IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 4, 2020, accepted August 3, 2020, date of publication August 10, 2020, date of current version August 21, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3015245

Reinforcement Learning-Based Path Generation Using Sequential Pattern Reduction and Self-Directed Curriculum Learning

TAEWOO KIM AND JOO-HAENG LEE
Department of Computer Software, Korea University of Science and Technology, Daejeon 34113, South Korea
Human-Robot Interaction Research Group, Electronics and Telecommunications Research Institute, Daejeon 34129, South Korea

Corresponding author: Joo-Haeng Lee (joohaeng@etri.re.kr)

This work was supported by the Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2017-0-00162, Development of Human-care Robot Technology for Aging Society).

ABSTRACT Recent advancements in robots and deep learning have led to active research in human-robot interaction. However, non-physical interaction using visual devices such as laser pointers has gained less attention than physical interaction using complex robots such as humanoids. Such vision-based interaction has high potential for use in recent human-robot collaboration environments such as assembly guidance, even with a minimum amount of configuration. In this paper, we introduce a simple robotic laser pointer device that follows an arbitrary planar path and is designed to be a visual instructional aid. We also propose an image-based automatic path generation method using reinforcement learning and a sequential pattern reduction technique. However, such vision-based human-robot interaction is generally performed in a dynamic environment, and it can frequently be necessary to calibrate the devices more than once. In this paper, we avoid the need for this re-calibration process through episodic randomization learning and improved learning efficiency. In particular, contrary to previous approaches, the agent controls the curriculum difficulty in a self-directed manner to determine the optimal curriculum. To our knowledge, this is the first study of curriculum learning that incorporates an explicit learning environment control signal initiated by the agent itself. Through quantitative and qualitative analyses, we show that the proposed self-directed curriculum learning method outperforms ordinary episodic randomization and curriculum learning. We hope that the proposed method can be extended to a general reinforcement learning framework.

INDEX TERMS Path generation, robotic laser pointer, deep reinforcement learning, curriculum learning.

I. INTRODUCTION

A. BACKGROUND

With the advancements in machine learning, research on deep learning-based human-robot interaction has attracted much attention recently [1]–[3]. Humans can interact with robots in various forms, and these methods can be categorized into physical interactions and non-physical (untact) interactions. For physical interactions (e.g., hand shaking, high fives, and manipulating objects), complex and high-cost robots such as humanoid robots are required. In contrast, non-physical interaction requires a relatively simple and low-cost robot such as a device that provides visual information using a beam projector [4]–[6] and an artificial intelligence speaker [7], [8]. In particular, unlike the recently commercialized artificial intelligence speaker, a visual-based interactive robot such as beam projector-based robot is highly expected to be useful in the future and should be studied and developed.

Such a visual-based interactive robot can be configured at a relatively lower cost and with a simpler structure than a humanoid robot or articulated manipulator. For human–robot visual interaction in three-dimensional (3D) space, the robot should have at least three degrees of freedom, and its simplest incarnation is a robotic laser pointer (RLP) device that consists of pan-tilt joints and a laser pointer. Although a laser pointer cannot provide high-dimensional visual information like a beam projector, effective provision of visual information and interaction with humans is possible when it is equipped with motorized laser control [9]. For example, the RLP can provide visual information such as a warning

The associate editor coordinating the review of this manuscript and approving it for publication was Joanna Kołodziej.

signal or emotional expressions to users by generating repetitive motor trajectories for specific patterns. Moreover, when combined with a camera sensor, it can be utilized in guidance for assembly, path planning, and object finding applications. For these applications, it is essential to equip the robot with image-based path generation skills, which generate laser trajectories from target patterns such as the contour of a target object.

In previous approaches, image-based path generation basically requires complex manual processes such as image processing, curve fitting [10], path planning [11], and calibration [12]. In a dynamic environment in particular, frequent recalibration is expected because the positions and orientations of the robot, camera sensor, and target objects are more likely to change. Therefore, there is a high need for a learning-based method so that the robot can perform path generation with much less effort than would be needed for manual programming in an uncalibrated environment.

In this paper, we propose an image-based path generation method that is able to generate paths in an uncalibrated environment using reinforcement learning. The path generation process can be formulated as a sequential visiting problem for each region of a given pattern, where the visitor (agent) and the path are regarded as the pixels of laser point and the pattern region respectively. We modeled this problem as a sequential pattern reduction (SPR) problem in which the laser pointer sequentially reduces the pattern (path). In addition, we designed a Markov decision process (MDP) that includes an SPR reward function to employ RL in our learning system.

In the image-based path generation problem, the agent should be able to generate paths even in an environment that is not calibrated. To this end, we applied an episodic randomization learning approach that randomizes the environment parameters in every episode [13], [14]. However, the method converges slowly because difficult tasks are given from the initial state. Moreover, local minima can be a problem. To solve these problems, this paper proposes a self-directed curriculum (SDC) based learning. Through the proposed SDC and reward function, the agent is able to choose an appropriate level of task difficulty for itself considering its capability and learns the target task effectively during training. We experimentally show that the proposed SDC outperforms other existing learning methods.

B. RELATED WORK

There have been many attempts to automatically generate paths from the visual data of target patterns. In robotic path generation based on 3D point cloud data [15]–[19], the target object's shape is scanned by a 3D scanner, the data are converted to point cloud data, and these data are used to generate the object's surface. The surface is used to generate tool paths in automatic computer numerical control (CNC) machining [15]. These methods require precise 3D sensors and reverse engineering, which transforms the point cloud data into a CAD model.

Automatic robot path generation from 2D images, which can be thought of as image-based visual servoing [20] have been studied intensively in many previous studies. Pachidis and Lygouras [21] studied a path generation method for robotic arc welding from 2D images. Given a stereo-vision image, the corresponding line segments are obtained by image processing and a correspondence algorithm. Then, from the detected edges, robot paths are generated by the path point calculation algorithm. Aritos *et al.* [22] proposed a robot path generation method from lines on a flat 2D planar surface image using image processing and robot-camera calibration. Chang *et al.* [10] proposed an image-based motion planning method using Pythagorean-hodograph (PH) splines. Using this approach, they are able to follow the contours of an object with proper acceleration/deceleration using a PH quantic spline interpolator in an eye-to-hand camera structure [23].

Recently, there have been attempts to reduce the number of calculations needed for curve fitting and path generation during runtime based on a learning approach. Li *et al.* [24] utilized RL to generate a fast and smooth tool path for a target pattern in a calibrated environment for CNC applications. Using deep RL, they replaced most steps of the previous pre-planning smoothing method with a neural-network based method and achieved the development of an algorithm involving fewer calculations than the usual solution. Jing *et al.* [25] proposed a computational framework for robot path generation for surface/shape inspection application. They used an RL-based tree search algorithm to efficiently generate online paths based on the proposed MDP formulation to solve the coverage planning problem [11]. In the research cases mentioned above, RL has shown remarkable performance in various studies including computer games [26]–[28] and graphics [29], [30]. Especially in the robotics field, RL is used to learn visuomotor skills [31], [32], path planning [33], navigation [34], [35], and human-robot interaction [36]. In this paper, we also used RL as our main learning method.

The aforementioned methods typically adopt the following set of steps: point data extraction from the image, curve generation using curve fitting methods (using, e.g., B-splines [37] or Bezier curves [38]), and then path planning for the curves. This procedure has become the general approach of applications that require high precision such as CNC and computer aided manufacturing [39], [40]. Moreover, in the learning-based methods, deep learning is used as a subprocess of the main algorithm. However, because our aim is to develop an end-to-end learning-based method in the context of human-robot interaction while minimizing the effort needed for manual programming in uncalibrated environments, an evaluation of the precision is outside the scope of this paper.

C. CONTRIBUTIONS

In this paper, we propose a deep RL-based robotic path generation method that is efficient at learning and does not require mathematical modeling for the target patterns or

calibration. Our main contributions can be summarized as follows:

- For given target patterns, we propose a deep RL-based SPR method for automatic path generation without any prior knowledge of mathematical and geometric models.
- We propose a system architecture characterized by a novel reward function and distributed learning framework for fast and effective learning.
- For learning unknown patterns in uncalibrated environments, we propose a SDC learning method that automatically controls the level of difficulty of the task considering the current capability of its own policy. We experimentally verified the superiority of the SDC learning method.

The remainder of the paper is organized as follows. Preliminaries for our algorithm are presented in Section II. In Section III, we describe the details of the proposed method, including the SPR algorithm, MDP design, distributed learning environment, and SDC learning. Experiments and an ablation study are presented in Sections IV and V, respectively, along with their results, and finally the paper is concluded in Section VI.

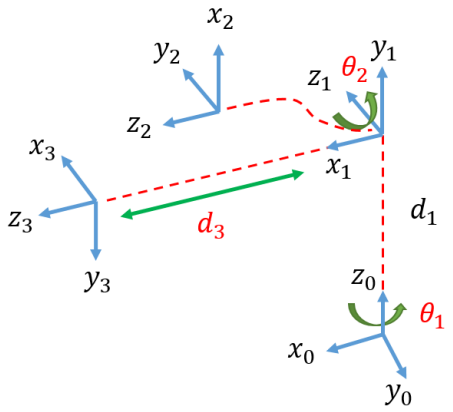


FIGURE 1. Kinematic diagram of the RLP. The kinematics of the RLP are described by two revolute joints and a prismatic joint, where θ_1 and θ_2 correspond to pan and tilt and d_3 is the laser pointer distance.

II. PRELIMINARIES

A. ROBOTIC LASER POINTER

We designed a simple RLP that consists of two revolute joints (for pan and tilt) and a laser pointer (see Figs. 1 and 2). The laser pointer's on/off action and the pan-tilt motors are controlled by an embedded processor board and the main PC controls the device through serial communication. To control the laser pointer in Cartesian space, we derived the analytic inverse kinematics of the RLP using trigonometric functions based on its kinematic structure. The kinematics are described by the Denavit–Hartenberg parameter [41] of Table 1 and Fig. 1 as follows:

$$\boldsymbol{x}_r^d = (R_z(-\pi/2) \cdot R_x(-\pi/2))^{-1} \cdot (\boldsymbol{x}_w^d - \boldsymbol{o}_w^{\text{tilt}}), \quad (1)$$

$$\theta_{\text{pan}} = \arctan 2(-x_{r.x}^d, x_{r.z}^d), \quad (2)$$

$$\theta_{\text{tilt}} = \arctan 2\left(-x_{r.y}^d, \sqrt{{x_{r.z}^d}^2 + {x_{r.x}^d}^2}\right), \quad (3)$$

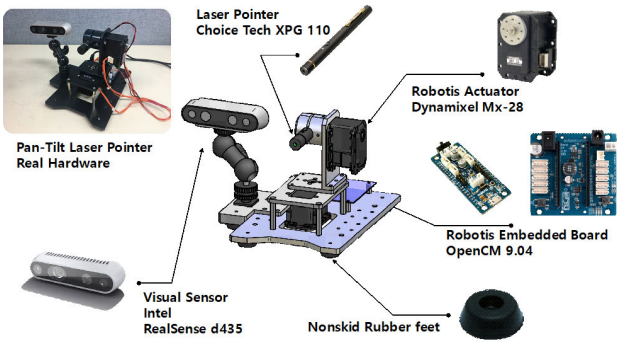


FIGURE 2. (Top left) RLP. (Bottom) Design of the RLP and its components.

TABLE 1. Denavit–Hartenberg parameter of the RLP device. The laser pointer movement can be modeled by a prismatic joint.

i	θ_i	α_i	d_i	a_i
1	θ_1	$\pi/2$	0.119	0
2	$\theta_2 + \pi/2$	$\pi/2$	0	0
3	$\pi/2$	0	d_3	0

where $\boldsymbol{x}_w^d = (x_{r.x}^d, x_{r.y}^d, x_{r.z}^d)$ represents the desired position of the laser pointer in Cartesian space and $\boldsymbol{o}_w^{\text{tilt}}$ is the origin of the tilt joint. Moreover, R_z and R_x are 3×3 rotation matrices and their inverse is multiplied by the desired position to produce that position in robot coordinates. We set the origin of the pan coordinate system to the origin of the world coordinates and the pan and tilt joint angles are calculated by an arc tangent function.

B. REINFORCEMENT LEARNING

In this paper, the robot agent generates paths only from the observed target pattern images. Therefore, we model the robotic pattern generation problem as a partially observable MDP [42], [43] with a tuple $\mathcal{M} = \{\mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{T}, r, \gamma, \mathbb{S}\}$, where each element represents a space for a partial observation of state, action, state transition probability $\mathcal{T}(s_{t+1}|s_t, a_t)$, reward function $r : S \times A \to \mathbb{R}$, discount factor $\gamma \in (0, 1]$, and initial state distribution $\mathbb{S}$, respectively. The agent learns a deterministic rule $\pi : O \to A$ that maximizes the expected discounted reward from the initial reward R_0 over a finite horizon:

$$J = \mathbb{E}_{\mathbb{S}}[R_0|\mathbb{S}]. \quad (4)$$

The return at time t is defined by the discounted reward:

$$R_t = \sum_{i=t}^{T} \gamma^{i-t} r(s_i, a_i), \quad (5)$$

where $r(s_i, a_i)$ returns a reward when the agent performs an action a_i at state s_i. The return R_t at time t is defined by the sum of discounted rewards during T. In recent RL-based studies, the actor–critic network [44] has become a popular method for building the agent's policy because its performance is more stable and better than when using the

actor policy alone. Based on this, we previously introduced an asymmetric actor–critic network [45] to our policy for Q-function-based policy evaluation. The critic network Q_ζ estimates the Q-function, which describes the expected return from action a_t at state s_t as follows:

$$\begin{aligned} Q^\pi(s_t, a_t) &= \mathbb{E}_\pi[R_t|s_t, a_t] \\ &= \mathbb{E}_\pi[r(s_t, a_t) + \gamma Q^\pi(s_{t+1}, a_{t+1})|s_t, a_t], \end{aligned} \quad (6)$$

where (6) is the Bellman equation, which represents the expected action-value (Q-value) of the current state s_t and action a_t estimated by the discounted Q-function of the next state s_{t+1} and action a_{t+1}. The input to the critic is a feature vector consisting of the current observation and state with current action generated by the actor π_ω. The experiences of the agent are gathered from the simulator and stored as a set of tuples $(o_t, s_t, a_t, q_t, r_t)$ in the rollout memory. Each element of the tuple represents the current observation, state, action, Q-value, and reward, which are described in a later section.

C. PROXIMAL POLICY OPTIMIZATION USING THE Q-FUNCTION

In recent RL-based studies, proximal policy optimization (PPO) [46] has shown superior performances especially in robotic tasks [47], [48] and character animation [29]. Because our agent is a kind of robotic device, we also trained using the PPO algorithm. PPO optimizes the actor–critic policy network based on the following conservative policy iteration L^{CPI}, clipped surrogate objective L^{CLIP}, and squared Bellman error loss L^{QF} as follows:

$$L^{CPI}(\omega) = \mathbb{E}_\pi[\varphi_t(\omega)A_t], \quad \varphi_t(\omega) = \frac{\pi_\omega(a_t|o_t)}{\pi_{\omega_{old}}(a_t|o_t)}, \quad (7)$$

$$\begin{aligned} L^{CLIP}(\omega) = \mathbb{E}_\pi[&\min(\varphi_t(\omega)A_t, \\ &\text{clip}(\varphi_t(\omega), 1-\epsilon, 1+\epsilon)A_t)], \end{aligned} \quad (8)$$

$$\begin{aligned} L^{QF}(\zeta) = &(r(s_t, a_t) \\ &+ \gamma Q_\zeta(s_{t+1}, a_{t+1}) - Q_\zeta(s_t, a_t))^2, \end{aligned} \quad (9)$$

where $\varphi_t(\omega)$ represents the ratio between the current and previous policy's action probability given an observation. Instead of the value function error used in [46], we define the Q-function error loss for evaluation by the critic network during training [45]. The final objective is defined by the weighted sum of those three objectives as

$$\begin{aligned} &L^{CLIP+QF+S}(\omega, \zeta) \\ &\quad = \mathbb{E}_\pi[L^{CLIP}(\omega) - c_1 L^{QF}(\zeta) + c_2 S(\pi_\omega, o_t)], \end{aligned} \quad (10)$$

where S denotes an entropy bonus term, $c1$ and $c2$ are weights for L^{QF} and S, which are set to 1.0 and 0.01, respectively, in this paper.

D. RANDOM ENVIRONMENTS AND CURRICULUMS

The episodically randomized environment (ERE) is widely used in RL to handle task variance [13], [14]. We also adopted ERE in our solution to eliminate the need for re-calibration in an unknown environment. However, the ERE makes policy convergence difficult and slow because challenging tasks are given from the early stages of learning and rewards are sparse. To solve this issue, curriculum learning has been used in many papers. However, most previous work manually determines the curriculum without considering the maturity of policy learning [49]–[52]. While some recent studies consider the policy capability [53]–[56], the update rules are still determined manually or based on temporal difference error [55].

Our proposed SDC learning, in contrast, focuses on self-directedness: the policy actively controls the curriculum difficulty by itself; it is not controlled by a human. In particular, our method is distinguished from others by the incorporation of an explicit control parameter τ for curriculum difficulty. To our knowledge, this is the first study of curriculum learning equipped with a self-directed control signal from an agent to the learning environment. Details are described in Section III-D.

TABLE 2. Property of each learning method in terms of random environments and curriculums: EIE, ERE, LIC, and SDC.

	EIE	*ERE*	*LIC*	*SDC*
Random Environment	No	Yes	Yes	Yes
Curriculum Support	No	No	Yes	Yes
Curriculum Control	-	-	Linear	Adaptive

In our experiments, we compare four methods characterized by their handling of random environments and curriculum. Table 2 lists the properties of each learning method. Episodically invariant environment (EIE) learning is based on a fixed learning environment without randomness or a curriculum. ERE learning episodically randomizes the environment without a curriculum. Linearly increasing curriculum (LIC) learning has both randomness and a curriculum, but the level of randomness (curriculum) is linearly increased. In SDC, randomized learning is performed with a curriculum that is determined by the policy itself. The criteria for self-curriculum determination is described in Section III-D.

For the random environments listed in Table 2, we randomly reset the positions and orientations of the target plane, camera, target pattern texture, and initial position of the laser point with respect to their initial state during training. For the target pattern texture, only the x- and y-directional position (x_p, y_p) and orientation with respect to the z-axis (z_p) in the local coordinate of the target plane are considered in the randomized reset (see Fig. 4). The randomization is triggered at the beginning of every episode and the range of values for each object is described in Table 3. We also randomized the colors and pattern geometries of the background plane with a Perlin noise texture [57].

III. METHOD

In this section, we describe the overall robotic path generation method. Beginning with the target pattern generation, we describe the SPR, MDP design including

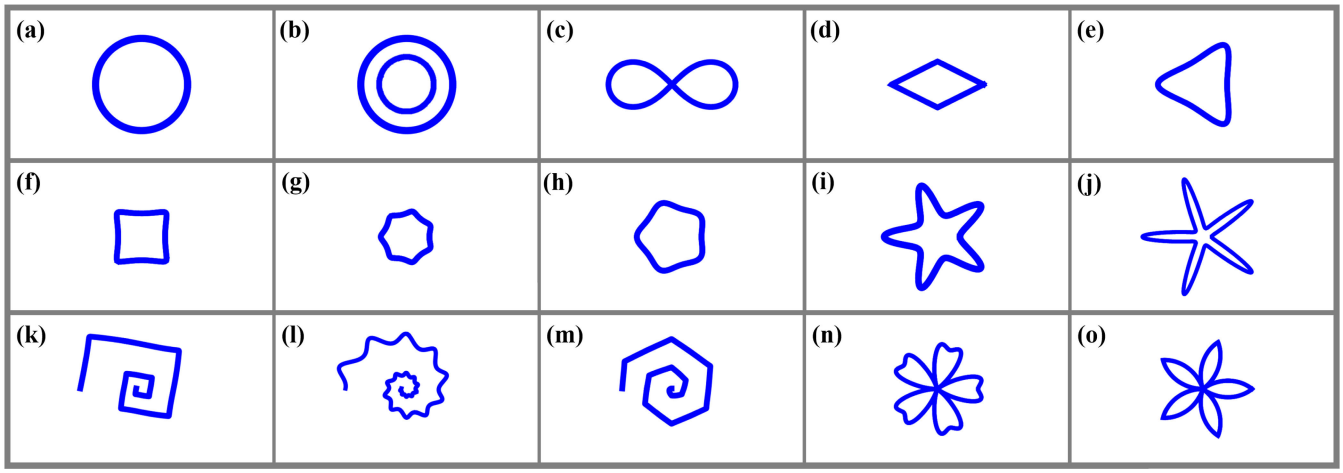


FIGURE 3. Fifteen target patterns generated by custom equations and the superformula. The (a) *Circle*, (b) *Double Circle* and (c) *Lemniscate of Bernoulli* were created using their pre-defined equations. The rest were created by the superformula with parameters ($\rho(\phi)$; $m, n_1, n_2 = n_3$). The scale value $a = b$ is set appropriately for each pattern. (d) *Diamond* (1; 4, 1, 1) (horizontally stretched); (e) *Nuphar Luteum* (1; 3, 4.5, 10); (f) *Scrophularia nodosa* (1; 4, 12, 15); (g) *Equisetum Stem* (1; 7, 10, 6); (h) *Raspberry* (1; 5, 4, 4); (i) *Starfish 1* (1; 5, 2, 7); (j) *Starfish 2* (1; 5, 2, 13); (k) *Spiral 1* ($e^{0.2\phi}$; 4, 100, 100, 100); (l) *Spiral 2* ($e^{0.2\phi}$; 10, 5, 5, 5); (m) *Spiral of Archimedes* (ϕ; 6, 250, 100, 100); (n) *Modified Rose 1* ($\cos(m\phi)$, 2.5, 1/1.3, 2.7); and (o) *Modified Rose 2* ($\cos(m\phi)$, 2.5, 5, 5).

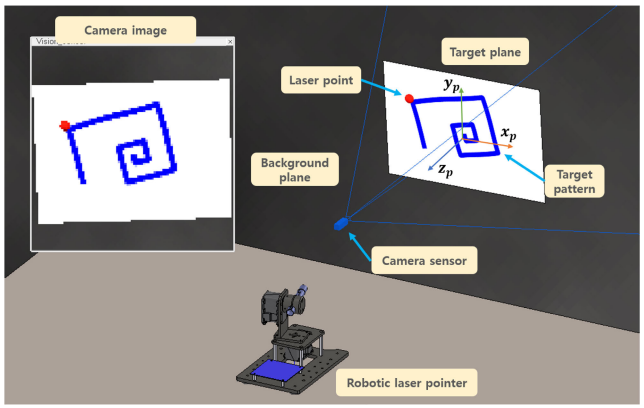


FIGURE 4. RLP simulation environment consisting of an RLP, camera, and target plane. The position and orientation of the target plane, camera sensor, target pattern texture, background plane color pattern, and initial laser position are randomly reset in every episode. The resolution of the camera image is 96 × 96 pixels.

network architecture, distributed learning framework, and SDC learning.

A. TARGET PATTERN GENERATION

To learn a robust path generation skill, various training datasets are required. We prepared 15 classes of patterns (Fig. 3) using custom equations and the following superformula [58]:

$$f(\phi) = \rho(\phi) \frac{1}{\sqrt[n_1]{\left(\left|\frac{1}{a}\cos\left(\frac{m}{4}\phi\right)\right|\right)^{n_2} + \left(\left|\frac{1}{b}\sin\left(\frac{m}{4}\phi\right)\right|\right)^{n_3}}}, \tag{11}$$

where $m, n_1, n_2 = n_3$ are the parameters used to form a particular pattern and $f(\phi)$ is represented using polar coordinates. Except for the patterns in Figs. 3 (a)–(c), all patterns are defined using [58], where the parameter details are given in the figure's caption.

TABLE 3. Randomization ranges for each target object. RLP: robotic laser pointer. Subscripts *w* and *p* represent the world coordinates and local coordinates of the target plane respectively.

Target objects	*Parameter*	*Range*
Target plane	$\{\boldsymbol{x}_{\boldsymbol{w}}^{\text{target}}\}, \{\boldsymbol{\omega}_{\boldsymbol{w}}^{\text{target}}\}$	$\pm 0.05\ m, \pm 10^\circ$
Camera sensor	$\{\boldsymbol{x}_{\boldsymbol{w}}^{\text{cam}}\}, \{\boldsymbol{\omega}_{\boldsymbol{w}}^{\text{cam}}\}$	$\pm 0.05\ m, \pm 10^\circ$
Pattern texture	$\{x_p^{\text{pattern}}, y_p^{\text{pattern}}\}, \{\gamma_p^{\text{pattern}}\}$	$\pm 0.03\ m, \pm 45^\circ$
Initial laser pose	$\{\boldsymbol{x}_{\boldsymbol{w}}^{\text{laser}}\}$	$\pm 0.04\ m$
Inital RLP pose	$\{x_w^{\text{RLP}}, y_w^{\text{RLP}}\}, \{\gamma_w^{\text{RLP}}\}$	$\pm 0.05\ m, \pm 10^\circ$

B. SPR

As the name suggests, SPR is a method to reduce the target patterns sequentially so that all pattern pixels are completely removed and the corresponding full robotic paths can be generated from the given images. SPR consists of pattern reduction and repaint (PRR) and a reward function-based rollout process.

1) PRR

In the initial state, the agent observes an initial target pattern image I_t at time t. Then, it performs an action and observes the next image frame I_{t+1}. At this stage, there are two important steps in PRR that the agent should perform, which we call pattern reduction and repaint, respectively (Algorithm 1 and Fig. 6). In the pattern reduction, the target pattern region that overlaps the region of the laser point is removed from image I_t. To determine the location of the laser point on the target plane, we first create a binary mask image B_t^{plane} of the target plane using image processing techniques [59]

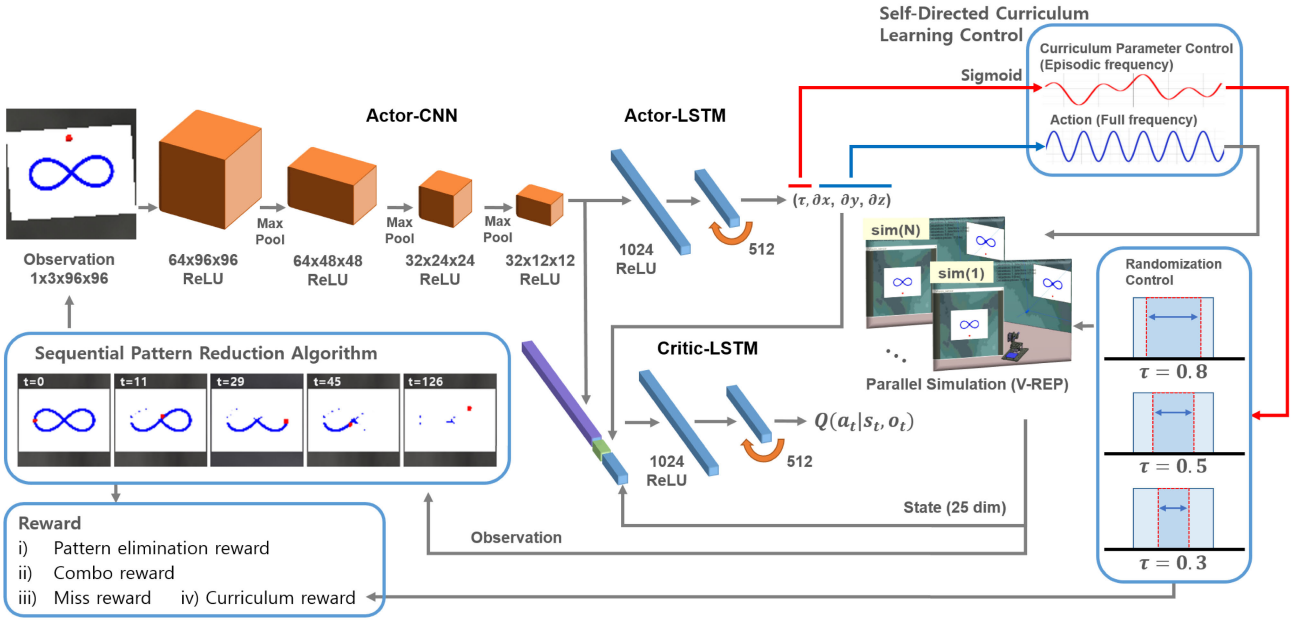


FIGURE 5. Overall system architecture including the actor–critic based SPR policy network. It consists of an Actor–CNN (convolutional neural network), Actor–LSTM (long short-term memory), and Critic–LSTM, for encoding the image, performing an action, and evaluating the actor network, respectively. The agent observes the pattern images from the simulator and sequentially reduces the pattern using the proposed algorithm. Based on the processed observation, the agent performs the next action. In SDC, the frequency at which each element of action $\hat{a}_t = \{\partial x_t^d, \tau_t\}$ is utilized differs. The positional action x_t^d is applied to the environment at full frequency whereas the curriculum control action τ_t is used at an episodic frequency to determine the degree of randomization.

Algorithm 1 PRR

Input: Observed image pair I_t and I_{t+1}

Output: Pattern-reduced and repainted image I_{t+1}^{reduce}

1 Convert I_t^{color} to I_t^{gray}
2 Apply Gaussian blurring to I_t^{gray} to obtain I_t^{blur}
3 Apply thresholding to I_t^{blur} to obtain binary image B_t^{i}
4 Apply morphology to B_t^{i} to obtain B_t^{plane}
5 Convert I_t^{color} to I_t^{hsv}
6 Apply colorslicing to I_t^{hsv} to obtain B_t^{laser}
7 Set $B_t^{\text{laser}} = B_t^{\text{laser}} \cap B_t^{\text{plane}}$
8 Do reduction with I_t^{color}, B_t^{laser} to obtain I_t^{-}
9 repeat 5 to 7 with I_{t+1}^{color} to obtain B_{t+1}^{laser}
10 Do repaint with I_t^{-}, B_{t+1}^{laser} to obtain I_{t+1}^{reduce}
11 **return** I_{t+1}^{reduce}

such as Gaussian blurring, thresholding, and morphological operations (lines 1 to 4). The region of the laser pointer is represented by a binary mask image B_t^{laser} and detected by color slicing [59], where the aim of this technique is to obtain a binary image of the target color distribution based on color thresholding.

The pattern-reduced image I_t^{-} is then obtained by making the pixel colors of some region that overlaps with B_t^{laser} in I_t^{color} equal to the background color (lines 5 to 8). The next step is repainting, which paints the laser point region of

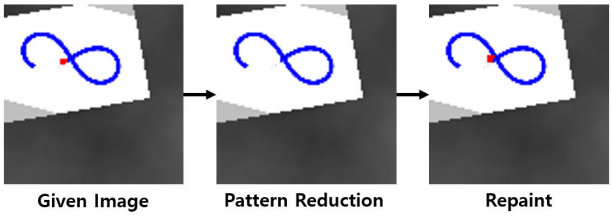


FIGURE 6. Pattern reduction and repaint process. For a given observation image, the agent reduces the pattern region that overlaps the pixel region of the laser point at time t. Then the pixel region of the laser point at time $t+1$ is repainted on the target pattern image.

the next observation I_{t+1} on the pattern-reduced image I_t^{-}. After repeating lines 5 to 7 on I_{t+1}^{color} to acquire the masked laser point image at the next time step, we use it to repaint the laser point on I_t^{-} (lines 9 and 10). The final repainted image is the pattern-reduced and repainted image I_{t+1}^{reduce} at time $t+1$.

2) OVERALL ALGORITHM FOR SPR

From the previous step, we obtain the pattern-reduced and repainted image I_{t+1}^{reduce}. The SPR reward is then calculated based on the I_{t+1}^{reduce}, where the reward consists of pattern reduction, *combo*, and *miss* rewards. Details of the reward calculation are described in Section III-C. In this step, the agent collects rollout experiences using PRR and then calculates the reward. The overall SPR process is described in Algorithm 2.

We incorporated an early termination technique [29] into our learning to avoid the collection of very poorly performing experiences (e.g., when the agent points to regions outside the target plane). Such experiences will hinder the robot from learning the optimal policy and consequently, the policy can overfit to local optima. Early termination was implemented by the following processes: If the agent points to a location inside the region of the target plane, a zero penalty $r^{-}_{t+1} = 0$ is given. In contrast, a negative penalty $r^{-}_{t+1} = -1$ is given and the environment is reset when the agent points to a location outside the target plane.

Whenever the agent succeeds in reducing the target patterns, the *combo* and *miss* count variables are increased and reset, respectively. If the agent fails, the reverse operation is conducted (lines 14 to 17 in Algorithm 2). If the *miss* count is over 40 or the episode ends, early termination is implemented so that the environment and all the variables are reset to the initial state (lines 19 to 21).

3) SYNCHRONOUS DISTRIBUTED LEARNING FRAMEWORK

Recent RL studies train their policy network using a distributed learning environment [60]–[62] to reduce the sample correlations and learning time. Our learning framework is based on a distributed system that is similar to the A3C algorithm [62]. However, ours is a synchronous distributed system, in which the learning framework simultaneously collects the experience data from N distributed processes and updates the policy using the merged data, whereas the A3C operates asynchronously. The authors of A3C claim that the asynchronous operation improves performance, but Wu *et al.* [63] disagreed and presented experimental results supporting this counter-argument. Considering their claim and the simplicity of synchronous implementation, we built a synchronous distributed learning framework (see Fig. 7) that can be regarded as a synchronously distributed version of A2C [64] using a message passing interface (MPI) and V-REP simulator [65]. During training, our framework creates multiple processes, each of which runs a V-REP simulator. The training data and network parameters of the agent in each process are shared via the MPI.

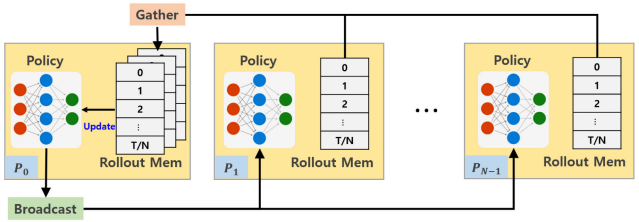


FIGURE 7. MPI-based distributed learning framework. Experience data of the agent are collected from distributed simulators and merged to update the root policy. The updated hyperparameter of the root (P0) policy is transferred to the other child policies.

In the main learning process, N distributed processes including the policy network and other variables are first created and initialized using the MPI. Given a total number of learning frames T and rollout frames per iteration M,

Algorithm 2 SPR With the Distributed Learning Framework

1 Create N distributed processes
2 Initialization: Parameters of policies $\{\pi_0, Q_0, \cdots, \pi_{N-1}, Q_{N-1}, \}$, rollout frames per iteration M, total number of learning frames T frames per episode S, rollout memory $\mathcal{D}_{0,\cdots,N-1}$
3 **for** $j \leftarrow 0$ ***to*** T/M **do**
4 $o_0, s_0 \leftarrow$ reset environment
5 Set combo $= 0$, miss $= 0$
6 **for** $t \leftarrow 0$ ***to*** M/N **do**
7 Select $a_t = \pi(o_t),\ q_t = Q(o_t, s_t)$
8 Execute action a_t in simulator and obtains observation o_{t+1}, state s_{t+1}, penalty reward r^{-}_{t+1}, done d_{t+1}
9 Convert tensors o_t, o_{t+1} to images I_t, I_{t+1}
10 PRR with I_t, I_{t+1} to obtain I^{reduce}_{t+1}
11 Convert image I^{reduce}_{t+1} to tensor o_{t+1}
12 Calculate reward using I_t, I_{t+1}, combo, miss to obtain r_{t+1}, reduction flag f_{reduce}
13 Set final reward $r^{\text{final}}_{t+1} = r^{-}_{t+1} + r_{t+1}$
14 **if** $f_{reduce} == true$ **then**
15 Increase combo by 1 and set miss $= 0$
16 **else**
17 Increase miss by 1 and set combo $= 0$
18
19 **if** miss > 40 *or* $d_{t+1} == true$ **then**
20 Reset environment to obtain o_{t+1}, s_{t+1}
21 Set $d_{t+1} = true$; combo $=$ miss $= 0$
22 Store transition $(o_{t+1}, s_{t+1}, a_t, q_t, r^{\text{final}}_{t+1})$ in $\mathcal{D}_n$
23 $\mathcal{D}_0 \leftarrow$ gather$(\mathcal{D}_{0,\cdots,N-1})$
24 $\pi_0, Q_0 \leftarrow$ update$(\mathcal{D}_0)$
25 $\pi^{\text{temp}}_n, Q^{\text{temp}}_n \leftarrow$ broadCast(π_0, Q_0)
26 $\pi_n, Q_n \leftarrow$ softTargetUpdate$(\pi^{\text{temp}}_n, Q^{\text{temp}}_n)$

T/M iterations are performed to update the policy. In each of the M rollout processes, the experience data is simultaneously acquired from the N distributed processes for rapid data collection; thus, the total number of rollout frames per iteration is M/N. From the initial observation and state, the agent performs an action in the environment and obtains the resulting observation, state, reward, and episode-end-flag d_{t+1} at the next time step. When the episode reaches the end or early termination is activated, the episode finish flag d_{t+1} is set to true and at this moment, the current target pattern is randomly changed to another one. The current and next observations are then converted to images and input to the PRR process (Algorithm 1) to obtain I^{reduce}_{t+1}. We defined the reward function for the SPR method so that it accepts images

corresponding to the current and next observations as well as additional variables such as *combo* and *miss* for learning sequential behavior generation.

After finishing M/N rollout steps, the rollout data collected by the N processes are merged to the root node P_0 and used to update the root policy. Then, the weight parameters of the root policy are broadcasted to all child nodes $(P_1, P_2, \cdots P_{N-1})$ and used for updating their policies using the following soft target update [66]:

$$\omega_{\text{new}} = \beta\ \omega_{\text{old}} + (1-\beta)\ \omega_{\text{new}}, \tag{12}$$

$$\zeta_{\text{new}} = \beta\ \zeta_{\text{old}} + (1-\beta)\ \zeta_{\text{new}}, \tag{13}$$

where $\beta(=0.05)$ is a balancing constant between the previous parameters, ω_{old} and ζ_{old}, and new parameters, ω_{new} and ζ_{new}, of the actor and critic network, respectively.

C. MDP DESIGN

1) POLICY NETWORK

As shown in Fig. 5, the policy network consists of Actor-CNN π_ω, Actor-LSTM $\pi_{\omega^\dagger}$, and Critic-LSTM Q_ζ networks, where the $\omega, \omega^\dagger$, and ζ represent the hyper-parameters of each policy network respectively. The Actor-CNN takes an image frame $o_t = \mathcal{I}_{(3\times96\times96)}$ of the target pattern at time t and encodes it to a feature vector $v_t^f = \mathcal{I}_{(1\times4608)}$ by flattening the last convolved image tensor $\mathcal{I}_{(32\times12\times12)}$. This feature vector is fed to the Actor-LSTM network and an action is sampled. The input of the Critic-LSTM is a concatenated vector $v_t^c = \mathcal{I}_{(1\times4637)}$ consisting of a feature vector, action $a_t = \mathcal{I}_{(1\times4)}$, and the agent's state $s_t = \mathcal{I}_{(1\times25)}$ obtained from the simulator. Then, it outputs an estimated Q-value $Q^\zeta(a_t|s_t, o_t)$. All networks use the ReLU activation function.

2) STATE AND ACTION

The state of the agent consists of the information of the agent and other related objects. Specifically, the 25-dimensional state $s_t = \mathcal{I}_{(1\times25)}$ includes the angular values and angular velocities of the pan and tilt joints, the position and orientation of the RLP device with respect to world coordinates, the position and orientation of the camera and the target plane, and the position of the laser point. This state information is only used in the training phase.

Given an observation, the Actor-LSTM samples an action $a_t = \{\partial \boldsymbol{x}_t^d, \tau_t\}$, where the first element represents the desired positional differences $\partial \boldsymbol{x}_t^d = \{\partial x_t, \partial y_t, \partial z_t\}$ of the laser pointer in world coordinates. To calculate the desired position, the sampled action, which is the output from the actor based on a Gaussian distribution, is scaled and added to the current position of the laser pointer using the following equation:

$$\mu_t, \sigma_t = \pi_{\omega^\dagger}(\pi_\omega(o_t)), \tag{14}$$

$$a_t = \mu_t + \sigma_t * \epsilon, \tag{15}$$

$$\bar{\boldsymbol{x}}_t^d = \boldsymbol{x}_{t-1} + \eta\ \partial \boldsymbol{x}_t^d, \tag{16}$$

where ϵ is a random noise constant to ensure exploration. During the test phase, the agent performs a deterministic action by considering μ only. The scale factor $\eta = 2.5 \times 10^{-3}$ was introduced to stabilize the learning. We found that without the proper scaling factor, the learning curve of training becomes very unstable and performance may degrade because of the distribution of actions specified by the initial weight parameters of the policy network. The desired relative position of (16) is input to the inverse kinematics of (1) to obtain the desired angles, which are then input to the PID controller of the RLP in the simulator. The last element of the action is the randomization control parameter τ_t, which is used in the proposed SDC learning method. We describe the details of τ_t in Section III-D.

3) REWARD

For RL-based path generation, the reward function should be able to determine the SPR skill of the agent. We defined the reward function in terms of three sub-rewards. The first is pattern reduction reward r_p, which is proportional to the area of the pattern reduction obtained by the laser pointer. Based on the binary images B, the ratio of the area of the reduced pattern to the area of the laser point is defined as

$$e = \frac{\sum B_{t+1}^{\text{laser}} - \sum \left(B_t^{\text{pattern}} \cap B_{t+1}^{\text{laser}}\right) + \epsilon}{\sum B_{t+1}^{\text{laser}} + \epsilon}, \tag{17}$$

$$r_{t+1} = \omega^p r_p + \omega^c r_c + \omega^m r_m, \tag{18}$$

$$r_p = \exp(-3.0 * e), \tag{19}$$

$$r_c = \exp(-5.0 * (\text{combo} + \epsilon)^{-1}), \tag{20}$$

$$r_m = \exp(-5.0 * \text{miss}), \tag{21}$$

$$\omega^p = 0.7, \quad \omega^c = 0.15,\ \omega^m = 0.15,\ \epsilon = 1.0 \times 10^{-6}, \tag{22}$$

where ϵ is used to avoid the division by zero. Moreover, B_{t+1}^{laser} and $B_t^{pattern}$ represent the binary image of the laser point at time $t+1$ and pattern at time t respectively. The second sub-reward is the *combo* reward r_c, which reflects the sequential reduction skill. If the agent consecutively succeeds in reducing a pattern, the *combo* count increases, and this in turn leads to a high *combo* reward, which is expressed as shown in (20). The last sub-reward is the *miss* reward r_m, which gives the agent a higher penalty if it fails to reduce the pattern (represented in (21)). All sub-rewards are normalized by an exponential function. The weighted sum of these three sub-rewards is the next reward r_{t+1} (line 13 in Algorithm 2). The final reward is calculated by adding the penalty term r_{t+1}^- to the reward, as described in line 13 of Algorithm 2.

D. SDC LEARNING

In SDC learning, the following modified reward function is used to learn the curriculum control skill:

$$\hat{r}_p = r_p * (1 - r_c), \tag{23}$$

$$r_c = \exp(-20.0 * \tau), \tag{24}$$

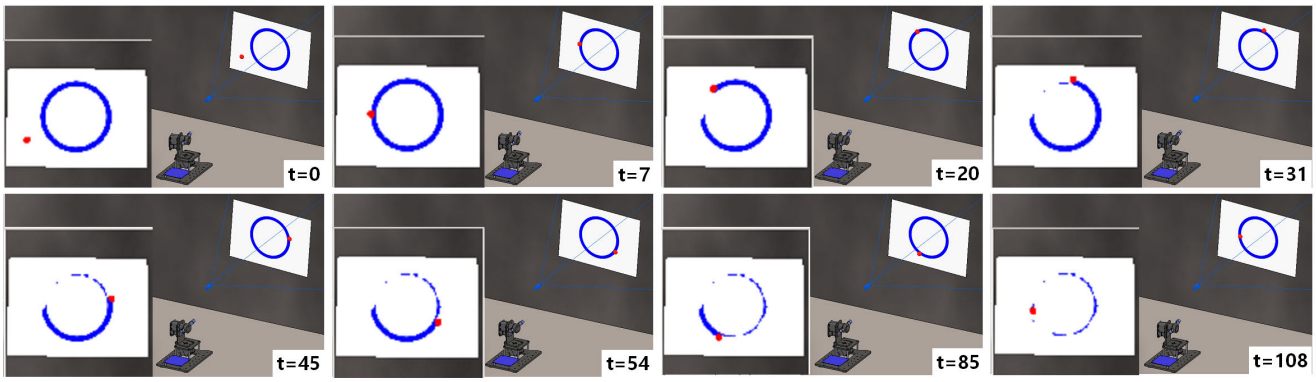


FIGURE 8. Path generation result for a circle pattern. The robot agent sequentially performs the target pattern reduction so that the series of actions form the complete path of the circle pattern.

where τ is the last element of an action $a_t = \{\partial \boldsymbol{x}_t^d, \tau_t\}$ to control the degree of randomization. The curriculum reward r_c is normalized between 0 and 1 using an exponential function, and the agent can obtain a higher reward if τ in (24) is close to 1. This means that if the agent succeeds in reducing the pattern in a more challenging environment, a high reward is given to take into account its level of difficulty. Therefore, the agent should be able to effectively learn the SPR policy in a relatively larger dynamic environment by controlling the curriculum for itself (see Fig. 5).

The value of τ is rescaled to a value between 0 and 1 by the sigmoid function when it is input to the environment. If this value is set to 1, the agent requests a full randomization of Table 3 for the environment. In contrast, a value of 0 indicates an initial fixed environment without randomization. Each element of action a_t is utilized at different frequencies, where the positional action $\partial \boldsymbol{x}_t^d$ is applied to the environment at the full frequency (every frame) whereas the randomization action τ_t is used at an episodic frequency (only at the beginning of every episode) because a randomized environment setup should be maintained for the duration of at least one episode. In short, for efficient learning, the agent learns the curriculum control skill by the curriculum reward r_c, which is a feedback from the environment incorporating the task difficulty control signal τ.

IV. EXPERIMENTS

In this section, we first describe the experimental details. Then, the SPR results in known and unknown environments are presented. We then describe the comparative analysis on learning strategy and quantitative analysis in terms of Hausdorff similarity and pattern reduction ratio.

A. EXPERIMENTAL DETAILS

For the experiments, experience data were simultaneously collected from eight distributed processes in a single machine equipped with an i7-8700K CPU, 64 GB RAM, and Titan Xp and 1080ti GPUs. Because of the memory capacity limitations of the GPUs, the rollout processes were conducted by splitting the tasks among the two GPUs and updating the network parameters using the merged rollout data.

For Algorithm 2, we set the rollout frames $M = 4,096$, total learning frames $T = 1.0 \times 10^7$, and frames per episode $S = 256$. The rest of the learning parameters were set as follows: the learning rate for actor $\eta_a = 5.0 \times 10^{-5}$, the learning rate for critic $\eta_c = 1.0 \times 10^{-3}$, mini batch size $= 256$, the number of PPO epochs $= 5$, discount factor $= 0.99$, and entropy coefficient $= 0.01$. We also used generalized advantage estimation [67] for temporal difference-based policy optimization.

If the training is conducted using a single process, the overall learning time for each policy will take almost 11 days for an LSTM with a single recurrent layer and a total number of rollout frames of 10M. Owing to the eight distributed processes, in contrast, we are able to reduce the overall training time to about 1.5 days for each policy on a single machine.

B. PATH GENERATION IN A KNOWN ENVIRONMENT

To verify the SPR algorithm, we first evaluated the EIE policy in a fixed environment. Evaluations were conducted for 50 episodes in the simulator (see Fig. 8) and the results are shown in Fig. 9. For each pattern, the left image shows the whole path of the laser pointer during one episode and the right image shows the visual patterns reconstructed by the laser pointer. Through the proposed SPR method, we show that the SPR policy is able to perform image-based path generation for various patterns including convex (*Circle*) and concave (*Starfish*) shapes without using additional manual programming, geometrical modeling of the target patterns, or curve fitting algorithms.

The proposed SPR method, however, has several limitations on some patterns that involve multiple intersecting points such as the modified rose (Figs. 9(n) and (o)), or elaborate patterns. In the former case, the agent may lose direction when it revisits the intersecting point because the pattern region is already removed in the previous visit. For this reason, the path generation results of those patterns are less accurate than those of the others. In the latter case,

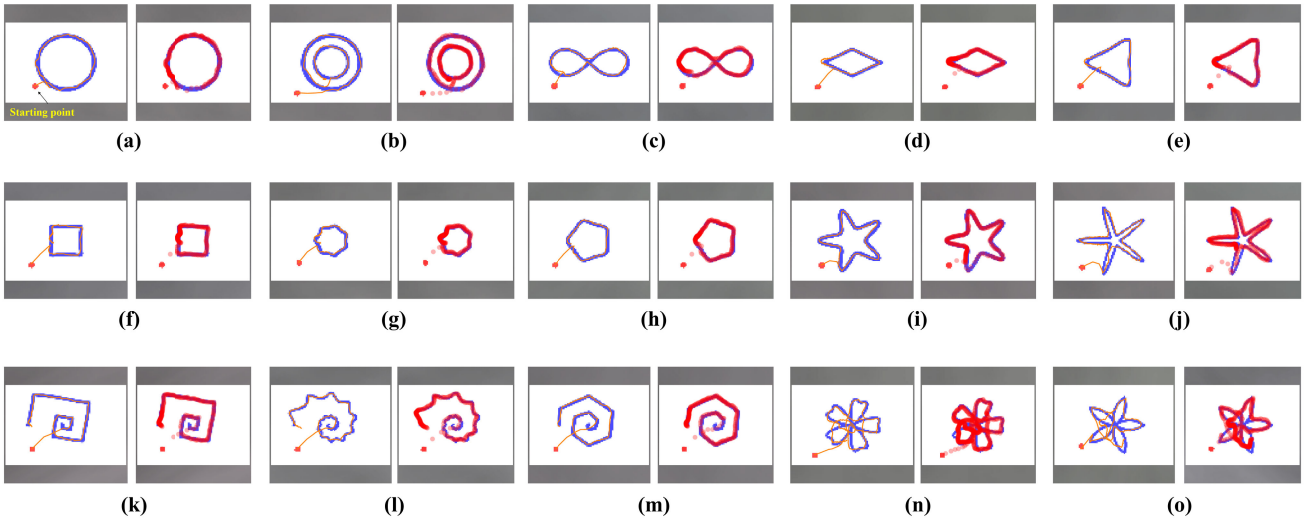


FIGURE 9. SPR results of the EIE policy in a fixed environment. For each pattern ((a)–(o)), the left image is the complete path for each pattern represented by connecting each point whereas the right image is the visualization of the laser pointer traces during one episode. The gray region represents the background.

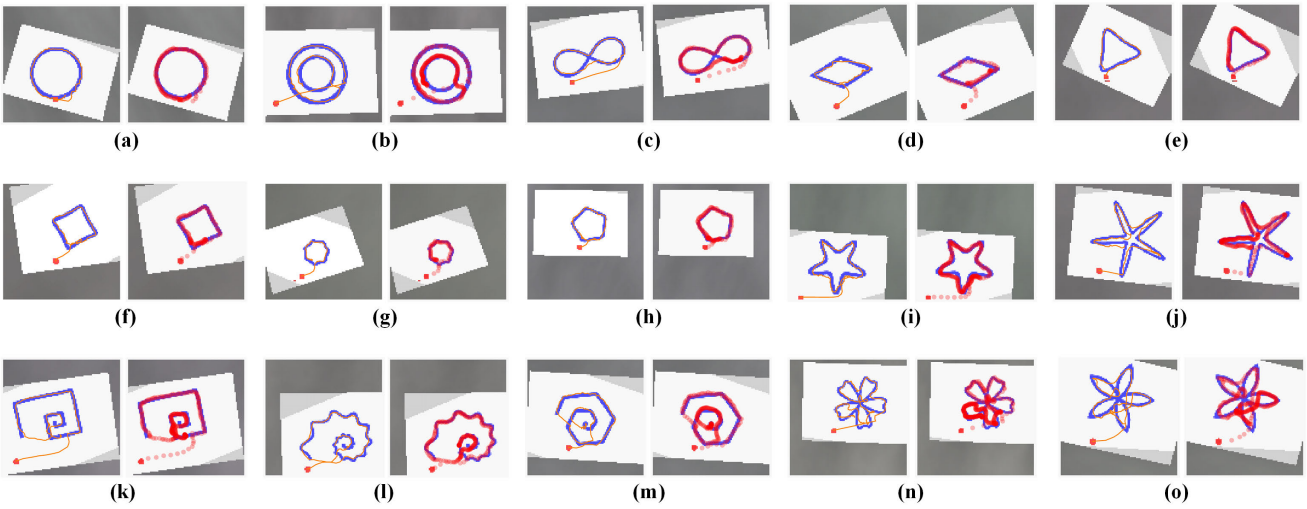


FIGURE 10. SPR results using the policy trained by the SDC learning in a randomized environment. For each pattern ((a)–(o)), the left side image shows the generated paths of the laser pointer for each target pattern placed in arbitrary positions and orientations whereas the right side image is the visualization of the laser pointer traces during one episode.

the scale difference between the pixel region of the pattern and the laser point is the cause for this problem. In *Starfish2* (Fig. 9(j)), only a single path of the bottom protrusion was generated because the relatively large pixel region of the laser point unintentionally reduced the neighboring pattern region so that the agent could not find the source for path generation when it revisited that place. We call this the *over-reduction* problem and in future work, we plan to find a solution for it.

During evaluation, we found that the agent accelerates the laser pointer at initial state until it reaches the pattern region (see Fig. 12). This is because the policy is trained to maximize the expected rewards by reducing the pattern as quickly as possible during an episode. Once the laser point contacts the pattern, the agent sequentially reduces it at a relatively slower speed due to the observation-action sampling time frequency (20 Hz).

C. PATH GENERATION IN AN UNKNOWN ENVIRONMENT

We evaluated the SPR policy in an uncalibrated environment, where the positions and orientations of the camera, target plane, RLP, and initial laser position were initialized to random values in each episode within the ranges listed in Table 3. Figure 10 shows the evaluation results of the SPR policy, which was trained by the SDC learning. In most cases, the agent was able to generate the paths for target patterns even though the robot, camera, and target objects

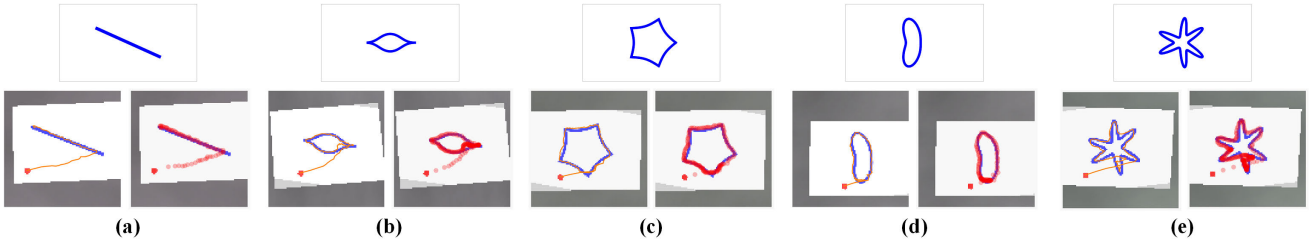


FIGURE 11. SPR results for unlearned patterns: (a) *Line*. The remaining patterns were created by the superformula with parameters ($\rho(\phi)$; m, n_1, n_2, n_3). (b) *Eye* (1; 2, 0.5, 0.5, 0.5); (c) *ConcaveStar* (1; 5, 1, 1, 1); (d) *Bean* (1; 2, 1, 4, 8); (e) *Spark* (1; 6, 1, 4, 8). The first row shows the unlearned target pattern images. In the second row, the left side image represents the paths of the laser pointer whereas the right side image is the visualization of the laser pointer traces during one episode.

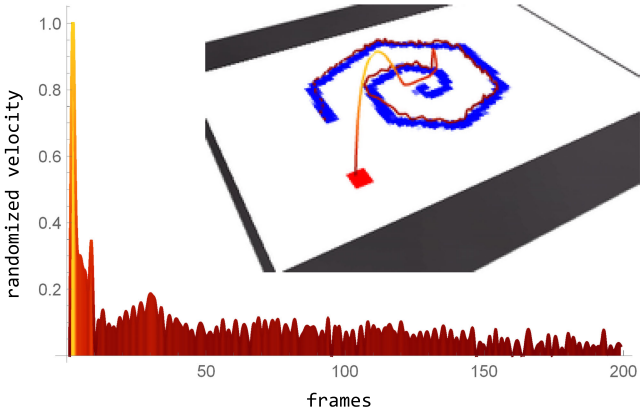


FIGURE 12. (Bottom left) Normalized velocity of the pixel laser pointer with respect to the *Spiral of Archimedes*. (Top right) 3D plot of the normalized velocity during one episode.

were uncalibrated. In terms of the pattern reduction ratio (described in Section IV-E), the average performance of the SDC greatly exceeds other methods (see Table 8 in the Appendix).

To evaluate the generality of the path generation skill, we further tested the SDC policy on unlearned patterns. Figure 11 shows the evaluation results for unlearned patterns such as *Line*, *Eye*, *ConcaveStar*, *Bean*, and *Spark*. Although the generated paths are not perfect, the overall performance seems better than other methods. Moreover, in Fig. 11(a), the agent attempted to begin the pattern reduction from the end of the line instead of the middle or somewhere close to the starting position. This indicates that the agent learns how to obtain higher rewards for a given pattern even if that pattern has never been observed before. From this experiment, we confirmed that the proposed SPR, reward design and SDC learning method are an effective way to learn a robust path generation skill for arbitrary target patterns in an uncalibrated environment.

D. COMPARATIVE ANALYSIS ON LEARNING STRATEGY

We compared the performances of the four models (EIE, ERE, LIC, and SDC) during the learning phase in a fully randomized environment. As shown at the top of Fig. 13, the LIC policy converges quickly in the early phase of training; however, the performance degenerates as learning progresses because of the discrepancy between the learning speed of the policy and the difficulty of the given task. As expected, ERE converges slowly whereas the proposed SDC outperforms others as learning progresses.

We analyzed the tendency of the randomization control τ to determine the relationship between the randomization constant and learning curve. Figure 14 shows the change in the randomization constant for each learning method for every 100 training epochs, where each epoch corresponds to 4,096 rollout frames. Because of its constant full randomization, the learning curve of ERE is the slowest to converge. The learning curve of LIC shows better performance in the beginning and poor performance at the end because LIC merely increases the randomness without consideration for the learning capability of the policy network. In the SDC learning, in contrast, the agent appropriately controls the randomization constant for itself by taking into account its learning capability. Thus, we expected SDC to perform the best.

We evaluated the four learning methods in the test phase to demonstrate the performance of the SDC in terms of reward. Evaluations were conducted using only pattern reduction reward r_p during 50 episodes, where each episode consists of 256 frames. The results shown at the bottom of Fig. 13 indicate a tendency that is identical to the one above. The proposed SDC outperforms the other methods, and the EIE policy performs the worst because it has never observed varied scenes during training. From these evaluation results, we can conclude that the proposed SDC learning approach could be effective in a randomized learning environment.

E. QUANTITATIVE ANALYSIS

1) HAUSDORFF SIMILARITY

For a more objective evaluation of the proposed method, we quantitatively analyzed the four policies (EIE, ERE, LIC, and SDC) using the *Hausdorff similarity measurement* [68]. We first extracted the point set of each target pattern and measured the Hausdorff distance between the extracted point set and the points of the generated path using *Euclidean* distance. Tables 6 and 7 respectively in the Appendix show the

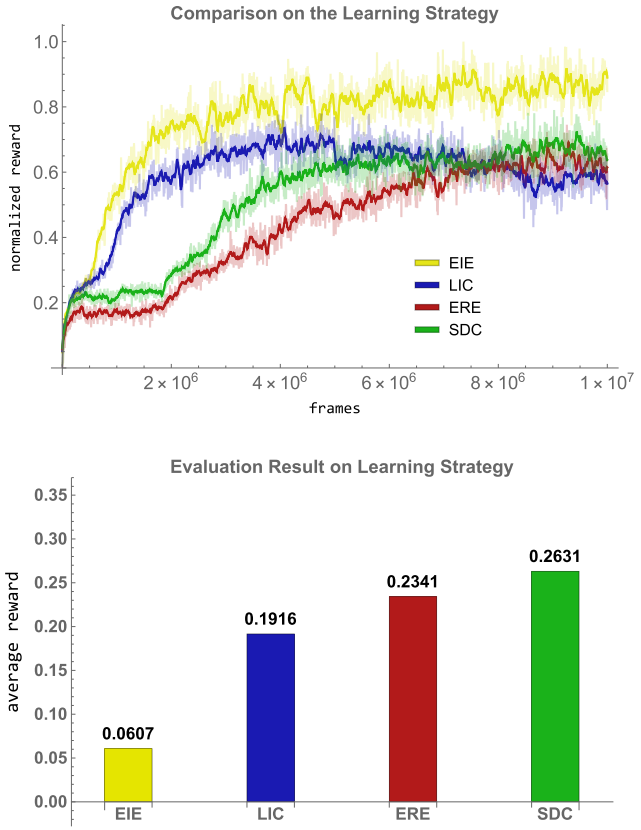


FIGURE 13. (Top) Comparison on the learning strategy including EIE, ERE, LIC, and SDC during training. The Y-axis represents the normalized reward, and each policy was trained by 1×10^7 rollout frames. (Bottom) Evaluation results of the learning strategy over 50 episodes. Each episode consists of 256 frames and the values represent the pattern reduction reward per frame. Evaluations were conducted in a fully randomized environment.

Hausdorff distances between the point sets of the learned and unlearned target patterns and the paths generated using the four learning methods. In each column of Table 6, *baseline* is the evaluation result using the EIE policy in the fixed environment and the distances are measured from the result of Fig. 9 for reference. We marked the minimum distance values in bold in all columns (excluding the *baseline*). The number of minimum distances for each learning method is shown in Table 4, which shows that the proposed SDC outperforms other methods on both the learned and unlearned patterns.

2) PATTERN REDUCTION RATIO

In addition to the Hausdorff similarity, we present a quantitative analysis of the four learning methods based on the ratio between the original and reduced pattern regions, calculated as follows:

$$\lambda = \frac{\sum I_i^{\text{reduce}}}{\sum I_j^{\text{pattern}}}, \tag{25}$$

where I_i^{reduce} and I_j^{pattern} represent each pixel of the reduced pattern and original pattern respectively. The pattern reduction ratio λ is calculated by the sum of their pixels.

TABLE 4. Number of maximum similarities measured by Hausdorff distance (see Tables 6 and 7, and Figs. 17–20 in the Appendix).

Learning method	*Learned patterns*	*Unlearned patterns*	*Total*
EIE	2	1	3
ERE	4	1	5
LIC	4	1	5
SDC	**5**	**3**	**8**

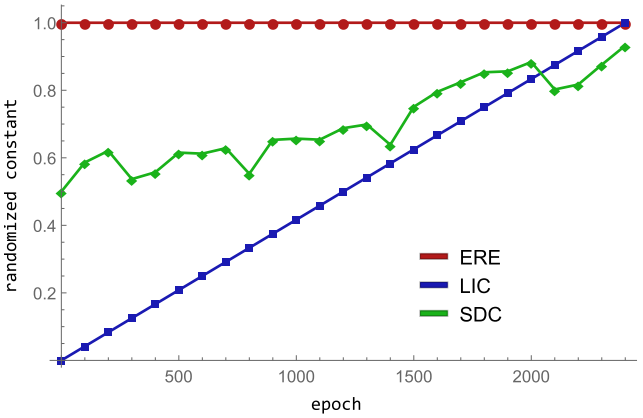


FIGURE 14. Progress of the randomization constant during a single rollout iteration of 4,096 frames with respect to every 100 training epochs.

TABLE 5. Number of minimum ratio values of the reduced pattern with respect to original pattern area (see Tables 8 and 9, and Figs. 17–20 in the Appendix).

Learning method	*Learned patterns*	*Unlearned patterns*	*Total*
EIE	0	0	0
ERE	2	0	2
LIC	1	0	1
SDC	**12**	**5**	**17**

Table 5 presents the number of minimum values of the pattern reduction ratio for each learning method, where the detailed data are shown in Tables 8 and 9 of the Appendix. The statistical data demonstrate that for the learned patterns, the SDC obtains average pattern reduction ratios that are 81.6%, 13.7%, and 20.6% less than those of EIE, ERE, and LIC, respectively. Similarly, it obtains ratios that are 81.5%, 37.8%, and 29.2% less for the unlearned patterns. The corresponding images for the reduced patterns are shown in Figs. 17–20 of Appendix.

V. ABLATION STUDIES

In this section, we present the ablation studies for components of the network architecture and reward function to verify their effectiveness in learning.

A. NETWORK ARCHITECTURE

Although the agent performs a frame-by-frame action, the SPR task should be considered in terms of a series

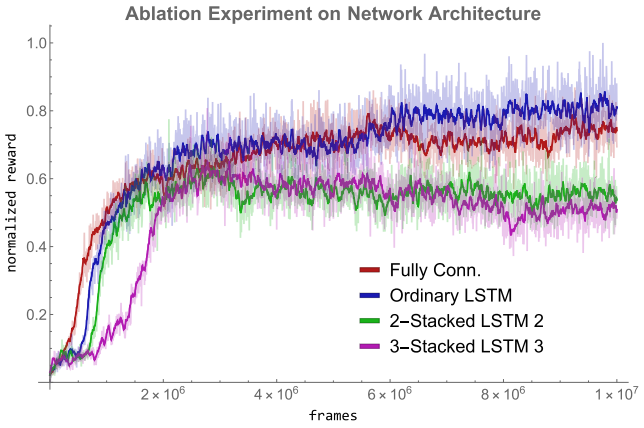


FIGURE 15. Training result with respect to the network architecture. The x-axis and y-axis represent the number of training frames and normalized reward, respectively. The network architectures are a fully connected network, ordinary LSTM, and two and three stacked LSTMs. The ordinary LSTM with a single recurrent layer performs the best.

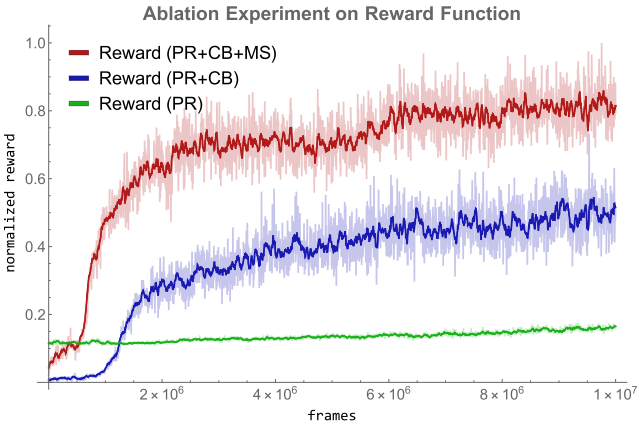


FIGURE 16. Training result represented by the normalized reward. The x-axis and y-axis represent the number of training frames and normalized reward, respectively. Reward (PR+CB+MS), which is a weighted sum of the three sub-rewards (pattern reduction, *combo*, and *miss* rewards), performs the best. Reward (PR+CB) is a weighted sum of the pattern reduction and *combo* rewards. Reward (PR) only considers the pattern reduction reward and performs the worst.

of actions. From this point of view, we assume two things: the first is that the LSTM network performs better than a fully connected network because of its recurrence property. However, it is not clear how many recurrent layers the LSTM should have for the best performance. This leads to the second assumption: increasing the number of recurrent layers as a stacked LSTM will improve the SPR performance because the multiple recurrent layers allow the agent to learn more abstract feature representations.

To verify these assumptions, we compared the training results obtained with fully connected, LSTM, and stacked LSTM layer architectures in a randomized environment. We verified the first assumption: the normal LSTM performs better than a fully connected network, as shown in Fig. 15. However, in the case of the stacked LSTM, the result disproves our second assumption. The result instead shows that as the number of recurrent layers increases, the performance decreases and is even worse than that of the fully connected network. It can be interpreted that using more than one recurrent layer increases the overfitting problem and thus is too much for our task. However, we still believe that introducing a multi-layered LSTM may be advantageous for longer and more complex patterns. To sum up, it is best to use a single-layered LSTM network in our SPR task.

B. REWARD FUNCTION

In addition to the network architecture, we performed an ablation experiment on the reward function to determine the effects of each term. As described in (18), the reward function consists of three sub-rewards: pattern reduction reward r_p (PR), *combo* reward r_c (CB), and *miss* reward r_m (MS). The experiments were conducted by manipulating the set of these sub-reward weights $\{\omega^p, \omega^c, \omega^m\}$ while keeping their range between 0 and 1, where (PR+CB+MS) : $\{\omega^p = 0.7, \omega^c = 0.15, \omega^m = 0.15\}$, (PR+CB) : $\{\omega^p = 0.7, \omega^c = 0.3, \omega^m = 0.0\}$, and (PR) : $\{\omega^p = 1.0, \omega^c = 0.0, \omega^m = 0.0\}$.

The results of the ablation experiments verify that each term of the sub-rewards has a positive effect on learning in the SPR policy. As shown in Fig. 16, the weighted sum of the three sub-rewards (PR+CB+MS) performs better than the other combinations. Moreover, (PR), which uses only the pattern reduction reward, performs the worst. From this, we can infer that r_p is an insufficient reward for learning the SPR policy effectively because the corresponding rewards of consecutive pattern reduction and independent pattern reduction are not distinguishable. Moreover, no penalties are imposed on the agent when it misses a pattern; thus, nothing exists to force the agent not to miss. Even though the agent can learn appropriate consecutive actions from the discounted rewards, this is very inefficient. Although (PR+CB) yields a better result than (PR) owing to the *combo* reward, its performance is lower than that of (PR+CB+MS). From this result, we can verify the positive effect of r_m in SPR policy learning.

VI. CONCLUSION AND FUTURE WORK

In this paper, we proposed a deep RL-based SPR method using a simple RLP device to learn the path generation skill for arbitrary target pattern. We also proposed the SDC learning method for effective learning in a randomized environment. Using our method, we were able to learn a robust path generation policy that can generate paths for arbitrary patterns in an uncalibrated environment. In particular, we experimentally verified that SDC learning is an effective way to learn randomized tasks and outperforms other ordinary and curriculum-based methods. We believe that the proposed device and learning-based path generation method could be used for non-physical human-robot interaction applications such as instructional aids for education and

TABLE 6. Quantitative evaluation of the generated paths. Hausdorff distances between point sets for each learned target pattern and the generated paths of Figs. 17–20 in the Appendix. The Hausdorff similarities were measured in terms of the *Euclidean* distance.

	(a)	*(b)*	*(c)*	*(d)*	*(e)*	*(f)*	*(g)*	*(h)*
Baseline	3.6	4.12	24.69	14.0	8.6	3.0	4.0	3.6
EIE	**22.47**	24.08	$\inf^+$	$\inf^+$	35.51	28.3	32.8	24.69
ERE	25.05	10.19	27.2	43.38	34.82	22.8	**32.55**	7.07
LIC	24.08	**8.94**	27.01	42.15	35.22	**21.09**	34.65	6.4
SDC	24.04	9.21	**26.24**	**41.78**	**31.4**	22.8	34.65	**5.38**
	(i)	*(j)*	*(k)*	*(l)*	*(m)*	*(n)*	*(o)*	
Baseline	10.0	9.21	13.6	11.4	8.48	7.07	7.28	
EIE	$\inf^+$	56.08	59.48	**19.1**	25.63	21.0	$\inf^+$	
ERE	54.64	**33.95**	21.09	21.54	**25.17**	20.12	**44.77**	
LIC	**42.04**	34.92	36.23	20.0	28.16	**18.78**	45.48	
SDC	50.99	39.21	**20.88**	19.41	27.2	19.69	45.8	

TABLE 7. Hausdorff distances between point sets for each unlearned target pattern and the generated paths of Figs. 17–20 in the Appendix. The Hausdorff similarities were measured in terms of the *Euclidean* distance.

	(a)	*(b)*	*(c)*	*(d)*	*(e)*
EIE	**44.38**	37.64	47.8	35.44	22.09
ERE	68.0	$\inf^+$	**44.04**	35.84	13.03
LIC	62.96	37.0	47.8	34.2	**11.4**
SDC	64.77	**36.24**	48.16	**30.08**	**11.4**

TABLE 8. Ratio of the reduced pattern area with respect to its original pattern in learned target patterns. Statistical evaluations were performed with respect to each learning method. The reduced pattern images for each learning method are shown in Figs. 17–20 in the Appendix.

	(a)	*(b)*	*(c)*	*(d)*	*(e)*	*(f)*	*(g)*	*(h)*
Baseline	0.0748	0.1154	0.0751	0.0213	0.0294	0.0428	0.0655	0.0561
EIE	0.6313	0.594	0.7325	1.0	0.5728	0.4202	0.28	0.6779
ERE	**0.147**	0.2486	0.1945	0.2114	0.1225	0.1787	0.3163	0.2796
LIC	0.1725	**0.125**	0.3829	0.1778	0.1456	0.1449	0.2363	0.3333
SDC	0.1509	0.1733	**0.0364**	**0.1577**	**0.1092**	**0.0531**	**0.1054**	**0.0564**
	(i)	*(j)*	*(k)*	*(l)*	*(m)*	*(n)*	*(o)*	*Average*
Baseline	0.0763	0.1221	0.0503	0.1142	0.1135	0.0871	0.1873	0.082
EIE	0.7844	0.9082	0.9145	0.3113	0.9399	0.7869	0.9137	0.6978
ERE	0.362	0.2933	0.2239	0.2146	0.103	**0.1723**	0.292	0.2239
LIC	0.6889	0.339	0.3846	0.3089	0.1802	0.2455	0.2168	0.2721
SDC	**0.2551**	**0.1504**	**0.1162**	**0.1108**	**0.0772**	0.2048	**0.1648**	**0.1281**

TABLE 9. Ratio of the reduced pattern area with respect to its original pattern in unlearned target patterns. Statistical evaluations were performed with respect to each learning method. The reduced pattern images for each learning method are shown in Figs. 17–20 in the Appendix.

	(a)	*(b)*	*(c)*	*(d)*	*(e)*	*Average*
EIE	0.7006	0.5787	0.6083	0.6714	0.9771	0.7072
ERE	0.3375	1.0	0.1863	0.2285	0.2379	0.398
LIC	0.7579	0.3148	0.1444	0.2	0.2677	0.3369
SDC	**0.2292**	**0.0936**	**0.057**	**0.0952**	**0.1762**	**0.1302**

assembly guidance. We expect that the SDC can be generally used to improve the performance of other RL problems.

We would like to note some limitations of our approach. The proposed SPR method still requires classic image processing. Moreover, the *over-reduction* problem still remains unsolved for some cases. These hinder the application of the proposed method to more complex patterns such as *Starfish2* with multiple intersection points, which is the immediate direction of our future work. Encouraged by the simulation results, we plan to conduct experiments applying the proposed method to a real world robot and target patterns.

APPENDIX

A. QUANTITATIVE EVALUATION OF GENERATED PATHS

In this section, we provide supplementary experimental results and analysis. Tables 6 and 7 show the quantitative

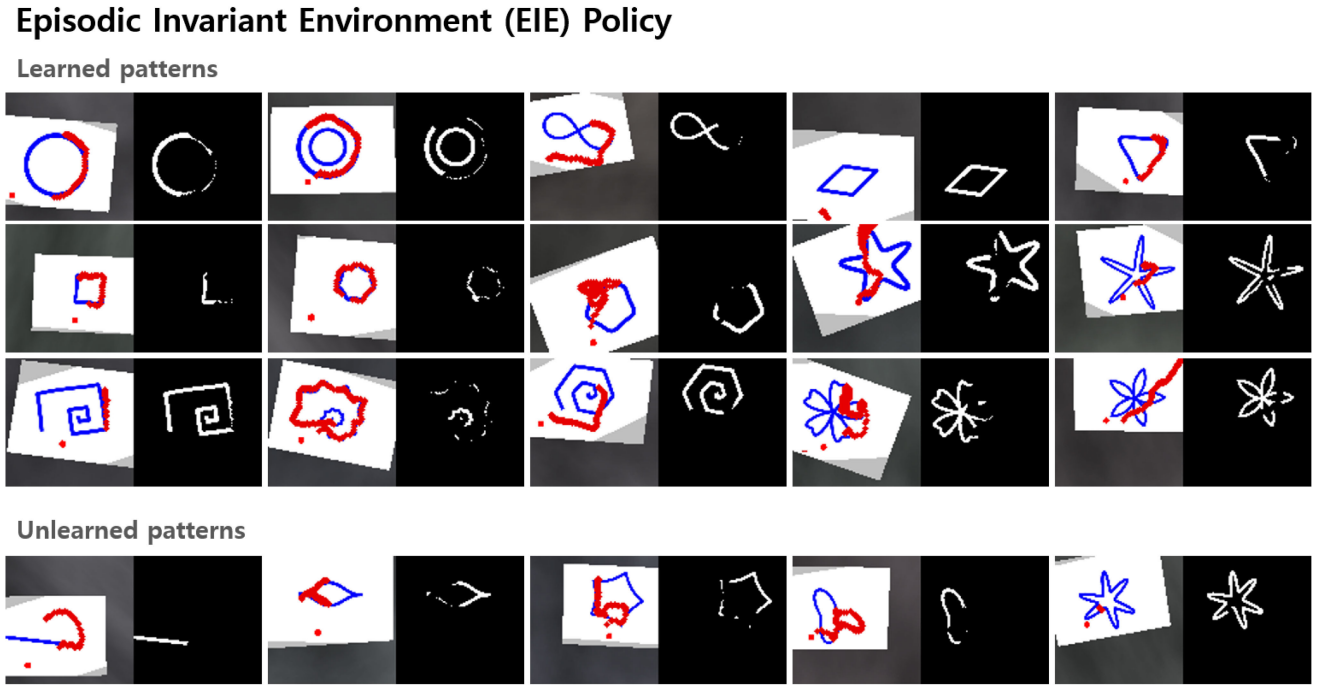


FIGURE 17. Evaluation results on the learned and unlearned target patterns using the EIE policy in a randomized environment. For each pattern, the generated paths are represented by red pixels in the left image. The right image shows the pattern image reduced by the laser pointer path of the left image.

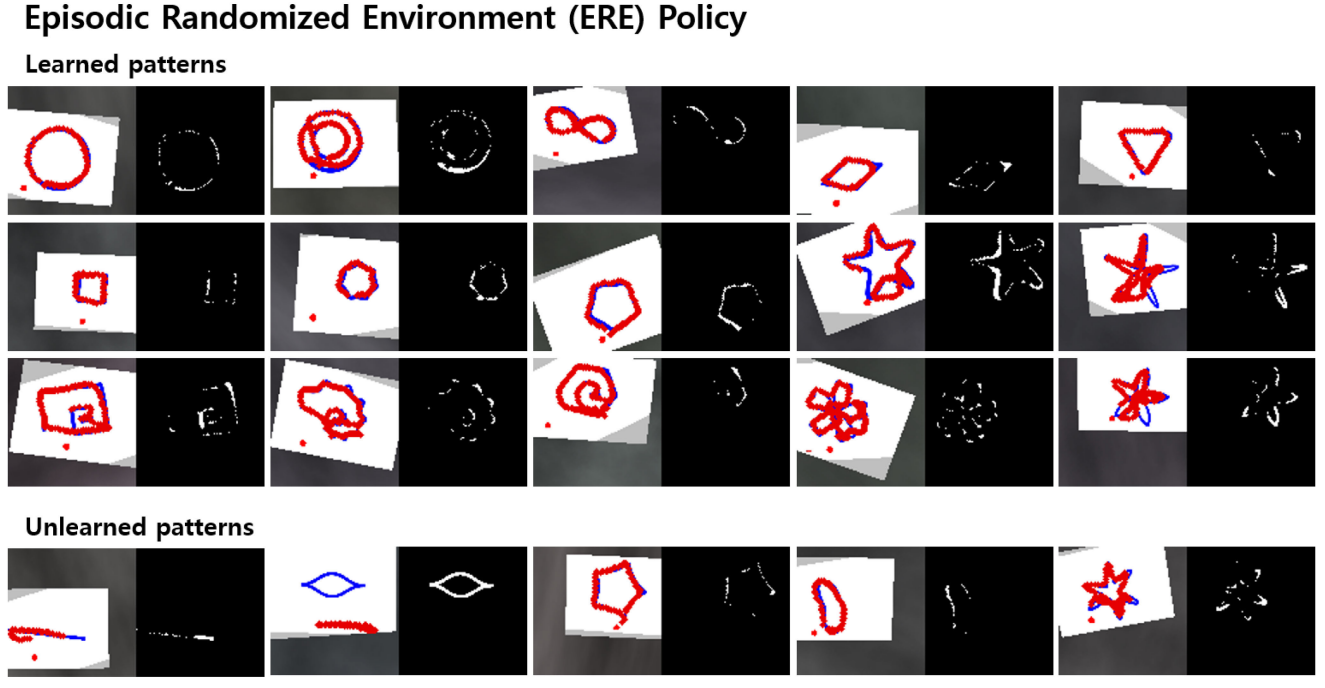


FIGURE 18. Evaluation results on the learned and unlearned target patterns using the ERE policy in a randomized environment. For each pattern, the left image shows the target pattern image and path generated on it. The right image shows the pattern image reduced by the laser pointer path of the left image.

evaluation results on the generated path using the *Euclidean* Hausdorff distance for learned and unlearned patterns, respectively. Tables 8 and 9 show the ratio values of the reduced patterns with respect to the original patterns in each learning method for learned and unlearned patterns, respectively. The data are visualized in Section *B*.

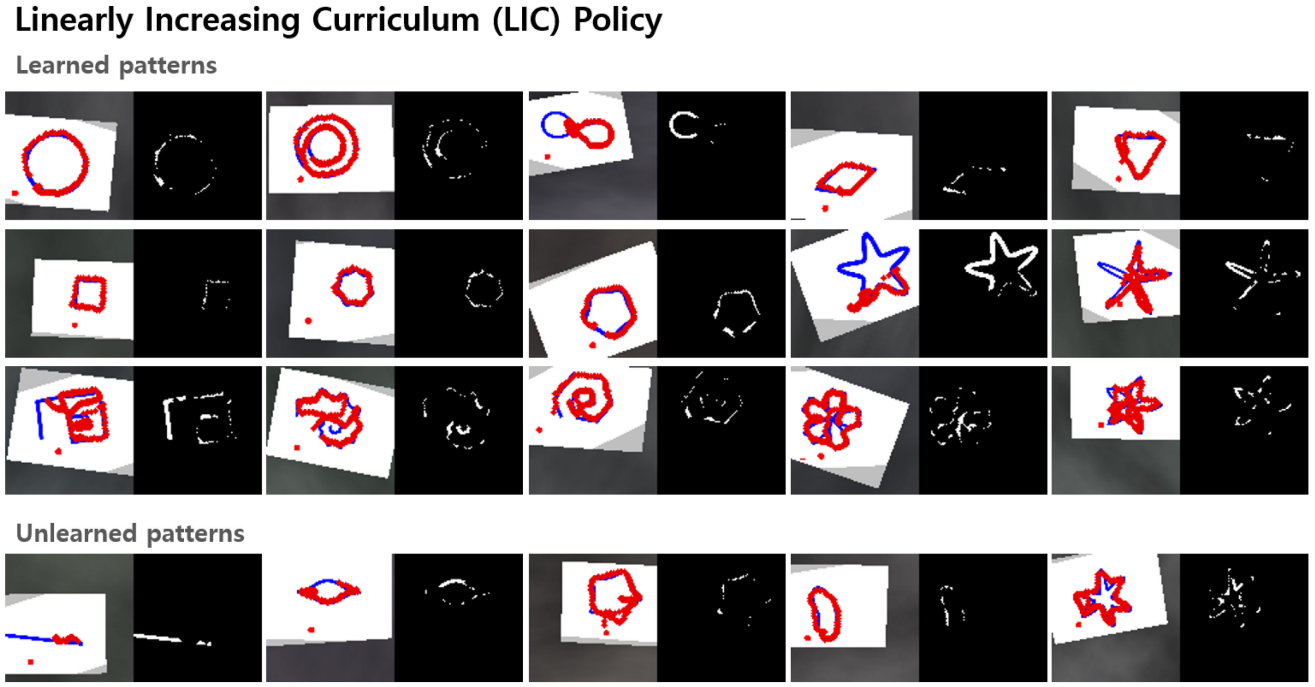


FIGURE 19. Evaluation results on the learned and unlearned target patterns using the LIC policy in a randomized environment. For each pattern, the left image shows the target pattern image and generated path on it. The right image shows the pattern image reduced by the laser pointer path of the left image.

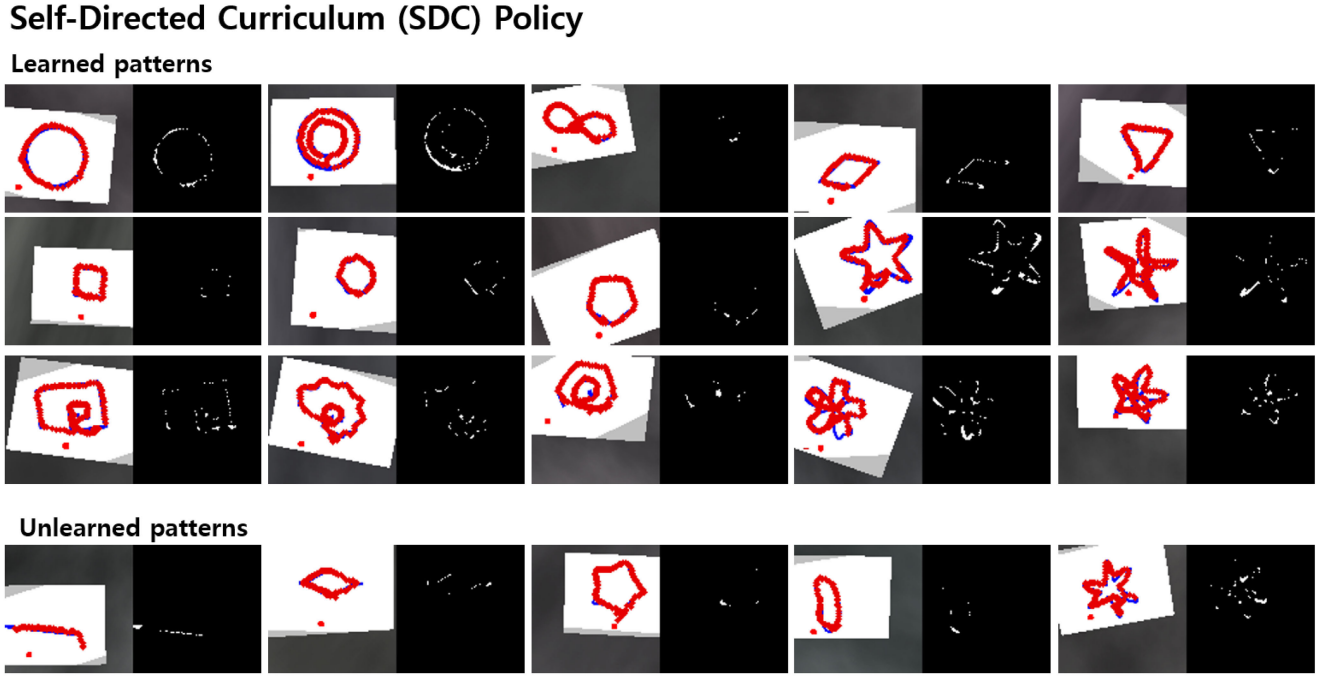


FIGURE 20. Evaluation results on the learned and unlearned target patterns using the SDC policy in a randomized environment. For each pattern, the left image shows the target pattern image and generated path on it. The right image shows the pattern image reduced by the laser pointer path of the left image.

B. VISUALIZATION OF GENERATED PATHS

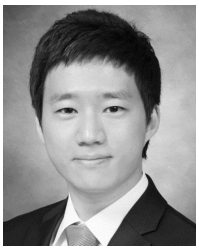
In this section, visualizations of the path generation results are provided.

REFERENCES

[1] M. A. Goodrich and A. C. Schultz, "Human-robot interaction: A survey," *Found. Trends Hum.-Comput. Interact.*, vol. 1, no. 3, pp. 203–275, 2007.

[2] M. Eppe, S. Trott, and J. Feldman, "Exploiting deep semantics and compositionality of natural language for Human-Robot-Interaction," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 731–738.
[3] L. Chen, M. Zhou, W. Su, M. Wu, J. She, and K. Hirota, "Softmax regression based deep sparse autoencoder network for facial emotion recognition in human-robot interaction," *Inf. Sci.*, vol. 428, pp. 49–61, Feb. 2018.
[4] J.-H. Lee, J.-H. Lee, and M. G. Chung, "Modeling and simulation of robotic projector for SAR: A preliminary study," in *Proc. IEEE Int. Conf. Consum. Electron.-Asia (ICCE-Asia)*, Oct. 2016, pp. 1–2.
[5] S. W. Lee, "Robot and method of providing guidance service by the robot," U.S. Patent App. 16 578 036, Jan. 9, 2020.
[6] A. Lee, J.-H. Lee, and J. Kim, "Data-driven kinematic control for robotic spatial augmented reality system with loose kinematic specifications," *ETRI J.*, vol. 38, no. 2, pp. 337–346, Apr. 2016.
[7] R. Kaur, R. S. Sandhu, A. Gera, T. Kaur, and P. Gera, "Intelligent voice Bots for digital banking," in *Smart Systems and IoT: Innovations in Computing*. Singapore: Springer, 2020, pp. 401–408.
[8] H. J. Saddler, A. T. Piercy, G. L. Weinberg, and S. L. Booker, "Intelligent automated assistant," U.S. Patent 10 553 215, Feb. 4, 2020.
[9] S. Ohta, H. Kuzuoka, M. Noda, H. Sasaki, S. Mishima, T. Fujikawa, and T. Yukioka, "Remote support for emergency medicine using a remote-control laser pointer," *J. Telemed. Telecare*, vol. 12, no. 1, pp. 44–48, Jan. 2006.
[10] W.-C. Chang, M.-Y. Cheng, and H.-J. Tsai, "Image feature command generation of contour following tasks for SCARA robots employing Image-Based Visual Servoing—A PH-spline approach," *Robot. Comput.-Integr. Manuf.*, vol. 44, pp. 57–66, Apr. 2017.
[11] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robot. Auto. Syst.*, vol. 61, no. 12, pp. 1258–1276, Dec. 2013.
[12] A. Freeman, "SAR calibration: An overview," *IEEE Trans. Geosci. Remote Sens.*, vol. 30, no. 6, pp. 1107–1121, Nov. 1992.
[13] 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," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 23–30.
[14] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-Real transfer of robotic control with dynamics randomization," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 1–8.
[15] S. Ghogare and S. S. Pande, "Efficient CNC tool path planning using point cloud," in *Proc. Int. Manuf. Sci. Eng. Conf.*, vol. 51388. American Society of Mechanical Engineers, Jun. 2018, Art. no. V004T03A019.
[16] A. Masood, R. Siddiqui, M. Pinto, H. Rehman, and M. A. Khan, "Tool path generation, for complex surface machining, using point cloud data," *Procedia CIRP*, vol. 26, pp. 397–402, 2015.
[17] Z. Teng, H.-Y. Feng, and A. Azeem, "Generating efficient tool paths from point cloud data via machining area segmentation," *Int. J. Adv. Manuf. Technol.*, vol. 30, nos. 3–4, p. 254, 2006.
[18] K. L. Chui, W. K. Chiu, and K. M. Yu, "Direct 5-axis tool-path generation from point cloud input using 3D biarc fitting," *Robot. Comput.-Integr. Manuf.*, vol. 24, no. 2, pp. 270–286, Apr. 2008.
[19] S. C. Park and Y. C. Chung, "Tool-path generation from measured data," *Comput.-Aided Des.*, vol. 35, no. 5, pp. 467–475, Apr. 2003.
[20] F. Chaumette, "Potential problems of stability and convergence in image-based and position-based visual servoing," in *The Confluence of Vision and Control* (Lecture Notes in Control and Information Sciences), vol. 237, D. Kriegman, G. Hager, and A. Morse, Eds. New York, NY, USA: Springer-Verlag, 1998, pp. 66–78.
[21] T. P. Pachidis and J. N. Lygouras, "Vision-based path generation method for a robot-based arc welding system," *J. Intell. Robotic Syst.*, vol. 48, no. 3, pp. 307–331, Feb. 2007.
[22] D. Aristos, T. Pachidis, and J. Lygouras, "Robot path generation by viewing a static scene from a single camera," in *Proc. IEEE Int. Symp. Robot. Autom. (ISRA)*, Sep. 2002, pp. 1–6.
[23] G. Flandin, F. Chaumette, and E. Marchand, "Eye-in-hand/eye-to-hand cooperation for visual servoing," in *Proc. ICRA. Millennium Conf. IEEE Int. Conf. Robot. Automat. Symp.*, vol. 3, Apr. 2000, pp. 2741–2746.
[24] B. Li, H. Zhang, P. Ye, and J. Wang, "Trajectory smoothing method using reinforcement learning for computer numerical control machine tools," *Robot. Comput.-Integr. Manuf.*, vol. 61, Feb. 2020, Art. no. 101847.
[25] W. Jing, C. F. Goh, M. Rajaraman, F. Gao, S. Park, Y. Liu, and K. Shimada, "A computational framework for automatic online path generation of robotic inspection tasks via coverage planning and reinforcement learning," *IEEE Access*, vol. 6, pp. 54854–54864, 2018.
[26] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 2013, *arXiv:1312.5602*. [Online]. Available: http://arxiv.org/abs/1312.5602
[27] G. Lample and D. S. Chaplot, "Playing FPS games with deep reinforcement learning," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 2140–2146.
[28] F.-Y. Wang, J. Jason Zhang, X. Zheng, X. Wang, Y. Yuan, X. Dai, J. Zhang, and L. Yang, "Where does AlphaGo go: From church-turing thesis to AlphaGo thesis and beyond," *IEEE/CAA J. Automatica Sinica*, vol. 3, no. 2, pp. 113–120, Apr. 2016.
[29] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Trans. Graph.*, vol. 37, no. 4, p. 143, 2018.
[30] X. B. Peng and M. van de Panne, "Learning locomotion skills using deeprl: Does the choice of action space matter?" in *Proc. ACM SIGGRAPH/Eurographics Symp. Comput. Animation*, 2017, pp. 1–13.
[31] C. Finn, X. Yu Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel, "Deep spatial autoencoders for visuomotor learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 512–519.
[32] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1334–1373, 2015.
[33] B. Kim and J. Pineau, "Socially adaptive path planning in human environments using inverse reinforcement learning," *Int. J. Social Robot.*, vol. 8, no. 1, pp. 51–66, Jan. 2016.
[34] A. Faust, K. Oslund, O. Ramirez, A. Francis, L. Tapia, M. Fiser, and J. Davidson, "PRM-RL: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 5113–5120.
[35] J. Choi, K. Park, M. Kim, and S. Seok, "Deep reinforcement learning of navigation in a complex and crowded environment with a limited field of view," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 5993–6000.
[36] L.-Y. Gui, K. Zhang, Y.-X. Wang, X. Liang, J. M. F. Moura, and M. Veloso, "Teaching robots to predict human motion," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 562–567.
[37] E. Catmull and J. Clark, "Recursively generated B-spline surfaces on arbitrary topological meshes," *Computer-Aided Design*, vol. 10, no. 6, pp. 350–355, Nov. 1978.
[38] T. W. Sederberg and R. T. Farouki, "Approximation by interval Bézier curves," *IEEE Comput. Graph. Appl.*, vol. 12, no. 5, pp. 87–95, Sep. 1992.
[39] C. Dripke, S. Höhr, A. Csiszar, and A. Verl, "A concept for the application of reinforcement learning in the optimization of CAM-generated tool paths," in *Machine Learning for Cyber Physical Systems*. Berlin, Germany: Springer-Vieweg, 2017, pp. 1–8.
[40] S. Mahadevan and G. Theocharous, "Optimizing production manufacturing using reinforcement learning," in *Proc. FLAIRS Conf.*, vol. 372, 1998, p. 377.
[41] R. P. Paul, *Robot Manipulators: Mathematics, Programming, and Control: The Computer Control of Robot Manipulators*. Los Angeles, CA, USA: Richard Paul, 1981.
[42] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, vol. 135. Cambridge, MA, USA: MIT Press, 1998.
[43] T. Jaakkola, S. P. Singh, and M. I. Jordan, "Reinforcement learning algorithm for partially observable Markov decision problems," in *Proc. Adv. Neural Inf. Process. Syst.*, 1995, pp. 345–352.
[44] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 2000, pp. 1008–1014.
[45] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel, "Asymmetric actor critic for image-based robot learning," 2017, *arXiv:1710.06542*. [Online]. Available: http://arxiv.org/abs/1710.06542
[46] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*. [Online]. Available: http://arxiv.org/abs/1707.06347
[47] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas, and N. Heess, "Reinforcement and imitation learning for diverse visuomotor skills," 2018, *arXiv:1802.09564*. [Online]. Available: http://arxiv.org/abs/1802.09564
[48] T. Li, H. Geyer, C. G. Atkeson, and A. Rai, "Using deep reinforcement learning to learn high-level policies on the ATRIAS biped," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 263–269.
[49] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 41–48.

[50] S. Braun, D. Neil, and S.-C. Liu, "A curriculum learning method for improved noise robustness in automatic speech recognition," in *Proc. 25th Eur. Signal Process. Conf. (EUSIPCO)*, Aug. 2017, pp. 548–552.
[51] C. Gong, D. Tao, S. J. Maybank, W. Liu, G. Kang, and J. Yang, "Multi-modal curriculum learning for semi-supervised image classification," *IEEE Trans. Image Process.*, vol. 25, no. 7, pp. 3249–3260, Jul. 2016.
[52] Y. Shi, M. Larson, and C. M. Jonker, "Recurrent neural network language model adaptation with curriculum learning," *Comput. Speech Lang.*, vol. 33, no. 1, pp. 136–154, Sep. 2015.
[53] C. Florensa, D. Held, M. Wulfmeier, M. Zhang, and P. Abbeel, "Reverse curriculum generation for reinforcement learning," 2017, *arXiv:1707.05300*. [Online]. Available: http://arxiv.org/abs/1707.05300
[54] M. Svetlik, M. Leonetti, J. Sinapov, R. Shah, N. Walker, and P. Stone, "Automatic curriculum graph generation for reinforcement learning agents," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 2590–2596.
[55] Z. Ren, D. Dong, H. Li, and C. Chen, "Self-paced prioritized curriculum learning with coverage penalty in deep reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2216–2226, Jun. 2018.
[56] S. Narvekar, J. Sinapov, and P. Stone, "Autonomous task sequencing for customized curriculum design in reinforcement learning," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Aug. 2017, pp. 2536–2542.
[57] K. Perlin, "Improving noise," in *Proc. 29th Annu. Conf. Comput. Graph. Interact. Techn.*, 2002, pp. 681–682.
[58] J. Gielis, "A generic geometric transformation that unifies a wide range of natural and abstract shapes," *Amer. J. Botany*, vol. 90, no. 3, pp. 333–338, Mar. 2003.
[59] R. C. Gonzales and R. E. Woods, *Digital Image Processing*, 2nd ed. Boston, MA, USA: Addison-Wesley, 2002.
[60] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, and A. Ray, "Learning dexterous in-hand manipulation," *Int. J. Robot. Res.*, vol. 39, no. 1, pp. 3–20, 2020.
[61] T. Bansal, J. Pachocki, S. Sidor, I. Sutskever, and I. Mordatch, "Emergent complexity via multi-agent competition," 2017, *arXiv:1710.03748*. [Online]. Available: http://arxiv.org/abs/1710.03748
[62] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.
[63] Y. Wu, E. Mansimov, R. B. Grosse, S. Liao, and J. Ba, "Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5279–5288.
[64] J. X Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick, "Learning to reinforcement learn," 2016, *arXiv:1611.05763*. [Online]. Available: http://arxiv.org/abs/1611.05763
[65] E. Rohmer, S. P. N. Singh, and M. Freese, "V-REP: A versatile and scalable robot simulation framework," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nov. 2013, pp. 1321–1326.
[66] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*. [Online]. Available: http://arxiv.org/abs/1509.02971
[67] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," 2015, *arXiv:1506.02438*. [Online]. Available: http://arxiv.org/abs/1506.02438
[68] A. A. Taha and A. Hanbury, "An efficient algorithm for calculating the exact Hausdorff distance," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 11, pp. 2153–2163, Nov. 2015.



TAEWOO KIM received the B.S. degree from the School of Mechatronics Engineering, Chungnam National University, Daejeon, South Korea, in 2011. He is currently pursuing the Ph.D. degree with the Department of Computer Software and Engineering, Korea University of Science and Technology, Daejeon. His research interests include robot kinematic control, motion retargeting, and RL in robotic systems.

JOO-HAENG LEE received the Ph.D. degree from the Computer Graphics Laboratory, Computer Science Department, POSTECH, in 1999. His thesis was General Sweep Boundary Construction Based on Envelope and Set Operations. He joined the Electronics and Telecommunications Research Institute (ETRI) to work on various research fields. As an Artist of code painting, he has held several personal and group exhibitions with images generated with his own algorithms. He is currently a Principal Research Scientist with ETRI and a Professor with the Korea University of Science and Technology (UST), South Korea. His research interests include geometric modeling, photorealistic rendering, augmented reality, human-robot interaction, deep learning, and artificial intelligence. He was a recipient of the Wolfram Innovator Award, in 2019.