End-to-End Visuomotor Learning from Virtual Environment to Real Robot

Kei Higuchi[1,2], Constantin Uhde[3], Gordon Cheng[3],
Ixchel G. Ramirez-Alpizar[2], Gentiane Venture[4,2], and Natsuki Yamanobe[2,1]

***Abstract*— Robots can acquire skills to accomplish a target task by learning human manipulations. In this study, we build an end-to-end visuomotor learning system for a robot to learn multiple tasks in a virtual environment, and then perform the same tasks in a real environment without re-training. We use domain randomization to improve the generalization performance of the learning model. To effectively tackle this challenge, we build an integrated learning system that jointly learns robot motions and visual features of the task. The experimental results show that our system can perform multiple tasks with a high success rate and is able to successfully bridge the Sim2Real gap, compared to learning motion and visual features separately.**

I. INTRODUCTION

Robots are expected to work alongside humans in human environments. In such environments, robots are no longer required to perform a single task but also multiple tasks in response to constantly changing situations. In addition, it will be necessary for people who are not robot engineers to generate robot motions.

Humans are able to learn behaviors by imitating the actions of instructors or by having their own bodies manipulated directly by the instructors. In this way, by imitating the actions of the instructor, humans can intuitively understand, learn, and practice without verbal instructions. If the robot could learn the motions in the same way, anyone could easily make the robot perform the desired motions. Motion generation by imitation can be realized by using a method called Learning from Demonstration (LfD) in which the robot acquires skills by being controlled to achieve correct motions by humans. There are high expectations for motion generation through end-to-end learning that learns the relationship between sensor input and output using human manual control as training data in LfD, such as scooping up something from a bowl [1], folding a cloth [2] and various other tasks are realized using this framework. However, learning a robot's behavior requires a large number of data, and using real robots is problematic in terms of the cost of building the environment, safety (not anyone would be able to manipulate the robot), and maintenance [3].

[1]Graduate School of Engineering, Mechanical Systems Engineering, Tokyo University of Agriculture and Technology, Japan
[2]National Institute of Advanced Industrial Science and Technology (AIST), Japan.
[3]Institute for Cognitive Systems, Technical University of Munich, Germany
[4]Department of Mechanical Engineering, Faculty of Engineering, The University of Tokyo, Japan

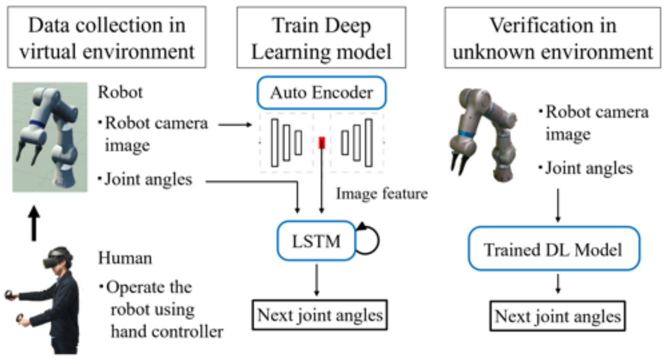


Fig. 1. Overview of the system. A human operates a robot to collect dataset in a virtual environment. The dataset is used to train a deep learning model which predicts the next joint angles. Finally, we assess our model in an unknown environment.

In this study, unlike other works and to avoid such problems, we utilize a robot that has been trained to perform multiple tasks using a virtual reality (VR) system instead of the real environment. We avoid possible damages to the real robot and make the robot perform desired behaviors intuitively by directly manipulating the robot in a virtual environment. This environment also allows the user to change viewpoints, allows the user to get as close as they want to the robot without any safety concerns, which consequently lets the user teach the robot in an easier way. We developed a system for integrated learning that allows the visual feature learner to be influenced by the results of the motion learner, with the expectation that the generalization performance of the motion generator can be improved by successfully learning task-specific features. We conducted a validation test using a reaching motion and a pushing motion, and confirmed that using the motion learning loss in the visual feature learner results in an improved success rate. Experimental results show that our system can learn multiple tasks simultaneously, and furthermore, that it can be applied to the real world.

II. RELATED WORK

A. Motion learning using reinforcement learning

Reinforcement learning enables robots to acquire appropriate behaviors based on rewards through repeated trial and error. Levine et al. [4] introduced reinforcement learning to achieve complex behaviors such as hanging coat hangers, tightening bottles, and removing nails. The advantage of reinforcement learning is that it does not require the preparation of a training dataset in advance, but it requires a large number of trials to learn the appropriate behavior. Levine et al. [5] and Gu et al. [6] have successfully acquired object grasping and door opening/closing behaviors without using prior demonstrations or manually designed representations. However, the acquisition of these behaviors implies using certain budget and the time cost of operating the robots. In addition, in the early stage of learning, when the control policy has not been established, there is a risk of outputting unexpected behaviors. In order to reduce the huge amount of time it takes for a robot to acquire effective strategies, robots can learn initial behaviors from demonstrations before reinforcement learning exploration [7]. Furthermore, to avoid dangerous behaviors in the early stages of learning, robots are set up in simulation [8]. However, in these studies, it is necessary for humans to design rewards adequately in order for the robot to acquire appropriate behaviors, and the design needs trial and error.

B. *Motion learning using imitation learning*

In addition to reinforcement learning, imitation learning, which uses human demonstrations, is often used for robot motion acquisition. Yang et al. [2] and Zhang et al. [9] have proposed a method in which a robot is teleoperated using a head-mounted display to obtain training data. Based on the obtained data, the robot learns folding and assembling motions by combining image feature extraction using convolutional neural networks and motion learning. In imitation learning, it is possible to associate not only the robot's sensor data but also human data performing the same task. Yu et al. [10] proposed a method of meta-learning, which allows a robot to perform the same motion as demonstrated by a human, using only one sample. Wu et al. [11] proposed GELLO, a controller with the same kinematic structure as the real robot for intuitive teleoperation to collect human demonstration data. However, all of these researches use real robots for learning, which places a heavy burden on the cost of environment construction, safety, and maintenance [12].

C. *Learning robot behavior in virtual environments*

Due to the heavy burden of robot learning in real environments, especially for learning social intelligence through the interaction with humans, Inamura et al. [13], [14] provide a cloud-based VR platform that enables virtual human-robot interaction. This platform can also store human behaviors in VR and create datasets to understand the behaviors. Uhde et al. [15] created a dataset of subjects performing housekeeping tasks in VR, which was later used to construct an action sequence that allows robots to perform similar tasks.

In a virtual environment, the large amount of data necessary for robot learning can be generated. However, a robot trained with this data may not operate properly in the real world due to the Sim2Real gap. In order to bridge this gap, a method called domain randomization has been proposed, in which the training data is varied by randomly setting the virtual environment [3], [12], [16], [17]. By providing enough variability in the virtual environment setting to include the deviation from the real world, the learning results in the virtual environment can be applied to the real world without additional learning.

Many studies for end-to-end learning using deep learning only deal with a single task [1], [2]. When dealing with multiple tasks, methods have been proposed that prepare a trained model for each task [10] or distinguish between tasks by providing task-switching inputs in addition to sensor inputs [18], [19]. In this study, a robot learns a policy for multiple tasks in a virtual environment and then, the learnt policy is applied on a real environment. We use a VR system to teach the motions, and the training data acquisition environment is randomly set based on Sim2Real research [12], [17]. The learning model is divided in two models, a visual feature learning model (for learning images) and a motion learning model (for learning time-series motions). In order to acquire task-specific features and learn multiple tasks without confusion, the two models are trained simultaneously in this study, whereas they have been trained independently in previous studies [1], [2].

III. Visuomotor learning system

A. *Overview of the system*

Fig. 1 shows an overview of the learning system, the demonstrator becomes a physical avatar in a virtual environment using a VR headset (Oculus) and teaches a robot arm (Torobo Arm [20]) how to move. For the virtual environment, we used SIGVerse [14], which can perform human-robot interaction using VR. Fig. 2 shows the environment for teaching the robot using VR. In the virtual environment, the robot arm can be grasped and moved directly by the human demonstrator using a VR controller. During the teaching, the images from the robot camera and the joint angles of the arm are collected. Next, we train a learning model using the collected data. The learning model consists of two Deep Learning models: an Auto Encoder (AE), which compresses the image and extracts features, and a Long Short-Term Memory (LSTM) model, which predicts the next joint angles of the robot using the current image features and joint angles as inputs [1]. In this study, the parameters of the two models are updated simultaneously during training to acquire task-specific features. Finally, we evaluate the models by generating behaviors in a new virtual or real environment created for evaluation using the trained models.

B. *Tasks*

In this study, we selected three types of tasks, reaching, pushing, and pulling as basic manipulation tasks. In the first

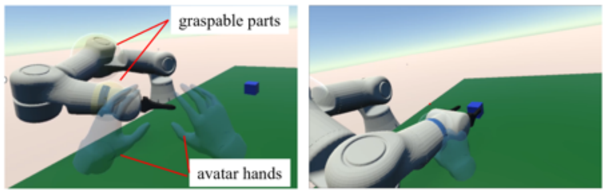


Fig. 2. An environment for teaching a robot using VR system.

verification, the robot learns reaching and pushing, and in the additional verification, it learns pulling. The contents of each task are shown in Fig. 3. In the reaching task, the robot hand reaches a blue cube randomly placed on the table. In the pushing task, a target position of a red square is added to the table, and the robot hand pushes the blue cube to reach the target position. In the pulling task, a hook-like tool is used to pull the blue cube to the target position. During teaching, RGB images, joint angles of the 7 DOF robot arm, and 0 or 1 values representing hand opening and closing (8 dimensions in total) were collected at 10 Hz for 7 seconds.

C. Data collection in virtual environment

We prepared an environment with a table in front of the robot and set up objects as targets for the three described tasks. In order to improve the generalization performance of the learning model, we decided to set up environments with different parameters and collect motion data [10]. Specifically, the parameters shown in Table I were all set randomly each time when the environment was generated. The blue cube, which is the target object for reaching and pushing, and the red square, which is the target position for pushing, were randomly placed within the reachable range of the robot hand. The colors of the objects were changed according to the values of the HSV color space. The domain randomization of the colors was set to a range that included the real-world environment. The coordinate system for each object is a left-handed coordinate system with the orientation from the ground to the base of the robot as the positive Y-axis and the orientation from the robot to the table as the positive Z-axis. For each training, the objects were randomly placed within the range of values shown in Table I. The ambient light source is placed directly above the robot. The darkness of the shadows created by the lights was set between 10% $\sim$ 50% in the *Unity3D* engine.

D. Learning model

The details of the learning model are shown in Fig. 4 The AE has four convolutional layers symmetrically for encoder and decoder, respectively. Batch normalization and max-pooling are applied to the encoder and decoder convolutional layers. Finally, the AE compresses the 80$\times$160 RGB image information to a total of 100 dimensions. Adam [21] was used as the optimization function with a learning rate of 1×10^{-3}, and the batch size was set to 32.

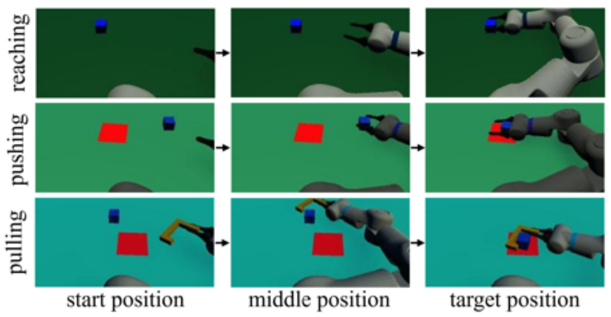


Fig. 3. Target tasks.

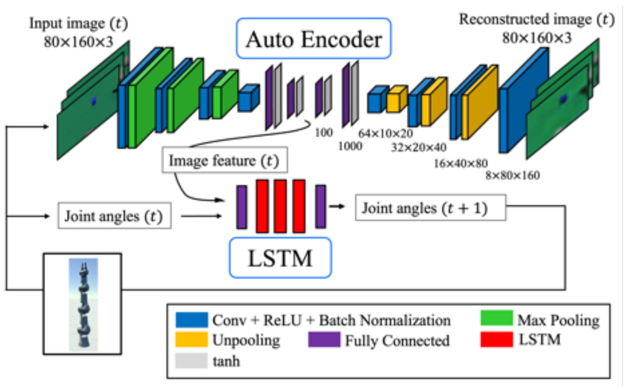


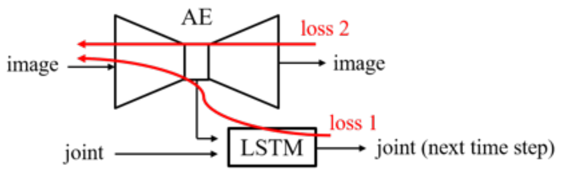
Fig. 4. Deep Learning architecture for end-to-end task learning, which consists of an Auto Encoder to extract image features and a LSTM to predict the robot's next joint angles.

The LSTM model uses an architecture with two fully connected layers and four LSTM layers. It takes, as input, 100 dimensions of image features and 8 dimensions of joint angles of the robot arm for 20 frames extracted from the training data and, and outputs 8 dimensions of predicted joint angles of the robot arm in the next step. A dropout wrapper is applied after the first full connected layer, so that 50% of the units in each layer are randomly ignored. Adam was used as the optimization function with a learning rate of 1×10^{-3}, and the batch size was set to 32.

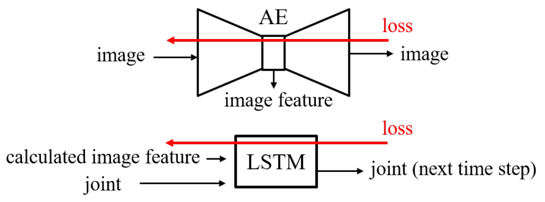
1) Integrated learning: In the previous studies [1], [2], the learning of visual features and motions was done independently. In this study, however, the parameters of the two deep learning models are updated simultaneously during training with the aim of acquiring task-specific features. The parameter update algorithm for integrated learning is shown in Fig. 5(a). Before the learning, the training of AE is completed using all the image training data. Then, the parameters from the encoder part of the AE to the LSTM are updated using the mean squared error of the predicted and true robot's joint angles. The parameters from the encoder part to the decoder part of the AE are then updated using the mean squared error of the reconstructed and true images.

TABLE I
ENVIRONMENTAL SETTING

Object	Color			Position	Angle
	H	S	V		
Cube	210∼250 (blue)	75∼100	75∼100	random on the table	Y:±5°
Target	340∼380 (red)	75∼100	75∼100	random on the table	Y:±5°
Table	145∼185 (green)	75∼100	75∼100	XZ:±1cm	Y:±1°
Floor	0∼360	0∼10	75∼100	-	-
Robot	0	0	50∼90	-	-
Camera	-	-	-	XYZ: ±1cm	XYZ:±1°
Light	-	-	-	-	X:±30° Y:±90°

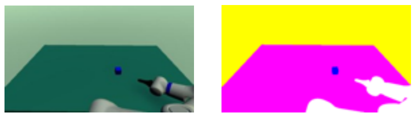


(a) Integrated learning



(b) Independent learning

Fig. 5. Integrated learning and independent learning models.



(a) original image (b) segmentation image

Fig. 6. Example of an object mask. Weights are used to represent the parts of the image that we want to draw attention to.

2) Independent learning: As a comparison, we also built an independently trained model (Fig. 5(b)). This method is similar to previous studies [1], [2]. First, the parameters of AE are updated using the mean squared error of the reconstructed and true images to complete the training. Next, we output the image features for all the training image data and store the standardized features. Finally, the LSTM is trained using the computed image features and the joint angles of the robot as input. The mean squared error of the predicted and true robot's joint angles is used to update the LSTM parameters.

3) Loss function during AE learning: When learning the AE, if the loss function is directly calculated based on the difference between the original image and the reconstructed image, the objects, such as cube, target, etc., are recognized as noise and the features cannot be acquired well, resulting in the loss of objects during reconstruction. Therefore, we decided to create a mask and set weights in order to learn the important parts of the tasks. To create the mask, segmentation images obtained by the VR are saved with the RGB images (Fig. 6). The weights are set to $w_{object} = 1.0$ for pixels with objects, and $w_{background} = 0.4$ for background pixels, and applied to the difference between the original and reconstructed images to calculate the loss function.

IV. Experiment

A. Experiments in virtual environment

We applied the learned model to a Torobo Arm and tested it in the virtual environment.

1) Virtual environment for verification: For verification, we prepared 15 different environments for each task. The positions of the objects were placed randomly in each environment. The color of the objects, the position of the camera, and the intensity and angle of the ambient light were fixed in a combination that was not present in the training data.

2) Learning model: For comparison, we prepared several types of learning models. There were six types of models, consisting of two patterns of learning methods (integrated learning and independent learning) and three patterns of training datasets (reaching(400), pushing(400), reaching(200)+pushing(200); the number in parentheses is the number of training data. Each training dataset has 70 steps of RGB images and joint angles.)

3) Target tasks and success conditions: Reaching and pushing tasks are used for the verification experiments. The success conditions of each task are shown below.

Reaching: The goal is to move the robot hand to a position where it can grasp the cube, i.e. the cube is within reaching distance of the fingers.

Pushing: The goal is for the cube to be pushed into the target square.

4) Results in virtual environment: The learned model takes the current RGB image and the joint angles of the robot as input to estimate the next joint angles of the robot. Using the learned model in the virtual environment, the robot performed its tasks in 15 different virtual environments for verification. The success rates of each model are shown in Table II. In the case of integrated learning, the average success rate for the two tasks is 86.7% for both the model that learns a single task and the model that learns multiple tasks, maintaining the same success rate. However, in the case of independent learning, the average success rate for the two tasks of the model that learns a single task is 53.4%, and that of the model that learns multiple tasks is 43.4%, indicating a decrease in the success rate. We believe that integrated learning emphasizes the differences in images for each task by feeding back the value of the loss function of the LSTM to the AE, allowing the model to correctly learn the relationship between visual information and tasks and to perform similar tasks without confusion.

When the learning model can correctly recognize the positional relationship of each object, the robot succeeds in

TABLE II
Average Success rate over 15 virtual environments

(a) Integrated learning

training data (number of data)	test task	success rate
Reaching (400)	Reaching	73.3%(11/15)
Pushing (400)	Pushing	100%(15/15)
Reaching (200) + Pushing (200)	Reaching	**86.7%(13/15)**
	Pushing	**86.7%(13/15)**

(b) Independent learning

training data (number of data)	test task	success rate
Reaching (400)	Reaching	46.7%(7/15)
Pushing (400)	Pushing	60.0%(9/15)
Reaching (200) + Pushing (200)	Reaching	46.7%(7/15)
	Pushing	40.0%(6/15)

the task. On the other hand, if the learning model cannot recognize the positional relationship of each object correctly, the robot will fail the task.

B. Experiment in real environment

In order to verify the generalization performance of the learned model, we applied the learned model to a real Torobo Arm and tested it in a real environment.

1) Real environment for verification: Fig. 7 shows the real environment prepared for the verification. The positions and angles of the camera and table were set in the same way as in the virtual environment. However, since the environment was randomized during training data acquisition, there is no need to be strict with the settings. For the camera, we used Microsoft's Kinect V2. The cube and target position were made of paper. They have the same size as those in the virtual environment and were placed so that the positions seen from the camera were the same as those verified in the virtual environment.

2) Learning model: We used the same model as in the virtual environment, without retraining.

3) Target tasks and success conditions: The contents and success conditions of each task are the same as those for the verification in the virtual environment.

4) Results in real environment: The success rates of each model are shown in Table III. Comparing the success rates of the integrated learning models with multiple tasks in the real environment with the success rates in the virtual environment, it is shown that the success rate decreases for both reaching and pushing. However, the number of failed tasks is only two-fifteenths each. Therefore, we believe that a learning model trained only in a virtual environment can be applied to a real environment and still perform the desired task with a sufficient success rate. We conclude that the Sim2Real gap can be successfully bridged by training a two-part model that handles spatial and temporal learning simultaneously with the help of domain randomization. The trends of the results for integrated and independent learning are almost the same as those for the virtual environment.

Fig. 8 shows successful and failed examples using integrated learning models trained with 200 each of reaching and pushing data. As in the results in virtual environments, when the learning model can correctly recognize the location of each object, the robot succeeds in the task. On the other hand, when the learning model cannot recognize the location of each object correctly, the robot will fail the task. In the independent learning model, the causes of failure were the same, but the success rate in the independent learning model was 26.7% for reaching and 13.3% for pushing, while the success rate in the integrated learning model was 73.3% for both reaching and pushing.

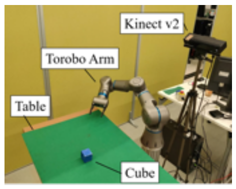


Fig. 7. Verification in real environment.

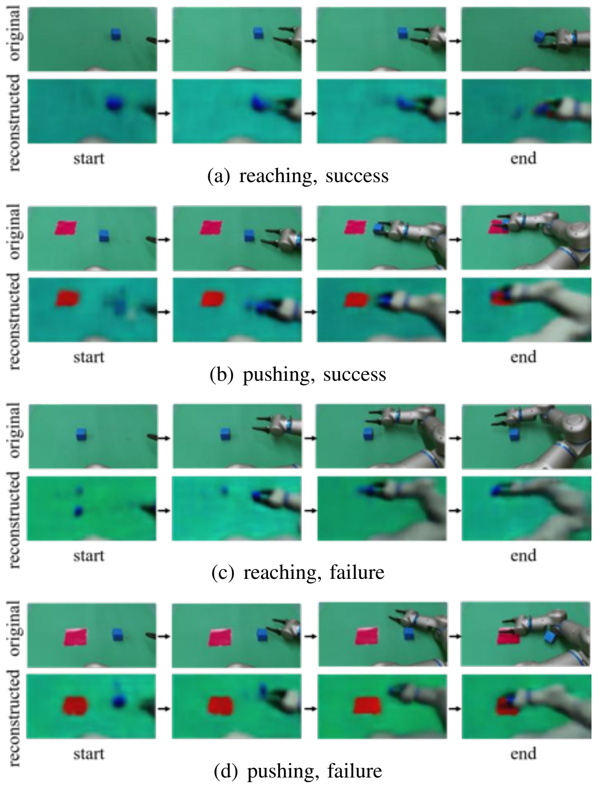


Fig. 8. An example of tasks when using the integrated learning model trained with reaching(200) and pushing(200) in real environment.

In addition, Fig. 9 shows an example of a failed reaching operation using the independent learning model. A non-existent object (red target) sometimes appeared, but this did not occur in the case of integrated learning. Since the motion is different between when the red target is present and when it is absent, we believe that the accuracy of feature extraction and image reconstruction was improved by incorporating this difference into the learning of the AE.

C. Additional task

1) Target task and success conditions: Since we were able to successfully learn two tasks using integrated learning, we increased the number of tasks with the addition of pulling

TABLE III

AVERAGE SUCCESS RATE IN REAL ENVIRONMENT OVER 15 TRIALS.

(a) Integrated learning

training data (number)	test task	success rate
Reaching (400)	Reaching	46.7%(7/15)
Pushing (400)	Pushing	60.0%(9/15)
Reaching (200) + Pushing (200)	Reaching	**73.3%(11/15)**
	Pushing	**73.3%(11/15)**

(b) Independent learning

training data (number)	test task	success rate
Reaching (400)	Reaching	26.7%(4/15)
Pushing (400)	Pushing	53.3%(8/15)
Reaching (200) + Pushing (200)	Reaching	26.7%(4/15)
	Pushing	13.3%(2/15)

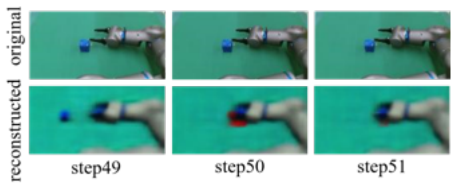


Fig. 9. An example of a failed reaching task when using the independent learning model trained with reaching (200) and pushing (200) in real environment.

TABLE IV
SUCCESS RATE OF INTEGRATED LEARNING.

(a) Virtual environment

training data (number)	test task	success rate
Reaching (200) +Pushing (200)+Pulling (200)	Reaching	73.3%(11/15)
	Pushing	80.0%(12/15)
	Pulling	80.0%(12/15)

(b) Real environment

training data (number)	test task	success rate
Reaching (200) +Pushing (200)+Pulling (200)	Reaching	**73.3%(11/15)**
	Pushing	**73.3%(11/15)**
	Pulling	**80.0%(12/15)**

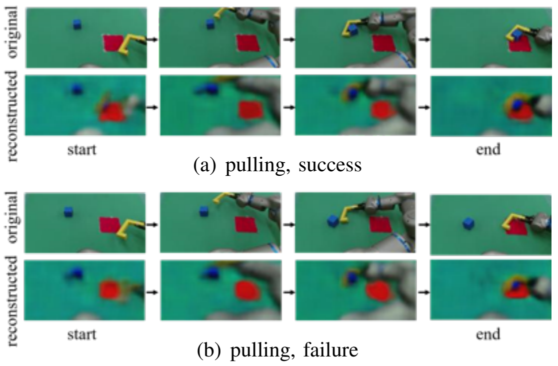
motion. In the pulling task, the blue cube is slid to the target position of the red square using a tool (Fig. 3). The task is determined as success when the final position of the cube is above the target square after the pulling operation.

2) *Results:* Table IV shows the success rate of the integrated learning model in the virtual and real environments. Comparing the success rate in the real environment with that in the virtual environment, the success rate decreases only for reaching, while it remains the same for pushing and pulling. In the case of reaching, where the success rate decreased, only one task failed. Therefore, we believe that a learning model trained only in a virtual environment can be applied to a real environment and still perform the target task with a sufficient success rate.

Fig. 10 shows the results of the successful and failed pulling task. The image reconstruction for the failed case shows that the robot does not correctly recognize the position of the object at the start of the motion. When the robot arm is positioned near the object, the image is reconstructed as if the object is inside the tool. This error in the image reconstruction may have caused the robot to fail to correct the motion. The results for the image reconstruction during the reaching and pushing motions are similar to the results shown in Fig. 8.

D. *Image feature visualization*

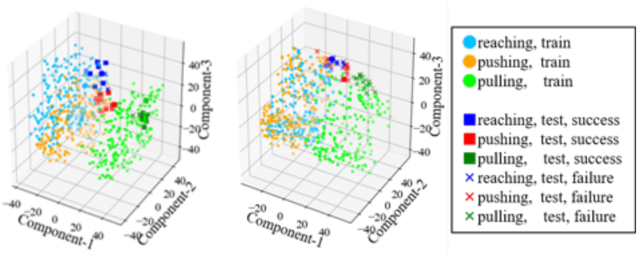
To visualize the results of integrated and independent learning, we used Principal Component Analysis (PCA) to map a series of image features for each of the three tasks into three dimensions (Fig. 11). The independent learning features are not well separated for each task, while the integrated learning features are well separated. We think this is the same reason why the non-existent red target is not reconstructed in the integrated learning as shown in Fig. 9. Incorporating the differences in motion between tasks into the training of



(a) pulling, success

(b) pulling, failure

Fig. 10. An example of pulling tasks when using the integrated learning model trained with reaching (200), pushing (200) and pulling (200) in real environment.



(a) Integrated learning (b) Independent learning

Fig. 11. Distribution of a series of image features for each task of the three tasks by PCA.

AE is considered to have improved the accuracy of feature extraction and image reconstruction. We believe that the AE of integrated learning clearly recognizes the difference between each task and is able to perform the task without confusion, thus improving the success rate.

V. CONCLUSIONS

In this study, we constructed an end-to-end visuomotor learning system using a virtual environment. The system consists of a spatial and a temporal network to combine visual with motor learning, and the generalization performance of the motion generation model was improved by successfully learning task-specific features. The experimental results show that our system performs better than previous research learning methods that learn vision and motion separately, especially when learning multiple different tasks. Furthermore, our system can accomplish the learned tasks without significant loss of success rate in real-world execution, compared to simulation.

In future work, we aim to improve the accuracy of the image reconstruction and to recognize when the system fails the task. When our system fails in a task, the visual learning model reconstructs the image as if it had succeeded in the task. For practical purposes, it is essential to recover from errors. We are also considering investigating how many different types of difficult tasks can be learned simultaneously.

ACKNOWLEDGMENT

We would like to thank Prof. Kensuke Harada at Osaka University and members of his laboratory for useful discussions.

References

[1] H. Ochi, W. Wan, Y. Yang, N. Yamanobe, J. Pan, and K. Harada, “Deep learning scooping motion using bilateral teleoperations,” *2018 3rd International Conference on Advanced Robotics and Mechatronics (ICARM)*, pp. 118–123, 2018.

[2] P. C. Yang, K. Sasaki, K. Suzuki, K. Kase, S. Sugano, and T. Ogata, “Repeatable folding task by humanoid robot worker using deep learning,” *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 397–403, 2018.

[3] M. Yan, I. Frosio, S. Tyree, and J. Kautz, “Sim-to-real transfer of accurate grasping with eye-in-hand observations and continuous control,” *arXiv preprint arXiv:1712.03303*, 2017.

[4] S. Levine, C. Finn, T. Darrell, and P. Abbeel, “End-to-End Training of Deep Visuomotor Policies,” *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[5] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, “Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection,” *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, 2018.

[6] S. Gu, E. Holly, T. Lillicrap, and S. Levine, “Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates,” *IEEE international conference on robotics and automation (ICRA)*, pp. 3389–3396, 2017.

[7] A. Zhan, P. Zhao, L. Pinto, P. Abbeel, and M. Laskin, “A Framework for Efficient Robotic Manipulation,” 2020.

[8] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox, “Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience,” *CoRR*, vol. abs/1810.05687, 2018.

[9] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel, “Deep Imitation Learning for Complex Manipulation Tasks from Virtual Reality Teleoperation,” *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–8, 2018.

[10] T. Yu, C. Finn, A. Xie, S. Dasari, T. Zhang, P. Abbeel, and S. Levine, “One-Shot Imitation from Observing Humans via Domain-Adaptive Meta-Learning,” *arXiv preprint arXiv:1802.01557*, 2018.

[11] P. Wu, Y. Shentu, Z. Yi, X. Lin, and P. Abbeel, “Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators,” 2023.

[12] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, “Domain randomization for transferring deep neural networks from simulation to the real world,” *IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 23–30, 2017.

[13] Y. Mizuchi, and T. Inamura, “Cloud-based multimodal human-robot interaction simulator utilizing ROS and unity frameworks,” *IEEE/SICE International Symposium in System Integration*, pp. 948–955, 2017.

[14] T. Inamura, and Y. Mizuchi, “SIGVerse: A cloud-based VR platform for research on social and embodied human-robot interaction,” *arXiv preprint arXiv:2005.00825.*, 2020.

[15] C. Uhde, N. Berberich, K. Ramirez-Amaro, and G. Cheng, “The Robot as Scientist: Using Mental Simulation to Test Causal Hypotheses Extracted from Human Activities in Virtual Reality,” *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8081–8086, 2020.

[16] S. James, A. J. Davison, and E. Johns, “Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task,” *1st Conference on Robot Learning*, 2017.

[17] A. Bonardi, S. James, and A. J. Davison, “Learning one-shot imitation from humans without humans,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3533–3539, 2020.

[18] R. Rahmatizadeh, P. Abolghasemi, L. Bölöni, and S. Levine, “Vision-Based Multi-Task Manipulation for Inexpensive Robots Using End-To-End Learning from Demonstration,” *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3758–3765, 2018.

[19] P. Abolghasemi, and L. Bölöni, “Accept Synthetic Objects as Real: End-to-End Training Attentive Deep Visuomotor Policies for Manipulation in Clutter,” *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6506–6512, 2020.

[20] Tokyo Robotics, “Torobo arm,” Accessed on: Dec. 28, 2020. [Online]. Available:. `https://robotics.tokyo/products/torobo_arm/`.

[21] D. Kingma, and J. Ba, “Adam: A method for stochastic optimization,” *Proc. Int. Conf. Learn. Representations*, pp. 1–13, 2014.