APPLIED SCIENCES AND ENGINEERING

Autonomous robotic nanofabrication with reinforcement learning

Philipp Leinen[1,2,3]*, Malte Esders[4]*, Kristof T. Schütt[4], Christian Wagner[1,2†], Klaus-Robert Müller[4,5,6†], F. Stefan Tautz[1,2,3]

The ability to handle single molecules as effectively as macroscopic building blocks would enable the construction of complex supramolecular structures inaccessible to self-assembly. The fundamental challenges obstructing this goal are the uncontrolled variability and poor observability of atomic-scale conformations. Here, we present a strategy to work around both obstacles and demonstrate autonomous robotic nanofabrication by manipulating single molecules. Our approach uses reinforcement learning (RL), which finds solution strategies even in the face of large uncertainty and sparse feedback. We demonstrate the potential of our RL approach by removing molecules autonomously with a scanning probe microscope from a supramolecular structure. Our RL agent reaches an excellent performance, enabling us to automate a task that previously had to be performed by a human. We anticipate that our work opens the way toward autonomous agents for the robotic construction of functional supramolecular structures with speed, precision, and perseverance beyond our current capabilities.

INTRODUCTION

The swift development of quantum technologies could be further advanced if we managed to free ourselves from the imperatives of crystal growth and self-assembly and learned to fabricate custom-built metastable structures on atomic and molecular length scales routinely (*1*–*7*). Metastable structures, apart from being more abundant than stable ones, tend to offer attractive functionalities, because their constituent building blocks can be arranged more freely and in particular in desired functional relationships (*7*).

It is well established that single molecules can be manipulated and arranged using mechanical, optical, or magnetic actuators (*8*), such as the tips of scanning probe microscopes (SPMs) (*9*–*12*) or optical tweezers (*13*, *14*). With all these types of actuators, a sequence of manipulation steps can be carried out to bring a system of molecular building blocks into a desired target state. The problem of creating custom-built structures from single molecules can therefore be cast as a challenge in robotics.

In the macroscopic world, robots are typically steered using either human expert knowledge or model-based approaches (*15*–*18*). Both strategies are not available at the nanoscale, because, on the one hand, human intuition is largely trained on concepts like inertia and gravity, which do not apply here, while, on the other hand, atomistic simulations are either too imprecise to be helpful or computationally too expensive to generate the large amount of sufficiently accurate data required for training. This is aggravated by the fact that actuators have variable and typically unknown structures and properties at the nanometer scale, making it extremely difficult, if not impossible, both to cover all relevant configurations of the actuator in the simulation and to establish a connection between the actual robotic process and the simulation. This leaves autonomous robotic nanofabrication as the preferred option.

In the current study, we show that reinforcement learning (RL) can be used to automate a manipulation task at the nanoscale. In RL, a software agent is placed in an environment at time $t = 0$ and sequentially performs actions to alter the state s_t of this environment (*19*, *20*). While executing random actions in the beginning, the agent will, based on its accumulated experience, incrementally learn a policy π for choosing actions a_t that maximize a sum over reward signals r_{t+1}. The reward signal is returned by the environment in a manner specified by the experimenter beforehand. The experimenter designs the reward signal such that behavior that solves the problem yields a high reward, whereas failure to do so yields a low reward. The advantage of this approach is that the experimenter does not have to specify how the agent needs to act, but instead only has to define the desired outcome.

Considering a compact layer of PTCDA (3,4,9,10-perylene-tetracarboxylic dianhydride) on an Ag(111) surface, we define removing individual molecules from this layer using an SPM as the RL agent's goal (see Fig. 1). This task is an example of a subtractive manufacturing process that starts from a self-assembled molecular structure. We note that subtractive manufacturing has been identified as key to nanoscale fabrication (*21*, *22*).

RESULTS

Robotic nanofabrication as an RL challenge

An RL task is usually modeled as a Markov decision process (*19*), which is a Markov process equipped with an agent that can perform certain actions at each state to influence the transition into the next state. In nanofabrication, a complete numerical representation of the environment state $\bar{s}_t$ would consist of the coordinates of all relevant atoms in the environment. The probability distribution $p(\bar{s}_{t+1}, r_{t+1} \mid \bar{s}_t, a_t)$, which defines the probability to reach state $\bar{s}_{t+1}$ and receive reward r_{t+1} after taking action a_t in state $\bar{s}_t$, is deterministic at low temperatures (in our example, at $T = 5$ K) and stochastic at temperatures where thermal fluctuations are enabled (*13*). The

[1]Peter Grünberg Institut (PGI-3), Forschungszentrum Jülich, 52425 Jülich, Germany. [2]Jülich Aachen Research Alliance (JARA)–Fundamentals of Future Information Technology, 52425 Jülich, Germany. [3]Experimentalphysik IV A, RWTH Aachen University, Otto-Blumenthal-Straße, 52074 Aachen, Germany. [4]Machine Learning Group, Technische Universität Berlin, 10587 Berlin, Germany. [5]Max Planck Institute for Informatics, 66123 Saarbrücken, Germany. [6]Department of Brain and Cognitive Engineering, Korea University, Seoul, South Korea.
*These authors contributed equally to this work.
†Corresponding author. Email: c.wagner@fz-juelich.de (C.W.); klaus-robert.mueller@tu-berlin.de (K.-R.M.)

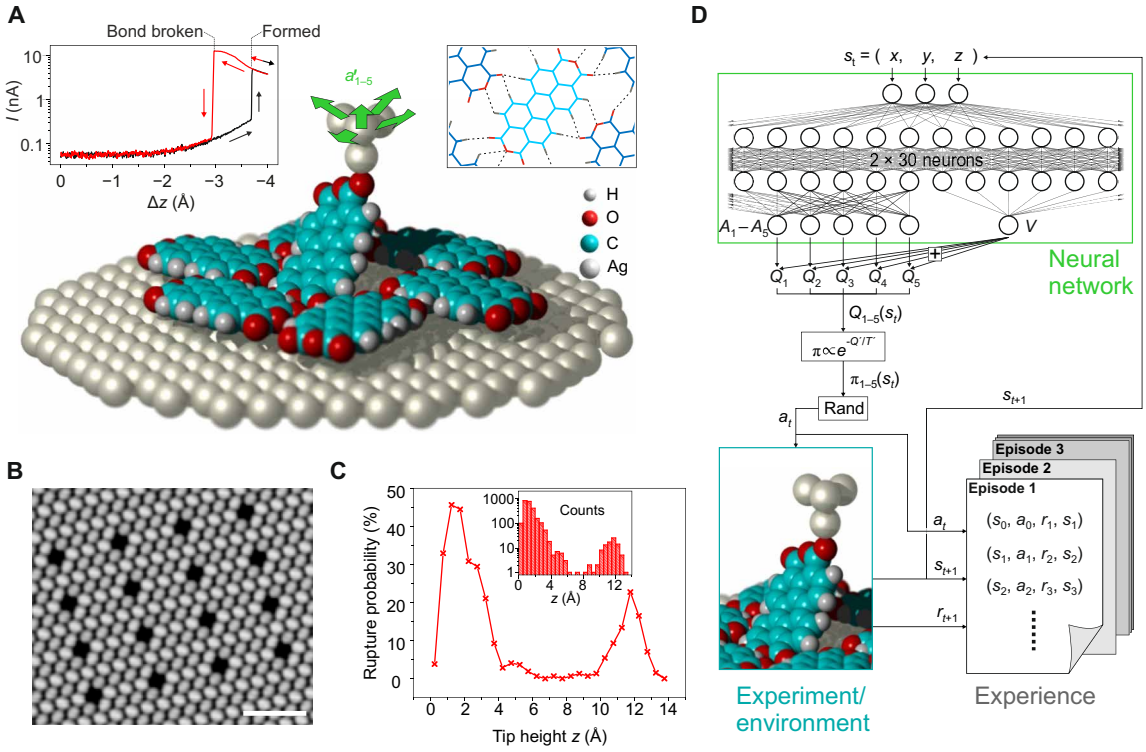

Fig. 1. Subtractive manufacturing with an RL agent. (**A**) PTCDA molecules can spontaneously bind to the SPM tip and be removed from a monolayer upon tip retraction on a suitable trajectory. Bond formation and breaking cause strong increases or decreases in the tunneling current (left inset). The removal task is challenging, because PTCDA is retained in the layer by a network of hydrogen bonds (dotted lines in right inset). The RL agent can repeatedly choose from the five indicated actions a'_{1-5} (green arrows) to find a suitable trajectory (action set $\mathcal{A}$: $\Delta z = 0.1$ Å step plus ±0.3-Å step in the x or y direction, or no lateral movement). (**B**) STM image of a PTCDA layer with 16 vacancies created by the RL agent (scale bar, 5 nm). (**C**) Probability of bond rupture in intervals of 0.5 Å around tip height z as a function of z, based on all bond-breaking events accumulated during the RL agent experiments (inset). (**D**) The Q function is approximated by a neural network with 30 neurons in the first and 2×15 neurons in the second hidden layer. This dueling network architecture (*39*) features separate outputs A_i and V, with $Q_i = V + A_i$ for actions $a'_{i=1\ldots5}$. The actually performed action is then randomly chosen from $\mathcal{A}$ with probabilities computed with the policy π.

complete state $\bar{s}_t$ of the environment is generally not observable at the current level of technology (*13*). To apply RL to nanofabrication, we therefore suggest using an approximate state definition s_t. There are several plausible elements to construct such a definition. First, there is the known actuator position. Second, there are typically some measurable quantities (like forces) in any robotic nanofabrication setup, which are functions of the complete state $\bar{s}_t$ of the environment and which could thus be used to approximate this state.

Any approximate state definition will be of much lower dimensionality than the complete state definition, such that transitions $\bar{s}_t \rightarrow \bar{s}_{t+1}$ in the complete state space cannot be captured fully by transitions $s_t \rightarrow s_{t+1}$ in the approximate state space. Hence, two states s_t and s'_t could be identical even when the underlying complete states $\bar{s}_t$ and $\bar{s}'_t$ are not. Using an approximate state description has several consequences: First, it could break the Markov property, because the history $s_0, \ldots, s_t$ could provide more information about $\bar{s}_t$ than s_t alone. Second, the problem could become effectively nonstationary because a change in the actuator (i.e., in the arrangement of its atoms) could change the entire probability distribution $p(s_{t+1}, r_{t+1} \mid s_t, a_t)$, without the approximate state definition being capable of capturing these changes. This could render the accumulated experience at least partially worthless. An additional source of nonstationarity of nanofabrication systems is parameters of the (external) macroscopic environment (the apparatus, the room, etc.), which are also varying slowly.

This nonstationarity is at the heart of the difficulty of autonomous nanofabrication, because it means that the successful policy is not static but must be evolved constantly. Furthermore, the speed at which a policy is learned needs to be faster than the rate at which $p(s_{t+1}, r_{t+1} \mid s_t, a_t)$ changes. In practice, this requires a substantial speedup of the standard RL algorithm, which is typically very slow because it needs a lot of training data. If this key advance was achieved, a policy $\pi(s_t)$ could be learned in the lifetime of the distribution $p(s_{t+1}, r_{t+1} \mid s_t, a_t)$. Moreover, the intrinsically adaptive RL agent would be able to deal with occasional hidden changes of $\bar{s}_t$. Below, we will demonstrate that RL can be sped up sufficiently to solve our exemplary nanofabrication task.

PTCDA lifting task as an RL challenge

We have previously studied the removal of single PTCDA molecules from condensed layers on Ag(111) by manual tip control (*21*, *23*–*27*). Using motion capture and virtual reality, we were able to identify specific three-dimensional tip trajectories that reach the target state in which the molecule is fully disconnected from the surface but still bonded to the tip. We stress that the bond between one of the carboxylic oxygen atoms and the tip (Fig. 1A) typically ruptures if a random retraction trajectory is chosen, because along most trajectories, the molecule-surface and intermolecular forces together exceed the strength of the tip-molecule bond. Thus, to be successful, the RL agent has to find trajectories on which the retaining force, which

holds the molecule in the layer, never surpasses the tip-molecule bond strength.

In our example, neither the atomic coordinates of the object system (PTCDA layer and manipulated PTCDA molecule) nor the atomic structure of the rest of the environment (SPM tip apex) is known precisely. For the definition of the approximate state s_t, we could exploit two measurable quantities: the tunneling current and the force gradient of the SPM. We tried to use these quantities together with the Cartesian coordinates of the tip apex as the state description of the environment (five dimensions), but we had to conclude that, given the limited time until the task needs to be solved, the measured quantities have too high a variance to be helpful in our RL setup (see Materials and Methods). We therefore chose to substantially reduce the state description and only include the Cartesian coordinates of the tip (three dimensions) in the state definition. Because the manipulated PTCDA molecule could assume different conformations at identical SPM tip positions if different tip trajectories led to this position, this approximate state definition is not guaranteed to have the Markov property. Because the given nanofabrication task could nevertheless be solved successfully, we chose to keep the complexity in our proof-of-concept study as low as possible and did not use strategies to attempt to restore the Markov property [see, e.g., (*19*), chap. 17].

In the PTCDA lifting task, the nonstationarity of $p(s_{t+1}, r_{t+1} \mid s_t, a_t)$ discussed above arises, for example, from abrupt changes in the atomic structure of the tip apex and from a slow drift of the piezo tubes controlling the SPM tip. Such changes influence both the measurable quantities and the trajectories on which it is possible to lift the molecule and thus also the rewards r_{t+1} in the distribution $p(s_{t+1}, r_{t+1} \mid s_t, a_t)$.

It should be noted that despite the nonobservability of the complete state, two important key events can be unambiguously detected: the undesired loss of contact to the tip and the desired loss of contact to the surface. In the former event, the tunneling current drops by at least an order of magnitude (Fig. 1A), which is well outside the range in which the current varies while the bond is still in place [see (*28*) for the case of an isolated molecule]. In addition, in both events, a clear signal can be observed in the SPM force gradient channel.

Formal RL setup

The RL agent steers the SPM with its actions. The environment is the actuators of the SPM and the object system. At time step t, the environment is in a state that we represent numerically by $s_t \in \mathcal{S}$. As noted above, we simplify the state representation to $\mathcal{S} \subset \mathbb{R}^3$, and the three components are the Cartesian coordinates of the tip apex. On the basis of the received state, the agent picks an action a_t from the set of actions $\mathcal{A}$. We specify $\mathcal{A}$ to consist of five possible actions, all of which move the tip in different directions (see Fig. 1A). The performed action, in turn, causes the environment to emit a new state s_{t+1} and also a reward $r_{t+1} \in \mathbb{R}$.

We design the reward system as follows: If the environment transitions to a nonterminal state, we assign a default reward of $r_{t+1} = 0.01$ (see Materials and Methods for a discussion). If transitioning into a state in which the SPM tip loses contact with the molecule, the agent is penalized with $r_{t+1} = -1$, and the current episode stops. Last, if transitioning into a state where the molecule has been lifted successfully, we assign a reward of $r_{t+1} = +1$, and the episode also stops. After each failed episode, the molecule, by virtue of the hydrogen bonds (Fig. 1), drops back to its original position in the PTCDA layer, and the tip is moved back to $s_0 = (0,0,0)$, where the tip-molecule bond reestablishes such that the next episode can start with identical conditions (provided that no change in the tip apex has occurred).

Central to RL is the Q-value function, which is learned (Fig. 2C) and which, in our case, is approximated by a neural network (NN) (Fig. 1D). $Q(s_t, a_t)$ is the agent's estimate of the expected discounted future reward $G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} r_k$ it will receive when performing action a_t in state s_t and afterward following its policy π. In a given state, the policy π assigns action-selection probabilities to each action depending on their respective Q values. In our case, π is computed from $Q' = -Q$ using the Boltzmann distribution

$$\pi\left(s_t, a_t, T\right) = \exp\left(-\frac{Q'(s_t,a_t)}{T}\right) \Big/ \sum_{a \in \mathcal{A}} \exp\left(-\frac{Q'(s_t,a)}{T}\right) \quad (1)$$

As is common in RL, Q appears with opposite sign in this equation, because a high Q means a high probability, opposite to the energy/occupation relation for which this distribution was initially derived. The "temperature" parameter T determines how greedily the agent chooses actions having higher Q values.

The interaction with the environment is organized into state-action-reward-state tuples $(s_t, a_t, r_{t+1}, s_{t+1})$ and stored in an experience memory to be used for training (Fig. 1D). During training, the Q values predicted by the NN are adjusted to the discounted future rewards (Fig. 2C). We use an off-policy variant (see below) of the Expected SARSA (state-action-reward-state-action) (*29*) algorithm, for which the loss is computed as

$$L(s_t, a_t) = \left(Q(s_t, a_t) - \left(r_{t+1} + \gamma \sum_{a \in \mathcal{A}} \pi(s_{t+1}, a)\, Q(s_{t+1}, a) \right) \right)^2 \quad (2)$$

and used to optimize the NN weights via gradient descent with samples obtained by prioritized sampling (*30*) from the experience memory. Note that the discounted ($\gamma = 0.97$) future reward at $t + 1$ is given by the Q-value function itself. This recursive formulation, called temporal difference learning, allows one to learn Q values particularly efficiently and propagate them through the state space (*19*).

Simulation results and RL adjustments

Before connecting the RL agent to the microscope, we benchmarked our RL setup on a simulated system using synthetic bond-breaking criteria (Fig. 2A) derived from prior lifting experiments (*21*). Note that the probability of bond rupture as a function of tip height is similar between the simulation and the real experiment (Figs. 1C and 2B). Specifically, there are two heights at which there is an increased chance of rupture in the experiment, and our synthetic bond-breaking criteria recover this pattern. Even in this stable simulation with no uncontrolled variability (and complete observability), the agent typically requires more than 150 episodes to find a successful policy (Fig. 2D). As discussed above, this low data efficiency would make it (almost) impossible to achieve the goal in the real-world experiment, where we expect a substantial degree of variability over this time scale of hundreds of episodes, rendering much of the collected experience worthless.

Driven by the need to solve the task more efficiently, we introduce two modifications to the standard Expected SARSA algorithm: First, we make use of our purely Cartesian coordinate state description to perform model-based RL similar to the Dyna algorithm (*31*).

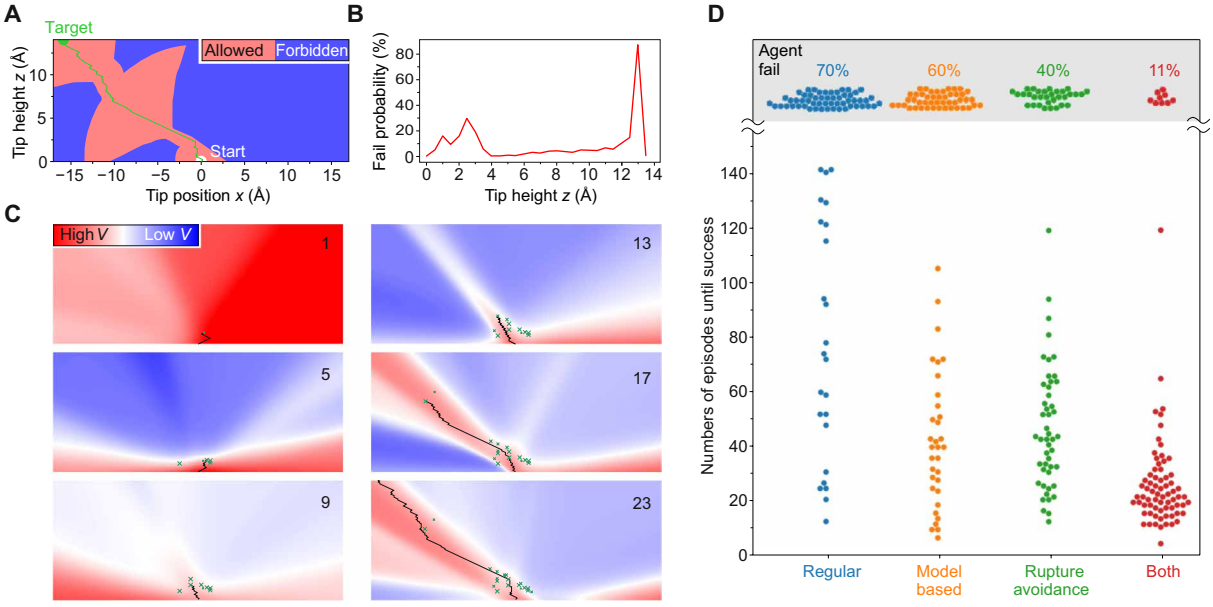

Fig. 2. Training and performance of RL agents. (**A**) Map [two-dimensional (2D) slice through the 3D system] of the synthetic bond rupture criteria used to study the RL agent's behavior under controlled conditions. The criteria are based on a successful experimental trajectory around which a corridor of variable diameter has been created (light red) beyond which the bond ruptures (blue). The corridor diameter is chosen to approximately reproduce the experimental bond-rupture probabilities (Fig. 1C). One successful trajectory [see (C)] is indicated in green. (**B**) Probability of agent failure in *z* intervals of 0.5 Å in the simulation in (A). (**C**) Learning progress of one RL agent. The six plots show 2D cuts ($y = 0$) through the color-encoded value function *V* after the number of episodes indicated in the upper right corner. A 2D projection of the agent's trajectory in each episode is shown as a black line. Crosses indicate bond-breaking events triggered according to the criteria in (A) (see Supplementary Animation for a 3D view). (**D**) Swarm plot comparing the performance of different types of RL agents acting in the simulation (A). Plotted is the number of episodes *n* required to accomplish the removal task for four sets of 80 simulated experiments each. An experiment was considered a failure after 150 unsuccessful episodes. The respective probabilities of agent failure are indicated in the upper part of the graph.

Dyna uses both actual experience from interaction with the environment and experience obtained from a learned environment model to update the *Q* values. In our case, learning an environment model is easy, because the state transition from s_t given a_t to s_{t+1} is deterministic. The environment model also needs to model the obtained reward. We implement our learned environment model such that it emits the default reward $r_{t+1} = 0.01$, unless the successor state s_{t+1} is a known failure state (bond previously ruptured at this position in the experiment), in which case it emits the failure-reward $r_{t+1} = -1.0$. We use this environment model to sample (s_t, a_t, r_{t+1}, s_{t+1}) tuples around states obtained from prior experience (see Materials and Methods) and train our NN with them. Second, we introduce a rupture avoidance mechanism by setting a negative temperature $T_{\text{train}} < 0$ in Eq. 1 during training. Using a negative temperature gives lower *Q* values at time step $t + 1$ in Eq. 2 more importance. Therefore, information about impending failure states is propagated much further toward previous positions, and the agent can use this information to avoid them. Of course, while acting in the environment, we still set a positive temperature $T_{\text{act}} > 0$.

We next benchmark the performance of the RL setup with our two modifications in the simulation. Figure 2D shows that especially in combination, the two modifications speed up the learning process markedly, to the extent that it now becomes possible to connect this modified RL agent to the real-world experiment.

SPM setup

While the RL agent controls our ultrahigh-vacuum low-temperature noncontact atomic force microscope/scanning tunneling microscope (STM) fitted with a qPlus tuning-fork sensor (*32*), the tunneling current through the junction ($V_{\text{bias}} = 10$ mV) is continuously monitored by our software (see Materials and Methods). When the bond to the tip ruptures, the increased tunneling barrier leads to a sudden drop in current, and the failure-reward is assigned. Because the length of the molecule is known (11.5 Å), the target state can also be automatically detected as any state with $z > 14$ Å, and the contact to the molecule still in place. Thus, the agent works autonomously until the point where the final success of the manipulation as reported by the agent is verified by the experimenter, who deposits the molecule from the tip elsewhere onto the Ag(111) surface (*21*) and images the vacancy that is created in the PTCDA layer (Fig. 1B). Tip changes that occasionally occurred either during an episode or when depositing the molecule back onto the surface have been identified by checking changes in the STM image contrast or position and in the tip-molecule bonding behavior.

DISCUSSION

Analysis of the learning process

We measure the performance of each RL agent by the number *n* of episodes that it requires to solve the removal task. To separate intrinsic RL stochasticity from the uncontrollable variability of the SPM manipulation, we plot in Fig. 3A all data points *n* of real-world experiments that were conducted with identical tips in the same color. The scatter is indeed smaller within groups of experiments with identical tips. Moreover, the difficulty of the removal task clearly depends on the tip. For example, with tip D, removal is easy, resulting, on average, in small *n*, while tip E, with which removal appears more difficult, results in larger *n*.

A small force threshold of the tip-molecule bond reduces the fraction of successful trajectories. For particularly weak tips (labeled tip failure in Fig. 3A), the removal task cannot be solved at all. One would expect that for larger n, i.e., weaker tips, successful trajectories cluster more narrowly in space. Figure 3B, in which we compare the (x, y) coordinates of successful trajectories at $z = 2$ Å for tip D and tip E, clearly confirms this effect. The distribution of the corresponding (x, y) coordinates for all tips (plotted as a grayscale background) is rather broad and indeed similar to the distribution for the strong tip D. For the weak tip E, however, the agent has to traverse a very specific region in the xy plane to avoid bond breaking. This naturally explains why tip E tends to require larger numbers n of episodes until success.

In Fig. 3A, we also compare randomly initialized (R) agents with pretrained (P) ones. R-agents start with random weights in the NN, while P-agents have already solved the removal task once with one particular tip. Initially, all P-agents are identical; i.e., they have the same experience and NN weights. On average, P-agents perform better than R-agents. This is evident both in the complete dataset and for individual tips (see, for instance, tip E). It clearly demonstrates that at least some knowledge about the removal task is universal and can be transferred to new tip configurations.

Figure 3C reveals the nature of this universal knowledge. In this figure, we have plotted the (x, y) coordinates of the rupture points (i.e., termini of unsuccessful episodes) for R-agents (black) and P-agents (red). The data are limited to the first 10 episodes of each experiment, in which the difference in training and experience between both types of agents is strongest. The plot shows that the randomly initialized agents explore all (x, y) directions rather uniformly, while the pretrained agents have a strong bias toward the lower left quadrant ($x < 0, y < 0$) through which almost all successful trajectories pass (Fig. 3B). This bias is the essence of the universally valid policy that, once learned, gives P-agents a performance edge over R-agents. We note that this policy is consistent with our general understanding that molecules have to be "peeled" out of the condensed layer along their long axis to break the hydrogen bonds sequentially and limit the associated retaining force (*21*).

Conclusion

Automatically fabricating complex metastable structures at the molecular level is a highly desirable goal. Given the limited observability and uncontrolled variability involved, this goal seemed out of reach until now. In this proof-of-concept study, we demonstrated that indeed autonomous robotic nanofabrication becomes possible

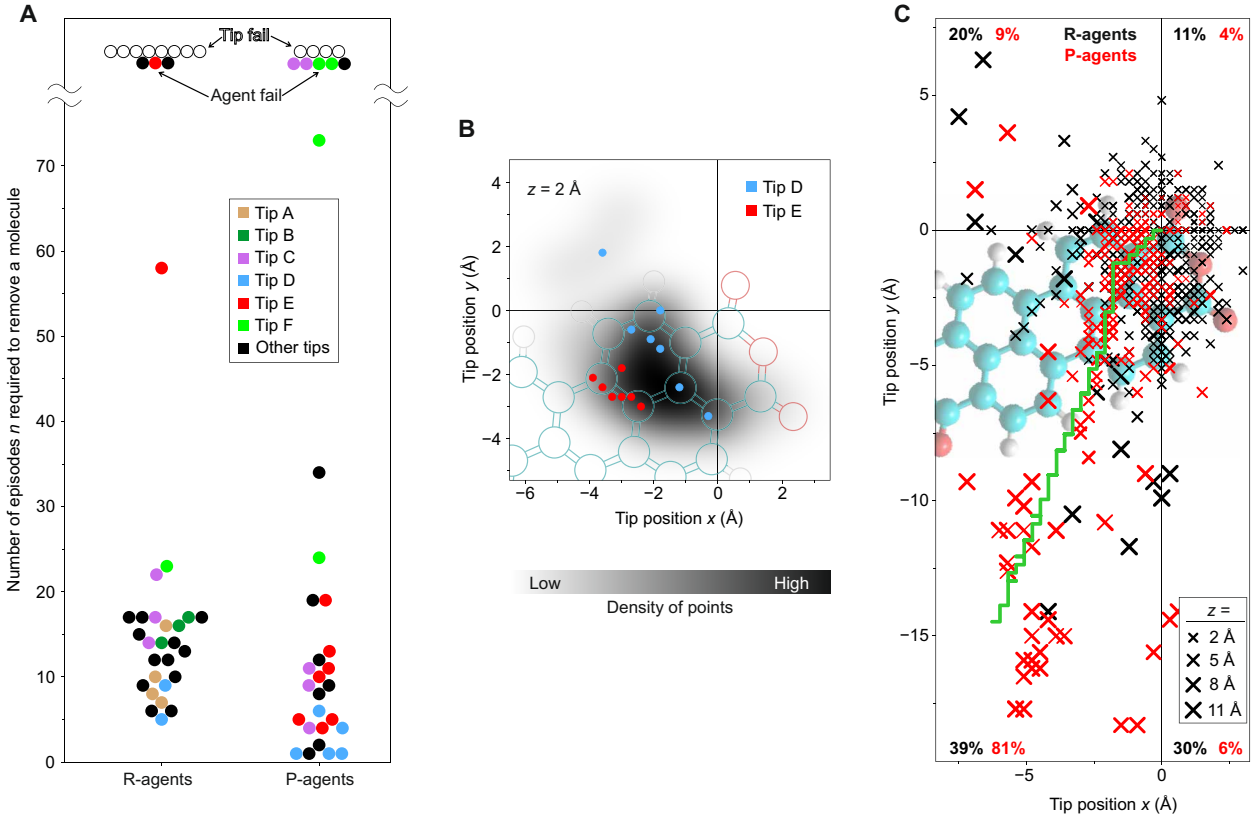

Fig. 3. Performance of the RL agent in experiment. (**A**) Swarm plot of the number of episodes n required to accomplish the removal task. Groups of at least three data points acquired with the same tip are identically colored (except black). If a tip capable of removal (proven by a successful experiment) failed in another experiment, the respective data point is labeled as "agent fail." Points labeled as "tip fail" denote tips with which the removal task has never been accomplished, notwithstanding that this could, in principle, also be a failure of the agent. (**B**) Density of (x, y) positions where all (ultimately successful) tip trajectories pass through the z region of highest bond-rupture probability ($z = 2$ Å; Fig. 1C). The positions for Tip D (strong tip) and Tip E (weak tip) are indicated by dots. (**C**) (x, y) projections of all bond ruptures occurring within the first 10 episodes for R- and P-agents. Cross sizes indicate rupture heights z. The quoted numbers give the percentage of rupture points located in each of the four quadrants of the coordinate system. The green curve shows the last trajectory chosen by the P-agent during its pretraining. Its direction indicates why the P-agents have a clear preference to explore the promising (B) lower left quadrant, which explains their performance edge (A).

using RL without the necessity of human intervention. We chose the real-world task of lifting a molecular structure off a material surface as an example. Because we used the RL framework, it was not necessary to specify how to solve the task—instead, only the goal had to be set, which is clearly easier. We showed not only that an RL agent in full control of an SPM setup is capable of reaching a real manipulation goal with a moderate number of trials but also that it is robust enough to transfer a previously learned policy to new object systems. Furthermore, other nanofabrication tasks, using different molecules, could also benefit from the accelerated learning approaches that we used to enable this performance, namely, negative training temperature and model-based learning.

The limited observability is perhaps the most severe limitation of RL at the nanoscale. Although RL can indeed work under partial observability and in stochastic environments, such conditions negatively affect the number of trials needed to solve a task. To alleviate this problem, one could try using a hybrid approach in which insight gained from atomistic simulations is used for guiding the RL agent in its exploration of possible solutions. While atomistic conformations may not be practically accessible in detail, related measurements like tunneling current and force (gradient) are. Hence, a future research direction could focus on including such helpful variables into an RL setup. Moreover, the combination of autonomous SPM-based nanofabrication with autonomous tip preparation (*33*–*35*) would be a logical development.

In conclusion, we demonstrated that autonomous robotic nanofabrication is viable. It enables immediate progress toward the freedom of designing quantum matter, beyond the constraints of even the most complex quantum materials.

MATERIALS AND METHODS

Experiments

PTCDA is deposited onto Ag(111) at room temperature and briefly annealed to 200°C. The PtIr tip of the qPlus sensor was cut by ion-beam etching and prepared via indentation into the uncovered Ag(111) surface. Because each indentation typically changes the tip apex structure, it affects the strength of the molecule-tip bond and thus the difficulty of the removal task. This allowed us to test the RL agent at various levels of difficulty. The RL agent controls the tip via a voltage source, the output of which is added to the piezo voltages of the SPM.

In principle, the primary criterion to quantify the performance of an agent should be the time the agent requires to accomplish a manipulation task. To assess agents for simulated and real systems on equal footing, we use the number of episodes n required for this task. This quantity n is not fully, but closely related to the time needed (episodes may take longer or shorter depending on the length of the trajectory). Using the wall-clock time as a criterion is moreover rather meaningless, because we have intentionally slowed down the removal process in the experiment to a point where we could carefully observe the actions of the agent to, for example, spot changes in tip apex structure immediately. A removal process took typically 5 to 10 min in the experiment. The tip apex changed in 20% of the removal experiments (not to be confused with episodes), which were excluded from the statistics. During redeposition of the removed molecule onto the surface, the apex changed with a probability of 15%.

Approximate state description

Generally speaking, the Markov property allows a much better theoretical treatment of RL, including, e.g., convergence proofs for the possibility of finding the optimal solution. Without the Markov property, convergence cannot be guaranteed on theoretical grounds. With our reduced state description, distinct complete states could, in principle, result in identical approximate states. In this case, the Markov assumption would be violated, because the past trajectory could be informative about future states. The inclusion of the measured I and Δf would allow discriminating a larger set of the complete states, thus reducing the scope of hidden parameters and associated apparent memory, making the process altogether more Markovian. As a consequence, fabrication tasks that involve processes that display, e.g., a strongly hysteretic behavior would benefit from including I and Δf, simply because the agent could make more informed decisions based on its ability to discriminate the respective complete states better. In principle, this could boost the learning efficiency, as the agent would not receive apparently contradicting information. However, there is also a downside: Generalizing a policy across regions of similar approximate states becomes much harder if it is of higher dimensionality (e.g., five versus three dimensions), especially if I and Δf vary strongly. This hinders the learning process. Our initial RL approaches included I and Δf in the approximate state; however, for the present manipulation task, the disadvantage of increased dimensionality with strongly varying I and Δf outweighed the advantage of being able to better distinguish between underlying complete states.

Reinforcement learning

The pure state transitions $p(s_{t+1} \mid s_t, a_t)$ in our setup are deterministic, because the SPM tip can be moved deterministically to a new position. We use this fact to introduce model-based RL. Note that this determinism is a result of our choice of restricting the state description to the coordinates of the tip. If we included the tunneling current or the force gradient measurements, model-based RL would not be possible anymore, unless one would learn to model these variables as well, which proved too unstable in our pilot experiments.

To prevent overly optimistic estimates of the Q values (*36*), we use the Double Q-learning approach (*37*), which also works with function approximation (*38*). Note that instead of Q-learning, we use Expected SARSA (*29*). While Expected SARSA is an on-policy algorithm, we make it off-policy by using different temperatures T_{train} and T_{act} for the Boltzmann distribution (Eq. 1) during training compared to when using the network to act in the environment. In Double Q-learning, two networks are used in parallel: They start out with equal weights, but in the subsequent training steps, only one network is updated (the "live network"), while the other is held fixed for a while (the "target network"). When computing Q-value estimates for time step $t + 1$ in the loss function (Eq. 2), the target network is used to obtain the actual Q values, while the live network is used for the probabilities of the Boltzmann distribution in the training policy (Eq. 1) and to compute $Q(s_t, a_t)$. Every 200 training steps, the weights of the target network are set to the weights of the live network, such that both networks once again have equal weights.

Rewards

There is a trade-off between sparse and shaped rewards. The former are only given once the agent either accomplishes its final goal or

ultimately fails, while the latter also reward (or punish) intermediate steps, thus directing the agent more efficiently toward its goal. If not chosen well, shaped rewards can induce unwanted agent behavior, while sparse rewards induce no such bias. We use sparse rewards because of the poor observability of the full state (atomic coordinates) of the object system and the resulting lack of information regarding the assessment of intermediate steps. Therefore, we give a reward of +1.0 for success (fully lifting the PTCDA molecule out of the surface) and −1.0 for rupture of the bond between the tip and PTCDA molecule. However, we do slightly shape the reward by giving a reward of +0.01 for each nonterminal step of the agent. Physically, this is motivated by the fact that each step separates the molecule 0.1 Å further from the surface. On the RL side, this small reward makes the agent prefer exploring trajectories on which it previously advanced very far before rupture. This happens because the (discounted) propagation of rewards to previous states during training (Eq. 2) makes states along longer trajectories have a higher value and therefore more "attractive" to the agent.

NN architecture

The NN used to approximate the Q function consists of the three-neuron input layer receiving (x, y, z), followed by a hidden layer with 30 neurons. After this intermediate layer, the network splits up into two separate streams, as inspired by the Dueling Network Architecture (*39*). Both streams consist of a hidden layer with 15 neurons and an output layer with either 1 neuron (V-value stream) or 5 neurons representing the five possible actions (A-value stream) with $Q_i = V + A_i$. The neurons in all hidden layers have tanh nonlinearities. The number of neurons in the hidden layers was tuned empirically in the simulation to be as low as possible while maintaining optimal performance.

Training details

The NN was trained after each episode and held fixed during episodes. The optimization method was "Adam" (*40*), with a constant learning rate of 10^{-3} and a batch size of 30. To avoid performing many training steps while little experience is present, the number of training steps was increased in the first 10 episodes, from 200 after the first episode to lastly 2000 training steps after 10 or more episodes. The discount for future rewards was $\gamma = 0.97$. The training temperature was $T_{\text{train}} = -0.1$, and the action temperature used to select actions during an episode was $T_{\text{act}} = 0.004$.

During training, 10% of the samples are chosen from actual experience that was obtained during any previous episode. These samples are drawn with prioritized experience replay (*30*). Ninety percent of the samples are obtained from the environment model. The values of all parameters given above were selected via grid searches in the simulation environment.

Rupture avoidance mechanism

Prior human trials of removing the PTCDA molecule from the surface show that if a bond rupture (failure) occurs at a given location, ruptures are likely to occur in the area around it too. Also, as stated previously, a viable trajectory needs to be found as fast as possible, because the SPM tip might change at any time. Thus, the agent needs to explore the state space quickly for a promising direction. Therefore, we implement a mechanism to make the agent rapidly avoid states where the tip-molecule bond previously ruptured and particularly also the states leading up to it.

Usually, RL algorithms like the one trained with Eq. 2 tend to "ignore" future negative rewards, if there is a strategy that narrowly avoids them: To compute the expected future reward at time $t + 1$, they weigh the highest Q values more than the lower ones. In the limit as $T \to 0$, all Q values but the highest one are ignored. Because of this, information about failure states barely propagates to any states that lie two or more steps away. In our case, this means that with a randomly initialized NN, an agent would try to lift the molecule on similar paths in each episode until it is absolutely certain that this path is not viable. This leads it to fail at nearly identical locations each time, and therefore, it does not explore efficiently. For general RL settings, this may be the desired behavior, but we need the agent to learn as quickly as possible to avoid the failure state and the states leading up to it. Our solution is to use a negative temperature T_{train} during training, which has the effect of inverting the importance of the Q values as computed by Eq. 1. Now, low Q values, which indicate danger of rupture, are given a high importance. This information about a rupture is therefore propagated much further.

There is a trade-off here, because the further away the agent tries to stay from known failure states, the more likely it becomes that it misses a viable trajectory. We can control this trade-off by changing the training temperature. In the simulation environment, we found that the optimal training temperature was $T_{\text{train}} = -0.1$.

We tested whether using RL is necessary at all, then if all the agent does is avoid the regions of rupture states. We conducted an experiment in which the agent tries a random trajectory in the first episode and, in all following episodes, chooses its actions such that it stays as far away as possible from all previously occurred ruptures. Despite substantial experimentation with this approach, the agent never reached the goal state but got stuck in dead ends. RL, on the other hand, can identify such dead ends and plan trajectories to avoid them.

Exploiting the Cartesian state description for model-based planning

We use a slight variation of Dyna (*31*) for model-based planning. Dyna, in general, updates Q values with state transitions sampled from a learned environment model. To learn an environment model, we make use of the fact that the result of performing an action from a known state deterministically results in a new state, because the state description includes only the Cartesian coordinates of the tip, and each action moves the tip by a specified amount. To obtain a state from our environment model, we first pick a random state from the unique set of actually visited states and sample a new position around it. To sample a new position, we randomly pick between zero and four actions from our action set $\mathcal{A}$ and use them to walk to a new location. At each step, we randomly either walk into the usual positive z direction or instead walk in the negative z direction. The new location becomes s_t. Then, we sample an action a_t and the resulting successor state s_{t+1}, which is easily computed given s_t and a_t. At this point, we only need a reward r_{t+1}. If the sampled successor state is a known failure state (where bond breaking was observed before), the environment model emits the failure-reward of −1. Otherwise, it emits the default step-reward +0.01. In this way, we generate new $(s_t, a_t, r_{t+1}, s_{t+1})$ tuples and use them for training (Eq. 2) in the same way as with regular samples. When training the NN, we use 10% of the samples from real experience and 90% of the samples from the environment model. In particular, in combination with the rupture avoidance mechanism, this propagates failure information

to a much larger number of states, which can then be avoided in the next episodes.

SUPPLEMENTARY MATERIALS

Supplementary material for this article is available at http://advances.sciencemag.org/cgi/content/full/6/36/eabb6987/DC1

REFERENCES AND NOTES

1. R. Temirov, S. Soubatch, O. Neucheva, A. C. Lassise, F. S. Tautz, A novel method achieving ultra-high geometrical resolution in scanning tunnelling microscopy. *New J. Phys.* **10**, 053012 (2008).
2. L. Lafferentz, F. Ample, H. Yu, S. Hecht, C. Joachim, L. Grill, Conductance of a single conjugated polymer as a continuous function of its length. *Science* **323**, 1193–1197 (2009).
3. M. Koch, F. Ample, C. Joachim, L. Grill, Voltage-dependent conductance of a single graphene nanoribbon. *Nat. Nanotechnol.* **7**, 713–717 (2012).
4. C. Wagner, N. Fournier, V. G. Ruiz, C. Li, K. Müllen, M. Rohlfing, A. Tkatchenko, R. Temirov, F. S. Tautz, Non-additivity of molecule-surface van der Waals potentials from force measurements. *Nat. Commun.* **5**, 5568 (2014).
5. M. Aono, K. Ariga, The way to nanoarchitectonics and the way of nanoarchitectonics. *Adv. Mater.* **28**, 989–992 (2016).
6. S. Kawai, A. Benassi, E. Gnecco, H. Söde, R. Pawlak, X. Feng, K. Müllen, D. Passerone, C. A. Pignedoli, P. Ruffieux, R. Fasel, E. Meyer, Superlubricity of graphene nanoribbons on gold surfaces. *Science* **351**, 957–962 (2016).
7. T. Esat, N. Friedrich, F. S. Tautz, R. Temirov, A standing molecule as a single-electron field emitter. *Nature* **558**, 573–576 (2018).
8. K. C. Neuman, A. Nagy, Single-molecule force spectroscopy: Optical tweezers, magnetic tweezers and atomic force microscopy. *Nat. Methods* **5**, 491–505 (2008).
9. M. Rief, F. Oesterhelt, B. Heymann, H. E. Gaub, Single molecule force spectroscopy on polysaccharides by atomic force microscopy. *Science* **275**, 1295–1297 (1997).
10. J. K. Gimzewski, C. Joachim, Nanoscale science of single molecules using local probes. *Science* **283**, 1683–1688 (1999).
11. P. S. Weiss, A molecular four-wheel drive. *Nature* **479**, 187–188 (2011).
12. N. Pavliček, L. Gross, Generation, manipulation and characterization of molecules by atomic force microscopy. *Nat. Rev. Chem.* **1**, 0005 (2017).
13. J. Stigler, F. Ziegler, A. Gieseke, J. C. M. Gebhardt, M. Rief, The complex folding network of single calmodulin molecules. *Science* **334**, 512–516 (2011).
14. J. Glückstad, D. Palima, A. Bañas, Light robotics: Aiming towards all-optical nano-robotics, in *The Proceedings of the Optical Manipulation Conference* (2017), p. 102520C.
15. C. G. Atkeson, J. C. Santamaria, A comparison of direct and model-based reinforcement learning. *Proc. Int. Conf. Robot. Autom.* **4**, 3557–3564 (1997).
16. M. P. Deisenroth, D. Fox, C. E. Rasmussen, Gaussian processes for data-efficient learning in robotics and control. *IEEE Trans. Pattern Anal. Mach. Intell.* **37**, 408–423 (2015).
17. S. Levine, C. Finn, T. Darrell, P. Abbeel, End-to-end training of deep visuomotor policies. *J. Mach. Learn. Res.* **17**, 1–40 (2016).
18. M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, W. Zaremba, Hindsight experience replay, in *31st Conference on Neural Information Processing Systems (NIPS 2017)*, Long Beach, CA, USA (2017).
19. R. S. Sutton, A. G. Barto, *Reinforcement Learning: An Introduction* (MIT Press, ed. 2, 2018).
20. G. Carleo, M. Troyer, Solving the quantum many-body problem with artificial neural networks. *Science* **355**, 602–606 (2017).
21. M. F. B. Green, T. Esat, C. Wagner, P. Leinen, A. Grötsch, F. S. Tautz, R. Temirov, Patterning a hydrogen-bonded molecular monolayer with a hand-controlled scanning probe microscope. *Beilstein J. Nanotechnol.* **5**, 1926–1932 (2014).
22. N. Kocić, D. Blank, P. Abufager, N. Lorente, S. Decurtins, S.-X. Liu, J. Repp, Implementing functionality in molecular self-assembled monolayers. *Nano Lett.* **19**, 2750–2757 (2019).
23. P. Leinen, M. F. B. Green, T. Esat, C. Wagner, F. S. Tautz, R. Temirov, Virtual reality visual feedback for hand-controlled scanning probe microscopy manipulation of single molecules. *Beilstein J. Nanotechnol.* **6**, 2148–2153 (2015).
24. C. Wagner, M. F. B. Green, P. Leinen, T. Deilmann, P. Krüger, M. Rohlfing, R. Temirov, F. S. Tautz, Scanning quantum dot microscopy. *Phys. Rev. Lett.* **115**, 026101 (2015).
25. P. Leinen, M. F. B. Green, T. Esat, C. Wagner, F. S. Tautz, R. Temirov, Hand controlled manipulation of single molecules via a scanning probe microscope with a 3D virtual reality interface. *J. Vis. Exp.*, 54506 (2016).
26. R. Temirov, M. F. B. Green, N. Friedrich, P. Leinen, T. Esat, P. Chmielniak, S. Sarwar, J. Rawson, P. Kögerler, C. Wagner, M. Rohlfing, F. S. Tautz, Molecular model of a quantum dot beyond the constant interaction approximation. *Phys. Rev. Lett.* **120**, 206801 (2018).
27. C. Wagner, M. F. B. Green, M. Maiworm, P. Leinen, T. Esat, N. Ferri, N. Friedrich, R. Findeisen, A. Tkatchenko, R. Temirov, F. S. Tautz, Quantitative imaging of electric surface potentials with single-atom sensitivity. *Nat. Mater.* **18**, 853–859 (2019).
28. N. Fournier, C. Wagner, C. Weiss, R. Temirov, F. S. Tautz, Force-controlled lifting of molecular wires. *Phys. Rev. B* **84**, 035435 (2011).
29. H. van Seijen, H. van Hasselt, S. Whiteson, M. Wiering, A theoretical and empirical analysis of Expected Sarsa, in *Proceedings of the 2009 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL 2009)* (2009), pp. 177–184.
30. T. Schaul, J. Quan, I. Antonoglou, D. Silver, Prioritized experience replay, in *4th International Conference on Learning Representations (ICLR 2016)*, San Juan, Puerto Rico (2016).
31. R. S. Sutton, Dyna, an integrated architecture for learning, planning, and reacting. *ACM SIGART Bull.* **2**, 160–163 (1991).
32. F. J. Giessibl, The qPlus sensor, a powerful core for the atomic force microscope. *Rev. Sci. Instrum.* **90**, 011101 (2019).
33. M. Rashidi, R. A. Wolkow, Autonomous scanning probe microscopy in situ tip conditioning through machine learning. *ACS Nano* **12**, 5185–5189 (2018).
34. O. Gordon, P. D'Hondt, L. Knijff, S. E. Freeney, F. Junqueira, P. Moriarty, I. Swart, Scanning tunneling state recognition with multi-class neural network ensembles. *Rev. Sci. Instrum.* **90**, 103704 (2019).
35. A. Krull, P. Hirsch, C. Rother, A. Schiffrin, C. Krull, Artificial-intelligence-driven scanning probe microscopy. *Commun. Phys.* **3**, 54 (2020).
36. S. Thrun, A. Schwartz, Issues in using function approximation for reinforcement learning, in *Proceedings of the 4th Connectionist Models Summer School* (Lawrence Erlbaum Publisher, 1993).
37. H. van Hasselt, Double Q-learning, in *Annual Conference on Advances in Neural Information Processing Systems* (2010), pp. 2613–2621.
38. 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-16)* (2016), pp. 2094–2100.
39. Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, N. de Freitas, Dueling network architectures for deep reinforcement learning, in *Proceedings of the 33rd International Conference on Machine Learning (ICML)* (2016), pp. 1995–2003.
40. D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in *3rd International Conference on Learning Representations (ICLR 2015)*, San Diego (2015).

Acknowledgments

Funding: We acknowledge support from the Institute for Pure and Applied Mathematics (IPAM) at the UCLA (program "Understanding Many-Particle Systems with Machine Learning"). C.W. acknowledges funding through the European Research Council (ERC-StG 757634 "CM3"). F.S.T. acknowledges financial support from the Deutsche Forschungsgemeinschaft through SFB 1083, project A12. K.-R.M. was partly supported by the German Ministry for Education and Research (BMBF) under grants 01IS14013A-E, 01GQ1115, and 01GQ0850; the German Research Foundation (DFG) under grant Math+, EXC 2046/1, project ID 390685689; and the Institute for Information and Communications Technology Planning and Evaluation (IITP) grant funded by the Korean government (no. 2017-0-01779). We thank T. Esat for supporting the development of the RL-SPM interface software. **Author contributions:** C.W., K.T.S., K.-R.M., and F.S.T. conceived and designed this research during the IPAM long program "Understanding Many-Particle Systems with Machine Learning." M.E. and K.T.S. designed and implemented the RL agent. P.L. performed the experiments. C.W., P.L., and M.E. analyzed the data. C.W., K.-R.M., and F.S.T. developed the conceptual framework. M.E., C.W., and F.S.T. wrote the paper, with input from all authors. **Competing interests:** The authors declare that they have no competing interests. **Data and materials availability:** All data needed to evaluate the conclusions in the paper are present in the paper and/or the Supplementary Materials. Code of the RL agent can be found at https://github.com/Maltimore/robotic_nanofabrication. Correspondence and requests for materials should be addressed to C.W. or K.-R.M.

Submitted 12 March 2020
Accepted 2 July 2020
Published 2 September 2020
10.1126/sciadv.abb6987

Citation: 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).

Science Advances

Autonomous robotic nanofabrication with reinforcement learning

Philipp Leinen, Malte Esders, Kristof T. Schütt, Christian Wagner, Klaus-Robert Müller and F. Stefan Tautz

Sci Adv **6** (36), eabb6987.
DOI: 10.1126/sciadv.abb6987

Science Advances (ISSN 2375-2548) is published by the American Association for the Advancement of Science, 1200 New York Avenue NW, Washington, DC 20005. The title *Science Advances* is a registered trademark of AAAS.