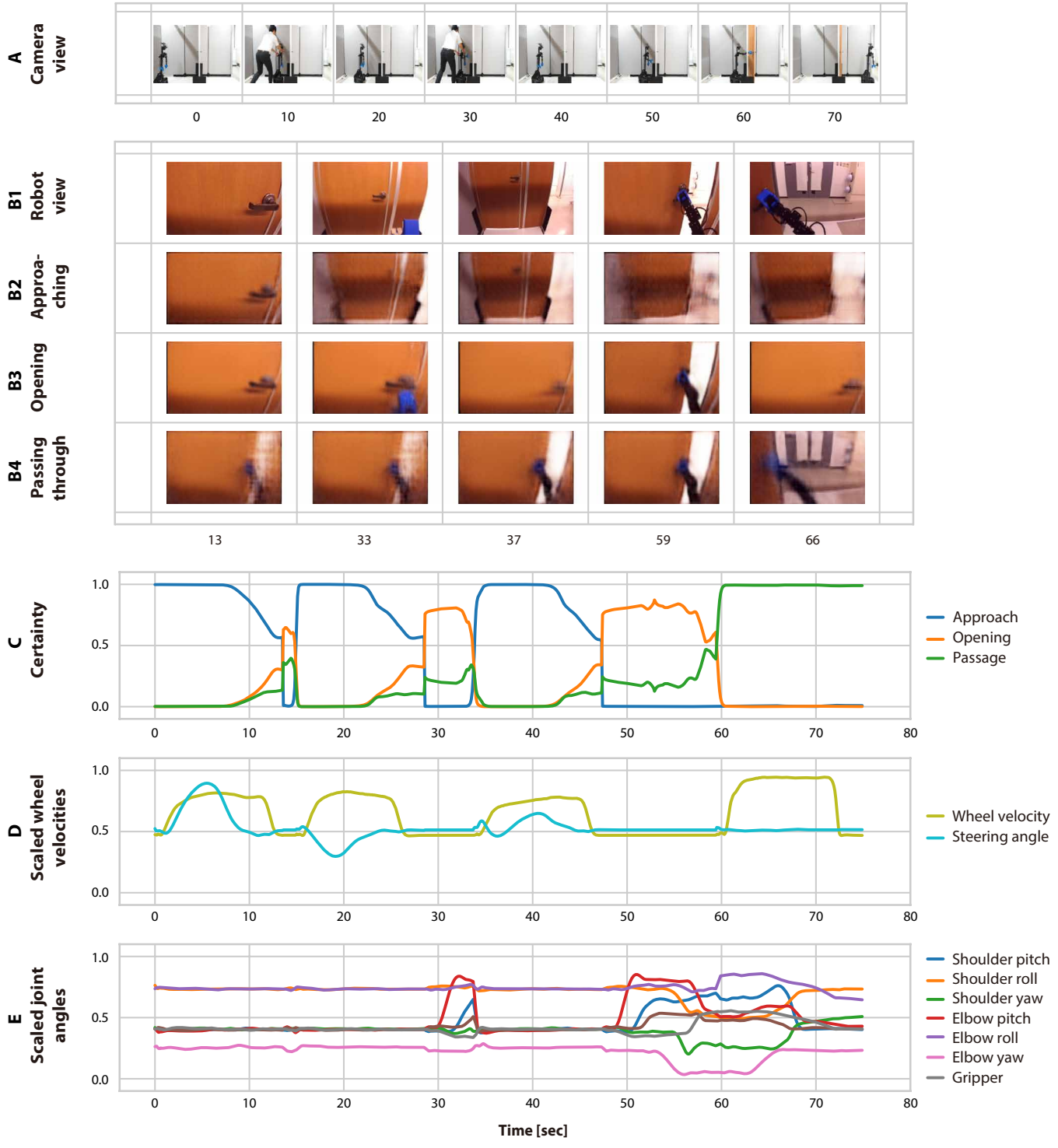


Fig. 8. Execution of interference task. (**A**) Person randomly moved robot to its initial position. (**B** to **E**) Each modules are switched automatically on the basis of certainty, it is possible to respond to sudden changes in a situation without design the switch timing strictly.

conditions as during training. The lighting conditions were changed to avoid a shadow under the handle. Judging by the human eye, the outward-door-opening module generated an image closer to the robot view. However, the inward-door-opening module had a smaller score value, so the wrong module was selected. As shown in Eq. 6, our method uses the error between the pixels of the robot's visual image and the module's predicted image to calculate the score, so the orientation of the door handle is not taken into consideration. Furthermore, because the color

of the door panel occupies a larger percentage of the image, the score is smaller for the inward-door-opening module with no shadow.

Module switching is unlikely to fail if the situation for each module is visually different, such as door approach, opening, and passage. However, when multiple modules have trained similar tasks, the possibility of failing to select a module increases depending on the experimental conditions.

Module switching against disturbance

In conventional robot systems, motion sequences must be programmed for the situation. To achieve robust motions in response to environmental changes, it is necessary to preprogram exception handles to deal with unexpected situations and disturbances. However, it is difficult to achieve this goal because of the huge amount of programming required for exception handling in a real environment with various changes. Here, we set up and evaluate the following two disturbance tasks to evaluate the robustness of motion sequences.

Interference task

Figure 8A shows a human interfering with a robot that is executing a task. Normally, the robot would perform the door-approach, door-opening, and door-passage motions in sequence, but to interfere with it, a person moved the robot to its initial position when it was in the middle of opening the door. The robot, once returned to its initial position, generated the door-approach motion again and eventually passed through the door. Figure 8B shows that the certainty factor of the door-approach module (blue line) gradually decreased as the robot approached the door. However, because of interference (movement to the initial position) at around 15 and 33 s, the certainty of the module rose sharply, and the door-approach motion was generated again. In comparison, the certainty factor of the door-opening module (orange line) gradually increased, and the modules switched at around 13, 28, and 48 s. Figure 8C shows the robot's joint angles, generating the door-opening motion during the second and third module switches. In particular, the module responded to the change in the situation and generated the door-approach motion immediately, although the door-opening motion was generated at the second module switching (around 33 s). Thus, it is possible to immediately switch to a module suitable for the surrounding situation, even in interference.

Sequence interrupt task

Figure S8A shows the state when a person opened the door just before the robot opened the door as a sequence interrupt process. Normally, the robot generates a door-opening motion after approaching the door, but here, the person caused a disturbance by opening the door first. Because of the sudden opening of the door, the robot skipped the door-opening motion and generated the door-passage motion. Figure S8C shows the certainty factor of each module at that time, and the certainty factor of the door-approach module (blue line) gradually decreased as the robot approached the door. In comparison, the certainty factor of the door-opening module (orange line) gradually increased. However, around 12 s after the person opened the door, the certainty factor of the door-passage module (green line) increased rapidly, and the modules switched. In addition, fig. S8B1 shows a camera image of the robot, and fig. S8B2 shows the predicted image of the door-passage module. The camera images at 15 and 20 s show the person, but the person was not in the predicted images of the door-passage module. Because this module had not learned situations where a person opens the door first, it cannot predict situations in which a person is seen. However, motion is generated by predicting the situation on the basis of the surrounding situation and past learning experiences (knowledge that it has). Thus, by comparing the present and past situations, it is possible to generate robust motions in response to changes in the surrounding environment. When a sudden change of state occurs in a conventional robot system, the robot requires time to re-recognize the environment and replan its operation. However, the proposed method can respond immediately to changes in a situation without programming exception handling. From the above, by preparing a large number of modules, the motion variation is increased by the number of combinations.

DISCUSSION

For robots to perform tasks that humans do daily, models need to be designed and modified according to the task, resulting in substantial development costs and long-term adjustments. This article proposes an approach to reduce the cost of model design and training data collection by using a DPL module that can simultaneously acquire recognition and motion generation models. As shown in table S1, the robot can respond to object position and background changes with only a maximum of 108 motion lessons (about 20 min) per task and 6 hours of module training. Furthermore, the design cost of the system was reduced by introducing multiple modules and introducing a mechanism shown in Algorithm 1 that switches modules in accordance with their prediction errors. Because the designer only has to design the competing part of each module, it is easy to delete or add modules. In general, deep reinforcement learning requires a huge amount of trial and error in a real environment (*11*). To reduce the number of trials, four robots were trained in a distributed and asynchronous manner to obtain feasible strategies with an average success rate of over 90% in about 50 trial-and-error cycles (*9*, *10*). Deep imitation learning is used to realize multiple tasks with a small number of motion teachings (*47*). In the case of the object-reaching task, the robot can achieve a task success rate of 90% after 200 motion lessons (about 30 min). However, neither method discusses robustness to environmental changes. In addition, an approach that integrates multiple modules is necessary to achieve more complex tasks. The following summarizes the advantages and future challenges of the methodology.

Module design

A person with a good understanding of both robotics and deep learning can complete the teaching and training of modules in about 2 days. A module trained with DPL can also generate motion for objects in untrained positions by interpolation at the location where it was taught. However, this teaching would greatly affect the motion accuracy of a robot. For example, when teaching a robot to approach a door, it is necessary to operate the robot to reach the goal range. If the amount of data outside the goal range increases, the desired motion accuracy cannot be obtained. Each motion basically depends on the diversity and accuracy of the teaching motion, making it difficult to evaluate optimality. In the future, integration with reinforcement learning methods should be considered to account for new generation and optimization of unlearned trajectories.

Module switching

In the conventional method, the design cost is an issue because it is necessary to separately program responses to external disturbances

in addition to the assumed sequence of operation. Furthermore, when a disturbance occurs, there is a problem in that the robot moves slowly because it takes time to recognize a situation and replan the trajectory. In comparison, in the proposed method, modules are switched automatically on the basis of certainty, so there is no need to design the switch timing strictly. Furthermore, because the system predicts multiple modules' certainty and motor commands in real time, it is possible to respond to sudden changes in a situation and changes in operating procedure. For multiple modules to switch autonomously, it is important to have a task design that considers where to divide the sequence of operations. Specifically, to switch modules, it is necessary to prepare two modules whose operation end and start statuses are close. For example, by preparing a module that approaches the front of the door and a module that opens the door while in front of the door, the combined operation of "approaching the door and opening the door" is performed. Our operation selector cannot handle tasks that look the same but have different motions. In this experiment, the modules for inward- and outward-opening doors were switched on the basis of the appearance of the surroundings such as the direction of the door handle and the edge of the door. Therefore, if there are multiple doors (inward/outward/pocket type) that look the same but the opening motion is different, it is necessary to prepare modules for each. However, because they look the same, the operation selector cannot determine which module to select. Most readers have surely experienced trying to open a push-type door when they thought it was a pull-type one. In this case, a human can switch between operations on the basis of visual, tactile, and force information. Other visual features, such as door dampers and hinges, allow us to determine the type of door on the basis of empirical rules, even if the door does not have a push or pull sign. Thus, a recovery mechanism for reselecting modules on the basis of multimodal feedback and the ability to select modules by focusing on the visual features of the target object will be interesting for future research.

Hardware constraints of the module

Each DPL module can infer motion at 10 to 20 Hz. In the proposed method, multiple modules are computed in parallel, and the modules are switched on the basis of the confidence level. In the case of the Jetson Xavier, the robot used in this research, the authors confirmed that up to six modules can be computed in parallel and in real time. However, the extraction of image features is time-consuming, so the robot cannot be used for tasks such as chip mounting of semiconductors or high-speed grasping of objects flowing on a conveyor belt. Hardware implementation of the DPL module in a field-programmable gate array (FPGA) (*48*) or improvement in the computing speed of computers may enable high-speed object manipulation and parallel computing of more modules.

Embedding multiple behaviors in a single model

As mentioned above, our method takes a local representation, which is a method for switching multiple modules. This has a good design prospect and is effective for full-body coordination of multi–degree-of-freedom robots like the present one. However, it is not easy to design a new module to be added when considering the generalization performance and competition of each module. Specifically, there is a possibility that the selection will be biased toward some modules and that motion switching will not occur. To avoid this, distributed representation is a method of embedding multiple behaviors in a single learning model. This method has the advantage of obtaining a meta-generalization structure that integrates and classifies each motion in the learning process. However, it is difficult to apply to multi–degree-of-freedom systems such as full-body coordination because it is generally difficult to obtain optimal solutions for large-scale models. We believe that an integrated framework of these two methodologies will be necessary in the future. We believe that our approach to realizing complex tasks in the real world with low design and motion teaching costs could be applied to tasks other than door opening and could be applied to a wide range of fields.

MATERIALS AND METHODS

Module overview

Figure 1B shows an overview of the motion generation module using DPL. The module is composed of (i) an encoder network, which extracts image feature values from the robot's visual information (camera images), (ii) a predictor network, which learns image feature values and robot-motion information (e.g., joint angles and vehicle speed) in a time series, and (iii) a decoder network, which generates an image from image feature values. The network parameters of the module are listed in table S2. The encoder/decoder network is composed of a convolution layer [convolutional neural network (CNN), Conv2D], a transposed convolution layer (transpose Conv2D), and a densely connected layer (Dense). The pooling layer, often used in general image recognition, can simultaneously execute position invariance and information compression. However, it has a problem in that the spatial position information of images is lost (*49*). On the contrary, the padding operation of the CNN is considered to learn information about absolute positions implicitly (*50*). As for robot motion using the CNN, spatial position information about, for example, the operation target and the robot hand is indispensable. Accordingly, in the encoder network, padding is introduced into all layers, and image feature values are extracted by changing the filter movement amount of the convolution layer. As for the decoder network, an image is generated by convolving image feature values in reverse. The predictor network uses an MTRNN (*44*).

Training method

First, as for the training process of the module, an operation to be learned by the robot is taught by teleoperation. The robot's motor information x_t and visual information u_t are collected when multiple conditions (e.g., robot position) are changed so that the robot can flexibly respond to changes in the environment. The collected data are divided into training data used for module training and evaluation.

Next, as a pretraining, the weights of the encoder/decoder network are trained. An autoencoder (*51*) is constructed by using table S2 (A and C). An autoencoder is an unsupervised learning that uses the same data for the input and output layers so that the middle layer can extract the features of the data. We input the collected visual information of the robot u_t to the module and output the predicted image $\widehat{u_t}$. The network weights are updated so that the input and output images have the same value. The MSE is used as the cost function. The encoder/decoder network is pretrained to extract robust image features for changes in the position of robots and objects. The images are preprocessed using data augmentation (e.g., changing brightness and contrast and shifting images left, right, up, or down). This enables image features to be extracted that are robust to changes in lighting conditions and object position.

Last, as end-to-end learning, the module is trained using the weights acquired in the pretraining. Using the collected training data, we input the robot's sensory-motor information (x_t, u_t) to the module and output the next motor command $\hat{x}_{t+1}$ and predicted image $\hat{u}_{t+1}$. We add Gaussian noise ($\mu = 0$, $\sigma = 0.1$) to the input data to denoise the real-world noise. The network weights are updated so that the output value of the module ($\hat{x}_{t+1}$, $\hat{u}_{t+1}$) and the true value (x_{t+1}, u_{t+1}) are the same. The network is trained using the MSE as the error function as shown in Eqs. 1 to 3, Adam (*52*) as the optimization method, and backpropagation through time (*53*) as the error-propagation method. Note that L^{img} is the prediction error of an image ($W \times H \times C$ pixel), and L^{motor} is the prediction error of the motor command (N dim). We train all our models on a PC with Intel Core i7-10700 CPU with 64 GB of memory and a graphics card NVIDIA GeForce RTX 2080Ti with 11 GB of memory.

$$L_t^{\text{img}} = \frac{1}{W \times H \times C} \|\hat{x}_{t+1} - x_{t+1}\|_2^2 \tag{1}$$

$$L_t^{\text{motor}} = \frac{1}{N} \|\hat{u}_{t+1} - u_{t+1}\|_2^2 \tag{2}$$

$$L = \sum_{t=0}^{T-1} (L_t^{\text{img}} + L_t^{\text{motor}}) \tag{3}$$

Motion generation method

In the process of executing the module, the robot predicts a motion command for the next time on the basis of the robot's sensor information. The sensor information (x_t, u_t) at time t is mixed with the predicted value ($\hat{x}_{t+1}$, $\hat{u}_{t+1}$) of the module at time $t - 1$ in a fixed percentage and input to the module, as shown in Eqs. 4 and 5. This process makes it possible to generate stable motion against real-world noise. The mixing coefficient α should be a value between 0.0 and 1.0, with smaller values emphasizing the prediction value of the module. Therefore, even if the robot's sensor values are noisy, it is possible to predict the next action on the basis of the predicted values of the previous time. However, if α is too small, it will be difficult to fine-tune the motion on the basis of real-world sensor information so that the robustness to position change is lowered. Here, α is set to 0.8

$$x_t = \begin{cases} (1-\alpha) \times \hat{x}_{t-1} + \alpha \times x_t, & t > 0 \\ x_t, & t = 0 \end{cases} \tag{4}$$

$$u_t = \begin{cases} (1-\alpha) \times \hat{u}_{t-1} + \alpha \times u_t, & t > 0 \\ \hat{u}_t, & t = 0 \end{cases} \tag{5}$$

Operation selector

The operation selector shown in Fig. 1A is described hereafter. The operation selector calculates certainty factor λ_i in real time on the basis of the prediction error of each module. It then selects the module with the highest certainty factor. For example, it is assumed that n modules exist. The i-th module receives sensor information x_t, u_t of the actual robot, and it predicts motion command $\hat{x}_{t+1}^t$ and predicted image $\hat{u}_{t+1}^t$ at the next time step. For each module, the prediction error between the actual camera image u_t and the predicted image $\hat{u}_{t+1}^t$ is calculated. Certainty factor λ_i is calculated by comparing the prediction errors of multiple modules. However, each module learns data from different states of the robot and the surrounding environment, which results in variations in learning performance. The prediction error of each module is standardized to allow for comparison between modules by using Eq. 6. The mean μ_i and variance σ_i used for standardization are calculated in advance using the evaluation data. For example, if there are 120 sequences of 9 pieces of evaluation data for the door-opening motion, the mean and variance are obtained from 1071 [$= 9 \times (120 - 1)$] image prediction errors. The closer the standardized prediction error ($z_{i,t}$) is to zero, the smaller the deviation from the training time, so the surrounding situation can be predicted correctly

$$z_{i,t} = \frac{L_{i,t}^{\text{img}} - \mu_i}{\sigma_i} \tag{6}$$

$$\lambda_{i,t} = \frac{\exp(-1 \times z_{i,t})}{\sum_{j=1}^{n} \exp(-1 \times z_{j,t})} \tag{7}$$

Equation 7 is used to determine whether the prediction is good or bad between each module. Because of the softmin function, the closer $z_{i,t}$ is to zero, the greater the certainty. The certainty factor of each module takes a value from 0 to 1, and the total certainty factor of all modules (n) is 1. If the prediction of one module is better than that of the others, the certainty factor is 1; if it is worse than that of the others, the factor is close to 0, and the operation selector selects the module with the highest factor. If all modules have a similar prediction accuracy, the factor will be $1/n$. At that time, the operation selector judges that there is no suitable module. In this way, by switching the module in accordance with the prediction error of an image, it is possible to generate a suitable operation for the current situation.

Overview of operation algorithm

The pseudo code for robot operation is shown in Algorithm 1 (shown below). First, initial hidden state H of the MTRNN constituting each module is read. The robot performs the motion when the maximum number of allowed steps T (maximum execution time) is exceeded or until the operation selector determines that no module is suitable (certainty factor $\simeq 1/n$). E, S, and X are local buffers that store the state of each module (prediction error, hidden state, and operation command, respectively) at each step. Next, the robot calculates the prediction value ($\hat{x}_{t+1}$, $\hat{u}_{t+1}$) and hidden state (s_{t+1}) of each module on the basis of sensor information (x_t, u_t), and those values are stored in a local buffer. Last, the softmin function shown in Eq. 7 is used to calculate the index of the appropriate module. Sending a motion command corresponding to obtained index i to the robot enables the robot to generate a motion suitable for the situation. At that time, the hidden state of the MTRNN of index i is updated. When the number of updates of the MTRNN at index i exceeds the maximum number of allowed steps T_i, the hidden state of the MTRNN is initialized. By updating only the hidden state of a specific MTRNN in this way, the other modules can wait for a situation that they know (namely, wait until they are sure that they can generate the desired action).

Algorithm 1: Prediction and data flow of modules.
Load initial state of RNN $\boldsymbol{H} = \{\boldsymbol{h}_{t=0}^1, \boldsymbol{h}_{t=0}^2, \ldots, \boldsymbol{h}_{t=0}^N\}$
Set max step of RNN $\boldsymbol{T} = \{\tau^1, \tau^2, \ldots, \tau^N\}$
while not task end condition **do**
  Load sensor data $\boldsymbol{x}_t$,$\boldsymbol{u}_t$
  Assign empty list to $\boldsymbol{E}$, $\boldsymbol{S}$, and $\boldsymbol{X}$

for n = 1,2...N **do**
Get predicted value and RNN state for n-th module:

$$\hat{\boldsymbol{x}}_{t+1}, \hat{\boldsymbol{u}}_{t+1}, s_{t+1} = M_n(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{H}_n)$$

Get standardized prediction error and append to E list:
$\boldsymbol{E}$.append(Standardizing (MSE ($\boldsymbol{u}_t$,$\hat{\boldsymbol{u}}_{t+1}$)))
Append predicted value and RNN state to the list:
$\boldsymbol{X}$.append($\hat{\boldsymbol{x}}_{t+1}$); $\boldsymbol{S}$.append($\boldsymbol{s}_{t+1}$)
end for
Get the index of the appropriate module: i = softmin(E)
Send motor command $\boldsymbol{X}_i$ to robot
Update the RNN state of index i: $\boldsymbol{H}_i \Leftarrow \boldsymbol{S}_i$
if Update count of $\boldsymbol{H}_{i>} \tau^i$ **then**
Reset the RNN state of index i: $\boldsymbol{H}_i \Leftarrow \boldsymbol{h}^i_{t=0}$
end if
end while

Task and dataset

To verify the effectiveness of the proposed method, we selected the "door-open/passage motion," which is an example of motion that requires coordination between two motions: movement and manipulation. The motion is composed of the following three suboperations. (i) Approaching the door: The robot moves to a position from where it can grasp the door handle from an arbitrary position. (ii) Opening the door: The right robot arm grasps the door handle and turns it to open the door. (iii) Passing through the door: The moving chassis and the robot arms work together to pass through the door.

The upper row of fig. S10 shows details on the teleoperation for the outward-opening door. Figure S10A shows a schematic diagram of the task, and fig. S10B shows a snapshot. Figure S10B1 shows the teaching position (six gray circles) at which the approach-door motion was taught. Figure S10B2 shows the position of the robot at which the door-opening motion was taught. The door-opening motion was taught at the nine gray circles in the figure. Figure S10B3 shows the teaching position at which the door-passage motion was taught. The robot was taught to move 1.5 m straight ahead from the door-opening position. Each position was taught 12 times, of which 9 were used as training data and 3 as evaluation data. We taught the door-approach motion 72 times, door-opening motion 108 times, and door-passage motion 108 times. Although this robot has a stereo camera, only the visual image on the right side was used. The camera parameters (brightness, contrast, and white balance) were fixed. Resized visual image (64 pixels by 128 pixels by 3 pixels) and motor information (21 dim) were collected at a sampling rate of 10 Hz and normalized to a value from 0.1 to 0.9. The lower row of fig. S10 shows the details of the teleoperation for the inward-opening door, and the experimental conditions were the same as above.

SUPPLEMENTARY MATERIALS

www.science.org/doi/10.1126/scirobotics.aax8177
Supplementary Text
Sections S1 to S9
Figs. S1 to S11
Tables S1 and S2
Movies S1 and S2
References (*54*–*62*)

REFERENCES AND NOTES

1. B. Drost, M. Ulrich, N. Navab, S. Ilic, Model globally, match locally: Efficient and robust 3D object recognition, in *Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (IEEE, 2010), pp. 998–1005.
2. J. J. Kuffner, S. M. LaValle, RRT-connect: An efficient approach to single-query path planning, in *Proceedings of the 2000 IEEE International Conference on Robotics and Automation* (IEEE, 2000), vol. 2, pp. 995–1001.
3. S. M. LaValle, J. J. Kuffner, Rapidly-exploring random trees: Progress and prospects. *Algorithmic Comput. Robot.* **5**, 293–308 (2000).
4. J. Redmon, A. Angelova, Real-time grasp detection using convolutional neural networks, in *Proceedings of the 2015 IEEE International Conference on Robotics and Automation* (IEEE, 2001), pp. 1316–1322.
5. J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, K. Goldberg, Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. arXiv:1703.09312 [cs.RO] (2017); https://arxiv.org/abs/1703.09312.
6. W. Wan, K. Harada, Integrated assembly and motion planning using regrasp graphs. *Robot. Biomimetics* **3**, 18 (2016).
7. C.-A. Cheng, M. Mukadam, J. Issac, S. Birchfield, D. Fox, B. Boots, N. Ratliff, RMPflow: A computational graph for automatic motion policy generation, in *Algorithmic Foundations of Robotics XIII: Proceedings of the 13th Workshop on the Algorithmic Foundations of Robotics* (Springer Nature, 2020), pp. 441.
8. S. Levine, C. Finn, T. Darrell, P. Abbeel, End-to-end training of deep visuomotor policies. *J. Mach. Learn. Res.* **17**, 1334–1373 (2016).
9. A. Yahya, A. Li, M. Kalakrishnan, Y. Chebotar, S. Levine, Collective robot reinforcement learning with distributed asynchronous guided policy search, in *Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE, 2017), pp. 79–86.
10. S. Gu, E. Holly, T. Lillicrap, S. Levine, Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates, in *Proceedings of the 2017 IEEE international conference on robotics and automation* (IEEE, 2017), pp. 3389–3396.
11. S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, D. Quillen, Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *Intl. J. Robot. Res.* **37**, 421–436 (2018).
12. J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, P. Abbeel, Domain randomization for transferring deep neural networks from simulation to the real world, in *Proceedings of the 2017 IEEE/RSJ international conference on intelligent robots and systems* (IEEE, 2017), pp. 23–30.
13. Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, D. Fox, Closing the sim-to-real loop: Adapting simulation randomization with real world experience, in *Proceedings of the 2019 International Conference on Robotics and Automation* (IEEE, 2019), pp. 8973–8979.
14. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, W. Zaremba, Learning dexterous in-hand manipulation. *Intl. J. Robot. Res.* **39**, 3–20 (2020).
15. R. Yokoya, T. Ogata, J. Tani, K. Komatani, H. G. Okuno, Experience-based imitation using RNNPB. *Adv. Robot.* **21**, 1351–1367 (2007).
16. Y. Duan, M. Andrychowicz, B. Stadie, O. J. Ho, J. Schneider, I. Sutskever, P. Abbeel, W. Zaremba, One-shot imitation learning, in *Proceedings of the 31st International conference on Neural Information Processing Systems* (Curran Associates Inc., 2017), pp. 1087–1098.
17. E. Uchibe, K. Doya, Imitation learning based on entropy-regularized forward and inverse reinforcement learning. arXiv:2008.07284 [cs.LG] (2020); https://arxiv.org/abs/2008.07284.
18. K. Friston, J. Kilner, L. Harrison, A free energy principle for the brain. *J. Physiol.* **100**, 70–87 (2006).
19. K. Friston, J. Mattout, J. Kilner, Action understanding and active inference. *Biol. Cybern.* **104**, 137–160 (2011).
20. A. Ahmadi, J. Tani, A novel predictive-coding-inspired variational RNN model for online prediction and recognition. *Neural Comput.* **31**, 2025–2074 (2019).
21. K. Noda, H. Arie, Y. Suga, T. Ogata, Multimodal integration learning of robot behavior using deep neural networks. *Robot. Autonom. Syst.* **62**, 721–736 (2014).
22. K. Kase, K. Suzuki, P.-C. Yang, H. Mori, T. Ogata, Put-in-box task generated from multiple discrete tasks by a humanoid robot using deep learning, in *Proceedings of the 2018 IEEE International Conference on Robotics and Automation* (IEEE, 2018), pp. 6447–6452.
23. H. Ichiwara, H. Ito, K. Yamamoto, H. Mori, T. Ogata, Spatial attention point network for deep-learning-based robust autonomous robot motion generation. arXiv:2103.01598 [cs.RO] (2021); https://arxiv.org/abs/2103.01598.
24. P.-C. Yang, K. Sasaki, K. Suzuki, K. Kase, S. Sugano, T. Ogata, Repeatable folding task by humanoid robot worker using deep learning. *IEEE Robot. Automation Lett.* **2**, 397–403 (2017).
25. K. Suzuki, M. Kanamura, Y. Suga, H. Mori, T. Ogata, In-air knotting of rope using dual-arm robot based on deep learning. arXiv:2103.09402 [cs.RO] (2021); https://arxiv.org/abs/2103.09402.

26. H. Ichiwara, H. Ito, K. Yamamoto, H. Mori, T. Ogata, Contact-rich manipulation of a flexible object based on deep predictive learning using vision and tactility. arXiv:2112.06442 [cs.RO] (2021); https://arxiv.org/abs/2112.06442.
27. R. Brooks, A robust layered control system for a mobile robot. *IEEE J. Robot. Automation* **2**, 14–23 (1986).
28. R. A. Jacobs, M. I. Jordan, S. J. Nowlan, G. E. Hinton, Adaptive mixtures of local experts. *Neural Comput.* **3**, 79–87 (1991).
29. M. I. Jordan, R. A. Jacobs, Hierarchical mixtures of experts and the EM algorithm. *Neural Comput.* **6**, 181–214 (1994).
30. D. M. Wolpert, M. Kawato, Multiple paired forward and inverse models for motor control. *Neural Netw.* **11**, 1317–1329 (1998).
31. J. Bohren, S. Cousins, The smach high-level executive [ros news]. *IEEE Robot. Automation Mag.* **17**, 18–20 (2010).
32. M. Colledanchise, P. Ögren, *Behavior Trees in Robotics and AI: An Introduction* (CRC Press, 2018).
33. M. Toussaint, Logic-geometric programming: An optimization-based approach to combined task and motion planning, in *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence* (IJCAI, 2015), pp. 1930–1936.
34. R. A. Brooks, Elephants don't play chess. *Robot. Autonom. Syst.* **6**, 3–15 (1990).
35. F. Xie, A. Chowdhury, M. Kaluza, L. Zhao, L. L. Wong, R. Yu, Deep imitation learning for bimanual robotic manipulation. arXiv:2010.05134 [cs.RO] (2020).
36. C. Paxton, Y. Barnoy, K. Katyal, R. Arora, G. D. Hager, Visual robot task planning, in *Proceedings of the 2019 International Conference on Robotics and Automation (ICRA)* (IEEE, 2019), pp. 8832–8838.
37. C. Paxton, Y. Bisk, J. Thomason, A. Byravan, D. Foxl, Prospection: Interpretable plans from language by predicting the future, in *Proceedings of the 2019 International Conference on Robotics and Automation (ICRA)* (IEEE, 2019), pp. 6942–6948.
38. D.-A. Huang, D. Xu, Y. Zhu, A. Garg, S. Savarese, L. Fei-Fei, J. C. Niebles, Continuous relaxation of symbolic planner for one-shot imitation learning, in *Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2019), pp. 2635–2642.
39. E. Klingbeil, A. Saxena, A. Y. Ng, Learning to open new doors, in *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE, 2010), pp. 2751–2757.
40. W. Meeussen, M. Wise, S. Glaser, S. Chitta, C. McGann, P. Mihelich, E. Marder-Eppstein, M. Muja, V. Eruhimov, T. Foote, J. Hsu, R. B. Rusu, B. Marthi, G. Bradski, K. Konolige, B. Gerkey, E. Berger, Autonomous door opening and plugging in with a personal robot, in *Proceedings of the 2010 IEEE International Conference on Robotics and Automation* (IEEE, 2010), pp. 729–736.
41. S. Chitta, B. Cohen, M. Likhachev, Planning for autonomous door opening with a mobile manipulator, in *Proceedings of the 2010 IEEE International Conference on Robotics and Automation* (IEEE, 2010), pp. 1799–1806.
42. S. A. Prieto, A. Adán, A. S. Vázquez, B. Quintana, Passing through open/closed doors: A solution for 3D scanning robots. *Sensors* **19**, 4740 (2019).
43. M. Arduengo, C. Torras, L. Sentis, Robust and adaptive door operation with a mobile manipulator robot. arXiv:1902.09051 [cs.RO] (2019); https://arxiv.org/abs/1902.09051.
44. Y. Yamashita, J. Tani, Emergence of functional hierarchy in a multiple timescale neural network model: A humanoid robot experiment. *PLoS Comput. Biol.* **4**, e1000220 (2008).
45. H. Hotelling, Analysis of a complex of statistical variables into principal components. *J. Educ. Psychol.* **24**, 417–441 (1933).
46. K. Suzuki, H. Mori, T. Ogata, Compensation for undefined behaviors during robot task execution by switching controllers depending on embedded dynamics in rnn. *IEEE Robot. Automation Lett.* **6**, 3475–3482 (2021).
47. T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, P. Abbeel, Deep imitation learning for complex manipulation tasks from virtual reality teleoperation, in *Proceedings to the 2018 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2018), pp. 5628–5635.
48. S. Ohara, T. Ogata, H. Awano, Binary neural network in robotic manipulation: Flexible object manipulation for humanoid robot using partially binarized auto-encoder on FPGA. arXiv:2107.00209 [cs.RO] (2021).
49. S. Sabour, N. Frosst, G. E. Hinton, Dynamic routing between capsules, in *Proceedings of the Advances in Neural Information Processing Systems* (NIPS, 2017), pp. 3856–3866.
50. M. A. Islam, S. Jia, N. D. Bruce, How much position information do convolutional neural networks encode? arXiv:2001.08248 [cs.CV] (2020); https://arxiv.org/abs/2001.08248.
51. G. E. Hinton, R. R. Salakhutdinov, Reducing the dimensionality of data with neural networks. *Science* **313**, 504–507 (2006).
52. D. P. Kingma, J. Ba, Adam: A method for stochastic optimization arXiv:1412.6980 [cs.LG] (2014); https://arxiv.org/abs/1412.6980.
53. P. J. Werbos, Backpropagation through time: What it does and how to do it. *Proc. IEEE* **78**, 1550–1560 (1990).
54. D. Erhan, Y. Bengio, A. Courville, P. Vincent, "Visualizing higher-layer features of a deep network" (Technical Report 1341, University of Montreal, 2009).
55. J. T. Springenberg, A. Dosovitskiy, T. Brox, M. Riedmiller, Striving for simplicity: The all convolutional net. arXiv:1412.6806 [cs.LG] (2014); https://arxiv.org/abs/1412.6806.
56. R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-cam: Visual explanations from deep networks via gradient-based localization, in *Proceedings of the IEEE International Conference on Computer Vision* (IEEE, 2017), pp. 618–626.
57. D. Smilkov, N. Thorat, B. Kim, F. Viégas, M. Wattenberg, Smoothgrad: Removing noise by adding noise. arXiv:1706.03825 [cs.LG] (2017); https://arxiv.org/abs/1706.03825.
58. M. Sundararajan, A. Taly, Q. Yan, Axiomatic attribution for deep networks, in *Proceedings of the International Conference on Machine Learning* (PMLR, 2017), pp. 3319–3328.
59. R. Brooks, L. Aryananda, A. Edsinger, P. Fitzpatrick, C. C. Kemp, U.-M. O'REILLY, E. Torres-Jara, P. Varshavskaya, J. Weber, Sensing and manipulating built-for-human environments. *Intl. J. Humanoid Robot.* **01**, 1–28 (2004).
60. S. Cousins, B. Gerkey, K. Conley, W. Garage, Sharing software with ros [ros topics]. *IEEE Robot. Automation Mag.* **17**, 12–14 (2010).
61. M. Wise, M. Ferguson, D. King, E. Diehr, D. Dymesich, Fetch and freight: Standard platforms for service robot applications, in *Workshop On Autonomous Mobile Service Robots* (Fetch Robotics Inc.,2016).
62. J. Pages, L. Marchionni, F. Ferro, Tiago: The modular robot that adapts to different research needs, in *International Workshop on Robot Modularity* (IROS, 2016).

Acknowledgments: We are grateful to A. Amino, chief researcher at Hitachi Ltd., for designing the robot. We also thank ExaWizards Inc., DENSO WAVE Incorporated, Beckhoff Automation GmbH, and TAISEI CORPORATION for performing the demonstration using the DPL module and providing the content of movie S1. **Funding:** This work was funded by Hitachi Ltd. and collaborated with Waseda University. **Author contributions:** All authors conceived the work. H.I. developed all the software required for this work, including the robot's control and teleoperation systems, proposed model, etc. H.I. collected and analyzed the experimental data, and K.Y. and T.O. verified all results. H.I. designed the figures and wrote the manuscript. All authors contributed to editing the manuscript. K.Y. and T.O. supervised the research. K.Y. acquired the research funding. **Competing interests:** The authors declare that they have no competing interests. **Data and materials availability:** All data needed to evaluate the conclusions in the article are present in the main text or the Supplementary Materials.

Submitted 16 December 2020
Accepted 8 March 2022
Published 6 April 2022
10.1126/scirobotics.aax8177

Science Robotics

Efficient multitask learning with an embodied predictive model for door opening and entry with whole-body control

Hiroshi Ito, Kenjiro Yamamoto, Hiroki Mori, and Tetsuya Ogata

Sci. Robot. **7** (65), eaax8177. DOI: 10.1126/scirobotics.aax8177

Science Robotics (ISSN 2470-9476) is published by the American Association for the Advancement of Science. 1200 New York Avenue NW, Washington, DC 20005. The title *Science Robotics* is a registered trademark of AAAS.