2025 IEEE/SICE International Symposium on System Integration (SII)
January 21-24, 2025, Munich, Germany

CNN-Based Motion Planning for Object Storage by Dual-Arm Robot

Satoshi Hoshino Yuusuke Yamada

Abstract— **In this paper, we focus on object storage by a dual-arm robot. The robot is required place objects grasped by its hands onto a shelf. The task is treated in terms of motion planning. For image inputs captured by a camera, the object storage motion outputs are determined by the robot itself based on a motion planner. For this purpose, we propose a motion planner based on Convolutional Neural Network, CNN. In order for the robot to plan the storage motion toward the shelf composed of multiple spaces, goal directions for both hands, indicating the center coordinates of target spaces, are used as inputs in addition to images and current hand positions. For these multimodal inputs into the motion planner, shortcut connections are further applied to convolutional layers in the CNN to adjust learning bias and extract image features. Through the experiments, we discuss the effectiveness of the motion planner with the goal directions and shortcut connections for object storage by the robot.**

I. INTRODUCTION

The authors have, thus far, worked on motion planning for dual-arm robots to handle objects by using the hands. In this paper, motion planning is defined as determining a set of 3D positions of both hands from the initial positions to the goal positions. A hand position is composed of the 3D pose, (ϕ, θ, ψ), in addition to (x, y, z). In previous works, this issue has been treated as a path planning problem rather than motion planning. Path planning requires a configuration space of the surrounding environment[1]. On the basis of the so-called C-space, for instance, a probabilistic roadmap planner[2] enabled robots to determine a global path from the initial position to the goal position. Many other path planners based on the C-space have been proposed. However, these path planners assumed accurate measurement of the environment, including objects, to build the C-space.

In recent years, therefore, motion planners based on deep neural networks, instead of the C-space, have been attracting attention. The deep neural networks continuously determine the next hand position outputs for the current sensor inputs. The robot then locally plans the motion to move the hands toward the positions. These motion planners are developed through training the deep neural networks.

For dual-arm robots, motion planners based on deep neural networks with multimodal inputs, such as raw RGB images, sound spectrum, and joint angles, have been proposed[3][4]. For these multimodal inputs, however, it is generally difficult for the deep neural networks to be trained. In contrast to the multimodal inputs, we have proposed a motion planner with only RGB image inputs[5]. For the image input, Convolutional Neural Network, CNN[6], was used as the deep neural network. As a result, the robot successfully identified target objects to grasp with both hands and planned reaching motions toward the objects. After the reaching motion, this paper focuses on the storage motion of the grasped objects onto a shelf, particularly in the context of factory automation.

For object storage, a dual-arm robot moves the hands grasping the objects toward a shelf. In doing so, the robot is required to plan the storage motion more precisely in 3D space, compared to the reaching motion, in order to avoid collisions between the hands, objects, and the shelf. Given a camera mounted on top of the robot, in this paper, CNN is used as a deep neural network for object storage motion planning. The CNN is trained through imitation learning[7] so as to play a role of the motion planner.

When the robot stores an object on a shelf, it is difficult for the right and left hands to determine the space where the object is to be placed if there are multiple spaces on the shelf, simply by using images as the inputs to the motion planner. For this challenge, the robot first detects spaces and objects on the shelf. In the detected image, the target spaces for both hands are determined, and the center coordinates are fed as the goal directions to the motion planner. However, these image and coordinate inputs involve only 2D information. Since the robot is required to plan the storage motion more precisely in 3D space, the current hand positions obtained by kinematic calculations from the joint angles of the robot are used as additional inputs.

Due to the multimodal inputs as described above, it might be difficult for the CNN, in the network training phase, to properly extract features important to determine motion outputs for the image inputs. In consequence, the robot is not able to plan the motion while paying attention to the hands, objects, and shelf in the images. For this challenge, we apply shortcut connections[8] to the convolutional layers in the CNN. Shortcut connections can prevent important image features from vanishing. For the hand position inputs, moreover, dropout[9] is applied to the input units. This allows the CNN to adjust learning bias for multimodal inputs of images and hand positions and extract image features.

In Section II, object storage motion planning by the dual-arm robot onto a shelf is described. Furthermore, goal directions for both hands, used as inputs to the motion planner, are calculated through space and object detection in an image. In Section III, a motion planner based on CNN with shortcut connections is proposed. In order for the robot to place the grasped objects, a hand state classifier is also described. In Section IV, the robot based on the motion planner attempts to store the objects. Through the experiments, we discuss the effectiveness of the motion planner for object storage. Finally, we conclude this paper in Section V.

S. Hoshino and Y. Yamada are with the Department of Mechanical and Intelligent Engineering, Utsunomiya University, 7-1-2 Yoto, Utsunomiya, Tochigi 321-8585, JAPAN hosino@cc.utsunomiya-u.ac.jp

II. MOTION PLANNING FOR OBJECT STORAGE

A. Object Storage by Dual-Arm Robot

The dual-arm robot moves the hands toward a shelf while grasping objects. At designated positions on the shelf, the robot opens the hands and places the objects. **Fig. 1** illustrates the overall flow of the object storage motion.

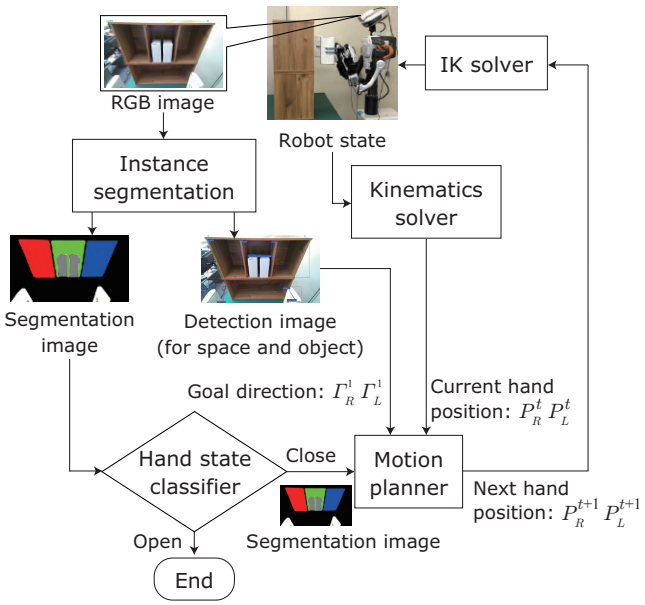


Fig. 1. Overall flow of object storage onto shelf by dual-arm robot

First off, an RGB image is captured by a camera mounted on top of the robot. Through instance segmentation[10], a space and object detection image is generated, in addition to the segmentation image, from the RGB image. In the segmentation image, each pixel is classified as a shelf, placed or grasped object, or hands[1]. The image is then fed as input to the hand state classifier. The output from the classifier is either "close" (to grasp) or "open" (to place). As long as the output is close, the segmentation image continues to be fed as input to the motion planner. Also, the current hand positions, $P^t = (x^t, y^t, z^t, \phi^t, \theta^t, \psi^t)$, obtained by kinematic calculations at step t, are used as additional inputs. Furthermore, goal directions for both hands, Γ^t, calculated from the space and object detection image, are used as inputs to the motion planner. As for the goal directions, Γ^1 at the first step ($t = 1$) is used in the following steps.

For these inputs, the hand positions, P^{t+1}, at the next step, $t + 1$, are derived as the output from the motion planner. Finally, the robot determines each joint angle through inverse kinematics calculations[11] for P^{t+1} and executes the motion. Meanwhile, once the output from the hand state classifier is open, the robot opens both hands and places the grasped objects onto the shelf, completing storage operation.

B. Goal Directions Γ in Detection Image

For a shelf composed of multiple spaces divided by partitions, it is difficult for the robot to determine the target spaces where the objects are to be placed by the right and left hands, simply by using images as the inputs to the motion planner. For this challenge, the goal directions of both hands are used as inputs to the motion planner, as described in II-A. In this section, therefore, the goal directions, Γ, are calculated from a detection image for spaces and objects in a shelf. **Fig. 2** shows an example detection image.

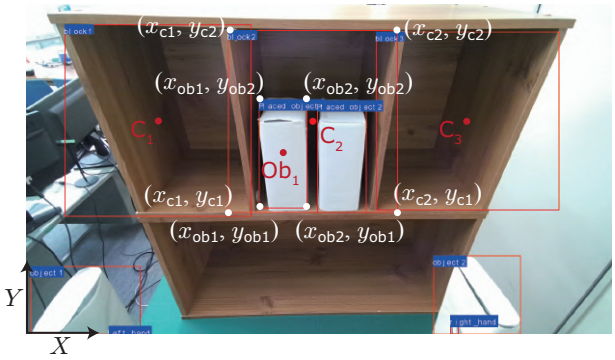


Fig. 2. Example of space and object detection image in shelf

In this image, the bottom left corner is considered the origin. The upper shelf consists of three spaces separated by two partitions. These spaces are detected and marked with red bounding boxes. Additionally, two objects placed in the center space are also detected. In the lower part of the image, parts of the hands and the grasped objects are detected. As for the center space, the coordinates of the four points of the bounding box are as follows: (x_{c1}, y_{c1}), (x_{c2}, y_{c1}), (x_{c1}, y_{c2}), and (x_{c2}, y_{c2}), respectively. For the two objects placed in the center space, the bounding box of the object on the left has the coordinates of its four points as follows: (x_{ob1}, y_{ob1}), (x_{ob2}, y_{ob1}), (x_{ob1}, y_{ob2}), and (x_{ob2}, y_{ob2}).

From the bounding boxes of the objects, the center coordinates are first calculated. For instance, the center coordinates of the left object are represented as: $\mathrm{Ob}_1(x_{\mathrm{Ob1}}, y_{\mathrm{Ob1}}) = (\frac{x_{ob1}+x_{ob2}}{2}, \frac{y_{ob1}+y_{ob2}}{2})$. Thus, by comparing the center coordinates of the objects with the x-coordinates of the bounding box of each space, the space in which the object is placed can be identified. The center of the left object, x_{Ob1}, is within the range $x_{c1} \leq x_{\mathrm{Ob1}} \leq x_{c2}$, corresponding to the bounding box of the center space. Therefore, the object is identified as being placed in this space.

The right object is similarly identified as being placed in the center space. Thus, from Fig. 2, it can be seen that among the three spaces, there are no objects placed in the right and left spaces. Therefore, these are determined as the target spaces for object storage. The center coordinates of each space, $\mathrm{C}_1(x_{\mathrm{C1}}, y_{\mathrm{C1}})$, $\mathrm{C}_2(x_{\mathrm{C2}}, y_{\mathrm{C2}})$, and $\mathrm{C}_3(x_{\mathrm{C3}}, y_{\mathrm{C3}})$, are then calculated. For the two spaces, excluding the center space, the x-coordinates of the centers, x_{C}, are compared to determine the goal directions. Then, the smallest one is assigned as the goal direction for the left hand, Γ_L, and the largest one is assigned as the goal direction for the right hand, Γ_R. Comparing C_1 and C_3, $x_{\mathrm{C1}} < x_{\mathrm{C3}}$. Therefore, the left-hand goal direction is determined as $\Gamma_L = (x_{\mathrm{C1}}, y_{\mathrm{C1}})$ and the right-hand goal direction as $\Gamma_R = (x_{\mathrm{C3}}, y_{\mathrm{C3}})$.

[1]The latest segmentation accuracies, also referred to as precisions, for the training and validation data sets are 98 [%] and 86 [%], respectively.

III. MOTION PLANNER BASED ON CNN

A. Motion Planner with Shortcut Connections Applied

As illustrated in Fig. 1, for the inputs composed of the current hand positions and goal directions, in addition to segmentation images, the next hand positions are derived from the motion planner. The relationship between the inputs and outputs is approximately represented by applying a deep neural network to the motion planner. For the image inputs, a CNN is used as a deep neural network. The CNN is trained through imitation learning from instructed storage motion. **Fig. 3** illustrates the structure of the motion planner based on CNN. The filters in the convolutional and pooling layers are represented as "size|stride|channels," respectively.

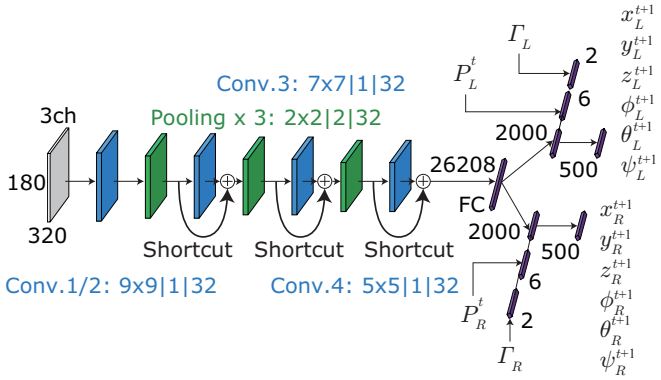


Fig. 3. Structure of motion planner based on CNN with shortcut connections

First off, a segmentation image in which each pixel is classified as a shelf, placed object, grasped object, or hands is fed as input to the motion planner. Through the convolutional and pooling layers, image features are extracted from the input. The extracted features are then flattened and used as input to the fully-connected layers. In the fully-connected layers, the network branches into right and left hands. For each branch, the current hand position, P^t, at step t and the goal direction, Γ, are further 6-dimensional and 2-dimensional inputs. In order for the CNN to extract the image features correctly from the multimodal inputs through imitation learning, shortcut connections are applied to the second, third, and fourth convolutional layers. Finally, the next hand positions, P^{t+1} at step $t+1$, are derived as the outputs from the right and left branches through a linear function. ReLU is used as the activation function in the convolutional and fully-connected layers.

The motion planner based on CNN, as illustrated in Fig. 3, is trained through imitation learning. For this purpose, the robot is instructed to store objects onto a shelf before network training. During the instruction, the robot registers the inputs to the motion planner and the corresponding motion outputs, which are next hand positions. These inputs and outputs are used as a training data set. **Fig. 4** illustrates an example of the training data set.

The training data set is composed of the inputs and outputs for each step. The input data include segmentation images, current hand positions, P_R^t and P_L^t, and goal directions, Γ_R and Γ_L of the hands calculated at the first step. For these inputs, the output data consist of the next hand positions, P_R^{t+1} and P_L^{t+1}, which are derived from the instructed motion trajectories.

Input	Segmentation image			...	
	Current hand position	P_R^1 P_L^1	P_R^2 P_L^2	...	P_R^N P_L^N
	Goal direction	Γ_R^1 Γ_L^1	Γ_R^1 Γ_L^1	...	Γ_R^1 Γ_L^1
Output	Instructed hand position	P_R^2 P_L^2	P_R^3 P_L^3	...	P_R^{N+1} P_L^{N+1}

Fig. 4. Example of training data set acquired by motion instruction

Network training of the motion planner proceeds by minimizing the output loss from the CNN for the input data in the training data set, as illustrated in Fig. 4. For this purpose, the output loss is calculated from the mean squared errors of both hands as follows: $\frac{1}{N}\sum^{N}(P^{t+1}-\hat{P}^{t+1})^2$, where P^{t+1} is the output data in the training data set, $\hat{P}^{t+1}$ is the output from the CNN during training, and N is the size of the data set. The convolutional filters and connection weights are optimized using Adam[12] based on the calculated loss. In the fully-connected layers, dropout is applied to each of the six input units corresponding to the current hand position, P^t. By doing so, each unit is deactivated with a probability of 0.5, thereby adjusting the learning bias for the multimodal inputs of images and hand positions.

B. Hand State Classifier

Before planning the motion, as illustrated in Fig. 1, the robot storing objects determines whether the hands' state is "close" (to grasp) or "open" (to place) for the segmentation image inputs. As long as the output from the hand state classifier is close, the robot continues to store the objects grasped by the hands. Meanwhile, once the output from the hand state classifier is open, the robot places the grasped objects onto the shelf. In order for the robot to determine the hands' state, close or open, from the segmentation image inputs, the hand state classifier is also developed based on CNN. **Fig. 5** illustrates the structure of the hand state classifier. In the storage motion, the robot is assumed to move the hands to the shelf simultaneously. Therefore, the hands close or open at the same time.

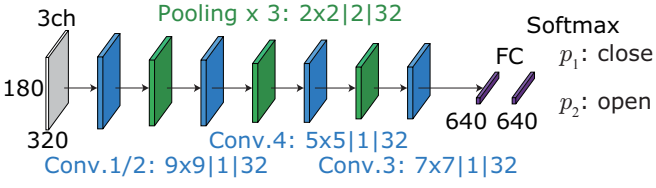


Fig. 5. Structure of hand state classifier based on CNN

The segmentation image input is processed through convolutional and pooling layers. The extracted image features are then flattened and fed as the input to the fully-connected layers. In the output layer, the softmax function is used to calculate the probabilities, p_1 and p_2, for the two classes of hand states: close and open. The class with the higher probability is output as the classification result. Therefore, if $p_1 > p_2$, the result is close, and if $p_1 < p_2$, the result is open. ReLU is used as the activation function in the convolutional and fully-connected layers.

The hand state classifier based on CNN, as illustrated in Fig. 5, is trained through supervised learning. In the training data set, hand states corresponding to the segmentation image inputs, as registered during the motion instruction described in III-A, are annotated as the output data. **Fig. 6** illustrates an example of the training data set.

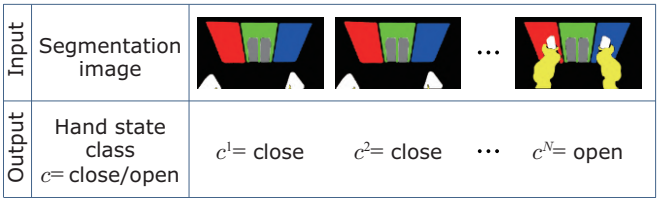


Fig. 6. Example of training data set for hand state classifier

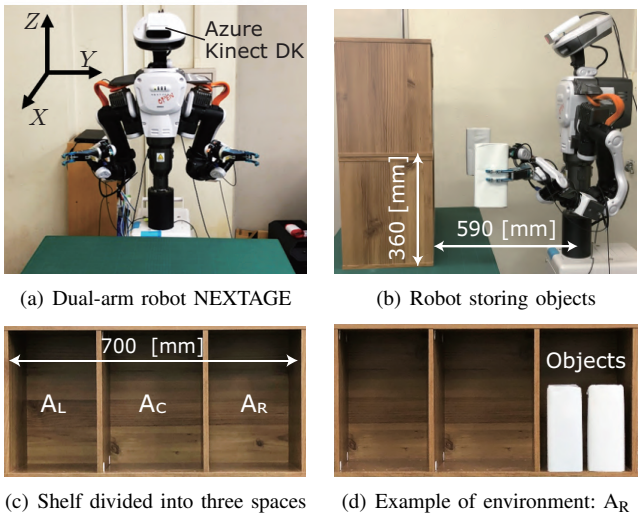
The training data set is composed of inputs and outputs for each step. The input data consist of segmentation images registered by the robot at each step. For the image inputs, the output data are annotated as one of the two classes, close or open, depending on the hand state during the instruction. In this regard, however, the number of hand states annotated with the close class is higher than those annotated with the open class. For this reason, data augmentation is applied to the hand states annotated with the open class to balance the proportions of the two classes.

Network training of the hand state classifier proceeds by minimizing the output loss from the CNN for the input data in the training data set, as illustrated in Fig. 6. For this purpose, the output loss is calculated from the cross-entropy errors as follows: $-\frac{1}{N}\sum_{t=1}^{N}\sum_{c=1}^{C} b_c^t \log \hat{p}_c^t$, where $\hat{p}_c$ is the output from the CNN, through the softmax function, during training. b_c is set to $b_c = 1$ if the classification result is correct, i.e., $\hat{c}^t = c^t$, and $b_c = 0$ if the result is incorrect, i.e., $\hat{c}^t \neq c^t$. N represents the size of the data set, and C denotes the number of classes for hand states, which in this case is $C = 2$. Thus, the states close and open are represented by $c = 1$ and $c = 2$, respectively. The convolutional filters and connection weights are optimized using Adam based on the calculated loss.

IV. OBJECT STORAGE EXPERIMENTS

A. Shelf Spaces without Placed Objects

In this experiment, we examine whether the robot, based on the proposed motion planner, is able to store objects on a shelf. The shelf is composed of multiple spaces. For object storage, spaces without objects are targets. **Fig. 7** shows the dual-arm robot and the environment used in this experiment.



(a) Dual-arm robot NEXTAGE (b) Robot storing objects

(c) Shelf divided into three spaces (d) Example of environment: A_R

Fig. 7. Object storage experiments for spaces without placed objects

The robot, NEXTAGE, shown in Fig. 7(a), has six degrees of freedom in each arm. A camera, Azure Kinect DK, is mounted on the robot head. The robot's hand is composed of three fingers. In Fig. 7(b), the robot is storing the grasped objects onto the upper shelf. The upper shelf is divided into three spaces, as can be seen in Fig. 7(c). These spaces are labeled A_R, A_C, and A_L, from right to left. In Fig. 7(d), two objects are already placed in space A_R. This environment is referred to as A_R. In environment A_R, therefore, the other spaces, A_L and A_C, are the targets for object storage.

The following three motion planners are applied to the robot for object storage to compare the performance depending on the presence or absence of goal direction inputs and shortcut connections with dropout:

1) Shortcut connections / no goal direction inputs
2) Goal direction inputs / no shortcut connections
3) Goal direction inputs / shortcut connections

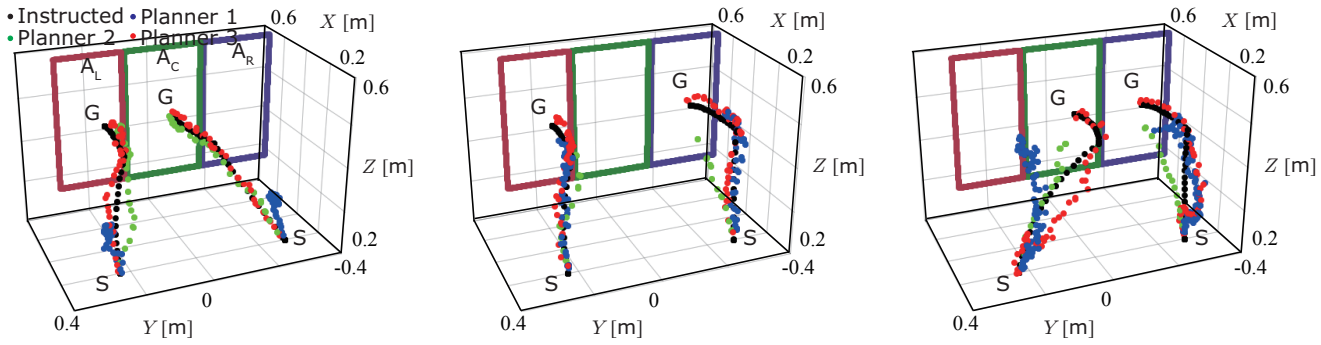
For imitation learning of each motion planner[2], the robot is instructed to store objects onto the shelf in environments A_C, A_L, and A_R. In environment A_C, two objects are placed in space A_C, and in environment A_L, two objects are placed in space A_L. For the motion instruction, the target spaces are given to the hands as listed in **Table I**.

TABLE I
TARGET SPACES GIVEN FOR MOTION INSTRUCTION

	Environment		
Hand	A_L	A_C	A_R
Left	A_C	A_L	A_L
Right	A_R	A_R	A_C

However, motion instruction to one position in the space causes an insufficient diversity in the training data set. As a result, the robot might fail in planning the storage motion due to a so-called covariate shift[13]. Even with limited motion instruction, we have thus far shown that applying

[2]The learning conditions for the three motion planners are the same.



(a) Environment A_R (b) Environment A_C (c) Environment A_L

Fig. 8. Comparison of hand trajectories depending on motion planners

data augmentation to the training data set has improved hand-reaching performance[14]. For these reasons, the robot is also instructed to store the objects to the positions at ± 20 [mm] in the Y-axis, in addition to the center position of each space, to increase the data diversity.

After network training, the robot based on each motion planner stores the objects in two spaces within the instructed environments, A_R, A_C, and A_L. The motion planners are employed together with the hand state classifier as described in III-B. The results are listed in **Table II**.

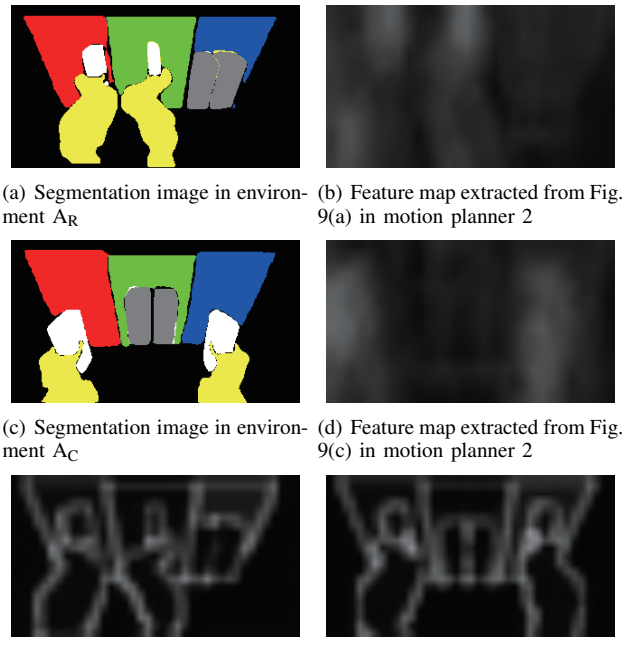
TABLE II
RESULTS OF OBJECT STORAGE EXPERIMENTS IN INSTRUCTED ENVIRONMENTS

Motion planner	Environment		
	A_L	A_C	A_R
1	×	×	×
2	×	×	◯
3	◯	◯	◯

As for motion planner 1, the robot failed to store the objects in all environments, as indicated by ×. In the row of motion planner 2, ◯ is marked only in environment A_R. This result indicates that the robot based on the motion planner succeeded in storing objects in this environment but failed in the other two environments. On the other hand, the robot based on motion planner 3 succeeded in storing objects in all the environments. To discuss these results, we compare hand trajectories, as shown in **Fig. 8**.

In Fig. 8(a), we can see that the robot based on motion planner 1 stopped its hands during the motion. In Fig. 8(c), although the target space for the left hand was A_C, the motion was incorrectly planned to move toward space A_L, where two objects were already placed. In environments A_R and A_L, the robot was allowed to store the objects in the center space on the shelf. Moreover, the robot could use either the right or left hand to store the objects in this space. In these cases, since goal directions were not used as inputs to motion planner 1, the robot could not determine the correct target spaces.

The robot based on motion planners 2 and 3, in contrast, successfully planned the storage motions toward the correct spaces with no objects in all the environments, as shown in Fig. 8(a) to Fig. 8(c). However, while the robot based on motion planner 2 succeeded in storing objects in environment A_R, it can be seen that the distance between the hands was narrower than during the motion instruction. Additionally, the hands deviated from the instructed trajectories, causing the grasped objects to collide with the shelf in environments A_C and A_L, as shown in Fig. 8(b) and Fig. 8(c). Compared to these results, the robot based on motion planner 3 was able to plan the storage motion generally along the instructed trajectories. Therefore, we further discuss the results of motion planners 2 and 3 based on the feature maps extracted by the convolutional layers in the CNN for image inputs. **Fig. 9** shows the feature maps extracted by the fourth convolutional layer in environments A_R and A_C.



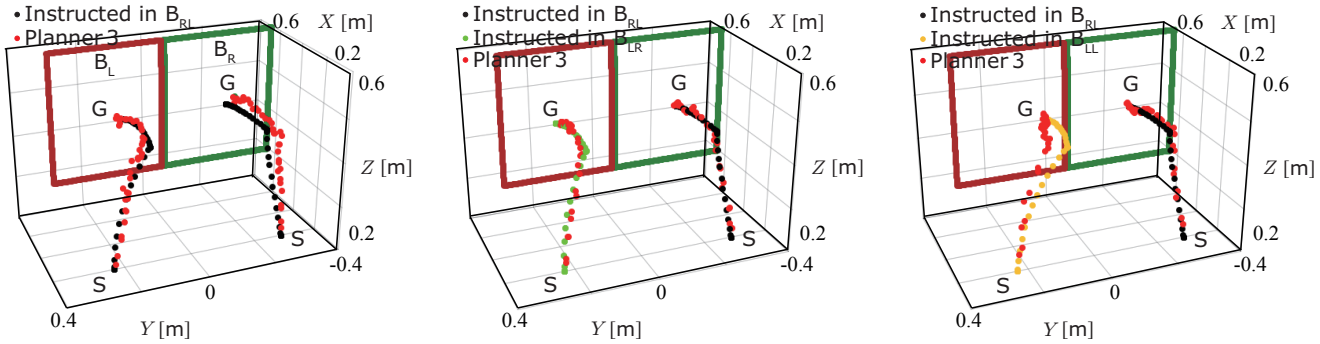
(a) Segmentation image in environment A_R (b) Feature map extracted from Fig. 9(a) in motion planner 2

(c) Segmentation image in environment A_C (d) Feature map extracted from Fig. 9(c) in motion planner 2

(e) Feature map extracted from Fig. 9(a) in motion planner 3 (f) Feature map extracted from Fig. 9(c) in motion planner 3

Fig. 9. Comparison of feature maps extracted in motion planners 2 and 3

Even in environment A_R, where the robot successfully stored the objects, motion planner 2 was unable to extract



(a) Known environment B_{RL} (b) Unknown environment $B_{LR\text{-}RL}$ (c) Unknown environment $B_{LL\text{-}RL}$

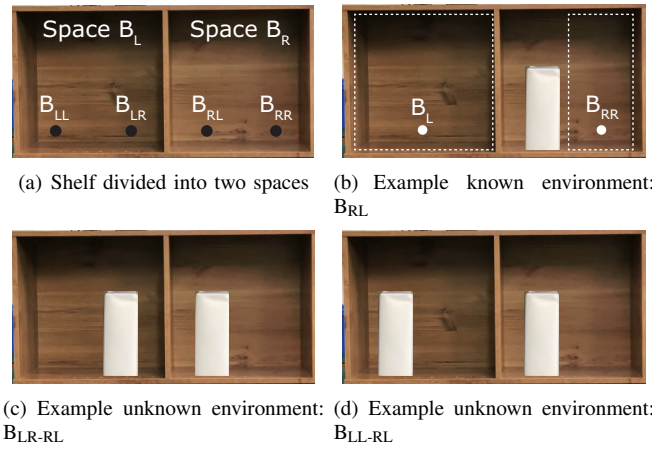
Fig. 11. Comparison of hand trajectories based on motion planner 3 for known and unknown environments

the features of the arms and hands, as shown in Fig. 9(b), for the image input in Fig. 9(a). As a result, the distance between the left and right hands became narrower when storing the objects. For environment A_C, where the robot failed to store the objects, the features of the shelf and the grasped objects vanished in Fig. 9(d), which shows the feature maps extracted from the image input in Fig. 9(c). The vanished features resulted in a collision between the object and the shelf. On the other hand, motion planner 3 clearly extracted the features of the arms and hands, in addition to the objects and shelf, as shown in Fig. 9(e) and Fig. 9(f), for similar image inputs as shown in Fig. 9(a) and Fig. 9(c). As a result, the robot successfully stored the objects into the correct target spaces without causing any collisions between the grasped objects and the shelf. From the results described above, it was shown that applying shortcut connections to the convolutional processes enabled the extraction of important features for object storage from images.

B. Shelf Spaces with Placed Objects

In IV-A, the robot stored the objects in spaces where no objects were previously placed. Moreover, since the three environments were used for motion instruction, these were known to the robot. In this experiment, the robot stores the objects in spaces that already contain objects. Furthermore, environments, where storage motions are not instructed, are used as unknown environments for the robot. The robot and shelf are the same as those shown in Fig. 7(a) and Fig. 7(b), but environments are different as shown in **Fig. 10**.

Unlike Fig. 7(c), the upper shelf in Fig. 10(a) is divided into two spaces. The right space is labeled B_R, and left space is labeled B_L. In each space of B_R and B_L, B_{RL} and B_{LL} represent the left-side positions, while B_{RR} and B_{LR} represent the right-side positions. For imitation learning of the motion planners, the robot is instructed to store objects in the environments, B_{RR}, B_{RL}, B_{LR}, and B_{LL}, where each object is already placed in one of these four positions. For instance, in environment B_{RL}, where an object is placed at position B_{RL} as shown in Fig. 10(b), the robot is allowed to store the object in space B_{RR}, as indicated by the dashed rectangle, in addition to space B_L. In this environment, therefore, the object grasped by the right hand is placed at position B_{RR},



(a) Shelf divided into two spaces (b) Example known environment: B_{RL}

(c) Example unknown environment: $B_{LR\text{-}RL}$ (d) Example unknown environment: $B_{LL\text{-}RL}$

Fig. 10. Object storage experiments for spaces with placed objects

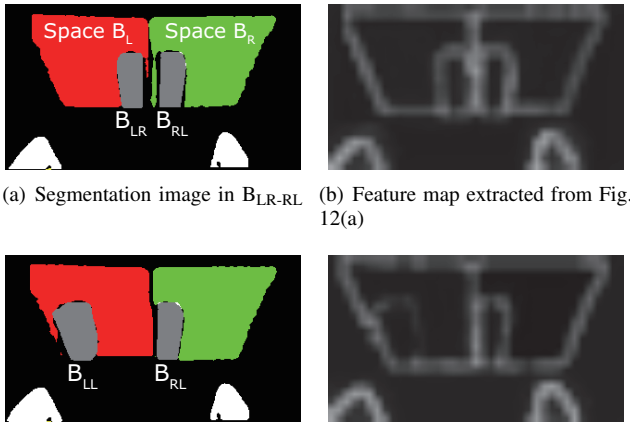
while the object grasped by the left hand is placed at B_L between B_{LR} and B_{LL}. In addition to these positions, the robot is instructed to store the objects at positions ± 20 [mm] along the Y-axis.

In unknown environments, an object is placed in each of the spaces B_R and B_L. For instance, Fig. 10(c) and Fig. 10(d) show unknown environments $B_{LR\text{-}RL}$ and $B_{LL\text{-}RL}$, where objects are placed at positions B_{RL} and B_{LR}, and positions B_{RL} and B_{LL}, respectively. In total, four environments, $B_{LR\text{-}RR}$, $B_{LL\text{-}RR}$, $B_{LR\text{-}RL}$, and $B_{LL\text{-}RL}$, are used, and all are unknown to the robot. Motion planner 3, described in IV-A, is applied to the robot along with the hand state classifier. Both the motion planner and the hand state classifier were retrained through imitation and supervised learning, as described in III-A and III-B, using the training data set acquired in the environments shown in Fig. 10(b). As the results, the robot based on motion planner 3 successfully stored the objects in both the four known and unknown environments. **Fig. 11** shows the hand trajectories in the known environment B_{RL} and the unknown environments $B_{LR\text{-}RL}$ and $B_{LL\text{-}RL}$.

In environment B_{RL}, as shown in Fig. 11(a), the position errors of the objects placed by the robot were 15 [mm] (right) and 25 [mm] (left) along the Y-axis. For reference, the average error for the known environments was 16 [mm].

The robot was instructed to store the objects at the center positions of the spaces and at positions ± 20 [mm] along the Y-axis. Therefore, the average error was within the instructed range. These results indicate that the robot planned the storage motions to closely imitate the instructed ones.

In environments $B_{LR\text{-}RL}$ and $B_{LL\text{-}RL}$, as shown in Fig. 11(b) and Fig. 11(c), the robot planned the storage motion for the right hand to place the object in the right-side position, B_{RR}, in space B_R, as instructed. On the other hand, in the same space, B_L, the storage motion for the left hand was planned to place the object in the left-side position, B_{LL}, in environment $B_{LR\text{-}RL}$, and in the right-side position, B_{LR}, in environment $B_{LL\text{-}RL}$. To discuss the different motions depending on the object placed in the space, **Fig. 12** shows the feature maps extracted by the fourth convolutional layer in the unknown environments $B_{LR\text{-}RL}$ and $B_{LL\text{-}RL}$.



(a) Segmentation image in $B_{LR\text{-}RL}$ (b) Feature map extracted from Fig. 12(a)

(c) Segmentation image in $B_{LL\text{-}RL}$ (d) Feature map extracted from Fig. 12(c)

Fig. 12. Feature maps extracted from segmentation image inputs in unknown environments $B_{LR\text{-}RL}$ and $B_{LL\text{-}RL}$

As shown in Fig. 12(a), which depicts the segmentation image used as input, the objects were placed at the left-side position, B_{RL}, in space B_R, and in the right-side position, B_{LR}, in space B_L. The features of these objects were clearly extracted, as shown in Fig. 12(b). In Fig. 12(c), while one object was also placed at the left-side position, B_{RL}, in space B_R, the other was placed at the different position, B_{LL}, in space B_L. The features of these objects were extracted, as shown in Fig. 12(d). This explains why the robot planned different motions depending on the object placement. As a result, while the robot stored the object in space B_{RR} with the right hand in both unknown environments, the other object was stored in space B_{LL} in environment $B_{LR\text{-}RL}$ and in space B_{LR} in environment $B_{LL\text{-}RL}$, respectively.

Compared to the shelf in IV-A, the shelf used in this experiment was divided into only two spaces. For the shelf composed of these two spaces, the center coordinates of the right space, B_R, and the left space, B_L, were continuously fed as inputs corresponding to the goal directions, Γ_R and Γ_L. Thus, in the three environments B_{RL}, $B_{LR\text{-}RL}$, and $B_{LL\text{-}RL}$, similar goal directions for the left hand, such as $\Gamma_L = (x_C, y_C) = (360, 253)$, $(359, 250)$, and $(367, 248)$, were fed as inputs to the motion planner, regardless of the placed object positions B_{LR} and B_{LL} in space B_L. This result indicates that the robot was not able to plan the storage motion correctly by using the goal directions alone for a shelf with spaces where objects were already placed. Therefore, the effectiveness of the motion planner, which utilized both the shortcut connections and goal directions for object storage, was demonstrated.

V. CONCLUSIONS

Focusing on object storage by a dual-arm robot onto a shelf, this paper proposed a motion planner based on CNN. Goal directions, along with camera images and current hand positions, were used as inputs to the motion planner. Shortcut connections were applied to the convolutional layers in the CNN to process these multimodal inputs. Through the experiments, it was shown that:

- the robot was able to determine the target spaces for storing the grasped objects.
- Clearly extracted image features enabled the robot to plan the storage motion depending on the positions of objects placed in the spaces.
- Finally, the robot successfully stored the objects onto the shelf not only in instructed, known environments but also in unknown environments.

From these results, the effectiveness of the motion planner for object storage by the dual-arm robot was demonstrated.

REFERENCES

[1] Lozano-Perez, "Spatial Planning: A Configuration Space Approach," *IEEE Transactions on Computers*, Vol. C-32, No. 2, pp. 108–120, 1983.

[2] L. E. Kavraki, *et al.*, "Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces," *IEEE Transactions on Robotics and Automation*, Vol. 12, No. 4, pp. 566–580, 1996.

[3] K. Noda *et al.*, "Multimodal Integration Learning of Robot Behavior Using Deep Neural Networks," *Robotics and Autonomous Systems*, Vol. 62, No. 6, pp. 721–736, 2014.

[4] P. Yang *et al.*, "Repeatable Folding Task by Humanoid Robot Worker Using Deep Learning," *IEEE Robotics and Automation Letters*, Vol. 2, No. 2, pp. 397–403, 2017.

[5] S. Hoshino and R. Oikawa, "Reaching Motion Planning with Vision-Based Deep Neural Networks for Dual Arm Robots," *International Conference on Intelligent Autonomous Systems 17*, pp. 455–469, 2022.

[6] Y. Lecun, *et al.*, "Gradient-Based Learning Applied to Document Recognition," *Proceedings of the IEEE*, Vol. 86, No. 11, pp. 2278–2324, 1998.

[7] B. D. Argall, *et al.*, "A Survey of Robot Learning from Demonstration," *Robotics and Autonomous Systems*, Vol. 57, pp. 469–483, 2009.

[8] K. He, *et al.*, "Deep Residual Learning for Image Recognition," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[9] N. Srivastava, *et al*, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *The Journal of Machine Learning Research*, Vol. 15, pp. 1929–1958, 2014.

[10] K. He, *et al.*, "Mask R-CNN," *IEEE International Conference on Computer Vision*, pp. 2961–2969, 2017.

[11] S. Chitta, *et al.*, "Moveit![ROS topics]," *IEEE Robotics and Automation Magazine*, Vol. 19, No. 1, pp. 18–19, 2012.

[12] D. P. Kingma and J. L. Ba, "Adam: A Method for Stochastic Optimization", *Proceedings International Conference for Learning Representations*, pp. 1–15, 2014.

[13] H. Shimodaira, "Improving Predictive Inference under Covariate Shift by Weighting the Log-Likelihood Function," *Journal of Statistical Planning and Inference*, Vol. 90, No. 2, pp. 227–244, 2000.

[14] S. Hoshino, *et al.*, "Imitation Learning based on Data Augmentation for Robotic Reaching," *Annual Conference of the Society of Instrument and Control Engineers of Japan*, pp. 417–424, 2021.