Synthesizing Depowdering Trajectories for Robot Arms using Deep Reinforcement Learning

Maximilian Maurer[1], Simon Seefeldt[1,2], Jan Seyler[1], and Shahram Eivazi[1,2]

***Abstract*— Research into robotics applications of deep reinforcement learning (DRL) has increasingly been focussed on learning precise object manipulation and trajectory planning. Extending these tasks to continuous robot-object interactions with the surface of complex geometries remains an open problem. In this paper we investigate end-to-end DRL solutions for depowdering tasks that work by directing a pressurized air stream onto the object's surfaces using a blast nozzle head mounted on a robotic arm. We develop a GPU accelerated vectorized cleaning effect for integration into RL training and consider ways to expose vision-less trajectory synthesis for surface treatment applications to the RL agent based on UV mapping. Our experimental evaluation demonstrates that DRL has the potential to be used for generating object-specific agents for depowdering tasks on a variety of 3D objects without requiring intermediate path planners even in a full 3D motion setup. Finally, we show that DRL-generated trajectories can be transferred to a real-world setup. Our task formulation lends itself to approximate a wide range of surface treatment applications (e.g., cleaning and spray painting) with various effects.**

I. INTRODUCTION

In 3D printing, the post-processing steps involving manual human interventions have become a highly time-consuming and costly bottleneck. Specifically, cleaning powder-based printers (e.g. Selective Laser Sintering) requires an operator to remove the residual powder from 3D-printed parts manually applying a combination of rotation, vibration, vacuum, and air blasting techniques [1], [2], [3], [4]. As such there has been a growing interest in the development of robotic systems that use a robot arm to automatically remove powder by using brushing, vacuuming, or air blasting. [5], [6], [7], [8]. For example, [5] presented a custom-made mechanical robotic station that uses Deep Learning for 3D perception at ICRA 2020. They used classical motion planning and force control to perform the depowdering task using an industrial robot arm.

In a recent study, [8] utilized object's mesh to obtain a complete coverage of surfaces. Then the task was formulated as a NP-hard coverage path planning (CPP) problem to find the shortest path for the robotic arm that visits each surface exactly once. They propose to decompose the computation on the whole mesh into subgraphs constructed by edges to its neighbor-connected clusters.

Determining a collision free path for robots while navigating around environments and objects and achieving a coverage objective has been well studied [9]. Studies on using deep reinforcement learning (DRL) techniques to solve CPP problems [10], [9], [11], [12] are highly relevant to this paper. For instance, [12] develop a generalized method for online CPP in unknown 2D environments where an RL agent directly predicts the control signals for the robot based on the environments' sensory data. To address the scalability problem, they proposed to use a multi-scale frontier map representation along with a scale-grouped convolutional neural network that independently processes the different scales in the map.

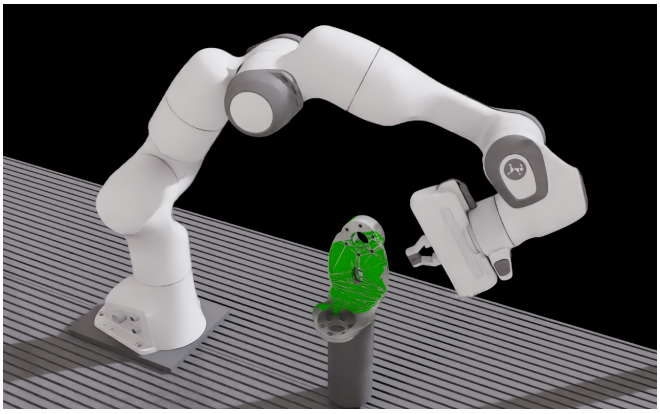


Fig. 1: Depowdering effect: When the virtual air nozzle points at the object the gray (dust) surface is removed revealing the green object surface.

We found that the majority of contributions in the DRL research either focus on the CPP problem for 2D navigation applications or, when dealing with 3D objects, rely on simplifying parts of the problem within restricted environments. Additionally, they do not provide an end-to-end solution but use DRL only for high-level decision-making while relying on traditional methods for low-level execution such as path planning.

A few studies have explored DRL solutions that encompass both CPP and surface treatment tasks, such as depowdering or spray painting. [13] propose PaintRL as a simplified version of spray painting, but limit the movement of the robot arm to four discrete actions (up, down, left, right). In another example, [14] proposed a two-step training strategy (pre-training and fine-tuning) to reduce the overall time needed for object-specific DRL training. However, their example includes only 2D rectangles shapes and 2D robot movements.

Tasks like spray painting and depowdering, while specialized, are applications of a broader robotic manipulation problem. They offer long-horizon decision problems involving

[1] Festo SE & Co. KG
[2] University of Tübingen

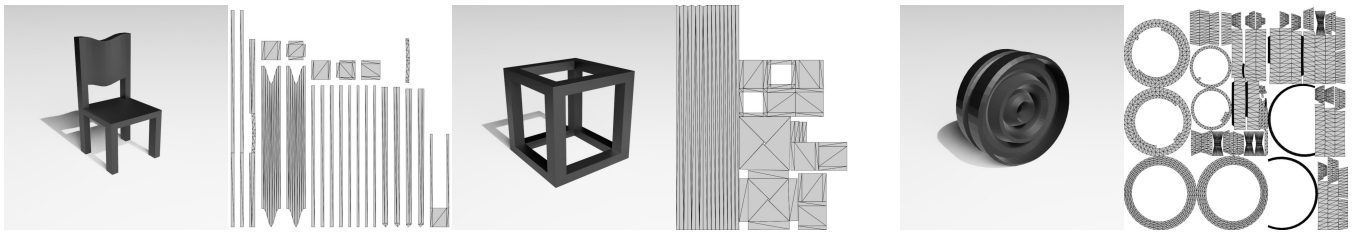


Fig. 2: Holdout objects used for evaluation. To the right side is the UV map for each object.

complex motion sequences, which present many challenges for DRL training.

In this paper, we take a step towards solving the depowdering task in an end-to-end manner in full 3D with the RL agent directly predicting joint position targets. We propose to use an end-to-end DRL method and also investigate a way of training it without the need for synthetic camera images. We train object-specific DRL agents for cleaning 3D objects that directly output control targets for the joint controllers of the 7 DOF robot arm in simulation. This yields simulated trajectories, which we then execute on a real robot arm. Our contributions are:

- We develop an integrated way to approximate the depowdering task in a GPU-accelerated manner suitable for vectorized DRL training that allows the expression of a broad range of scenarios.
- We investigate a lightweight way to train the DRL agent directly on a representation of the object cleaning state that does not require synthesizing camera images.
- We present end-to-end DRL agents that learn to control the simulated robot arm at the joint level, allowing the target surfaces to be cleaned while avoiding collision.

II. RELATED

The deep reinforcement learning (DRL) framework is considered a promising end-to-end solution for solving robotic tasks. Over the past decade, developments in simulation and computing technology have led to the growth of DRL robotic research [15], [16]. The interest in DRL research in the robotic domain has been accompanied by the growth of simulation software such as MuJoCo [17], Pybullet [18], and Nvidia Isaac [19] due to their ability to quickly synthesize large amounts of data. Among these studies, over the past decade, increasing use of mobile robots and articulated arms has led to the growth of DRL-based path planning methodologies [20]. Concurrently, DRL has become a popular method for solving the application of the Coverage Path Planning (CPP) problem [21], [9], [22].

The majority of DRL-based solutions for the CPP problem have been researched in the context of autonomous mobile robots (AMRs) [23], [12] or unmanned aerial vehicles (UAVs) [11], [24]. For example, in a recent study, [12], developed an online coverage path planning system that utilizes DRL in unknown 2D environments. They analyze various components of DRL, such as observation, action space, neural network architecture, and reward function to efficiently learn CPP while avoiding obstacles.

In contrast to AMRs and UAVs, there have only been a few studies that apply DRL-based methods specifically for learning CPP problems in manipulation tasks with a robot arm [13], [14]. The tasks often involve the use of a robotic arm for surface treatment applications, such as object cleaning, depowdering, or spray painting.

For example, in the context of a spray painting task, [13] approximate the spray gun as an array of rays in the Pybullet [18] simulation and calculate the intersection with the object for each ray. They propose to use the UV map as an input for the DRL agent (see related work [25], [26]). While [13] demonstrated that the proximal policy optimization (PPO) [27] DRL algorithm could achieve complete paint coverage of a car door, their approach was limited by a simplified version of spray painting, as their robot could only move in four discrete directions, and was provided with a lower-dimensional input based on identifying the achieved coverage in circular sectors around the current area.

In another spray painting example, [14] proposed an unsupervised DRL pre-training stage to reduce the overall time needed for object-specific training. They focus on task-agnostic training guided by a state-entropy maximization objective to solve the object-conditioned exploration problem. However, their example includes only 2D rectangles shapes and 2D robot movements.

In contrast, here, we unify various aspects of the previous works: we combine the use of a DRL agent, on a full 3D version of the task and cleaning effect, with a continuous control space and the usage of the UV mapping rather than synthetic camera images. Our environment involves additional geometry that the agents have to learn to avoid, and we use challenging holdout-objects to evaluate the method. We highlight in particular the non-convex nature of our objects with hard-to-access angles. It is generally not possible for the agent to clean our evaluation objects by going over them hemispherically. Additionally, our agents start from random initial configurations. As we do not train on camera images, our agents are object-specific.

III. METHOD

A. Vectorized Cleaning Effect

A scalable way to expose the dynamics of the cleaning task to the agent is necessary for DRL. Using fully accurate classical, or physically plausible, simulation techniques this can be a problem, as full simulation models modelling all aspects of the behavior of dust particles and their interaction

TABLE I: Sampling performance of the environment

BS	Resolution	Samples	$\frac{\text{Mean Wall Time}}{\text{Simulated Sample}}$	$\frac{\text{Simulated time}}{\text{Wall time}}$
256	128x128	100000	$2.2318\text{ms} \pm 0.0101$	44.8
512	128x128	250000	$2.0836\text{ms} \pm 0.0542$	48.0
256	256x256	100000	$2.6680\text{ms} \pm 0.0086$	37.5
512	256x256	250000	$2.5852\text{ms} \pm 0.0121$	38.7

The environment sampling performance was measured by running 3 sample experiments per row using a set batch size (number of parallel robot setups) and cleaning state texture resolution with an agent taking random actions, and then aggregating results. The number of samples per experiment was chosen based on the batch size to better reflect time overheads for resetting the episode on contacts. The sample collection was run until at least as many samples as requested were collected, rounded to the next increment of the batch size. The experiment ran on a Nvidia RTX A6000 GPU and an Intel Xeon W7-3445 CPU. For better comparability we report the time to provide the samples on the GPU. Usually, on top of this an additional performance penalty is incurred for transfer to host memory and conversion to the input format of the RL algorithm implementation. Not included is shader compilation time, and setup time to instantiate geometry and presample collision free reset poses for use during the training. We use a physics timestep of $\frac{1}{60}$ seconds, and run 6 physics timesteps per environment step. Sampling performance is reported as the speedup over real time, ie how many seconds of simulated interaction can be simulated in one real-world second.

with the geometry and airflow are prohibitively computationally expensive for use in this style of DRL training. Instead, here, we propose to train on an only semi-realistic task proxy, which we call the "cleaning effect" to distinguish it from a true simulation. Its computation is vectorized over n parallel robots within the scene.

We formulate this cleaning effect on arbitrary watertight triangulated geometry. We assume each object has an UV mapping available that maps the object surface to a 2D texture. During evaluation we consider examples of automatically generated UV maps as well as hand-made UV maps. The coverage of the object surface is represented by a single-channel 8 bit texture (the "cleaning state texture"). The cleaning effect is the interaction that mediates changes to the cleaning state texture based on the current pose of the robot and object, as well as a description of the desired effect.

We decompose the cleaning effect on the cleaning state texture's texels t and their corresponding positions on the object surface in 3D p into two stages: a refinement phase that applies various criteria to a binary mask determining which texels will be affected by the cleaning effect, and a successive phase which determines the integral magnitude of the change to the cleaning state texture. Given an end-effector position e, and the forward direction from it d, during the refinement phase we apply the following masks, which together effectively shape the effect:

- whether p is within a certain distance range of e. Cleaning from too close or too far away does not effectively clean the object.
- whether p is within a certain minimum distance from any point on the line extending forward in direction d from e. This effectively creates a cylinder shape.
- whether there is a direct line of sight from p to e that is not interrupted by any other face of the cleaning object. This prevents the effect from cleaning through the object or across corners. Here, the line of sight is validated exclusively on the geometry of the object itself, but not on other parts of the scene. Consequently, the agent could theoretically clean through other objects. Due to the geometry of our setup, we did not observe this to be a problem, as the table surface in combination with the robot arm's reach, effectively prevents poses from which this would be possible.

The final refinement mask is the logical conjunction of the partial masks. While a large number of different effects can be formulated within this framework, we consider here the simplest possible scenario where the change is constant for all eligible pixels, irrespective of direction or distance.

Additionally, we add special handling for mounting: in practical applications the object can be mounted, which means there are parts of the object surface that either cannot be cleaned, or that it is not desirable for the robot to attempt to clean (for example the parts that are attached to a baseplate in 3D printing). We express this by manually creating a masking texture that excludes some regions of the object from the calculation of the cleaning effect.

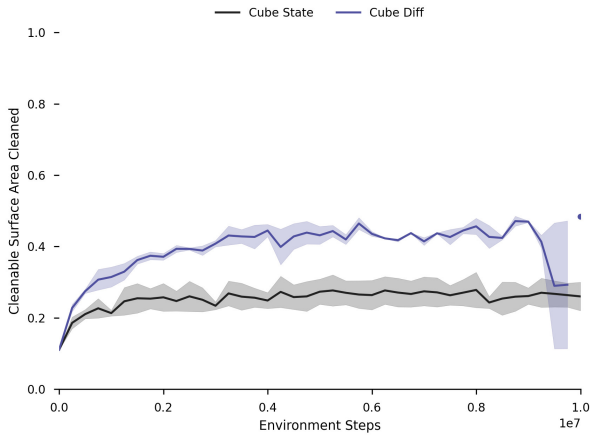
This calculation is performed in-parallel across the texels on the GPU. We implement it inside of NVIDIA Omniverse Isaac [19] using torch [28] for the bulk of the effect calculation, and NVIDIA Warp [29] for raycasting and geometry interaction. The performance is characterized in Table I.

The advantage of this formulation of the cleaning effect is its ability to approximate a wide variety of applications with minimal changes. We define the cleaning problem as the problem of reducing the values of the cleaning state texture as close to 0 as possible, without any penalty for over-cleaning. Painting could be expressed in this framework by instead defining a target and penalizing for both over- and under-applying the effect. Extensions like local surface properties such as different materials, a different reaction of the effect to creases, or different kinds of dirt, as well as different shapes and intensities of the effect are straightforward to implement in this framework.

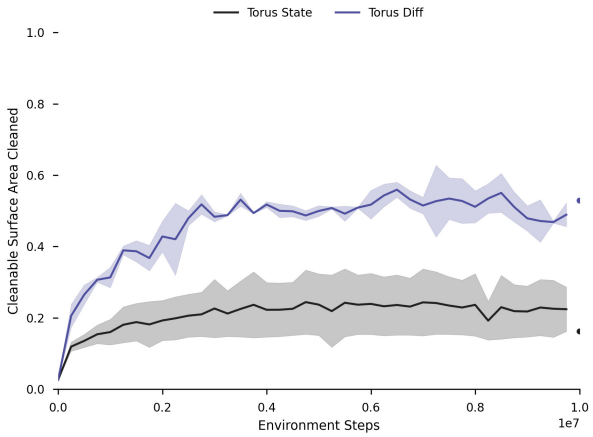
B. Cleaning Task

Our setup is a Franka Emika Panda robot arm mounted on a table. The object to clean is mounted in front of the robot arm. In this context the cleaning task has an intuitive formulation as moving the robot arm so as to effect the cleaning of the object while avoiding collisions with the object and scenery, while observing the scene through one or more cameras. We instead consider a different variation of the task. In particular, we are interested in synthesizing non-reactive cleaning trajectories using RL without having a simulated camera view of the scene available. Doing so avoids the computationally expensive process of rendering images during training at the cost of the agent not being reactive.

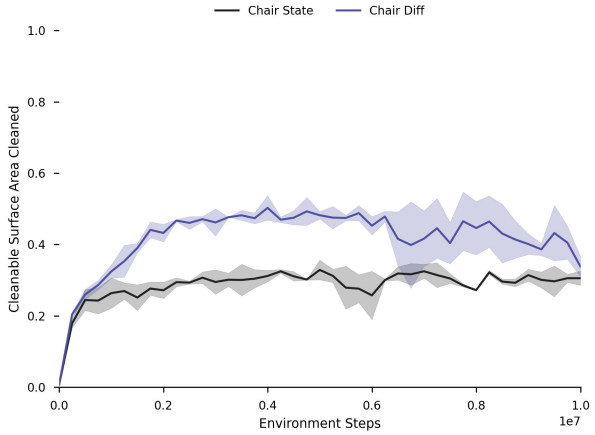
Concretely the agent controls the robot by setting the 7D control targets for the robot joint controllers. It starts



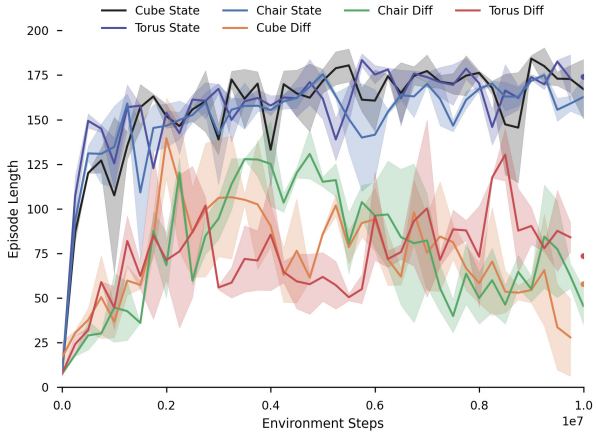
(a) Evaluation results for the cube object



(b) Evaluation results for the torus object



(c) Evaluation results for the chair object



(d) Episode lengths

Fig. 3: Evaluation results for the three objects. Each line represents the aggregate performance of the agents that have been trained with the indicated reward (stateful, or stateless based on the difference). Due to a technical error during training the final datapoint for one seed is missing from some training runs, we have indicated this with a dot that shows the data for the single remaining seed.

from an initial position that is uniformly sampled over the allowed joint movement range, and then rejection sampled to eliminate positions where the robot starts out in a collision. It observes the joint positions, the joint velocities, the world-space position of the end-effector, the world-space rotation matrix of the end-effector frame [30], the time that has passed within the episode as well as the current joint motor controller targets. We additionally provide the agent with a downscaled version of the cleaning state texture. This is its main way to track the cleaning interaction.

We simulate the environment at a simulation timestep of $\frac{1}{60}$ seconds. A DRL environment step as visible to the agent consists of 6 such simulation substeps. The cleaning effect is applied at each substep. An episode ends after a time limit or if the robot arm collides. To avoid too large inaccuracies given the fast movement of the robot arm, RL parameters related to termination, such as collision, are computed at the resolution of the substep rather than the DRL environment step.

C. Reward Fomulation

We use two different reward formulations expressing two different practical objectives. Considering the transition from state s to state s' in a setting with a maximum episode length l_e, the mean observed value of the cleanable pixels of the texture in a state $c(s_i)$, a flag value $e \in \{0, 1\}$ that indicates whether the transition was free of any collision or self-collision (in that case $e = 1$ else it is 0) and a reward scaling factor r_s, we consider a stateful reward

$$r_{\text{stateful}} = \frac{r_s}{l_e} e \cdot c(s')$$

Since in our task formulation, dirt cannot accumulate again, it follows that in this way removing dirt at a location is worth more if it happens earlier in the episode, and if the episode lasts longer. As an alternative we consider a stateless reward formulation as used in [13] where the agent is rewarded based on the scaled magnitude of the change to the cleaning state during that step

$$r_{\text{stateless}} = \frac{r_s}{l_e} e \cdot (c(s') - c(s))$$

This removes timing aspects from the consideration. In contrast to [13], we do not apply an additional time penalty to this reward. In our experiments, we apply reward scaling with $r_s = 1000$ for both rewards [31].

IV. EXPERIMENT

A. Training

Training was conducted using the Stable Baselines 3 [32] implementation of Soft Actor Critic (SAC) [33], training on 256 parallel vectorized instances using an episode length of 200 steps (corresponding to 20s simulated time per episode). We use a 1024x512x512x512 MLP for the actor and critic and set $\gamma = 0.9$, the learning rate to $3 \cdot 10^{-4}$, and the batch size to 256. The cleaning state texture is 256x256 pixels large. We set a minimum nozzle-to-surface-distance of 10cm below which no cleaning may happen. The parameterization of the cleaning device was chosen before the evaluation to be sufficiently large to enable the objects to be cleaned within the length of the episode based on the overall cleanable surface area.

We train for 10 million environment interaction steps performing 640 gradient steps per single vectorized step of the 256 instances. Thus training consists of 25 million gradient updates. Training was performed on Nvidia A10G and Nvidia RTX A6000 GPUs. On both these setups training was conducted with 2 parallel training runs per GPU, with each doublet taking 151 hours (A10G) and 101 hours (RTX A6000) for training.

B. Evaluation

We evaluate on a holdout set of 3 complex objects that were not previously used during development of the method (Figure 2):

- A chair, used here as a proxy for a general object. The UV map for this object was created manually.
- A toroidal wave-like shape, used here as a proxy for an object that is potentially hard to clean with classical techniques due to its many concavities, and potentially hard to access angles. The UV map for this object was created with Blender's Smart UV project feature [34].
- A cube-frame, used here because it creates a challenging movement scenario for the robot. The UV map for this object was created manually, since the Smart UV projection did not achieve a satisfactory result.

Evaulation was performed with 2 seeds per object and reward. Each agent is trained on a specific object. We perform our evaluation inside of the simulation.

C. Real-world execution

We additionally export some synthesized trajectories from the simulation to illustrate them on a real version of the robot setup. Nvidia Cortex was used to execute the synthesized trajectories on the real robot. For safety reasons we use Rmpflow [35] to perform the underlying control. It is configured with the acceptable safety boundaries of the setup. We try to achieve the steps from the synthesized trajectory sequentially to an acceptable fidelity (1cm position and 0.09 rad end-effector rotation). Poses which cannot be reached within a defined time-frame are reattempted from a recovery position, and if still not possible are filtered out. Results of these real-world demonstrations are shown in the supplementary material.

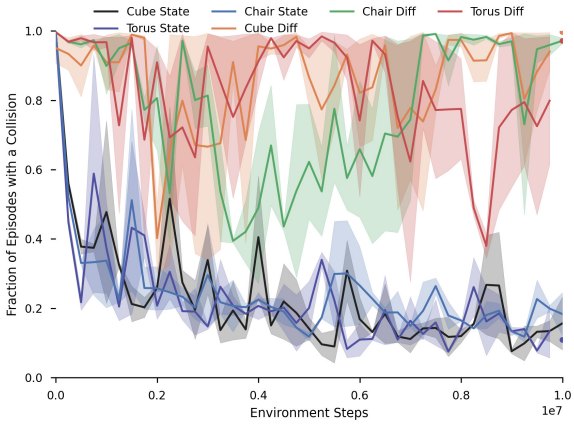


Fig. 4: Fraction of evaluation episodes that end in a contact.

V. RESULTS

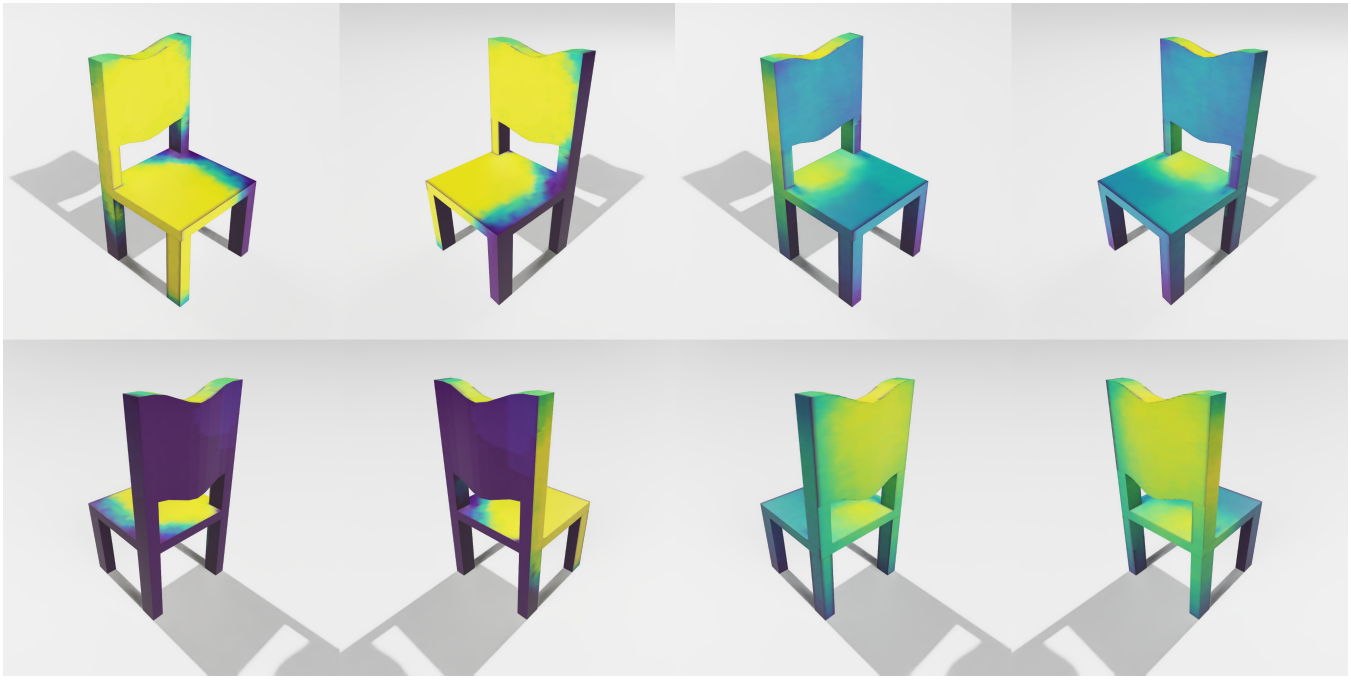
We use three held-out 3D objects to evaluate the performance of the DRL agents on the task. Figure 3 shows the results.

The stateless reward outperforms the stateful reward formulation initially for all objects. The bias in the stateful reward towards keeping the episode going for longer is visible in the results, where agents trained on this reward consistently result in much longer episodes. This does not result in a better cleaning result though.

For training runs using the stateless rewards there is a long-term instability on one of the three holdout objects that occurs in the later phases of the training (Figure 3) where an initial better approach disappears in a manner that the training process does not recover from during the rest of the training. Further experimentation was performed to assess the representativeness of this failure-mode for this object, and could consistently reproduce the issue. It corresponds to a resulting policy which crashes into the object, and is associated with a catastrophic decline in the entropy target.

We illustrate qualitative insights on the chair in Figure 5. Videos for all objects are provided in the supplementary material and provide a richer qualitative insight.

While both agents overall learn approaches that are more intuitively human-like in that they move over the object coarsely as opposed to more closely following the underlying geometry representation as would be typical for non-RL solutions. The objects represent challenging real-world kinematic scenarios, and this is where the agents struggle. The stateful agent chooses a strategy where it initially quickly cleans the large surfaces of the backrest and one side of the seat, as well



(a) Intensity map for the stateful reward

(b) Intensity map for the stateless reward

Fig. 5: Intensity maps of 192 episodes each illustrating the different approaches that the agents have learned. A yellow hue indicates a higher fraction of episodes where the surface was cleaned. Overall magnitude and intensity is not comparable between the two sides. The stateful agent has learned a strategy that largely stays on one side of the object, whereas the agent trained with a stateless reward has learned a much larger range of motion.

as a bit of the legs. It then gets stuck in a problematic pose from which it does not manage to further pursue cleaning. The stateless agent initially selects a kinematically much more favourable configuration that allows it to access both sides of the chair object and hemispherically move for better access.

Both agents get stuck during later parts of the episode where they just oscillate and are not able to make use of the rest of the episode to clean more of the object. Neither manages perform cleaning in the kinematically extremely unfavourable area at the lower side of the seat plate. These spots require a sustained and precise long-horizon motion sequence to be cleaned correctly. As a result none of the agents are able to fully clean the object.

VI. DISCUSSION

This work shows that it has become feasible to train with modern vectorized simulations on a depowdering task in full 3D in combination with reinforcement learning without an intermediate path planner or discretizing the action space. The main drawback of the approach is its simulation accuracy, as it cannot model powder accumulation, secondary effects like dust-clouds, and its resolution depends critically on the underlying UV representation as well as the resolution of the cleaning state texture.

DRL solutions differ from classical (model-based) approaches based on a more direct analysis of the geometry such as [8]. Model-based approaches are generally much more resource-efficient and yield reliably collision free trajectories. Conversely, the much slower end-to-end RL methodology allows for more flexibility: it can easily express different tasks, scenery and auxiliary priorities without fundamental changes to the underlying algorithms, at the cost of no longer guaranteeing collision freeness.

We establish that nonetheless, this camera-less task formulation is sufficiently challenging to provide an interesting research setup for surface treatment application with DRL. In particular learning the kinematic strategies required for collision free access to obstructed parts of real-world objects remains challenging.

The quantitative results show that fully solving this task formulation requires further techniques beyond out-of-the box RL methods, but at the same time the performance of these methods is sufficient to form a basis for further research. Additionally, it is clear that kinematic challenges can be investigated on this task without the need for synthetic camera images.

The task admits several areas of further work: improving the accuracy of the simulation, applying further techniques to improve the performance of the DRL agent, as well as task-specific exploration of the ability of agents to leverage different cleaning devices through different strategies.

ACKNOWLEDGMENT

We thank Jakob Dannel for his contribution to the setup of the real-world experiments.

References

[1] L. W. Hunter, D. Brackett, N. Brierley, J. Yang, and M. M. Attallah, "Assessment of trapped powder removal and inspection strategies for powder bed fusion techniques," *The International Journal of Advanced Manufacturing Technology*, vol. 106, pp. 4521–4532, 2020.

[2] P. Dunst, P. Bornmann, T. Hemsel, and W. Sextro, "Vibration-assisted handling of dry fine powders," in *Actuators*, vol. 7, no. 2. MDPI, 2018, p. 18.

[3] A. Ju, A. Fitzhugh, J. Jun, and M. Baker, "Improving aesthetics through post-processing for 3d printed parts," *Electronic Imaging*, vol. 2019, no. 6, pp. 480–1, 2019.

[4] Solukon, "Machines for powder removal," 2024, www.solukon.de, Visited on 12.02.2024.

[5] H. Nguyen, N. Adrian, J. L. X. Yan, J. M. Salfity, W. Allen, and Q.-C. Pham, "Development of a robotic system for automated decaking of 3d-printed parts," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 8202–8208.

[6] J. X.-Y. Lim and Q.-C. Pham, "Automated post-processing of 3d-printed parts: artificial powdering for deep classification and localisation," *Virtual and Physical Prototyping*, vol. 16, no. 3, pp. 333–346, 2021.

[7] Z. Liu, J. Geng, X. Dai, T. Swierzewski, and K. Shimada, "Robotic depowdering for additive manufacturing via pose tracking," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 770–10 777, 2022.

[8] V.-T. Do and Q.-C. Pham, "Geometry-aware coverage path planning for depowdering on complex 3d surfaces," *IEEE Robotics and Automation Letters*, 2023.

[9] C. S. Tan, R. Mohd-Mokhtar, and M. R. Arshad, "A comprehensive review of coverage path planning in robotics using classical and heuristic algorithms," *IEEE Access*, vol. 9, pp. 119 310–119 342, 2021.

[10] A. K. Lakshmanan, R. E. Mohan, B. Ramalingam, A. V. Le, P. Veerajagadeshwar, K. Tiwari, and M. Ilyas, "Complete coverage path planning using reinforcement learning for tetromino based cleaning and maintenance robot," *Automation in Construction*, vol. 112, p. 103078, 2020.

[11] M. Theile, H. Bayerlein, R. Nai, D. Gesbert, and M. Caccamo, "Uav coverage path planning under varying power constraints using deep reinforcement learning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 1444–1449.

[12] A. Jonnarth, J. Zhao, and M. Felsberg, "Learning coverage paths in unknown environments with deep reinforcement learning," in *International Conference on Machine Learning, 21-27 July 2024, Vienna, Austria*. PMLR, 2024, pp. 22 491–22 508.

[13] J. Kiemel, P. Yang, P. Meißner, and T. Kröger, "Paintrl: Coverage path planning for industrial spray painting with reinforcement learning," in *RSS Workshop on Closing the Reality Gap in Sim2real Transfer for Robotic Manipulation*, 2019.

[14] M. Prattico, "Towards autonomous robotic spray painting with unsupervised reinforcement learning," Ph.D. dissertation, Politecnico di Torino, 2023.

[15] B. Singh, R. Kumar, and V. P. Singh, "Reinforcement learning in robotic applications: a comprehensive survey," *Artificial Intelligence Review*, vol. 55, no. 2, pp. 945–990, 2022.

[16] S. Gu, E. Holly, T. P. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation," *arXiv preprint arXiv:1610.00633*, vol. 1, p. 1, 2016.

[17] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 5026–5033.

[18] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," 2016.

[19] N. Corporation, "Nvidia isaac simulation," 2023, accessed: 2023-10-01. [Online]. Available: https://developer.nvidia.com/isaac-sim

[20] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[21] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[22] J. P. Carvalho and A. P. Aguiar, "A reinforcement learning based online coverage path planning algorithm," in *2023 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. IEEE, 2023, pp. 81–86.

[23] A. Jonnarth, O. Johansson, and M. Felsberg, "Sim-to-real transfer of deep reinforcement learning agents for online coverage path planning," *arXiv preprint arXiv:2406.04920*, 2024.

[24] C. Zhao, J. Liu, S.-U. Yoon, X. Li, H. Li, and Z. Zhang, "Energy constrained multi-agent reinforcement learning for coverage path planning," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 5590–5597.

[25] Z. Chen, K. Yin, and S. Fidler, "Auv-net: Learning aligned uv maps for texture transfer and synthesis," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 1465–1474.

[26] W. Yang, Z. Chen, C. Chen, G. Chen, and K.-Y. K. Wong, "Deep face video inpainting via uv mapping," *IEEE Transactions on Image Processing*, vol. 32, pp. 1145–1157, 2023.

[27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[28] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[29] M. Macklin, "Warp: A high-performance python framework for gpu simulation and graphics," https://github.com/nvidia/warp, March 2022, nVIDIA GPU Technology Conference (GTC).

[30] A. R. Geist, J. Frey, M. Zobro, A. Levina, and G. Martius, "Learning with 3d rotations, a hitchhiker's guide to so (3)," *arXiv preprint arXiv:2404.11735*, 2024.

[31] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[32] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.

[33] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[34] Blender Online Community, *Blender - a 3D modelling and rendering package*, Blender Foundation, Amsterdam, The Netherlands, 2023, version 3.5. [Online]. Available: https://www.blender.org

[35] C.-A. Cheng, M. Mukadam, J. Issac, S. Birchfield, D. Fox, B. Boots, and N. Ratliff, "Rmpflow: A computational graph for automatic motion policy generation," 2019.