Knowledge-Based Reinforcement Learning for Industrial Robotic Assembly

1st In Jun Park
Digital Convergence Research Laboratory
Electornics and Telecommunications Research Institute
Daejeon, South Korea
ijpark@etri.re.kr

2nd Hyonyoung Han
Digital Convergence Research Laboratory
Electornics and Telecommunications Research Institute
Daejeon, South Korea
hyonyoung.han@etri.re.kr

3rd Joonmyun Cho
Digital Convergence Research Laboratory
Electornics and Telecommunications Research Institute
Daejeon, South Korea
jmcho@etri.re.kr

4th Jun Hee Park
Digital Convergence Research Laboratory
Electornics and Telecommunications Research Institute
Daejeon, South Korea
juni@etri.re.kr

***Abstract*—Although reinforcement learning has shown promise in solving industrial assembly tasks, it still faces challenges such as poor sample efficiency and sparse rewards that limit its learning capability. To address these challenges, we split multi-step assembly tasks into modular sub-tasks and use CAD-based prior knowledge to facilitate sub-task learning. The geometric information from the knowledge extraction module accelerates the learning of grasping and placing. Instead of considering multiple possible placement errors in a jig-free environment during training, which would significantly increase training time, our method uses a compensation module with a spatial transformer network to deal with errors. We evaluated our method on two 3D-printed models with different materials and achieved a completion success rate of 96% for the plastic model and 94% for the metal model. Our results indicate that our model can be robustly applied to products with similar geometry without requiring additional model updates.**

Keywords—Industrial Assembly, Robotic Manipulation, Reinforcement Learning

I. Introduction

Industrial robotic assembly remains a challenging problem, as it involves multi-step tasks that require high precision. With the shift towards more flexible and dynamic manufacturing environments, where expensive jigs and fixtures are no longer used for accommodating various product types, finding effective solutions has become even more important [1]. Deep reinforcement learning methods have been applied to solve robotic manipulation tasks, such as industrial insertion, due to their flexibility and adaptability [2, 3, 4, 5]. However, an end-to-end reinforcement learning approach with sparse rewards has limitations in solving multi-step tasks as it often wanders in large state-action spaces, performing mostly unsuccessful actions. Some recent studies have attempted to use imitation learning [6, 7, 8] to address the issue of sparse rewards, as it provides guidance for robots to take correct actions. Although these studies have shown successful results in various insertion tasks, they rely on expert demonstrations, which may not be feasible in an industrial manufacturing setting. Meta-reinforcement learning is another approach to solving multiple insertion tasks by learning common knowledge across different but similar types of tasks and adapting to new tasks with a small amount of data [5, 9]. However, this method has barriers to being applied in a manufacturing system, as assembly of industrial products is performed sequentially, meaning that tasks change over time. It is impractical to have an adaptation phase every time a new task is given in the middle of the assembly process.

Most studies on insertion tasks have been based on contact-rich manipulation [7, 8, 10, 11, 12] using force sensors, which requires one part to be fixed so that proper actions can be learned from the interactions between the parts. As we aim to target a manufacturing environment without jigs or fixtures, where parts can move in the XY-plane during insertion, we adopt a vision-based system, which is more capable of handling relatively large placement errors. We only use a single camera mounted on the robot arm, which is a simple and cost-effective configuration.

In this work, we propose a data-efficient method for multi-step industrial assembly using knowledge-based reinforcement learning. Our model, aiming to complete assembly tasks for a finished product, consists of distinct modules rather than an end-to-end solution. Utilizing the step design file common in industry, we developed an analyzer to extract knowledge about product structures and assembly tasks, reducing training time for skills like grasping and placing. The grasping and placing models were trained in simulation under standard conditions, ignoring placement errors, with a separate compensation module for unexpected errors. Our method, tested on two 3D-printed product models of different materials, achieved a 96% success rate with plastics and 94% with metal, without further adaptation.

II. Related Works

CAD-based knowledge is proven effective in robotic assembly tasks. In [13], the CAD file was used to create a motion plan guiding reinforcement learning. [14] used the CAD file for instruction generation, perception, and planning. In [15], CAD data-derived component poses combined with compliance control adapted quickly to new assembly tasks. Similarly, our method leverages knowledge from a standard CAD file (STEP file) to enhance reinforcement learning for manipulation skill acquisition.

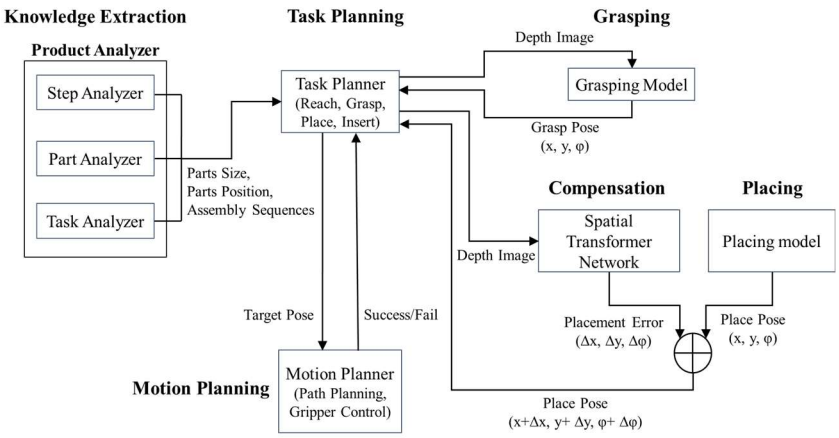

Fig. 1. System overview: The system is composed of modular sub-tasks that interact with each other. Each module is responsible for a specific task, including knowledge extraction, task planning, motion planning, grasping, placing, and error compensation.

Numerous studies have explored robotic grasping with reinforcement learning. QT-Opt [16] developed a system learning grasping from extensive real robot data, though its hardware and computational resources are not broadly applicable. Conversely, [17] proposed a simulated grasping benchmark using off-policy learning, highlighting deep Q-learning's superiority in low data scenarios. VPG [18] improved grasping in cluttered environments using deep Q-learning and pushing. However, most models overlook grasping quality, often dropping objects into bins. We adopt VPG for our grasping model, modifying environmental configurations and design, such as the reward function, to emphasize quality. Our placing model shares the grasping model's architecture but alters the exploration strategy with an action prior based on heightmap, akin to [19], without replacing the temporal difference target to prevent fade-out during training.

III. Methods

To make the assembly task more manageable, we break it down into modular sub-tasks instead of relying solely on reinforcement learning. Furthermore, we integrate the knowledge derived from the 3D product STEP file to streamline RL training by minimizing learning parameters and enhancing sample efficiency. Our system consists of six modules, as shown in Fig. 1: a knowledge extraction module, a task planning module, a motion planning module, a grasping module, a placing module, and a compensation module.

A. Knowledge Extraction Module

Industrial products use the STEP (Standard for the Exchange of Product Data) file for 3D product modeling [20]. We extract product structures and assembly descriptions from STEP using a part analyzer from [21]. The extraction involves several steps. Initially, the STEP analyzer reorganizes product details into a database using a regular expression library. The part analyzer then extracts all parts, analyzing their edges to compute min/max values in 3D space, determining size and position. Considering only stack-structured industrial products, the assembly analyzer sequences parts by their bottom line height, starting with the lowest. Parts are paired by their sequence, and the assembly analyzer identifies joint types like peg-in-hole or snap-fit in the final analysis step.

The analyzers' data is vital for various modules. For example, the part analyzer's height data aids the grasping module in reducing the action space dimension of the reinforcement learning-based grasping model, accelerating training. The part position, also from the part analyzer, informs the placing module's reward function. This reward is calculated based on successful part joining, with the final part position indicating success. The task planning module assigns tasks for each step, relying on assembly sequences and joint types from the task analyzer to execute the assembly task.

B. Task/Motion Planning Module

The task planning module oversees the system, accessing data on part size, position, initial location, and assembly sequence to determine tasks. It assigns basic actions like reaching, grasping, placing, and inserting, each characterized by the gripper's 6D pose. For the reach action, the wrist camera centers on the workspace for a top-view depth image of the target part. The grasp and place actions use the gripper's pose from respective modules, while the insert action is a vertical movement from the place position. Joint types are sequences of basic actions: reach, grasp, place, and insert. The force during the insert action differentiates peg-in-hole (no force, loose coupling) from snap-fit (force for tight joining). The motion planning module, using the Moveit library [22] based on OMPL algorithms [23], manages the robot arm and gripper, including path planning and gripper actions. Grasp success is gauged by the gripper fingers' distance, with success under a specific threshold.

C. Grasping Module

The grasping model is based on vision-based reinforcement learning, where the z-axis of the grasping pose can be determined by the part size extracted from the part analyzer. Thus, we assume that planar motion and rotation around the z-axis of the gripper are sufficient to complete the task, and reduce the dimension of the action space from 6D to 3D, defined by (x, y, φ). This reduction saves a significant amount of training

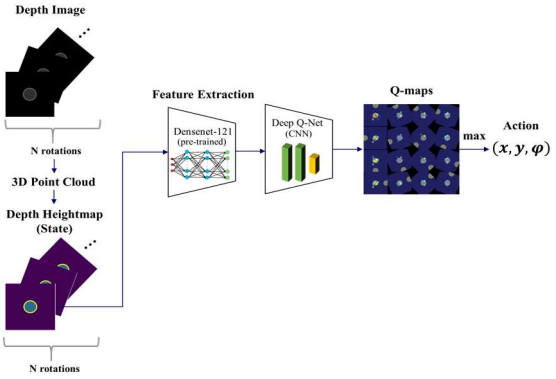

Fig. 2. The architecture of the grasping and placing model based on deep-Q learning.

time and increases the chance of successful training. We modify the grasping model from VPG [18], as depicted in Fig. 2, by incorporating reward shaping and using only the depth image as input. Additionally, instead of using a fixed global camera to cover the entire workspace and randomly grasping objects, we mount the wrist camera on the robot arm and move it to the target assembly part assigned by the task planner to obtain the depth image. Since the grasping model is trained separately from the placing model, the placing model does not receive any feedback on the grasp pose. Therefore, the gripper must grasp the part stably, with no shifting or tilting allowed. We define the reward function as (1), which values grasping with minimal distance between the gripper's position and the part's center position in x and y coordinates.

$$\boldsymbol{R}(s_t, a_t) = \begin{cases} \frac{1}{1+\alpha \cdot d((x,y)_{\text{gripper}},(x_c,y_c)_{\text{part}})}, & \text{if } \frac{1}{1+\alpha \cdot d((x,y)_{\text{gripper}},(x_c,y_c)_{\text{part}})} > \varepsilon_{th} \\ \mathbf{0}, & \text{otherwise} \end{cases} \quad (1)$$

D. Placing Model

The place action is the primary action that must be performed before insertion. Since insertion, which is the fundamental action for peg-in-hole and snap-fit, is defined as a top-down movement from the placing pose, it is crucial to estimate the precise placing pose in order to successfully complete the assembly task. The placing model has the same architecture as the grasping model, as it infers the best placing pose (x, y, φ) from the depth heightmap of the target part, except that the reward function and exploration strategy are defined differently. Since each part position of the completed product is known and extracted from the part analyzer, the success or failure of the assembly can be determined based on whether the x and y positions of two parts match and the height of the part coincides with the one from the analyzer. If the grasped part A is successfully placed and inserted into part B, the reward function is formulated as follows:

$$R(s_t, a_t) = \begin{cases} 1, \text{ if } |x_{\text{part A}} - x_{\text{part B}}| < \varepsilon_{th}, |y_{\text{part A}} - y_{\text{part B}}| < \varepsilon_{th}, |z_{\text{part A}} - z_{\text{height A}}| < \varepsilon_{th} \\ 0, \quad \text{otherwise} \end{cases} \quad (2)$$

The state of the model is represented by the heightmap image, which provides information about the area where the part exists. As exploring empty areas is a waste of time, this insight can be reflected in the action prior defined as (3).

$$AP(s_t, a_t) = \begin{cases} 1, & \text{if } s_t > 0 \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

The action prior is applied to the Q-map and limits the action space by incorporating it into the action selection process. The modified action selection process, which leads to an improvement in sample efficiency, can be represented as:

$$a_t = \max_{a_t} Q(s_t, a_t)\, AP(s_t, a_t) \quad (4)$$

E. Placement Error Compensation

In a jig-free environment, the assembly parts can move in the x-y directions while being inserted into the target part. This movement can cause errors and potentially lead to assembly failure, as our placing model is trained under the assumption that the target part is always located at the center of the workspace. Therefore, if the target part is shifted in the x or y-axis, the performance of the placing model may degrade, resulting in a failure of the assembly process.

To address this issue, we incorporate a compensation module using a spatial transformer network [24], which learns the affine transformation of the input image. The affine transformation is a linear transformation that includes translation, scaling, rotation, shears, and their combinations, preserving points, lines, and planes [25]. The spatial transformer network is originally designed for learning models that are geometrically invariant to diversely transformed images, facilitating tasks such as image classification and localization [24]. We use this network to learn how much the input image is transformed from the reference image in an unsupervised manner by reconstructing the target reference image. This approach is similar to the work in [26], but we do not apply the reference image to the network. Instead, we only use it for computing loss during the training. Fig. 3 illustrates the architecture of the spatial transformer network applied to our product model. The localization network takes the transformed input image and produces a 2x3 parameter matrix related to scaling, rotation, and translation. The grid generator transforms the regular grid over the output image to the sampling grid. The sampler produces the transformed output image by applying the sampling grid to the input image. Instead of computing the loss from the label like image classification, we use the mean squared error between the reference image and the output image as the loss for training. Thus, the loss function is given by:

$$\text{Loss} = \frac{1}{K}\sum_{k=1}^{K}\left(\frac{1}{N}\frac{1}{M}\sum_{i=1}^{N}\sum_{j=1}^{M}\left(\text{I}_{\text{Ref}}(i,j) - \text{I}_{\text{Trans}}(i,j)\right)^2\right) \quad (5)$$

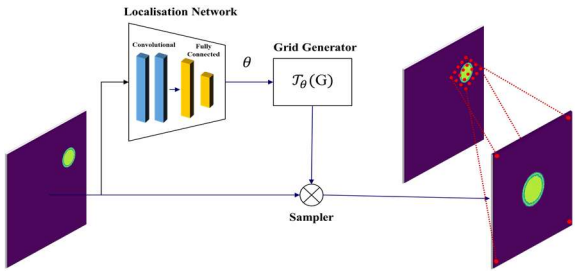

Fig. 3. Placement error compensation using STN.

Here, K denotes the batch size of input images, and N, M are the width and height of the image, respectively. In our product model, I_{Ref} represents the image of the target part located at the center of the workspace, which is used during the training of the placing model. The learned placing pose can then be compensated with the parameters estimated from the spatial transformer network to perform the insert action properly. The transformation parameters can be expressed as a 2x3 matrix:

$$\begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} = \begin{bmatrix} s \cdot \cos \Delta\varphi & -s \cdot \sin \Delta\varphi & \Delta x \\ s \cdot \sin \Delta\varphi & s \cdot \cos \Delta\varphi & \Delta y \end{bmatrix} \quad (6)$$

where the scaling factor is given by $s = \sqrt{\theta_{11}{}^2 + \theta_{12}{}^2}$, the rotation angle $\Delta\varphi$ is computed as $\Delta\varphi = \tan^{-1}\left(-\frac{\theta_{12}}{\theta_{11}}\right)$, and the translation values in the x and y directions are denoted by Δx and Δy, respectively. Using these parameters, we can compute the compensated placing pose as follows:

$$(x_{comp}, y_{comp}, \varphi_{comp}) = (x_{train} + \Delta x, y_{train} + \Delta y, \varphi_{train} + \Delta\varphi) \quad (7)$$

IV. Experiments

A. Experimental Setup

Fig. 4 illustrates the experimental setup and the test objects used in our real-world experiments. The multi-step assembly task was performed using a UR5 robot arm equipped with a Robotiq 2F-85 gripper. To capture the depth image of the target part, we mounted a RealSense L515 camera on the robot wrist. We evaluated our method using two 3D-printed product models made of different materials, namely plastics and metal. This allowed us to assess the generalization capability of our assembly model to similar types of products without requiring additional adaptation.

B. Knowledge Extraction Using Product Analyzer

To generate knowledge about the test product, we used the product analyzer, which comprises step, part, and task analyzers. Fig. 5 displays the extracted information about the parts composing the test product from the test product step file. The product's structure describes the list of joints and parts comprising each joint. The test product has four joints where each joint is composed of a pair of parts, except that the first joint is just a single part. The size of each part is expressed in

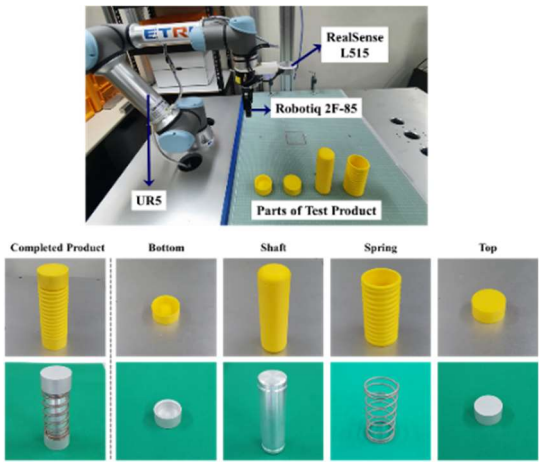

Fig. 4. Real-world experiment setup (upper) and test products (lower).

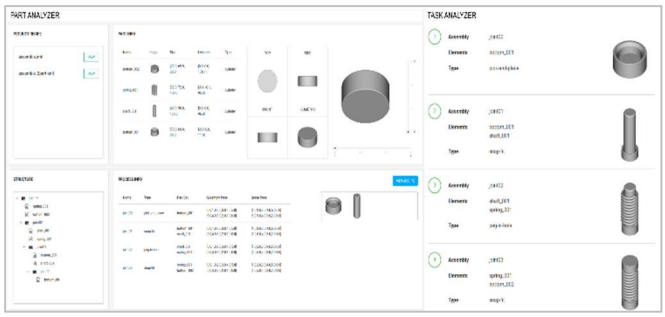

Fig. 5. Knowledge extraction: part analyzer(left) and the task analyzer

Cartesian coordinates. The extracted locations of the parts indicate the part position (x, y, z) when the assembly is successfully completed. The task analyzer analyzes the assembly sequences, which are also shown in Fig. 5. Each step is specified with the parts required for conducting the task and the necessary joint type. The test product requires four steps to finish the tasks. At the first step, the bottom is picked and placed at the center of the workspace. Then, the shaft is snap-fitted to the bottom. Next, the peg-in-hole between the spring and shaft is performed. At the last step, the top is snap-fitted to the shaft.

C. Training in Simulation

Grasping Model: The grasping model was trained in a simulated CoppeliaSim environment [27]. The product's 3D CAD file was converted to an STL format for CoppeliaSim. Each part was assumed to have a fixed position, depicted in Fig. 4. In each training episode, the wrist camera approached the target part, capturing a 224x224 pixel cropped depth image. This image was transformed into a heightmap and input into the grasping network to determine the most stable grasp pose. The reward function, given by (1), gauged the grasp quality, with the model learning the most stable grasps without compromising placing performance. Fig. 6 displays the grasping test results, showing all parts centrally grasped, the optimal position. Given the cylindrical nature of all parts, grasp rotation didn't impact performance.

Placing Model: The training of the placing model was also conducted in simulation, assuming a stably grasped part and a centrally positioned target part, with no target part movement considered. In each episode, the wrist camera's depth image of the target part is converted into a heightmap. The placing model learns the pose for successful part mating, performing random tasks from the assembly sequences in each episode, as depicted in Fig. 7. Along with the inserting action, the model learns three different assembly types. After convergence, successful placing and joining of parts occur, as shown in Fig. 7. The learning curve in Fig. 8 illustrates the assembly success rate over training, with the action prior defined in (3) accelerating learning compared to

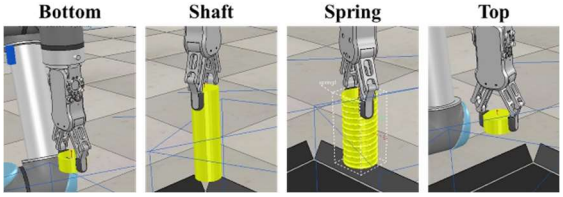

Fig. 6. Grasping test results in simulation

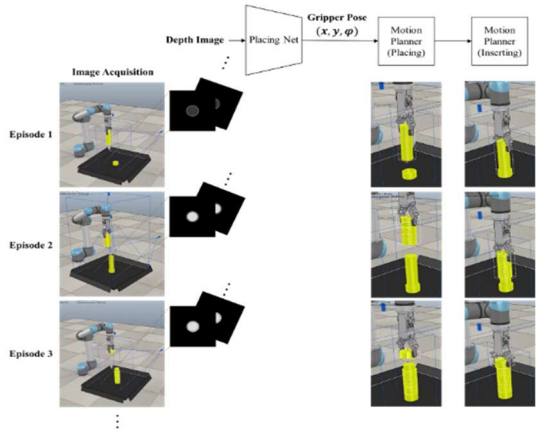

Fig. 7. Training the placing model in simulation

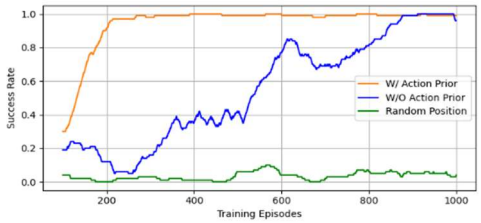

Fig. 8. Learning curve for the placing model. The model using the action prior learns faster. Training fails under the random setting.

without it. Training was also attempted with target part movement in the x-y axis during insertion for a jig-free environment, but the model failed to learn the placing pose even after extended training. The high precision required for assembly makes finding a solution in random settings highly challenging.

Compensation Model: The compensation model employs a spatial transformer network to learn translations and rotations compensating for target part spatial transformations. For training, 60,000 transformed images were generated from three reference images used in the placing model training. These images were top-down views of centered target parts. Scaling ranged from 0.5 to 1.2 in 0.1 increments, translation from -5 cm to 5 cm in 1 cm steps, and rotation from 0° to 360° in 45° steps. The model was assessed with 5,000 transformed images. Table 1 displays the root mean squared error of estimated parameters, excluding rotation due to cylindrical part shapes. The average translation error was under one pixel, suggesting accurate placing pose compensation. Fig. 9 presents test results from 24 images, confirming the model's precise spatial parameter learning and its ability to match input images to reference images.

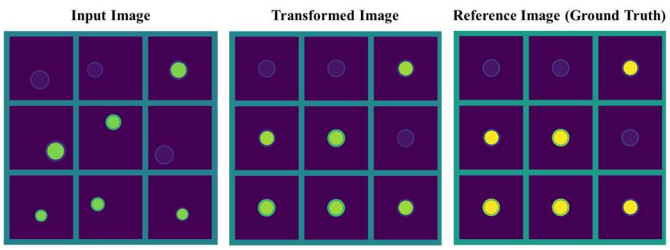

Fig. 9. The results of spatial transformer network.

TABLE I. RESULTS OF COMPENSATION MODEL EVALUATION

Transformation Parameters		RMSE
Scaling		0.0149
Translation	Δx	0.6722
	Δy	0.6061

D. Real-World Results

We tested our system in a real-world setting, depicted in Fig. 10, with the setup detailed in section A. Aiming to successfully complete a multi-step assembly for a finished product, we evaluated grasping, placing, and overall completion. We examined two product models with similar structures but slightly different sizes, shapes, and materials (plastics and metal). Across 100 trials (50 for each product), we intentionally displaced the target part up to 5 cm from the center in 80% of trials to induce placement errors. The results in Table 2 revealed a high assembly accuracy of 96% for plastics and 94% for metal. Failures stemmed from unstable grasping, where the estimated grasp pose deviated from the part's center. We concluded that further training was unnecessary for products with similar shapes and sizes.

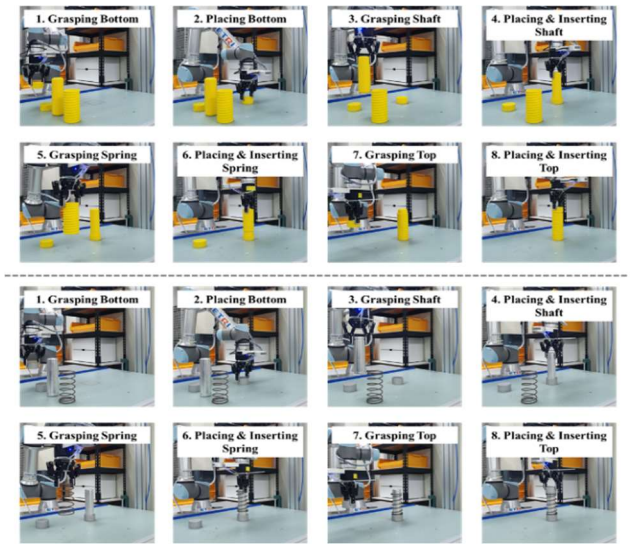

Fig. 10. Multi-step assembly execution using UR5 with two products.

TABLE II. ASSEMBLY TASK SUCCESS RATE (%)

Success Rate (%)		Product Material	
		Plastics	Metal
Grasping	Bottom	100	100
	Shaft	96 (48/50)	94 (47/50)
	Spring	100	100
	Top	100	100
Placing & Inserting	Bottom-Shaft	100	100
	Shaft-Spring	100	100
	Spring-Top	100	100
Completion		96	94

V. Discussion and Future Work

We present a method for tackling multi-step industrial assembly tasks using knowledge-based reinforcement learning, which successfully completed the tasks with a high accuracy of 96%. For future work, we aim to examine this method on various product types with more complex structures and shapes. Additionally, we plan to consider a more flexible environment where parts are randomly placed on a workspace or stacked in bins, as seen in real manufacturing settings. This will require a perception module capable of identifying each part's ID and estimating its position, allowing the task planner to assign appropriate parts for grasping and placing. Furthermore, we will study the vision-based decision module to determine the success or failure of the assembly at each step.

Acknowledgment

This work was supported by Electronics and Telecommunications Research Institute (ETRI) grant funded by the Korean government (23ZR1130, A Study of Hyper-Connected Thinking Internet Technology by autonomous connecting, controlling and evolving ways) and Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (No.2022-0-00187, Development of an edge brain framework to make manufacturing equipment and robots intelligent and No.2022-0-01049, Development of teaching-less product assembly system for smart factory based on autonomous robot task planning and manipulation). We would like to extend our special thanks to Jae Young Lee, a Ph.D. student in SSIT Lab. at KAIST, for his valuable suggestions and insightful discussions.

References

[1] H. Lasi, P. Fettke, and H.-G. Kemper, T. Feld, and M. Hoffmann, "Industry 4.0," in *Business & Information Systems Engineering*, 2014, vol. 6, no. 4, pp. 239–242.

[2] G. Schoettler, A. Nair, J. Luo, S. Bahl, JA. Ojea, E. Solowjow, and S. Levine, "Deep reinforcement learning for industrial insertion tasks with visual inputs and natural rewards," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5548–5555.

[3] T. Inoue, G. De Magistris, A. Munawar, T. Yokoya and R. Tachibana, "Deep reinforcement learning for high precision assembly tasks," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 819-825.

[4] J. Luo, E. Solowjow, C. Wen, J. A. Ojea, A. M. Agogino, A. Tamar, and P. Abbeel, "Reinforcement learning on variable impedance controller for high-precision robotic assembly," in *2019 International Conference on Robotics and Automation (ICRA) IEEE*, 2019, pp. 3080–3087.

[5] G. Schoettler, A. Nair, J. A. Ojea, S. Levine and E. Solowjow, "Meta-reinforcement learning for robotic industrial insertion tasks," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 9728-9735.

[6] L. Berscheid, P. Meißner, T. Kröger, "Self-supervised learning for precise pick-and-place without object model," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4828-35, June 2020.

[7] Y. Wang, CC. Beltran-Hernandez, W. Wan, K. Harada, "An adaptive imitation learning framework for robotic complex contact-rich insertion tasks," *Front Robot AI*, vol. 8, no. 777363, Jan. 2022.

[8] O. Spector, V. Tchuiev, and D. Di Castro, "InsertionNet 2.0: minimal contact multi-step insertion using multimodal multiview sensory input," in *2022 International Conference on Robotics and Automation (ICRA*), 2022, pp. 6330-6336.

[9] TZ. Zhao, J. Luo, O. Sushkov, R. Pevceviciute, N. Heess, J. Scholz, S. Schaal, and S. Levine, "Offline meta-reinforcement learning for industrial insertion," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 6386-6393.

[10] S. Levine, N. Wagener, and P. Abbeel, "Learning contact-rich manipulation skills with guided policy search," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 156-163.

[11] T. Davchev, K. S. Luck, M. Burke, F. Meier, S. Schaal and S. Ramamoorthy, "Residual learning from demonstration: adapting DMPs for contact-rich manipulation," in *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4488-4495.

[12] O. Spector and M. Zacksenhouse, "Deep reinforcement learning for contact-rich skills using compliant movement primitives," arXiv preprint arXiv:2008.13223, 2020.

[13] G. Thomas, M. Chien, A. Tamar, J. A. Ojea and P. Abbeel, "Learning robotic assembly from CAD," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 3524-3531.

[14] R. A. Knepper, T. Layton, J. Romanishin, and D. Rus, "Ikeabot: An autonomous multi-robot coordinated furniture assembly system," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 855–862.

[15] G. Gorjup, G. Gao, A. Dwivedi and M. Liarokapis, "A flexible robotic assembly system combining CAD-based localization, compliance control, and a multi-modal gripper," in *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 8639-8646.

[16] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine, "Scalable deep reinforcement learning for vision-based robotic manipulation," in *Conference on Robot Learning (CoRL)*, 2018, pp. 651-673.

[17] D. Quillen, E. Jang, O. Nachum, C. Finn, J. Ibarz, S. Levine, "Deep reinforcement learning for vision-based robotic grasping: A simulated comparative evaluation of off-policy methods," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 6284-6291.

[18] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser, "Learning synergies between pushing and grasping with self-supervised deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 4238-4245.

[19] A. Hundt, B. Killeen, N. Greene, H. Wu, H. Kwon, C. Paxton, GD. Hager, "Good robot!": Efficient reinforcement learning for multi-step visual tasks with sim to real transfer", *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6724-6731, 2020.

[20] "STEP-file, ISO 10303-21", Library of Congress. Jan. 2017.

[21] H. Han, H. Kim and J. Son, "Product description recipe generation from 3D STEP model for autonomous task planning," in *2021 21st International Conference on Control, Automation and Systems (ICCAS)*, 2021, pp. 852-856.

[22] D. Coleman, I. Sucan, S. Chitta, N. Correll. "Reducing the barrier to entry of complex robotic software: a MoveIt! case study", *Journal of Software Engineering for Robotics*, vol. 5, no. 1, pp. 3-16, May 2014.

[23] IA. Sucan, M. Moll, LE. Kavraki, "The open motion planning library," *IEEE Robotics & Automation Magazine*, vol. 19, no.4, pp. 72-82, Dec. 2012.

[24] M. Jaderberg, K. Simonyan, A. Zisserman. "Spatial transformer networks,", *Advances in neural information processing systems*, vol.28, 2015.

[25] EW. Weisstein, "Affine transformation,", from MathWorld--A Wolfram Web Resource, URL: https://mathworld.wolfram.com/AffineTransformation.html.

[26] G. Balakrishnan, A. Zhao, M. R. Sabuncu, A. V. Dalca and J. Guttag, "An unsupervised learning model for deformable medical image registration," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9252-9260.

[27] E. Rohmer, SPN. Singh, and M. Freese, "CoppeliaSim (formerly V-REP): a Versatile and Scalable Robot Simulation Framework (PDF)," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 1321–1326.