ARTIFICIAL INTELLIGENCE

An adaptive deep reinforcement learning framework enables curling robots with human-like performance in real-world conditions

Dong-Ok Won[1], Klaus-Robert Müller[1,2,3], Seong-Whan Lee[1,4]*

The game of curling can be considered a good test bed for studying the interaction between artificial intelligence systems and the real world. In curling, the environmental characteristics change at every moment, and every throw has an impact on the outcome of the match. Furthermore, there is no time for relearning during a curling match due to the timing rules of the game. Here, we report a curling robot that can achieve human-level performance in the game of curling using an adaptive deep reinforcement learning framework. Our proposed adaptation framework extends standard deep reinforcement learning using temporal features, which learn to compensate for the uncertainties and nonstationarities that are an unavoidable part of curling. Our curling robot, Curly, was able to win three of four official matches against expert human teams [top-ranked women's curling teams and Korea national wheelchair curling team (reserve team)]. These results indicate that the gap between physics-based simulators and the real world can be narrowed.

INTRODUCTION

Applying artificial intelligence (AI) technologies to the real world (*1*–*7*) is a challenging problem. Operating beyond the confines of a laboratory means dealing with unknown factors, such as an environment that varies over time, that can have an effect on the performance of an AI system (*8*, *9*). Moreover, the real world offers many uncertainties, which may be too complex and ill defined to be modeled with sufficient accuracy (*10*). Thus, it is necessary to incorporate uncertainty into modeling efforts and to measure and approximate changes in real-world environments (*5*–*7*).

Although deep reinforcement learning (DRL) methods have been successfully applied to games in a discrete virtual environment (*11*–*13*), numerous real-world reinforcement learning (RL) applications require an agent to select optimal actions from a continuous environment. A strength of RL is that it can learn by itself through trial and error without specific previous knowledge, such as a complete understanding of principles and rules (for example, a game or a robotics task). Therefore, RL systems typically learn from simulations in virtual environments, and considerable efforts have been made to apply an RL model, trained from simulations, to the real world (*5*, *6*, *14*–*18*). Note that an RL system with well-designed rewards can adapt well to a certain stationary learning environment (i.e., fixed surface friction). However, when the environment changes, even a well-trained DRL model is unable to adapt to a changing environment; thus, producing reasonable results will be challenging because such models are typically only adapted to a specific (stationary) environment. Such models then require a process of relearning each time when the environment changes (*19*).

Many studies have focused on the simulation–to–real world (sim-to-real) transition to narrow the reality gap in various robot control problems (*5*, *6*, *16*, *17*, *20*–*27*). Arndt *et al.* (*20*) studied quick adaptation strategies to unknown conditions, for example, a sim-to-real transfer under unknown friction. Tan *et al.* (*5*) designed a highly precise simulation and randomized the physical parameters to learn agile locomotion. After learning in a simulation environment, a quadruped robot could successfully perform two gaits in a real environment. These studies showed that the gap between simulation and reality can be narrowed. Peng *et al.* (*16*) demonstrated an object-pushing task with a robotic arm using dynamics randomization. The experiments with a real-world pushing task demonstrated performance comparable with that of a simulation and the ability to adapt to changes in contact dynamics. The paper claimed that the robotic arm can adapt to various environment changes, such as mass, surface friction, and so on, through learning with dynamics randomization. However, here also, the real test appears to have been conducted in a stable and stationary environment with limited changes. Song *et al.* (*21*), in their study, made changes to the environment and experimented with its adaptability. Their study addressed the problem of adapting to 50 episodes (for a total of 150 s) of real-world data in a mass-voltage task [the battery voltage dropped from 16.8 to 10 V and added a mass of 0.5 kg (about 8% of robot mass)] or a friction task (the friction coefficient reduced from rubber feet to tennis ball feet). Sufficient time was allocated to adapt to the changing environment. The work of Hwangbo *et al.* (*17*) dealt with the pose of legged robots using DRL. Their paper introduced a method to train a neural network policy in simulation and then transfer this to a legged system. This showed that the learned policy can successfully recover its pose from a random initial configuration in less than 3 s. Please refer to table S1 for a detailed comparison of transferring methods that bridge the reality sim-to-real gap.

In our study, we consider the game of curling as a test bed for demonstrating the interaction between an AI system and a highly nonstationary real-world scenario. Curling is challenging because it requires precise throwing (robot control problem) and strategic planning to win. For this, we had to address the issue of transferring from a simulation to a real environment with high uncertainty and needed to propose adaptation to a dynamically changing environment where relearning is not a possibility. In curling, relearning is

[1]Department of Brain and Cognitive Engineering, Korea University, Seoul, Republic of Korea. [2]Machine Learning Group, Department of Computer Science, Berlin Institute of Technology, Berlin, Germany. [3]Max Planck Institute for Informatics, Saarbrücken, Germany. [4]Department of Artificial Intelligence, Korea University, Seoul, Republic of Korea.
*Corresponding author. Email: sw.lee@korea.ac.kr

not feasible due to the time constraints of the tournament rules; in addition, very little to no information is available on the nature of the environmental change. Furthermore, every throw (episode) has a large impact on the outcome of the match. This case study contains challenges related to using robotic systems in real-world environments: strong temporal variability, uncertainties, and continuousness (*28*).

Curling has been described as a combination of bowling and chess (*29*); it is a turn-based game in which two teams play alternately on the ice sheet, requiring a high level of strategic thinking and performance. There are a variety of strategic nuances and subtleties, but as in most strategic games, there are two basic strategies: play cautiously, waiting for an opportunity to exploit the opposition's mistakes, or play aggressively, where the "best defense is a strong offense," like in chess or the game of go. From an AI perspective, curling—often referred to as "chess on ice" (*30*–*32*)—is different compared with board games such as chess or go (*12*, *13*) in multiple ways (fig. S8). Curling has a considerably higher number of allowed moves because the game progresses in a continuous environment (*2*, *3*, *33*). Because of stones' collisions, curling modeling requires a complex and time-consuming physics-based simulation process to accurately describe and simulate the possible allowed moves for the next step (*31*, *34*). Furthermore, the real curling game is played on an ice sheet covered with small pebbles. The pebbles change their conditions depending on factors such as the temperature, humidity, maintenance skill of ice makers, elapsed time since the conclusion of the maintenance, the previous stone trajectories, and finally the amount of sweeping done during the game (*35*, *36*)—all of which neither are under control nor can be measured. These conditions imply that when delivering curling stones using exactly the same direction, force, and curl, the trajectory of the stones will inevitably vary over time (Fig. 3). In addition, most of the strategic plays take place within a 1.83-m radius called the house region, about 40 m away from the hogline (the place where a curler releases the stone), so inaccuracies and strategic misplannings could be amplified.

In recent years, algorithms for curling strategy were analyzed just in a virtual simulator environment (*2*, *3*, *32*). In the real world, however, ice has varying conditions (even on short time scales), so the uncertainties and nonstationarities are higher than what may be expected and encountered when dealing with simulations in the virtual world. Therefore, the strategy algorithms developed using only simulator data are difficult to adapt to the real ice sheet environment. Uncertainty in curling can be caused by internal factors of the agent (robot control inaccuracies, etc.) or external environmental factors (temperature, humidity, friction, etc.). The external factors are intrinsically unpredictable and have strong nonstationarity because the friction of the ice essentially changes with every throw. In this uncertain environment, it becomes necessary (after learning) to adapt the agent in real time to perform the curling tasks well enough so that it could provide a very high competitive performance (see Materials and Methods).

Here, we introduce Curly, a robot that we have taught using an adaptive DRL framework to excel in the Olympic discipline of curling (Figs. 1 and 2). Compared with other studies, the main difference in our work is that Curly performs adaptive actions that can respond to the environment changes that occur continuously with every shot. These changes have a notable influence on the performance if not compensated appropriately in a continuous manner. Our robot Curly was able to win three of four official matches against expert human teams [top-ranked women's curling teams and Korea national wheelchair curling team (reserve team)], thus demonstrating human-level performance. Although we focus on curling, our framework is readily transferable to other complex real-world applications.

RESULTS

The curling ice sheet is an environment with highly varying uncertainty that has a large effect on the throw performance. Human players require many years of practice to master the game, which has complex strategic elements and a challenging throw technique. For example, world class–level curling players from the national wheelchair curling teams recorded approximate distance gaps of 0.8 to 1.3 m between the goal and reached positions in the Paralympic Winter Games 2018 (https://curlit.com/PDF/PWG2018_ResultsBook.pdf). This highlights the very challenging task of throwing with high accuracy to achieve small distance gaps in real curling. Our proposed adaptation framework can compensate for uncertainties and nonstationarities that are unavoidable in the curling game by augmenting standard DRL through temporal features, which helps to lower the distance gaps to competitive levels. Specifically, we could show that the adaptation works well and compares very favorably to transferring a model trained on curling physics-based simulations

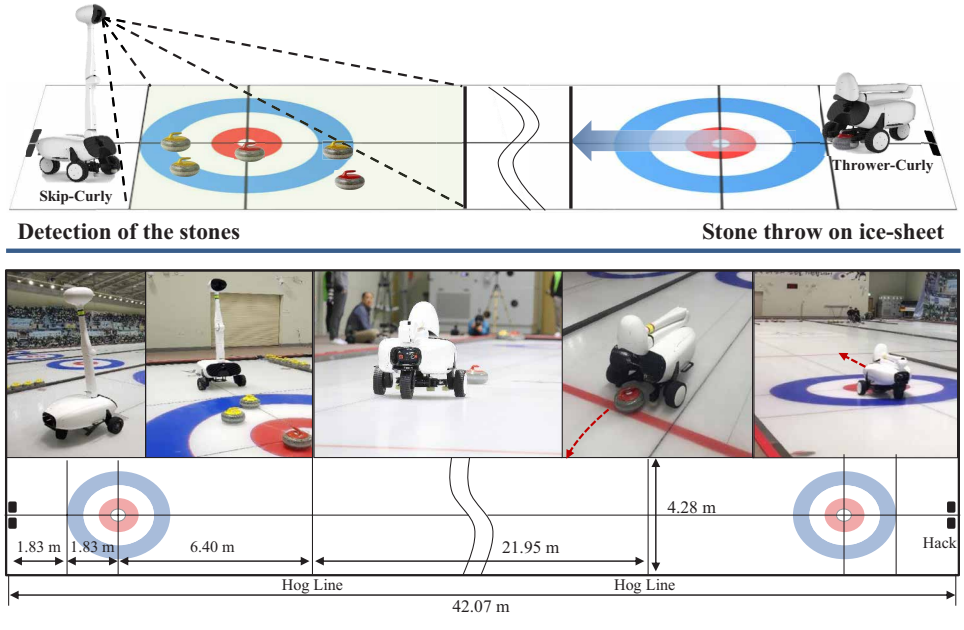

Fig. 1. AI curling robot system. The system consists of thrower/skip-Curly and curling AI (adaptation of the real icy environment, strategy planning, and curling simulator) (movie S6).

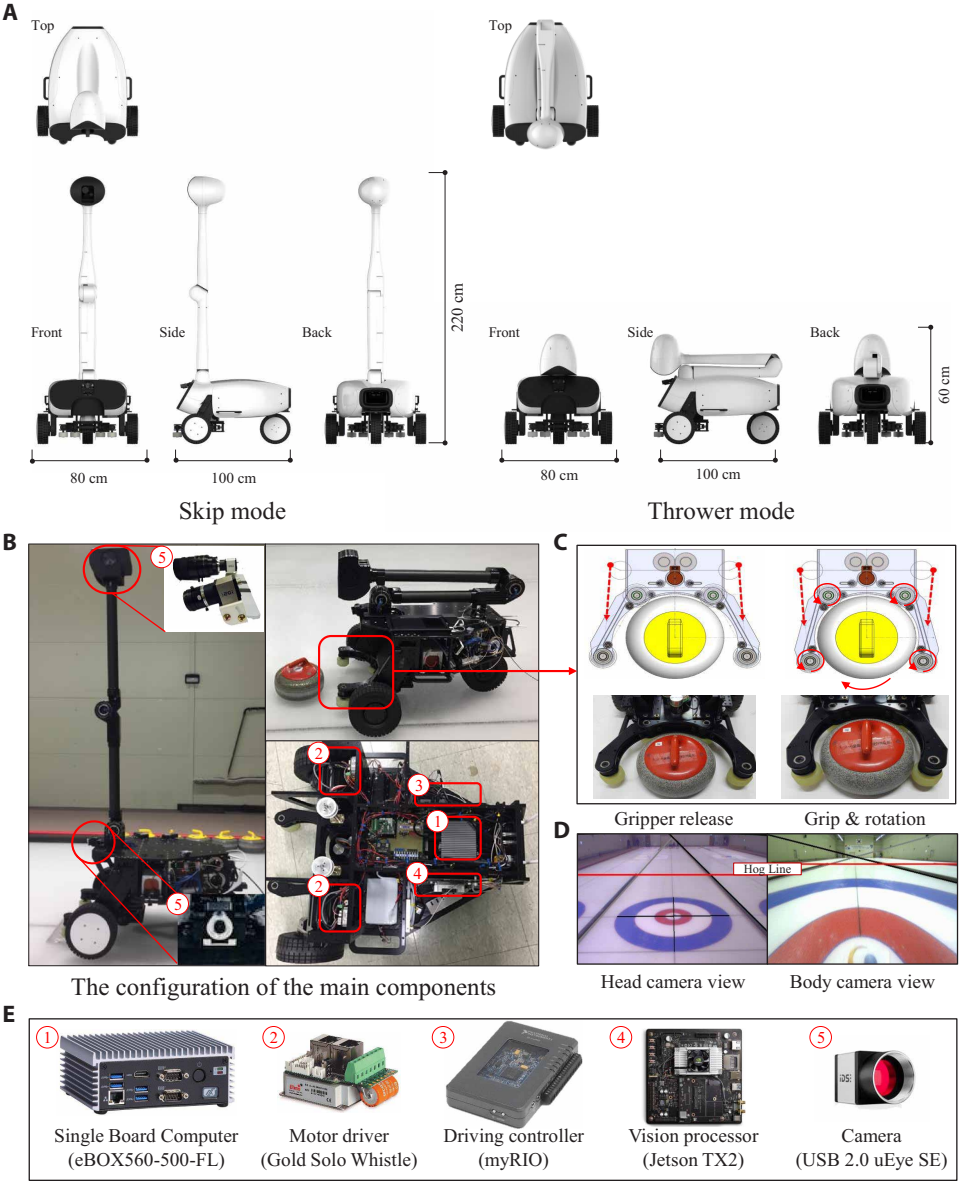

Fig. 2. Specification of the Curly. (**A**) (Left) skip mode and (right) thrower mode in the Curly. (**B**) Configuration of the main components in the skeleton Curly. (**C**) Operation of the gripper: (left) gripper open for stone release and (right) stone grip and rotation. (**D**) Camera view in head and body. Head camera: Stone detection and current location (i.e., orientation) of itself in skip-Curly and thrower-Curly, respectively. Body camera: Hog line detection for hog line release. (**E**) Main controllers and components.

only. An adaptation to the continuous changes occurring in the curling environment can be performed by adapting the whole control architecture of the robot (e.g, relearning) as a reaction to this change (our aim is to avoid this case). Or, alternatively, the control architecture stays untouched, and the coordinates of the target are adapted such that the control using this "adaptive action from the adaptation DRL model" yields a throw at the desired true target coordinate as if there was no change.

We will first discuss a simulation scenario that will serve as a proof of concept to quantitatively show the usefulness and limits of the proposed learning and adaptation procedures. In addition, we will show real curling results from an experimental setup to measure the performance of our proposed models in a reproducible real-world environment and determine their limitations. Last, we report on a series of tournament games that are unique events against human opponents, which present a real practical challenge for our robot, Curly. In the coming sections, we will conduct a comparison of the DRL adaptation, rule-based adaptation (*33*), no adaptation on simulated curling (see the "Simulated curling" section and Fig. 4), and a real curling ice sheet (see the "Test throw experiment on a real curling ice sheet" section and Fig. 5), introducing algorithmics and experimental findings. We have also used DRL adaptation in official real curling matches (see the "Real curling game with human teams" section and Fig. 6). The DRL adaptation model consists of model-free DRL based on the policy gradient (PG) algorithm, where we exploit the past distance gaps between the target and reached positions for learning the underlying dynamical changes of the environment. Furthermore, the rule-based adaptation method is based on previous knowledge of capturing the relationship between the target and reached positions (see also section A.1 in the Supplementary Materials). The no adaptation method is not learning about changes in the environment. All of these methods are first trained in a curling simulation environment and then tested on a real curling ice sheet (see the "Test throw experiment on a real curling ice sheet" section) and real icy curling matches against human teams (see the "Real curling game with human teams" section). Specifically, on a real curling ice sheet, the proposed adaptation DRL framework is only allowed to calibrate using a few throws before the test experiment or the start of the curling tournament (e.g., four preparation shots are used for initial adaptation; Fig. 3) and then continuously adapts using past distance gaps.

Simulated curling

In Fig. 4, the model-free DRL with adaptation and no adaptation condition (see Materials and Methods) are compared with respect to the cumulative distance errors under environmental changes (i.e., ice sheet friction variable and wear of pebbles). As shown in Fig. 4A, our proposed DRL method learned to curl at high accuracy, essentially reflecting the changes of the ice condition in the virtual environment. For the proposed model-free DRL framework with adaptation (see the "Adapting deep RL" section), the mean of the distance error is reduced to approximately one-third of the error of the no adaptation

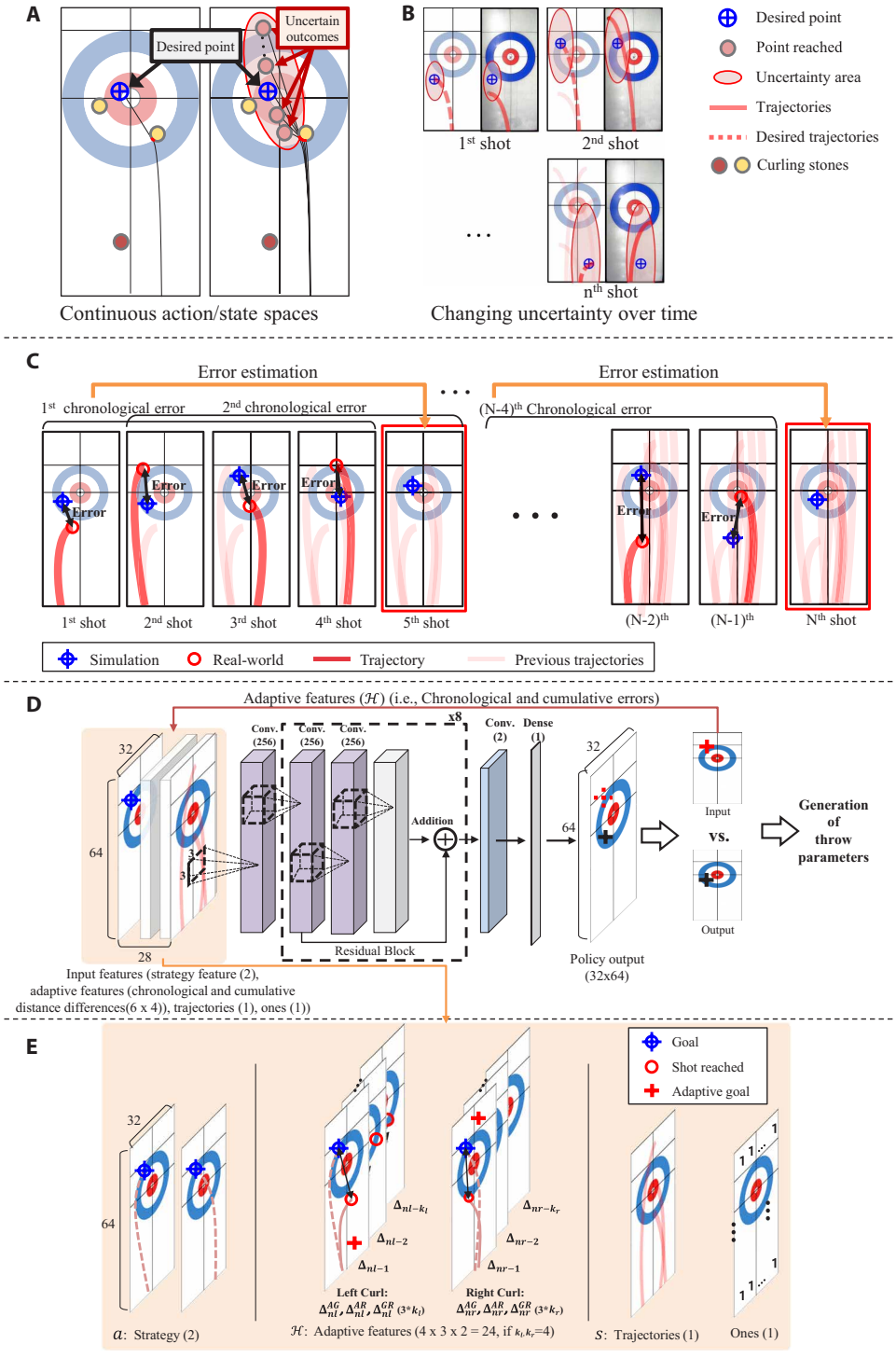

Fig. 3. Overview of concept and proposed framework to adapt deep RL from simulation to real curling with its uncertainties. (**A**) Distance gap (error) caused by continuous action/state spaces. (**B**) Changing uncertainty over time in the real world. (**C**) Conceptual gap estimation between simulation and real world with uncertainty using environment features (i.e., chronological distance errors and cumulative trajectories). (**D**) Architecture of our network for adaptation. (**E**) Various features for learning, including features from past trajectories for adaptation.

method: 3.12 ± 0.20 m (no adaptation) → 0.83 ± 0.41 m (DRL adaptation) (table S2). Moreover, as shown in Fig. 4C, the time-varying distance errors in the no adaptation method are higher than those in the proposed DRL adaptive method. Moreover, we also compared nonstationary and stationary conditions for the real icy environment condition and an ideal (normal uncertainty environment) condition, respectively. The DRL method is also more stable than the other methods in the nonstationary condition; we studied this as a function of the elapsed time since the last ice maintenance. Also, the adaptive DRL method obtained not only the lowest cumulative errors compared with others in nonstationary condition (Fig. 4B) but also stable low distance errors in the stationary condition (Fig. 4C). Thus, we could demonstrate that external environmental factors are well addressed in our adaptive framework (Figs. 1 and 3 and see also algorithm S1).

The DRL method is also found to be more stable than the other methods in nonstationary condition (i.e., an environment with dynamic changes); the rule-based method is compensated in the stable environment but does not work in the nonstationary condition (Fig. 4C).

Test throw experiment on a real curling ice sheet

As shown in Fig. 5, we compared the adaptation (the DRL and rule-based methods) and no adaptation method under environment changes from various uncertainties (temperature, humidity, pebble, ice status by the ice maker, etc.). Our adaptation model greatly reduces the error scores compared with the other methods in Fig. 5 and table S2: 1.38(±0.67) m, 2.11(±0.85) m, and 3.61(±0.87) m for adaptive DRL, rule based, and no adaptation, respectively. Furthermore, the distance errors were accumulatively validated through time adapting for change in the environmental parameters (e.g., ice sheet friction variable and wear of pebbles) (Fig. 5B).

This finding reflects that the accuracies resulting from the above strategies can differ depending on the degree of preciseness used to address the uncertainty. Moreover, a statistically significant difference is found between the observed distance error of these methods [analysis of variance (ANOVA) (*37*, *38*): $F = 33.82$, $P < 0.01$; *t* test: no adaptation > rule based > DRL adaptation, Bonferroni-corrected $P < 0.05$] (Fig. 5). The results indicate that the DRL adaptation method shows the best performance in terms of the correction of the distance errors in real curling.

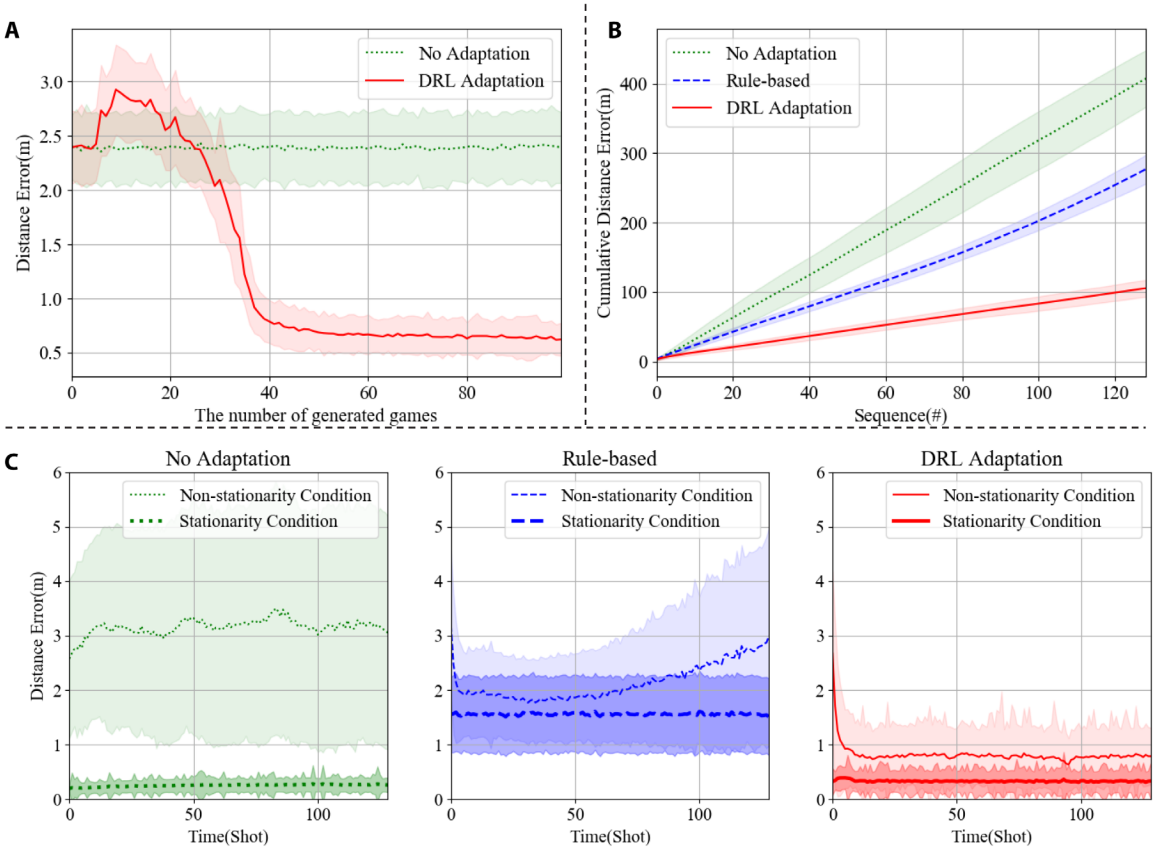

Fig. 4. Comparison of the deep RL adaptation, rule-based adaptation, and no adaptation on simulated curling. (**A**) Learning curve between desired and reached points for the adaptative deep RL method in a simulation [mean distance difference about 1,280,000 shots in each epoch (10,000 epoch × 128 shots)]. (**B**) Comparison of the cumulative distance error according to the elapsed time in nonstationarity condition. Test match has 10,000 sessions [=1,280,000 shots; one session consisted of 128 shots (8 ends × 16 shots)]. (**C**) Comparison of the averaged distance error in nonstationarity (thick line) and stationarity (thin line) conditions according to the elapsed time since the maintenance ended. Test match has 10,000 sessions [=1,280,000 shots; one session consisted of 128 shots (8 ends × 16 shots)].

Real curling game with human teams

We performed four official matches with a top-ranking Korean women's curling team and the Korean national wheelchair curling team. Snapshots of the curling matches are available in Fig. 6 in addition to videos of the official curling matches. These videos show the highlights and also the entire matches [videos for highlights (<3 min): match A (movie S1), match D (movie S2); videos for representation of strategies and throw: match C (movie S3), match D (movie S4); video links for entire matches: match A (https://youtu.be/1ZQOo0H6_FA) (16 min), match D (https://youtu.be/Cvp5nUu_GTM) (44 min)]. The matches allow only the thrower and skip player without sweepers (similar to wheelchair curling). Curly played the real curling game on par with the top-ranked players in Korea. In the course of the game, we have assumed the same fair play conditions as the top-ranked human team without giving the opportunity to repeat throws, even if Curly made mistakes (slight slip, little angle/speed of thrower errors, stone location detection, etc.). As seen in Fig. 6C, we succeeded not only in terms of strategic planning but also with respect to the real-time adaptation within the real curling game setting. We find that our robot Curly was able to win three of four official matches against human expert player teams [i.e., top-ranked women's curling teams and Korea national wheelchair curling team (reserve team)].

Furthermore, we analyzed the distance gaps during the curling game between non-adaptation (i.e., it would have occurred without using adaptation methods) and adaption (see Fig. 6B). In particular, the variation of the distance gaps in the number of throws was analyzed by comparing the cumulative error between the adaptation and non-adaptation. Despite the large uncertainty in the later part of the game, we observe that the target coordinates and distance differences have been narrowed reliably through adaptation. As a result, Curly was enabled to win against human players with a high level of performance.

DISCUSSION

Curling has been presented in this work as a test bed scenario to AI systems, where deviations from a controlled environment pose intricate challenges. Directly applying a model learned from virtual curling to the real environment has been unsuccessful. We could, however, demonstrate that a successful throw with good accuracy is possible with simulations of a real ice sheet environment and additional compensation that needs to be learned. As shown in Fig. 5, the error trends represent the uncertainty of all the thrown stones by the different control techniques in an online test. As the experiment progressed, we observed that various uncertainties accumulated,

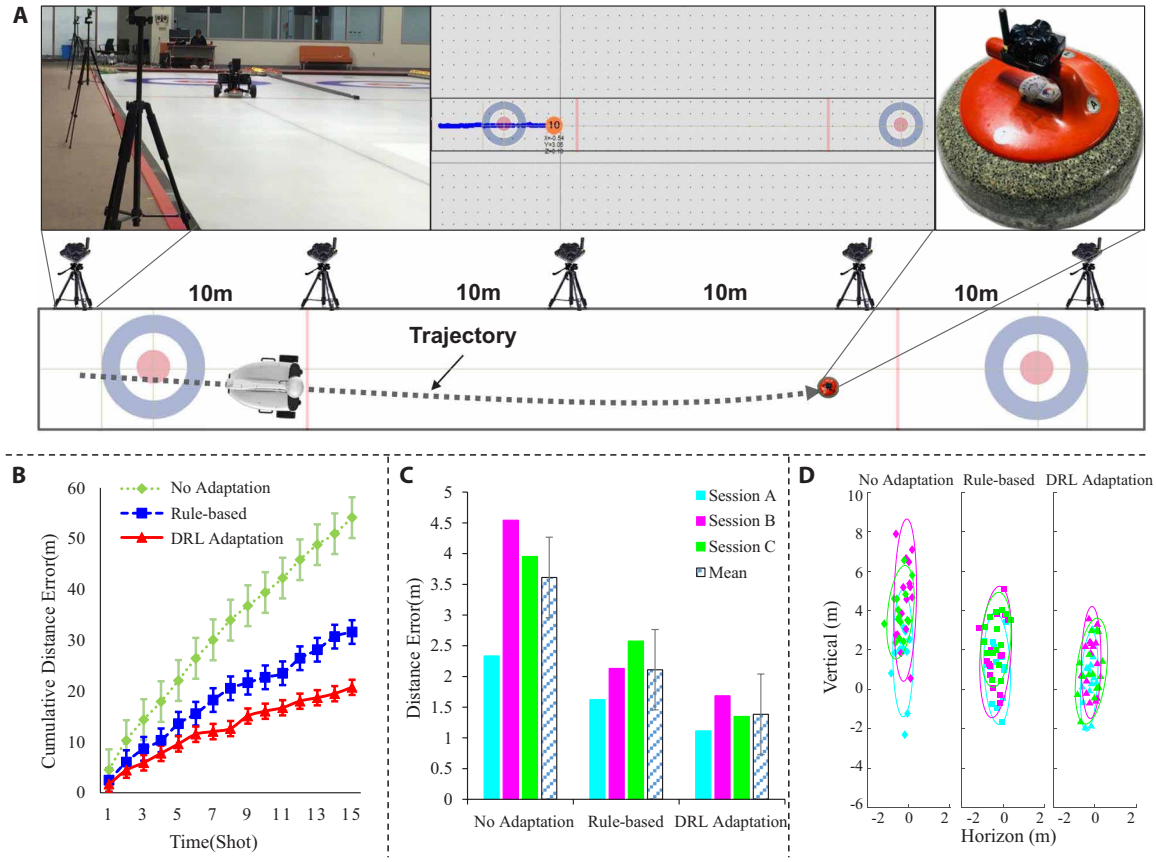

Fig. 5. Experimental setup for online test on real curling ice sheet. (**A**) Thrown stone's trajectory data captured by mounting a sensor (indoor GPS device; 12 Hz, ± 2 cm) attached on the stone (right). The thrower-Curly is throwing to deliver the stone to the target location chosen by the curling AI (left). (**B**) For the online test in the real curling ice sheet, comparison of the cumulative distance difference according to the elapsed time since the maintenance ended (mean and SE). (**C**) Mean of distance error between goal and reached coordinates for no adaptation, rule-based, and proposed adaptation DRL method in real curling environment. (**D**) Distance error between goal and reached coordinates for all sessions and methods.

gradually causing errors. Thus, it is important to compensate by adapting to the current environment. The online adaptation to the changing real environment performed by Curly uses recent trajectories and arrival position statistics. Because only few throws per game are made (the real curling match takes a total of 160 throws during about 120 min), adaptation needs to be performed in a data-efficient manner.

To explore the limits in accuracy of our various proposed AI controllers, we performed systematic test throws on the actual curling ice sheet for comparison. Unfortunately, Curly takes about 1 to 2 min for one shot, including whole throw procedures such as stone grasp, recognition of own coordinates, strategy planning/adaptation, and throwing and return. Having these experimental limits in mind, we designed an experiment with a total of 45 throws in one session (15 throws for each method) to measure success statistics; we repeated each session three times and thus have three sessions with altogether 45 × 3 throws. The above 45 throws were all targeted to random positions near the house (score zone) to obtain a target-independent assessment of the throw quality.

The best average accuracy of about 1.3 m for Curly (see Fig. 5C) is of the same order as that of a national wheelchair curling teams 0.8 to 1.3 m in the Paralympic Winter Games 2018. Sweeping allowed further compensation of differences up to 0.2 to 0.8 m in the Olympic Winter Games 2018 (https://curlit.com/PDF/PWG2018_ResultsBook.pdf, https://curlit.com/PDF/OWG2018_ResultsBook.pdf). Note that the throw experiments designed to test a special throw, i.e., the draw strategy, exhibit vertical errors (errors in the direction of the long axis of the curling sheet) that are roughly four times larger than the horizontal errors (Fig. 5D). Because of this error distribution, the success rate of the draw strategy can be expected to be lower than that of a takeout throw that just needs to hit the opponent's stone hard enough to remove it from the ice and depends only on the much smaller horizontal errors. Similarly to Curly, takeouts can also be much more accurately delivered by human players without sweeping. The average success rates were about 57% for draw and 62% for takeout in the Paralympic Winter Games 2018. Here, the medalist teams achieved about 61 and 66% in draw and takeout shot success rates, respectively. In the Olympic Winter Games 2018 involving sweeping, national human players showed success rates about 79% between draw and takeouts; i.e., sweeping can greatly affect shot accuracy.

We demonstrated the high performance of Curly through official matches with top-ranked human teams (Fig. 6). However, the precise reason for the competitiveness of Curly against human teams is not clear. Three possibilities for explaining this effect exist in principle: (i) the human team lacks the social competitive edge and is

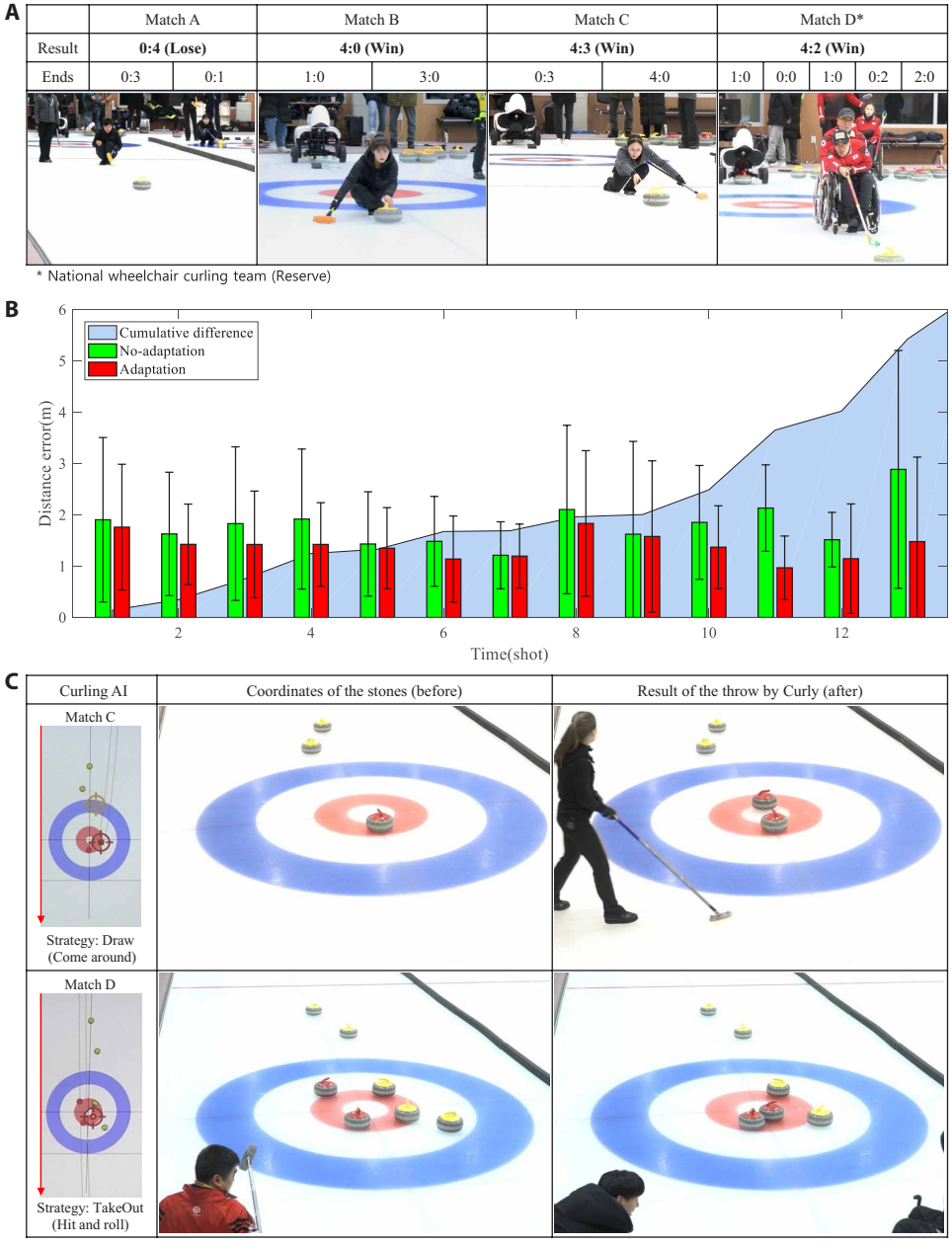

Fig. 6. Real curling match with human teams. (**A**) Results of curling matches with top-ranking Korean curling team and national wheelchair curling team (reserve). (**B**) Performance evaluation of adaptation deep RL during real curling game matches. Each error bar is the SD of the mean for each shot in all curling matches. (**C**) Representation of strategies and throw performance with Curly (see also movies S3 and S4 for matches C and D, respectively).

too relaxed to the outcome of a match against a robot opponent; (ii) humans tend to become nervous during matches, whereas robots do not; and (iii) the AI strategy component is superior to the human strategic insight because it is able to consider more rare events and can factor elements of uncertainty more successfully. We conducted an interesting case study of a game where Curly collaborated with human players. As well as the throw itself, the sweeping also has a large effect on the curling game. The human players played the role of sweepers in the Curly team instead of relying on the sweeping robots (fig. S1, movie S5, and https://youtu.be/1rocA7SLl2Y). Curly showed the human players a visualization of its throw strategies. This resulted in an interesting match between the Curly team mixed with sweeping human players and an all-human team (fig. S1). These special matches are a good example of collaborative robots in the real world. Curling is certainly a very challenging environment, and we found that humans and robots can cooperate very well, as was also observed in previous collaborative robot scenarios (*39*, *40*).

Typically, RL learns states from millions of actions in simulations. As discussed, in the real world, we may not even be able to perform hundreds of actions for the purpose of learning in each case (*11*–*13*, *18*, *23*, *41*). Also, application to nonstationary scenarios, namely, an environment that changes frequently, poses a challenge. Moreover, various uncertainty factors in unknown combinations contribute to performance errors. In this study, we therefore proposed a DRL framework that performs actions adaptively for states that are assumed uncertain because of the unknown environmental factors. The proposed framework carried out an estimation of the distance (to intended target position) gap based on cumulative trajectories and the chronological errors.

Systematic studies of Curly's performance were enabled by creating an integrated system platform—a task that required an extensive engineering effort. After numerous tests, investigations, and adaptation algorithms that could alleviate the challenges, we could reach an overall system stability and performance level that enabled Curly to play official curling matches with the best players in Korea. Most competitions were conducted in a wheelchair curling style without sweeping, and the content and results of the competition were successful and highly encouraging (Fig. 6 and movie S6).

The use of explainable AI techniques [e.g. (*42*–*45*)] to gain a further understanding of critical shot impacts, thus allowing the curling AI and its creators to learn better from their mistakes, will be interesting for future studies. Moreover, cases of extreme changes, such as sweeping in curling, deserve further investigation. Last, we note that the insights obtained within our framework on how to alleviate challenges such as strong temporal variability, uncertainties, and continuousness are readily transferable for contributing to other real-world applications of comparable complexity in robotics and beyond.

MATERIALS AND METHODS

In the following, we will introduce methods for adapting the deep reinforcement learner that was in a first step trained on simulation data and that will subsequently harvest the scarce data available in real-world conditions such as during a tournament.

Adapting deep RL

The main difference between curling and other games is that it is very difficult for a human or robot thrower to send the stone to the desired location due to the uncertainty and variability of the ice sheet condition, with variation caused by factors such as pebble condition, humidity, and temperature (i.e., external uncertainty). Also, robot throwers have inherent errors such as initial velocity, curl, and angles (i.e., internal uncertainty). Moreover, the large difference in the trajectory between a virtual physics simulator environment and the real ice sheet needs to be decreased to reach a competitive level. To perform this adaptation, we propose in this work to use the error information provided by the last throws for adapting the DRL model. To practically validate the resulting extended DRL framework (denoted later as adaptation DRL), we adapt the PG method (*13*, *46*) and also consider a simpler algorithm denoted as rule-based method as a baseline (see the Supplementary Materials for details) (*33*). We have introduced the system with the rule-based method (*33*). The main difference between the proposed adaptation and the rule-based method is whether they have dealt with the test bed environment with changes and full of uncertainty.

RL framework

Let us consider the standard RL framework, in which a learning agent interacts with a Markov decision process (MDP) [see, e.g., (*46*–*50*)]. To clarify the following notation, we have the state, action, and reward at each time $t \in 0,1,2,\ldots$ denoted by $s_t \in S$, $a_t \in A$, and $r_t \in R$, respectively. Also, the action a is defined as throw to the two-dimensional coordinate a on the ice sheet (*12*, *13*). Its dynamics are characterized by state transition probabilities

$$P^{a}_{ss'} = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a) \tag{1}$$

The decision-making procedure for Curly at each time is characterized by a policy

$$\pi(a_t \mid s_t, \boldsymbol{\theta}) = \Pr(a_t = a \mid s_t = s, \boldsymbol{\theta}_t = \boldsymbol{\theta}), \forall s \in S, a \in A \tag{2}$$

where $\boldsymbol{\theta} \in \mathbb{R}^{l}$, for $l \ll |S|$, is the policy's parameter vector. We assume that π is differentiable with respect to its parameters such that $\frac{\partial \pi(a \mid s)}{\partial \boldsymbol{\theta}}$ exists.

Here, we briefly describe the PG method that is used in our study (see also algorithm S1). The PG update $\boldsymbol{\theta}$ is defined as

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \left[r_t \nabla_{\boldsymbol{\theta}} (\ln \pi(a_t \mid s_t, \boldsymbol{\theta})) \right] \tag{3}$$

where the reward value r_t is defined as

$$r_t = \begin{cases} +2 & \text{if } |\Delta_t^{\mathrm{GR}}| < \text{half of the stone diameter (0.15m)}, \\ +1 & \text{if } 0.15\text{m} \le |\Delta_t^{\mathrm{GR}}| < 0.4\text{m}, \\ +0.5 & \text{if } 0.4\text{m} \le |\Delta_t^{\mathrm{GR}}| < 0.8\text{m}, \\ -1 & \text{otherwise} \end{cases} \tag{4}$$

and the r_t is determined by using the distance error Δ_t^{GR} between the desired position and the actual position and it is realized through execution. If Δ_t^{GR} is within half of the stone diameter (0.15 m), 0.4 m, or 0.8 m, then r_t is increased by +2, +1, or +0.5 for positive RL, respectively. Alternatively, r_t is decreased by −1 for negative RL. In curling, throwing a stone within 0.4 m of the target is regarded as very successful (curling stone diameter: 0.3 m). Furthermore, throwing it within 0.8 m is still considered to be successful because this is less than half of the house, or score zone (0.9 m). Even the distance gaps of the national team in the recent Paralympic Winter Games 2018 were 0.8 to 1.3 m. Reflecting the discussed curling game context, we therefore designed a discrete reward scheme (Eq. 4). In addition, the state s_t is defined as

$$s_t = \{\mathcal{H}_t, \mathcal{Y}_t^{G}, \mathcal{T}_t\} \tag{5}$$

where the state s_t is composed of the cumulative trajectories of the thrown stone ($\mathcal{T}$) on the curling ice sheet, the strategy (desired coordinates given by strategy planning model) ($\mathcal{Y}^{G}$), and the adaptive factor matrix ($\mathcal{H}$). In particular, learning to adapt to the unpredictable environmental changes was implemented by adding $\mathcal{H}$ to s.

DRL framework to adapt to uncertainties

We enrich the neural network for estimating the policy with temporal information to model nonstationary and uncertain environments (see Fig. 3D; for an overview of all input features into the DRL, see Table 1). To reflect these temporal changes and uncertainties, we define a matrix $\mathcal{H}$ of distance errors Δ from the previous throws and our prediction/simulation model

$$\mathcal{H}_n = \begin{Bmatrix} \Delta_{n_l-1}, \ldots, \Delta_{n_l-k_l} \\ \Delta_{n_r-1}, \ldots, \Delta_{n_r-k_r} \end{Bmatrix}, \text{where}(k_l < n_l; k_r < n_r; n = n_l + n_r) \tag{6}$$

where $\mathcal{H}_n$ is obtained for action a in state s and n is the nth action state (throw). n_l and n_r denote the number of left and right curl throws, respectively. k is the number of cumulative sequences used for training. k_l and k_r are the number of left and right curl cumulative sequences, respectively. We consider left and right curls separately for adaptation, because there is an asymmetry due to the different interaction of the rotating stone and the ice. For example, the maintenance skill of the ice maker gives rise to asymmetry in the pebbles or other aspects of the curling ice sheet condition.

The distance differences Δ (x,y coordinates) for left curl (Δ_{nl}) and right curl (Δ_{nr}) are defined as

$$\Delta_{n_l} = \begin{cases} \Delta_{n_l}^{\mathrm{AG}} = \mathcal{Y}_{n_l}^{A} - \mathcal{Y}_{n_l}^{G} \\ \Delta_{n_l}^{\mathrm{AR}} = \mathcal{Y}_{n_l}^{A} - \mathcal{Y}_{n_l}^{R} \\ \Delta_{n_l}^{\mathrm{GR}} = \mathcal{Y}_{n_l}^{G} - \mathcal{Y}_{n_l}^{R} \end{cases} \tag{7}$$

$$\Delta_{n_r} = \begin{cases} \Delta_{n_r}^{\mathrm{AG}} = \mathcal{Y}_{n_r}^{A} - \mathcal{Y}_{n_r}^{G} \\ \Delta_{n_r}^{\mathrm{AR}} = \mathcal{Y}_{n_r}^{A} - \mathcal{Y}_{n_r}^{R} \\ \Delta_{n_r}^{\mathrm{GR}} = \mathcal{Y}_{n_r}^{G} - \mathcal{Y}_{n_r}^{R} \end{cases} \tag{8}$$

where $\mathcal{Y}_n^{G}$ denotes the nth desired coordinates given by the strategy planning model, $\mathcal{Y}_n^{A}$ is the adaptive action (coordinate) a_n obtained from the adaptation DRL model, and $\mathcal{Y}_n^{R}$ results from a_n in real curling. The Δ superscripts denote the respective distance errors of the throws between G, R, and A, respectively. In addition, the cumulative

Table 1. Input features used in the network for our adaptive deep RL.

Features	Number of planes	Description
Strategy (coordinates)	2	Desired coordinates given by strategy planning model (strategy for left and right curls, respectively)
Adaptive factors ($\mathcal{H}$)	4 × 3 × 2	Chronological and cumulative errors, four previous sequences × distance errors Δ^{AG}, Δ^{AR}, and Δ^{GR} (two each for left and right curls)
Trajectories ($\mathcal{T}$)	1	Cumulative trajectories of the thrown stones

trajectories $\mathcal{T}_n$ of the previously thrown stones are used to reflect the change of the curling ice sheet caused over time. The environment and state s that contains the cumulative trajectories of the thrown stones $\mathcal{T}_n$ is defined as

$$\mathcal{T}_n = T_1 + T_2 + \ldots + T_n \tag{9}$$

where T is the trajectory of the thrown stone in a 237 × 1622 coordinate system (i.e., the trajectory path is marked as a series of ones, filling in the respective grid squares of the 237 × 1622 grid over which the stone passed and setting the rest of the squares to zero); we have reduced the resolution to 32 × 64 in the training phase. In the training phase, we furthermore add a random perturbation to the output coordinate point (with respect to the ideal simulation) to increase resilience to the uncertainties and changing environmental factors. This stabilizes training when going beyond the idealized noise-free curling simulation. The randomized coordinate point $\mathcal{Y}^{G'}_{x,y}$ is defined as

$$\mathcal{Y}^{G'}_{x,y} = \left(\mathcal{Y}^{G}_{x} + \epsilon_{x_1} \cdot \sin\left(\frac{1}{6}\cdot\epsilon_{x_2} + \frac{\pi}{4}\cdot\epsilon_{x_3}\right)\langle e_x\rangle, \mathcal{Y}^{G}_{y} + \epsilon_{y_1} \cdot \sin\left(\frac{1}{6}\cdot\epsilon_{y_2} + \frac{\pi}{4}\cdot\epsilon_{y_3}\right)\langle e_y\rangle\right) \tag{10}$$

where $\mathcal{Y}^{G}_{x,y} = (\mathcal{Y}^{G}_{x}, \mathcal{Y}^{G}_{y})$ represents the desired coordinates given by strategy planning model. In addition, ϵ_{x1}, ϵ_{x2}, ϵ_{x3}, ϵ_{y1}, ϵ_{y2}, ϵ_{y3} are appropriately scaled random perturbations and $\langle e_x\rangle$, $\langle e_y\rangle$ are averaged distance errors in real curling with no adaptation (i.e., vertical and horizontal errors are approximately 3 and 0.5 m, respectively).

The movements of the left/right curls are substantially different due to various external factors (amount of pebble uneven skewed in some areas, inclined ice sheet, etc.). To respond appropriately to this phenomenon, $\mathcal{Y}^{G'}_{x,y}$ need to distinguish between the left and right curl cases. Figure 3 shows an overview of our concept to adapt DRL from simulation to real curling with its uncertainties: The action space is, in principle, continuous; as an approximation for efficient learning in the neural networks, we use a discretization (*32*, *51*). We use a ResNet architecture (eight basic blocks) [c.f., the alphago zero or other well-known AI systems (*13*, *52*)], where the input feature maps (32 × 64) include the desired position given by strategy planning model (two planes), the trajectories of the past thrown stones ($\mathcal{T}$ from Eq. 9) (one plane), and error estimations Δs from Eqs. 7 and 8 [e.g., sequential distance gaps (4 × 3 × 2) planes] (Table 1). As shown in Eq. 6, for $\mathcal{H}$, the updates of the cumulative sequence differ for each left (k_l) and right (k_r) curl depending on the strategic target. Also, we need to adjust the number of cumulative sequences that are used for learning according to the application situation. For the real curling game, say in a tournament, we need to adapt to the new ice environment in the curling arena at the match time, where we are allowed to use about only 10 min of ice reading time (international curling match rules, seven to eight shots are available to Curly before the start of the game). So, we used this time to calibrate our adaptation using four throws for each left (k_l) and right (k_r) curl and then proceed to the curling match.

Essentially, the temporal features (i.e., the trajectories of the thrown stones) and cumulative distance errors are the essential features in our framework for adapting the simulation model estimates.

Note that retraining beyond a mere adaptation is not a practical option within the ruleset of curling matches.

AI curling robot system "Curly"

Curly, the AI curling robot system, consists of the curling AI (strategy planning model) (see Fig. 7 and fig. S2), adaptation DRL model to the dynamically changing environment in real icy world, and a curling simulator (see Figs. 3D and 8 and fig. S2) and two curling robots (skip-/thrower-Curly) (see Fig. 2). Curly operates in four steps: (i) The skip-Curly recognizes the coordinates of the stones on the ice sheet through active vision and transmits them to the curling AI. (ii) After receiving the current game status with the coordinates, the strategy planning (*32*) establishes a strategy within a curling simulator environment already factoring in uncertainties to compute the best throw (Fig. 7) and transmits them to the adaptation DRL model. (iii) After receiving the throw strategy, the adaptation DRL model computes an adaption to the coordinate provided by the strategy planning model, and then the curling simulator transmits the throw parameters of the adapted coordinate to the "thrower-Curly." Note that, here, our proposed framework (main contribution of this work) is to incorporate the diverse uncertainty factors into the real curling ice sheet parameter estimates, which plays a crucial role (Fig. 3). (iv) The thrower-Curly precisely delivers the actual stone with the parameters received from the curling AI, and when all the stones have stopped, the opponent takes his strategic move again.

Moreover, for a wireless control of the robot, a data communication module was used to transmit the acquired information (i.e., stones' coordinates) to the curling AI and to exchange the throwing parameters from the curling AI (i.e., the velocity, angle of stone, and the direction of curl). More details can be found in the Supplementary Materials (section A.5).

Curling robots: Skip and thrower

We constructed two identical robots (operated in skip and thrower modes), each equipped with video analysis, data communication, and throwing control modules including traction control (as we need to quickly accelerate a curling stone weighing about 20 kg) (see Fig. 2). The thrower-Curly implements our AI-based strategy on the ice sheet, i.e., holding and rotating a curling stone, accelerating it, and then releasing the stone from the gripper with the appropriate speed and angle before the hogline (Fig. 2). The skip-Curly can recognize the coordinates of all stones and the trajectories of the moving stones by using image processing techniques.

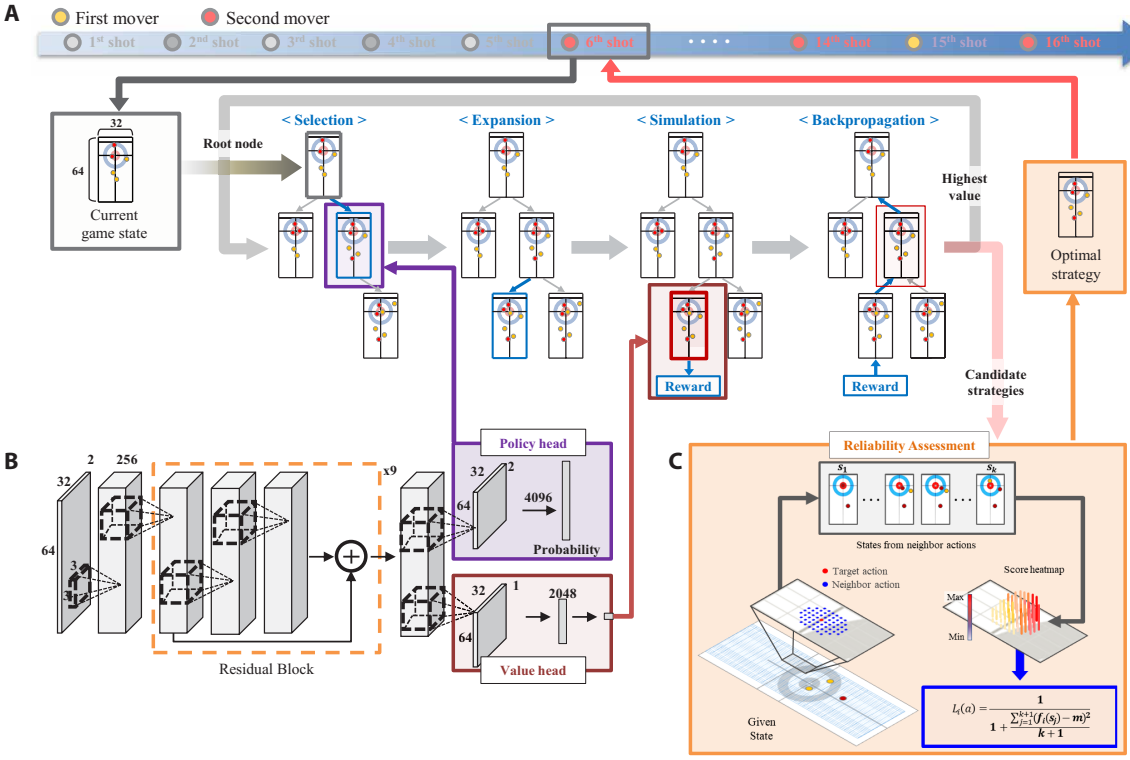

Fig. 7. Overview of strategy planning. (**A**) Steps of the MCTS algorithm. (**B**) Architecture of the strategy network. The network consists of the policy network head N_p and value network head N_v. (**C**) Policy RA (final selection).

Figure 2 shows the whole structure of the proposed robot, which consists of two type cameras (on the body and head), one neck (similar to giraffes or horses), two front wheels, one rear caster, and a gripper. The neck can lift the head camera up and bring it down. In the skip mode, the robot can recognize the stone by lifting the head. In the thrower mode, the head cameras on the neck are to detect the location of robot itself (localization), and the body camera is to detect the hog line. Although the head needs to be raised to recognize the stone positions accurately (because they are roughly 40 m away), the robot in crouched state enables stable throwing by the head down position and the respective lower center of gravity. Two front wheels are connected to two brushless DC motors and work as driving wheels, while one rear wheel is a caster that only supports the body. In addition, Curly uses the gripper to generate curls of the stone while driving and releases the stone before the hog line (rules of the competition). The gripper consists of two motors: One is used to grip and release the stone, and the other one rotates a belt that moves the stone to generate the necessary curl directions and speed of the stone (i.e., clockwise or counterclockwise rotation). Note that our mobile robots can perform a curling game without any external installation or cables.

Precision-controlled throwing in thrower-Curly

Curly can automatically make precise throws based on the current location estimate (i.e., coordinate on the ice sheet) and orientation, the desired velocity/angle control at the point of release time, and stable release moment control. Location and orientation are estimated using line detection and the crossing point of the curling ice sheet (house, side lines, etc.) while correcting the distortions according to the oblique view through a mapping of the actual ground-truth information and the obtained image from the head camera.

In a first step, the thrower-Curly travels approximately 10 m (from hack to hog line) to precisely control the target velocity and angle of the curling stone. The target velocity and angle are received from the curling AI, which is, as we indicated earlier, composed of strategy planning, adaptation, and curling simulation. Because the point of arrival (length of the throw) for the stone in the house depends mainly on the velocity at the moment of release, the thrower-Curly has to perform very precise velocity control. To achieve high precision driving, the first and most important technique is an efficient anti-slip control to prevent any slip between the rubber wheel and the ice, i.e., maintaining the slip ratio within a certain range. There are many anti-slip control algorithms to achieve high traction. Here, we apply to the thrower-Curly the so-called model following control suggested by Hori *et al.* (*53*), because its structure is straightforward and its tuning parameters are simple enough. The thrower-Curly driving is achieved by applying yaw moment observer (*54*) for a robust head angle control. As a result, our robot has an accuracy of 0.1° and 0.01 m/s error with respect to target velocity and angle [c.f. (*55*) for further details about the precision driving control applied to Curly]. The third step is the hog line release phase, one of the important rules to obey in the curling game. For stable release control, the thrower-Curly predicts the distance to the hog line based on the image processing technique (i.e., detection of the Hog line and side line) using the image of the front camera. Then, the image processing module sends a release signal to the gripper about 30 cm before the hog line; immediately, the gripper releases the stone.

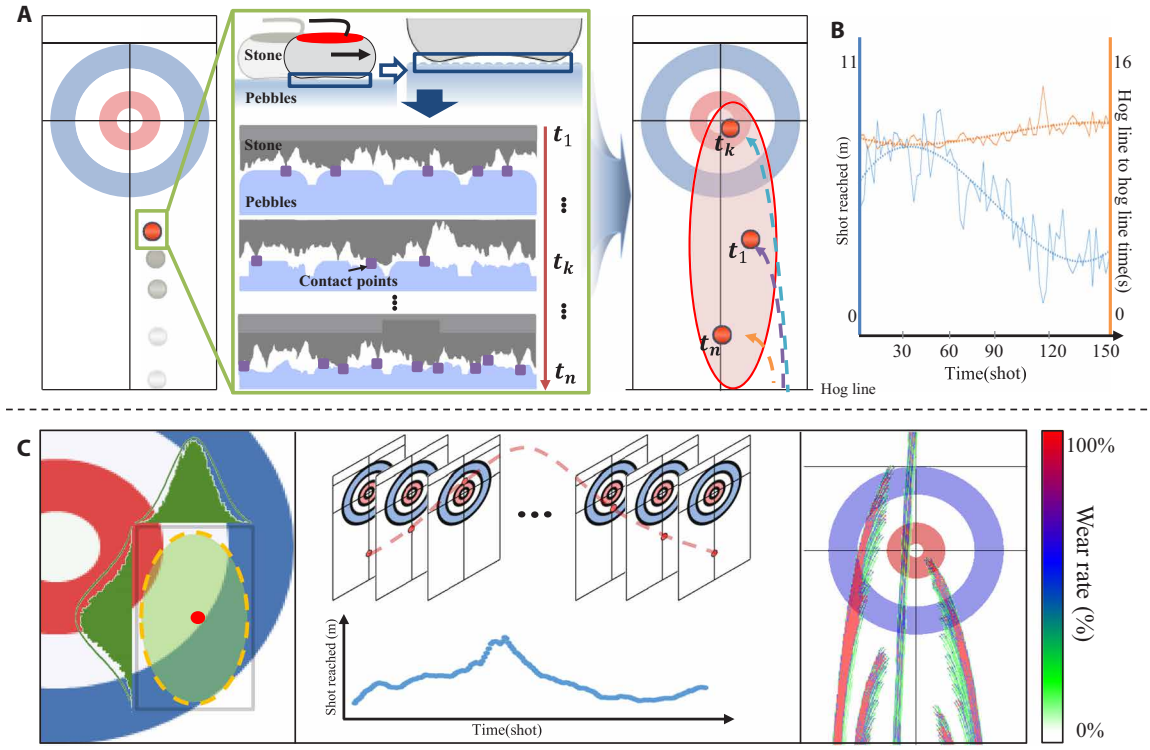

Fig. 8. Experimental setup for analyzing several uncertainties in simulated curling. (**A**) Overview of our curling simulation based on the curling stone's running band and pebbles on the ice (t_n is the nth shot). (**B**) Example of the reaching distance as the throwing progresses in implemented curling simulation. The shot reached in the y axis of the curling ice sheet (the zero on the y axis is the hog line). (**C**) Example of external environmental uncertainties (i.e., temperature, humidity, maintenance skill of ice makers, and elapsed time since the maintenance ended) and changed wear rate by thrown stones.

Detection of the curling stone location by the skip-Curly

Precise coordinate estimation of the stones is one of the important cornerstones for a successful curling game. The skip-Curly matches the coordinates of the actual curling ice sheet by recognizing the house circles, side line, back line, etc. After the newly thrown stone has stopped completely, its position is computed within the inferred coordinates and sent to the curling AI.

Realistic curling simulation

Our physics-based curling simulator is designed to simulate outcomes for a given parameter set, i.e., throw angle, velocity, and curl direction (Fig. 8). It can provide a starting point (although in the real world inaccurate) for a subsequent intelligent compensation, which then allows the robot to throw stones precisely on desired target points (Fig. 1), although all individual subcomponents of the Curly system are intrinsically carrying large uncertainties.

We implemented a curling stone curl simulation based on the representative physical mechanisms, known as "front-rear asymmetry" model and "scratch guiding" model (fig. S6, algorithms S2 and S3) following (*36*, *56*). Moreover, the implemented curling simulation can render the reaching distance different as the game progresses (Fig. 8).

Strategy planning in simulation

In our previous study, an optimal curling strategy was established within a simulation environment only (*32*). Curly's strategy planning uses a policy-value network (DNN) along with a Monte Carlo tree search (MCTS) using kernel regression upper confidence bound applied to trees (KR-UCT) (*32*). The search procedure follows the MCTS algorithm, which comprises selection, expansion, simulation, and backpropagation. (See also section A.3 in the Supplementary Materials for more details.) Moreover, we also implemented the reliability assessment (RA) step to achieve a stable strategy in the final selection. In addition, as an approximation, a discretization of the continuous action space is used for the MCTS search and the learning in neural networks.

Policy-value network

Our network takes the following inputs: the stones' location, stone order in terms of proximity from the tee (the center point of the house), number of shots, and flags to indicate whether each grid cell within the house, comprising three concentric circles where points are scored, is occupied by a stone (*32*). After a first convolutional block, nine residual blocks follow (*52*), which are shared during neural network training procedure (Fig. 7).

Monte Carlo tree search

Our MCTS method, referred to as KR-UCT, has been successfully applied to the simulated curling game (*32*). As shown in Fig. 7B, the curling strategy DNNs (policy-value networks) consist of the policy network head N_p and value network head N_v. The output of the neural network policy head N_p is applied for the selection and expansion of the actions in the MCTS steps. The value computed as the output of head N_p is used instead of an MCTS simulation (i.e., rollout policy).

Reliability assessment

In the curling simulation, the ice conditions are assumed unchanged with only a small range of uncertainties added (i.e., Gaussian noise),

and the throw stone trajectory shape is fixed (*32*). However, the real environment has a substantially larger range of uncertainties and also becomes more nonstationary. To provide more stable strategies in this study, the changes and uncertainties that emerge in the real world need to be anticipated, and an appropriate strategy that matches the real-time constraints for the simulation needs to be established accordingly. Moreover, when a stone is thrown to the target point in real curling, it will often deviate. Even small distance errors can have a substantial impact on the outcome of the curling match. To compensate for these issues, we also propose and apply an RA $\mathcal{L}(a)$ step to achieve a stable and robust strategy for the final selection after the MCTS step (Fig. 7C). It is possible to capture the uncertainty of the outcomes by simulating additional actions on neighboring target points. Because it has been proven that sufficient information can be obtained by sparse sampling (*57*), uncertainty can be measured through an analysis of few neighboring points. During search, the proposed algorithm performs stochastic simulations of stone throwings for a test group consisting of a target point and some k neighbors (slightly distorted targets). The $\mathcal{L}(a)$ is then approximated as a summation over the individual shot reliabilies $L_i(a)$ for the target action (a) from the given state s

$$\mathcal{L}(a) = \sum_{i=1}^{n} L_i(a) \tag{11}$$

$$L_i(a) = \left(\frac{1}{1 + \frac{\sum_{j=1}^{k+1} (f_i(s_j) - m)^2}{k+1}} \right) \tag{12}$$

where n is the number of functions f_n evaluating a position and m denotes the average of the evaluation values over the distorted targets. The evaluation functions f are determined by representative curling rules (i.e., f_1: score variation, f_2: changes in the number of guard stones, f_3: change in the number of stones in house). A large variance of evaluation values means that the difference between each distorted throw is large, and thus, the reliability $\mathcal{L}(a)$ goes to 0. Figure 8C shows the process of applying the proposed assessment method for curling. The RA evaluates the recommended strategic candidates by MCTS with the policy-value network in the curling simulation environment. Last, the $\mathcal{L}(a)$ model selects a stable strategy with the highest RA value. Insisting on throws with large reliability allows to achieve stable performance under nonstationary uncertainties, and thus, more robust strategies are created.

SUPPLEMENTARY MATERIALS

robotics.sciencemag.org/cgi/content/full/5/46/eabb9764/DC1
Summary of notation
Supplementary background, materials, and methods
Fig. S1. Match results between the top-ranked Korean women's team and cooperative team (robot and humans performing throwing and sweeping, respectively).
Fig. S2. Curly operates in main four steps.
Fig. S3. Sequence diagram of the throwing process in our system.
Fig. S4. The configuration of our curling simulator (server).
Fig. S5. Representation of the curling stone running band and pebbles on the ice.
Fig. S6. Overview of the asymmetrical friction mechanism on the rotating and sliding running band.
Fig. S7. Comparison of trajectories between actual case, our simulation, and Ito's simulation.
Fig. S8. Comparison of the go and curling (Curly).
Table S1. Comparison of transfer methods for bridging reality gap with physical robot.
Table S2. Comparison of the averaged distance error between no adaptation, rule-based, and model-free DRL-based adaptation in simulated curling and real curling.
Table S3. Comparison of curling simulation methods for the dynamics of the traveling curling stone.
Movie S1. Highlights of the official curling match A.
Movie S2. Highlights of the official curling match D.
Movie S3. Representation of strategies and throw performance with Curly in match C.
Movie S4. Representation of strategies and throw performance with Curly in match D.
Movie S5. Representation of collaboration throw with Curly and top-ranked humans.
Movie S6. Summary of AI curling robot.
References (*58*–*79*)

REFERENCES AND NOTES

1. M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, S. Schaal, Learning, planning, and control for quadruped locomotion over challenging terrain. *Int. J. Rob. Res.* **30**, 236–258 (2011).
2. T. Yee, V. Lisý, M. H. Bowling, Monte Carlo tree search in continuous action spaces with execution uncertainty, in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI)* (AAAI Press, 2016), pp. 690–697.
3. Z. F. Ahmad, R. C. Holte, M. Bowling, Action selection for hammer shots in curling, in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI)* (AAAI Press, 2016), pp. 561–567.
4. J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, P. Abbeel, Domain randomization for transferring deep neural networks from simulation to the real world, in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (24 to 28 September 2017), pp. 23–30.
5. J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, V. Vanhoucke, Sim-to-real: Learning agile locomotion for quadruped robots. arXiv:1804.10332 [cs.RO] (27 April 2018).
6. P. Christiano, Z. Shah, I. Mordatch, J. Schneider, T. Blackwell, J. Tobin, P. Abbeel, W. Zaremba, Transfer from simulation to real world through learning deep inverse dynamics model. arXiv:1610.03518 [cs.RO] (11 October 2016).
7. K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, S. Levine, V. Vanhoucke, Using simulation and domain adaptation to improve efficiency of deep robotic grasping, in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2018), pp. 4243–4250.
8. B. Hilleli, R. El-Yaniv, Toward deep reinforcement learning without a simulator: An autonomous steering example, in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence* (AAAI Press, 2018), pp. 1471–1478.
9. N. Fulton, A. Platzer, Safe reinforcement learning via formal methods: Toward safe control through proof and learning, in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence* (AAAI Press, 2018), pp. 6485–6492.
10. R. B. Duncan, Characteristics of organizational environments and perceived environmental uncertainty. *Adm. Sci. Q.* **17**, 313–327 (1972).
11. V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, Human-level control through deep reinforcement learning. *Nature* **518**, 529 (2015).
12. D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, D. Hassabis, Mastering the game of Go with deep neural networks and tree search. *Nature* **529**, 484–489 (2016).
13. D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, D. Hassabis, Mastering the game of Go without human knowledge. *Nature* **550**, 354–359 (2017).
14. M. E. Taylor, P. Stone, Transfer learning for reinforcement learning domains: A survey. *J. Mach. Learn. Res.* **10**, 1633–1685 (2009).
15. K. Doya, Reinforcement learning in continuous time and space. *Neural Comput.* **12**, 219–245 (2000).
16. X. B. Peng, M. Andrychowicz, W. Zaremba, P. Abbeel, Sim-to-real transfer of robotic control with dynamics randomization, in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2018), pp. 1–8.
17. J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, M. Hutter, Learning agile and dynamic motor skills for legged robots. *Sci. Robot.* **4**, eaau5872 (2019).
18. P. Leinen, M. Esders, K. T. Schütt, C. Wagner, K.-R. Müller, F. S. Tautz, Autonomous robotic nanofabrication with reinforcement learning. *Sci. Adv.* **6**, eabb6987 (2020).
19. N. Murata, K.-R. Muller, A. Ziehe, S.-i. Amari, Adaptive on-line learning in changing environments, in *Proceedings of the Advances in Neural Information Processing Systems* (MIT Press, 1997), pp. 599–605.
20. K. Arndt, M. Hazara, A. Ghadirzadeh, V. Kyrki, Meta reinforcement learning for sim-to-real domain adaptation. arXiv:1909.12906 [cs.CV] (16 September 2019).
21. X. Song, Y. Yang, K. Choromanski, K. Caluwaerts, W. Gao, C. Finn, J. Tan, Rapidly adaptable legged robots via evolutionary meta-learning. arXiv:2003.01239 [cs.RO] (2 March 2020).

22. A. B. L. B. Rouhollah Rahmatizadeh, Pooya Abolghasemi, Meta reinforcement learning for sim-to-real domain adaptation, in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence* (AAAI Press, 2018), pp. 6524–6531.
23. L. Tai, G. Paolo, M. Liu, Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation, in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2017), pp. 31–36.
24. J. Nakanishi, J. Morimoto, G. Endo, G. Cheng, S. Schaal, M. Kawato, Learning from demonstration and adaptation of biped locomotion. *Robot. Auton. Syst.* **47**, 79–91 (2004).
25. T. Matsubara, J. Morimoto, J. Nakanishi, M. Sato, K. Doya, Learning CPG-based biped locomotion with a policy gradient method. *Robot. Auton. Syst.* **54**, 911–920 (2006).
26. M. Riedmiller, T. Gabel, R. Hafner, S. Lange, Reinforcement learning for robot soccer. *Auton. Robots* **27**, 55–73 (2009).
27. P. Stone, R. S. Sutton, G. Kuhlmann, Reinforcement learning for RoboCup soccer keepaway. *Adaptive Behav.* **13**, 165–188 (2005).
28. O. M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. M. Grew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, W. Zaremba, Learning dexterous in-hand manipulation. *Int. J. Robot. Res.* **39**, 3–20 (2020).
29. K. J. Kostuk, K. A. Willoughby, A. P. Saedt, Modelling curling as a Markov process. *Eur. J. Oper. Res.* **133**, 557–565 (2001).
30. N. W. Stewart, C. Hall, The effects of cognitive general imagery use on decision accuracy and speed in curling. *Sport Psychol.* **30**, 305–313 (2016).
31. T. Ito, Y. Kitasei, Proposal and implementation of "digital curling", in *Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG)* (IEEE, 2015), pp. 469–473.
32. K. Lee, S.-A. Kim, J. Choi, S.-W. Lee, Deep reinforcement learning in continuous action spaces: A case study in the game of simulated curling, in *Proceedings of the Thirty-Fifth International Conference on Machine Learning (ICML)* (PMLR, 2018), pp. 2943–2952.
33. D.-O. Won, B.-D. Kim, H.-J. Kim, T.-S. Eom, K.-R. Müller, S.-W. Lee, Curly: An AI-based curling robot successfully competing in the olympic discipline of curling, in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI)* (AAAI Press, 2018), pp. 5883–5885.
34. K. Ohto, T. Tanaka, A curling agent based on the Monte-Carlo tree search considering the similarity of the best action among similar states, in *Proceedings of the Fifteenth International Conference on Advances in Computer Games* (Springer, 2017), pp. 151–164.
35. M. Denny, Curling rock dynamics: Towards a realistic model. *Can. J. Phys.* **80**, 1005–1014 (2002).
36. H. Nyberg, S. Alfredson, S. Hogmark, S. Jacobson, The asymmetrical friction mechanism that puts the curl in the curling stone. *Wear* **301**, 583–589 (2013).
37. R. A. Fisher, *Statistical Methods for Research Workers* (Springer, 1934).
38. J. Kaufmann, A. G. Schering, Analysis of variance ANOVA (John Wiley & Sons Inc., 2014).
39. L. Rozo, S. Calinon, D. G. Caldwell, P. Jimenez, C. Torras, Learning physical collaborative robot behaviors from human demonstrations. *IEEE Trans. Robot.* **32**, 513–527 (2016).
40. A. Bauer, D. Wollherr, M. Buss, Human–robot collaboration: A survey. *Int. J. Human. Robot.* **5**, 47–66 (2008).
41. V. Mnih, A. Puigdomènech, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu, Asynchronous methods for deep reinforcement learning, in *Proceedings of the Thirty-Third International Conference on Machine Learning (ICML)* (PMLR, 2016), pp. 1928–1937.
42. S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE* **10**, e0130140 (2015).
43. M. T. Ribeiro, S. Singh, C. Guestrin, "Why should I trust you?" explaining the predictions of any classifier. in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Association for Computing Machinery, 2016), pp. 1135–1144.
44. W. Samek, G. Montavon, A. Vedaldi, L.-K. Hansen, K.-R. Müller, *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning* (Springer Nature, 2019), vol. 11700.
45. S. Lapuschkin, S. Wäldchen, A. Binder, G. Montavon, W. Samek, K.-R. Müller, Unmasking clever hans predictors and assessing what machines really learn. *Nat. Commun.* **10**, 1096 (2019).
46. R. S. Sutton, D. A. McAllester, S. P. Singh, Y. Mansour, Policy gradient methods for reinforcement learning with function approximation, in *Proceedings of the Advances in Neural Information Processing Systems (NIPS)* (MIT Press, 1999), pp. 1057–1063.
47. R. S. Sutton, A. G. Barto, *Introduction to Reinforcement Learning* (MIT Press, 1998), vol. 135.
48. R. S. Sutton, A. G. Barto, *Reinforcement Learning: An Introduction* (MIT Press, 2018).
49. K. Arulkumaran, M. P. Deisenroth, M. Brundage, A. A. Bharath, Deep reinforcement learning: A brief survey. *IEEE Signal Process. Mag.* **34**, 26–38 (2017).
50. V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, J. Pineau, An introduction to deep reinforcement learning. *Found. Trends Mach. Learn.* **11**, 219–354 (2018).
51. S. James, A. J. Davison, E. Johns, Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task. arXiv:1707.02267 [cs.RO] (7 July 2017).
52. K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, 2016), pp. 770–778.
53. Y. Hori, Future vehicle driven by electricity and control-research on four wheel motored "UOT electric march II", in *Proceedings of the 7th International Workshop on Advanced Motion Control* (IEEE, 2002), pp. 1–14.
54. H. Fujimoto, T. Saito, T. Noguchi, Motion stabilization control of electric vehicle under snowy conditions based on yaw-moment observer, in *Proceedings of the 8th IEEE International Workshop on Advanced Motion Control (AMC)* (IEEE, 2004), pp. 35–40.
55. J. H. Choi, C. Song, K. Kim, S. Oh, Development of stone throwing robot and high precision driving control for curling, in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2018), pp. 2434–2440.
56. A. R. Penner, The physics of sliding cylinders and curling rocks. *Am. J. Phys.* **69**, 332–339 (2001).
57. M. Kearns, Y. Mansour, A. Y. Ng, A sparse sampling algorithm for near-optimal planning in large Markov decision processes. *Mach. Learn.* **49**, 193–208 (2002).
58. S. Verma, G. Novati, P. Koumoutsakos, Efficient collective swimming by harnessing vortices through deep reinforcement learning. *Proc. Natl. Acad. Sci. U.S.A.* **115**, 5849–5854 (2018).
59. H. van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double Q-learning, in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence* (AAAI Press, 2016), pp. 2094–2100.
60. I. Clavera, D. Held, P. Abbeel, Policy transfer via modularity and reward guiding, in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2017), pp. 1537–1544.
61. M. Hausknecht, P. Stone, Deep recurrent Q-learning for partially observable MDPs. arXiv:1507.06527 [cs.LG] (23 July 2015).
62. G. Lample, D. S. Chaplot, Playing FPS games with deep reinforcement learning, in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence* (AAAI Press, 2017), pp. 2140–2146.
63. M. Jaderberg, W. M. Czarnecki, I. Dunning, L. Marris, G. Lever, A. G. Castaneda, C. Beattie, N. C. Rabinowitz, A. S. Morcos, A. Ruderman, N. Sonnerat, T. Green, L. Deason, J. Z. Leibo, D. Silver, D. Hassabis, K. Kavukcuoglu, T. Graepel, Human-level performance in first-person multiplayer games with population-based deep reinforcement learning. arXiv:1807.01281 [cs.LG] (3 July 2018).
64. J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms. arXiv:1707.06347 [cs.LG] (20 July 2017).
65. R. J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.* **8**, 229–256 (1992).
66. E. P. Lozowski, K. Szilder, S. Maw, A. Morris, L. Poirier, B. Kleiner, Towards a first principles model of curling ice friction and curling stone dynamics, in *Proceedings of the Twenty-fifth International Ocean and Polar Engineering Conference* (International Society of Offshore and Polar Engineers, 2015).
67. N. Maeno, Dynamics and curl ratio of a curling stone. *Sports Eng.* **17**, 33–41 (2014).
68. V. Honkanen, M. Ovaska, M. J. Alava, L. Laurson, A. J. Tuononen, A surface topography analysis of the curling stone curl mechanism. *Sci. Rep.* **8**, 8123 (2018).
69. M. R. A. Shegelski, R. Niebergall, M. A. Walton, The motion of a curling rock. *Can. J. Phys.* **74**, 663–670 (1996).
70. E. T. Jensen, M. R. A. Shegelski, The motion of curling rocks: Experimental investigation and semi-phenomenological description. *Can. J. Phys.* **82**, 791–809 (2004).
71. M.-H. Heo, D. Kim, The development of a curling simulation for performance improvement based on a physics engine. *Procedia Eng.* **60**, 385–390 (2013).
72. I. Parberry, *Introduction to Game Physics with Box2D* (CRC Press, 2017).
73. S. Jackson, *Unity 3D UI Essentials* (Packt Publishing Ltd., 2015).
74. J. W. Johnston, The dynamics of a curling stone. *Can. Aeronaut. Space J.* **27**, 144–161 (1981).
75. M. Yamamoto, S. Kato, H. Iizuka, Digital curling strategy based on game tree search, in *Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG)* (IEEE, 2015), pp. 474–480.
76. E. A. Nadaraya, On estimating regression. *Theory Probab. Its Appl.* **9**, 141–142 (1964).
77. G. S. Watson, Smooth regression analysis. *Sankhyā Ser. A*, 359–372 (1964).
78. L. Kocsis, C. Szepesvári, Bandit based Monte-Carlo planning, in *Proceedings of European Conference on Machine Learning* (Springer, 2006), pp. 282–293.
79. R. Coulom, Efficient selectivity and backup operators in Monte-Carlo tree search, in *International Conference on Computer and Games* (Springer, 2006), pp. 72–83.

Acknowledgments: We appreciate the support from our AI curling robot project consortium (NT Robot, Yeungnam University, DGIST, UNIST, Manongdanto, DOSANET, Seoul Curling

Federation) for this research. We would like to thank M. Esders for valuable comments on the manuscript. **Funding:** This work was supported by the Institute for Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (nos. 2017-0-00521, 2017-0-01779, and 2019-0-00079) and partially funded by Microsoft Research Asia, and K.-R.M. was partly supported by the German Ministry for Education and Research (BMBF) under grants 01IS14013A-E, 01GQ1115, 01GQ0850, 01IS18025A, and 01IS18037A. This study was also supported by the German Research Foundation (DFG) under Grant Math+, EXC 2046/1, project ID 390685689. **Author contributions:** D.-O.W. and S.-W.L. designed curling AI, the experiment, and system; D.-O.W., K.-R.M., and S.-W.L. performed research; D.-O.W., and K.-R.M. contributed to data analysis; and D.-O.W., K.-R.M., and S.-W.L. wrote the paper. **Competing interests:** D.-O.W. and S.-W.L. are inventors on patent applications (Korean patent nos. 10-2045567, 10-2045566, and 10-2071782) held/submitted by Korea University that cover the AI curling robot. **Data and materials availability:** All data needed to evaluate the conclusions are present in the paper or Supplementary Materials.

Submitted 31 March 2020
Accepted 18 August 2020
Published 23 September 2020
10.1126/scirobotics.abb9764

Citation: D.-O. Won, K.-R. Müller, S.-W. Lee, An adaptive deep reinforcement learning framework enables curling robots with human-like performance in real-world conditions. *Sci Robot.* **5**, eabb9764 (2020).

Science Robotics

An adaptive deep reinforcement learning framework enables curling robots with human-like performance in real-world conditions

Dong-Ok Won, Klaus-Robert Müller, and Seong-Whan Lee

Sci. Robot. **5** (46), eabb9764. DOI: 10.1126/scirobotics.abb9764

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