PaintRL: Coverage Path Planning for Industrial Spray Painting with Reinforcement Learning

Jonas C. Kiemel[1], Peiren Yang[1], Pascal Meißner[1] and Torsten Kröger

***Abstract*—We present PaintRL, a framework that enables research on optimizing industrial spray painting for arbitrary objects with reinforcement learning. PaintRL implements a toolkit to simulate and visualize spray painting based on the physics engine PyBullet. By means of this toolkit, we train neural networks to predict coverage paths and evaluate the results on two objects: a quadratic sheet and a real car door. Our initial results show that the generated coverage path of the car door performs on a par with a manually implemented zigzag baseline. To allow sim2real transfer for non-resettable tasks like spray painting, we replace paint by light using projection mapping. This approach opens up new possibilities to visualize the results from simulation, collect human demonstrations and capture real-world images. PaintRL is part of our endeavor to utilize the recent advances in deep reinforcement learning for economically important industrial tasks.**

I. Introduction

Reinforcement learning (RL) offers huge potential to allow mass customization and line production from batch size one. In recent years, significant progress has been made in applying RL to real-world scenarios [1]–[4]. However, due the large amount of data required by modern RL algorithms, industrial applications have hardly be addressed.

Data generation for industrial tasks is complex in simulation as well as in the real world. Simulators like Mujoco [5] or Bullet [6], which are widely used by the RL community, do not offer out-of-the-box support for industrial tasks like painting, grinding or welding. Application-specific commercial simulators, however, are typically not designed for RL, meaning that they lack programming interfaces, are hardly adaptable, or simply too slow to generate sufficient data. We address this issue with a fast and scaleable spray painting simulation based on the physics engine PyBullet [7]. When collecting real-world data, previous work has focused either on tasks that can be easily reset by the robot [4], [8] or on those, in which every resting state can be considered as a starting point of a new rollout [1], [9], [10]. However, spray painting alters the workpiece in a non-resettable way, making autonomous data collection impractical. By utilizing projection mapping, we replace paint by light, meaning that real-world data can be collected without the need of a costly cleaning procedure.

[1]Intelligent Process Automation and Robotics Lab (IPR), Karlsruhe Institute of Technology (KIT) {jonas.kiemel, pascal.meissner}@kit.edu

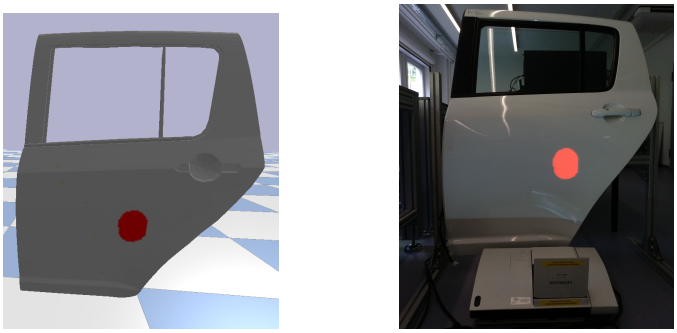


Fig. 1: Spray painting simulation environment and sim2real transfer with projection mapping

II. Spray painting simulation

A paint spray gun can be either modeled by a numerical or by a geometrical approach. The numerical approach is based on computational fluid dynamics (CFD) and aims to calculate the movement of each droplet with respect to gravity, electric fields and air flow [11]–[13]. While commercial simulators using CFD exist [14], a single simulation typically requires several hours, which strongly inhibits the generation of sufficient data for RL experiments. The geometrical approach assumes that the relationship between the spray gun and the droplets is not influenced by external forces. In [15], the spray gun is modeled as a cone with a parabolic paint distribution. Balkan and Arikan [16] proposed to use the beta distribution as a more general approach to model spray guns. We adopt the idea of using a beta distribution for our simulation, since it allows to model a wide range of spray guns while still being computationally efficient. In a simulator like PyBullet, rigid bodies are typically represented by a polygon mesh defined by the vertices, edges and faces of the object. To colorize the object, a projection from a 2D texture map to a 3D surface is needed. This projection is called UV mapping. We use Blender [17] to generate the UV mapping and the wavefront .obj file to store the polygon mesh.

Our workflow to simulate the spray painting process can be described as follows:

1) As shown in figure 2, approximate the spray gun as an array of rays and calculate the intersection with the workpiece for each ray.
2) Search the mesh for the three vertices that constitute the triangle face of the intersection and read their corresponding UV coordinates.

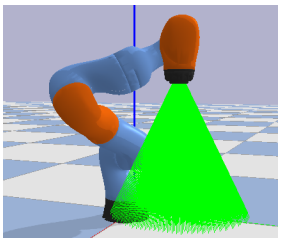


Fig. 2: Ray array to approximate a spray gun

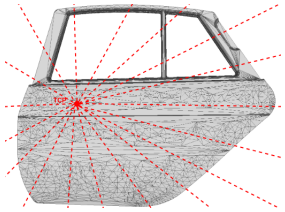
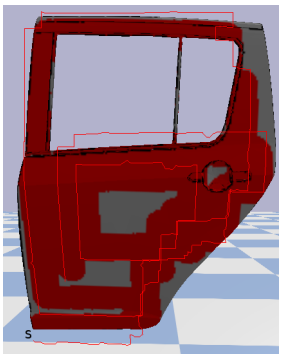
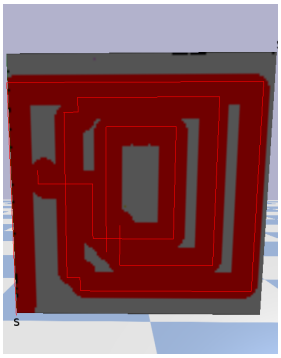


Fig. 3: Circular sectors used as low-dimensional observation



(a) Car door



(b) Quadratic sheet

Fig. 5: The generated coverage path after 150 action steps

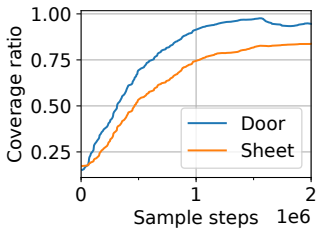
3) Use the UV coordinates from step 2 to calculate the pixels of the texture map that correspond to the point of intersection.
4) Change the color intensity of the affected pixels according to the paint thickness profile defined by the beta distribution.
5) Apply the modified texture map to the object.

III. Experiments

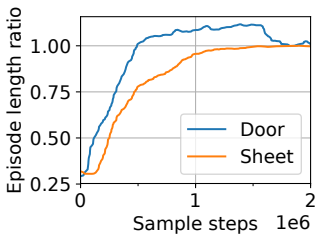
In order to use our spray painting simulation for RL experiments, we formalize the coverage path planning as a Markov Decision Process $(\mathcal{S}, \mathcal{A}, P_a, R_a)$. As depicted in figure 3, we subdivide the workpiece into circular sectors relative to the tool center point (TCP) and calculate the ratio of unpainted and total pixels for each sector. The state is composed of these ratios and the TCP position. We use Proximal Policy Optimization (PPO) [18] with four discrete actions (move up, left, down, right) to train a fully connected neural network consisting of two hidden layers with 256 and 128 neurons. We assume that a pixel can either be painted or unpainted and allow overlapping. Reward is given according to the number of newly painted pixels per action step deducting a time penalty to encourage low cycle times. An episode is terminated if the spray paint does not hit the workpiece for more than five action steps in a row. We train one network on a car door and a second network on a quadratic sheet.

IV. Results

After training the networks for two million steps, we check the performance against a zigzag baseline with respect to the coverage ratio and cycle time. For both, the car door and the quadratic sheet, 235 action steps are required to achieve full coverage with the baseline. As can be seen in figure 4(b), the mean time required to paint the workpieces is equivalent to the baseline. In case of the car door, a path that leads to full coverage can be found when evaluating the policy without exploration noise. The maximum coverage ratio achieved on the quadratic sheet is 93 %. The results are plausible since a zigzag pattern performs optimal on a sheet, but less-than-ideal on a car door due to the door window. Figure 5 shows the paths generated by the neural networks after 150 action steps. The path follows a spiral pattern and the network has learned to avoid the door window. However, since the robot can only move along four directions, rounded shapes lead to a decline in performance. We plan to address this issue in future work by increasing the dimensionality of the action space.



(a) Mean coverage ratio



(b) Mean episode length relative to zigzag baseline

Fig. 4: Network performance during the training process

V. Projection mapping for Sim2Real transfer

We use the projection mapping software Splash [19] to transfer the results from simulation to a real car door. As can be seen in figure 1, an ultra short throw projector is required in order to avoid shadowing. In future work, we intend to use projection mapping to gather human demonstrations by guiding the end effector of a robot and capture real-world images to learn a policy from pictures rather than a low-dimensional observation.

VI. Conclusion

We presented a framework which enables RL experiments for industrial spray painting and a way to transfer the results to the real world. With the simplifications mentioned above, a policy can be learned that leads to complete paint coverage of a car door. Nevertheless, due to overlap, the path is not optimal. Further research is required to evaluate the full potential of RL-based optimization for industrial spray painting.

References

[1] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–8.

[2] M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray *et al.*, "Learning dexterous in-hand manipulation," *arXiv preprint arXiv:1808.00177*, 2018.

[3] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," *arXiv preprint arXiv:1804.10332*, 2018.

[4] L. Berscheid, T. Rühr, and T. Kröger, "Improving data efficiency of self-supervised learning for robotic grasping," *arXiv preprint arXiv:1903.00228*, 2019.

[5] 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.

[6] (2019) Bullet physics sdk. [Online]. Available: https://github.com/bulletphysics/bullet3

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

[8] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox, "Closing the sim-to-real loop: Adapting simulation randomization with real world experience," *arXiv preprint arXiv:1810.05687*, 2018.

[9] S. James and E. Johns, "3d simulation for robot arm control with deep q-learning," *arXiv preprint arXiv:1609.03759*, 2016.

[10] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel, "Asymmetric actor critic for image-based robot learning," *arXiv preprint arXiv:1710.06542*, 2017.

[11] J. Domnick, A. Scheibe, and Q. Ye, "The simulation of the electrostatic spray painting process with high-speed rotary bell atomizers. part i: Direct charging," *Particle & Particle Systems Characterization*, vol. 22, no. 2, pp. 141–150. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/ppsc.200400968

[12] A. Mark, B. Andersson, S. Tafuri, K. Engstrom, H. Sorod, F. Edelvik, and J. S. Carlson, "Simulation of electrostatic rotary bell spray painting in automotive paint shops," *Atomization and sprays*, vol. 23, no. 1, 2013.

[13] J. Domnick, A. Scheibe, and Q. Ye, "The simulation of electrostatic spray painting process with high-speed rotary bell atomizers. part ii: External charging," *Particle & Particle Systems Characterization*, vol. 23, no. 5, pp. 408–416. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/ppsc.200601018

[14] (2019) Ips virtual paint spray website. [Online]. Available: http://www.fcc.chalmers.se/software/ips/ips-virtual-paint-spray

[15] H. Chen, W. Sheng, N. Xi, M. Song, and Y. Chen, "Cad-based automated robot trajectory planning for spray painting of free-form surfaces," *Industrial Robot: An International Journal*, vol. 29, pp. 426–433, 10 2002.

[16] T. Balkan and M. S. Arikan, "Modeling of paint flow rate flux for circular paint sprays by using experimental paint thickness distribution," *Mechanics research communications*, vol. 26, no. 5, pp. 609–617, 1999.

[17] (2019) blender.org - home of the blender project. [Online]. Available: https://www.blender.org/

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

[19] (2018) Splash, a multi-projector video-mapping software. [Online]. Available: https://github.com/paperManu/splash