Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 134 (2025) 939–944

www.elsevier.com/locate/procedia

Proceedings of the 58th CIRP Conference on Manufacturing Systems 2025

Enhancing Visual Inspection in Remanufacturing: A Reinforcement Learning Approach with Integrated Robot Simulation

Dominik Koch[a,*], Jan-Philipp Kaiser[a], Florian Stamer[a], Rainer Stark[c], Gisela Lanza[a,b]

[a]*wbk Institute of Production Science, Karlsruhe Institute of Technology (KIT), Kaiserstr. 12, 76131 Karlsruhe, Germany*
[b]*Global Advanced Manufacturing Institute (GAMI), KIT China Branch, Suzhou, 215123, China*
[c]*Chair of Industrial Information Technology, TU Berlin, Germany*

Abstract

This paper presents an approach for solving the View Planning Problem (VPP) in robotic inspection using Reinforcement Learning (RL). Building on a prior framework, this work takes a significant step forward by integrating a detailed robotic simulation environment with essential modules for trajectory and reachability planning. This allows for the development of an RL agent that not only selects adaptive viewpoints but also considers kinematic constraints and collision-free paths, which are crucial for practical, real-world inspections. The study specifically targets the initial inspection of returned products with high variability, demonstrating the feasibility of RL to manage complex tasks in remanufacturing. The RL-based solution is evaluated using Soft Actor-Critic (SAC) and Proximal Policy Optimization algorithms, with SAC showing superior performance. The learned strategies were validated on a real inspection station, showing the capability of using RL based inspection strategies. This research offers a robust, adaptable solution for inspection challenges, bridging the gap between theoretical models and application-ready inspection systems.

Keywords: Visual inspection; Automation; Reinforcement Learning; Robotics

1. Introduction

Companies are increasingly tasked with integrating remanufacturing strategies due to the anticipated scarcity of global resources [1], [2]. Remanufacturing focuses on restoring used products to a like-new condition, requiring accurate inspection and reconstruction of products to assess their suitability for further remanufacturing steps [3]. A crucial initial phase in remanufacturing is the optical inspection to form a preliminary assessment of its condition, guiding subsequent processing decisions.

Robotic inspection has emerged as a pivotal research area to enhance the efficiency and accuracy of this initial assessment [4]. Traditionally, these systems use predefined tool paths based on digital models. However, returned products often deviate from their original CAD models due to wear, modifications, or damage [3], necessitating adaptive strategies where fixed-path methods fall short [5]. Addressing these challenges centers around solving the View Planning Problem (VPP) for optimal coverage and efficient inspection. Building on previous research that developed a RL framework for the VPP [6, 7], this paper extends this framework by integrating a detailed robot simulation and trajectory planning module as well as validating the method on the real inspection station.

2. State of the Art

This section provides an overview of the current research landscape related to view planning and the application of RL for adaptive robotic inspection.

2.1. View Planning

View planning is the process of determining optimal sensor poses for inspecting or reconstructing an object. The objective

* Corresponding author. Tel.: +49-1523-9502626
E-mail addresses: dominik.koch@kit.edu (Dominik Koch)., jan-philipp.kaiser@kit.edu (Jan-Philipp Kaiser)., florian.stamer@kit.edu (Florian Stamer)., rainer.stark@tu-berlin.de (Rainer Stark)., gisela.lanza@kit.edu (Gisela Lanza).

is to achieve complete coverage of the target object while minimizing the number of views and total path length [8].

This problem is inherently sequential, as each new view must build upon previously acquired data to optimize coverage and efficiency. RL has proven effective for sequential decision-making problems [9], offering an agent-based approach where agents interact with an environment, perceive its state, make decisions, and receive feedback based on their actions [10]. The effectiveness of RL has been demonstrated in various applications within robotics, including path planning, object manipulation, and complex production system control [11, 12].

2.2. View Planning using Heuristic and Analytical Solutions

Heuristic and analytical solutions, such as those benchmarked in [6], offer fixed-path strategies for view planning using predefined models. Heuristic methods use preset rules for the selection of a sensor pose, e.g. positioning sensor poses based on basic geometric structures, while analytical approaches often involve extensive sampling of viewpoints to identify optimal poses. However, these methods face notable limitations: Heuristic solutions may struggle with adaptability for complex geometries, and analytical methods can require substantial computation times due to the exhaustive evaluation of multiple viewpoints. RL offers a promising alternative, enabling adaptive decision-making with reduced computation time.

2.3. View Planning using Reinforcement Learning

RL has emerged as a powerful tool for acquisition planning in various domains. Early works in active object recognition, such as [13] and [14], focused on next-best-view strategies to improve object classification by using RL to select optimal viewpoints. Kaba et al. [15], who formulated the VPP as a Markov Decision Process, and applied RL to solve it, demonstrating the potential of RL in view planning tasks. Building on these foundations, subsequent studies explored different representations of the environmental state to enhance VPP resolution. For example, [16] utilized occupancy grids and sensor poses, allowing more precise planning in complex environments. However, these approaches often oversimplified the problem by relying on low-resolution grids or fixed sensor poses, which could hinder the development of optimal strategies for intricate geometries. More recent work, such as [17], [6], and [7], has addressed some of these limitations. These studies demonstrated that RL agents could achieve competitive surface coverage with fewer acquisitions and substantially faster computation times. This efficiency makes them suitable for runtime use in remanufacturing, where variability and model-free scenarios are common. Despite significant advancements, existing methods have yet to fully integrate realistic robotic simulations that account for trajectory planning and kinematic constraints, key aspects needed for practical, model-free runtime scenarios in remanufacturing.

3. Approach

Building upon the foundation laid in previous publications [6] and [7], which introduced a comprehensive framework for addressing the VPP through RL approaches, this document extends the framework by integrating a robot simulation and modules essential for trajectory planning. The objective of this extension is to illustrate the practical applicability of RL within a simulated environment and to bridge the Sim2Real gap, enhancing the transferability of learned strategies to real-world applications.

The framework employed for VPP solution is depicted in Figure 1. Central to this approach is the simulation environment S_{Sim}, which embodies a detailed representation of a real-world system comprising a robot, a structured light 3D scanner, and a rotary clamping system. This scanner is capable of capturing data with a level of precision comparable to human visual assessment, enabling informed decision-making regarding the product.

The agent interface, denoted as I_{Agent}, serves as a crucial bridge, translating the actions generated by the planning agent A_{Plan} into executable poses for the acquisition system. Additionally, it is responsible for computing the system's status and determining the reward. The individual components of S_{Sim}, I_{Agent}, and A_{Plan} are further elaborated in the subsequent sections.

3.1. Simulation Environment

S_{Sim} contains a scanning module and uses sensor and object models to perform simulative acquisitions. Both are explained in detailed in prior work [6].

The object model is specified in STL format, representing the object´s surface as triangular facets. Ray tracing is used to simulate a 3D acquisition process.

An acquisition at step t triggers an observation update from step $t-1$ to t. The observation vector $\mathbf{o}_t$ stores all information given to I_{Agent} to deduce states $\mathbf{s}_t$ and to calculate the reward $\mathbf{r}_t$ given to the agent. This information is then used for training of a RL agent.

The simulation environment represents a core contribution of this work, establishing a highly detailed yet efficient virtual inspection station. This station mirrors the setup as shown in Figure 1. Positioned on one of these modules is a UR10e robot equipped with a Zivid RGB-D camera to enable adaptive acquisition view planning. This arrangement allows comprehensive inspections from a range of perspectives.

The simulation environment is constructed within the *ROS* (Robot Operating System) framework and utilizes *MoveIt* for motion planning. Key to accurately replicating the physical station is the definition of kinematic relationships and coordinate systems. The robot operates with two primary coordinate systems: C_R for the base and C_E for the end-effector. The rotary clamping system has a designated coordinate system, C_{Rot}, while the 3D acquisition setup is defined by C_A. Additionally, each inspection object is assigned a coordinate system, C_0.

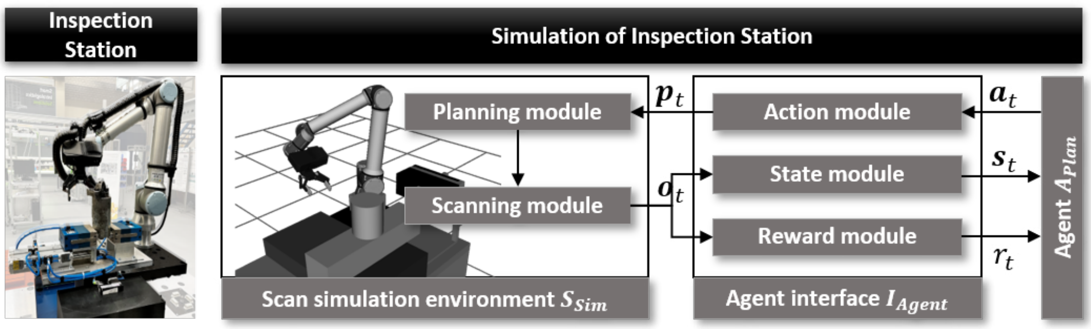

Fig. 1. Left: Real setup of the reconstruction station including robot, acquisition system, and rotary clamping system. Right: Overview of the simulation framework proposed in this work.

Transformations between these coordinate systems, such as $T_{B\rightarrow Rot}$ (from the robot base to the rotary table) and $T_{E\rightarrow A}$ (from the end-effector to the acquisition system), are calculated using hand-eye calibration techniques. These calibrated transformations ensure the virtual environment accurately emulates the physical station's spatial relationships, providing a reliable training platform for RL agents.

3.2. Planning Module

The planning module within the simulation environment is essential for the RL agent's training, offering trajectory planning and collision-free reachability assessment. To facilitate this, the module comprehensively models the kinematic behavior of the inspection station, leveraging *ROS* and *MoveIt* to implement efficient motion planning algorithms.

Given the transformations between coordinate systems, the planning module can dynamically assess a target pose $\mathbf{p}_t$ for feasibility based on the current configuration of the robotic system. The module performs collision checks and plans trajectories that ensure safe navigation of the end-effector while avoiding any obstructions in the simulated space. Only poses that meet these criteria are returned as valid, allowing the agent to explore potential strategies without risk of unrealistic or damaging operations.

The efficiency of the planning module is of critical importance due to the high sampling rates required for RL processes. By streamlining the trajectory calculations and minimizing computational overhead, the module supports rapid pose validation and trajectory generation. This is achieved through optimized path planning strategies within *MoveIt*, enabling the environment to maintain low computation times and ensure the RL training progresses smoothly without delays.

3.3. State Module

The state module encodes the information contained in $\mathbf{o}_t$ into a format suitable for the RL agent A_{Plan}. It is important to note that various state modeling strategies were explored in prior research, as described in [6]. The approach yielding the best results was selected for this work, ensuring an effective state representation tailored to the RL agent's requirements. Specifically, the state modeling $\mathbf{s}_t$ involves providing the agent with a downsampled representation of the cumulative point cloud $\mathbf{P}_{\text{cov},t-1}$ gathered up to step t in the current episode. The state representation $\mathbf{s}_t$ is structured as a matrix that encodes the coordinates of each point m in the downsampled point cloud. Each row of the matrix corresponds to an individual point, defined by its 3D coordinates (x, y, z).

3.4. Action module

The action module is responsible for managing the actions defined by the RL agent. Various mapping strategies and action types were explored in prior work [6], and the action variants selected for this study showed the best performance in terms of coverage and efficiency. These approaches were expanded here to incorporate control over the rotary system, ensuring adaptability in the acquisition process. In this work, two action variants are evaluated: 3T1R and 3T3R.

3T1R: The 3T1R action representation consists of an action vector with four components, each constrained to the range $[-1, 1]$. The first three components represent the translational parameters, expressed in spherical coordinates ϕ, θ, and r, which are used to calculate the x, y, and z coordinates for the camera pose. The orientation of the camera is determined heuristically, using the Euler angles α and β in the x-y'-z'' convention, while the angle γ is fixed at zero to account for the near rotational symmetry of the camera's frustum. The fourth component (1R) of the action vector controls the rotation angle ϑ of the rotary clamping system, enabling the agent to adjust the object's positioning on the turntable.

3T3R: The 3T3R action representation expands upon the 3T1R structure by including additional control over the camera's orientation. Here, the agent outputs an action vector with six components. The first three entries are identical to those in 3T1R, representing the spherical coordinates ϕ, θ, and r for translational positioning. The additional three components include two entries that allow the agent to adjust the Euler angles α and β for fine-tuning the camera's orientation, and one entry for controlling the rotation angle ϑ of the rotary clamping system (3R).

The agent's output, limited to the range $[-1, 1]$, is mapped to the actual operational range of the parameters. For instance, angular corrections for α and β are mapped to $[-10°, 10°]$. The spherical coordinates are restricted to $\phi \in [45°, 135.5°]$ and $\theta \in [22.5°, 67.5°]$, reflecting the robot's limited range of motion. Despite these limitations, the full surface of the object can still be covered through the use of the rotary table, which allows for complete rotation with $\vartheta \in [0°, 360°]$. The radius parameter r is mapped to the interval [0 cm, 100 cm], corresponding to the working range of the 3D scanner.

3.5. Reward module of I_{Agent}

The agent learns based on the reward. The aim is to achieve a given surface coverage of the object while minimizing the number of images required. In this work, dense and sparse reward functions to achieve this goal are evaluated and compared. Dense rewards R_{Dense} are given to the agent after each individual action, while sparse rewards R_{Sparse} are only provided at the end of an episode.

$$R_{Dense} = \frac{\Delta\Psi_t}{100 - \Psi_{t-1}} \quad (1)$$

$$R_{Sparse} = \begin{cases} \frac{\Psi_t}{n_t} & ,goal\ reached \\ -1 & ,else \end{cases} \quad (2)$$

R_{Dense} rewards the agent based on the percentual coverage of surface area of the inspection object $\Delta\Psi_t$ in the last acquisition that has not yet been covered by previous acquisitions, thus aiming at maximizing object surface coverage. Furthermore, R_{Dense} scales $\Delta\Psi_t$ by the maximum possible surface area of the inspection object to be covered in the last acquisition to always set the maximum reachable reward to 1. In a sparse reward setting, the agent only receives a reward once at the end of an episode when reaching the learning goal. R_{Sparse} is based on the coverage Ψ_t divided by the number of acquisitions n_t which were needed to reach the goal. The goal of an agent is reached when the surface coverage exceeds a threshold of 90% ($\Psi_t \geq 90\%$). An episode is additionally terminated when the agent has performed 10 successive acquisitions and did not reach the learning goal. In that case, the agent does not receive a reward.

3.6. Agent A_{Plan}

Due to the continuous state and action space, *Proximal Policy Optimization* (PPO) and *Soft Actor-Critic* (SAC) are evaluated as suitable state-of-the-art methods for this task. These algorithms are implemented using the RL library stable-baselines3 [18], offering seamless compatibility with OpenAI Gym environments.

Both PPO and SAC employ an actor-critic policy framework, leveraging neural networks for action selection. For feature extraction from the input point clouds, we use the Point Completion Network encoder [19], which is a lightweight variant of PointNet. This encoder is integrated to efficiently process point cloud data. Fully connected layers are appended to the feature extraction network to facilitate the agent's action output. The feedforward network structure is configured to output the appropriate number of free parameters, with four parameters for 3T1R and six for 3T3R.

Table 1. Hyperparameters of the learning algorithms

Learning parameters	
Learning rate	8e-5*
Gamma	0.9
Train frequency	8 episodes
Number of training steps	150,000
Environment modelling	
Required object coverage ψ	90%
Maximum acquisitions	10
Feed forward structure	256-128-64-[output dim]**

* Linear decay, with final value of 0

** Output dimension for 3T1R is 4 and for 3T3R the dimension is 6

In prior work detailed in [6], various network architectures were evaluated to identify the optimal configuration. The chosen architecture demonstrated superior performance in terms of achieving high coverage while minimizing the number of scans and computation time.

4. Use Case: Starter Motor

A data set of synthetically generated starter motors is used to represent the large number of product variants that can occur in remanufacturing. The data set is generated based on a presented pipeline (cf. [20]). The motors are randomly generated based on 9 basic components (e.g. solenoid, gear housing and flange). The individual components are further varied by 28 different parameters (e.g. length or diameter). Realistic parameter limits and defined relationships to each other guarantee that a diverse and realistic data set is generated. We generated 100 such engines with random parameters in STL format for agent training. Two examples are shown in Figure 2.

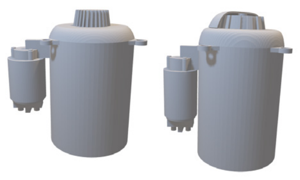

Fig. 2. Synthetically generated starter motors

4.1. Learning setup and evaluation criteria

To investigate the extent to which RL is suitable for solving the problem, the RL algorithms PPO and SAC are compared with each other. In addition, a random agent is used, which chooses a random action in each step. The agent is also benchmarked against an often-used analytical solution method that formulates the VPP as a Set Covering Problem (SCP) and solves it. VPP solution via solving the SCP involves discretizing and evaluating multiple acquisition system poses, and then

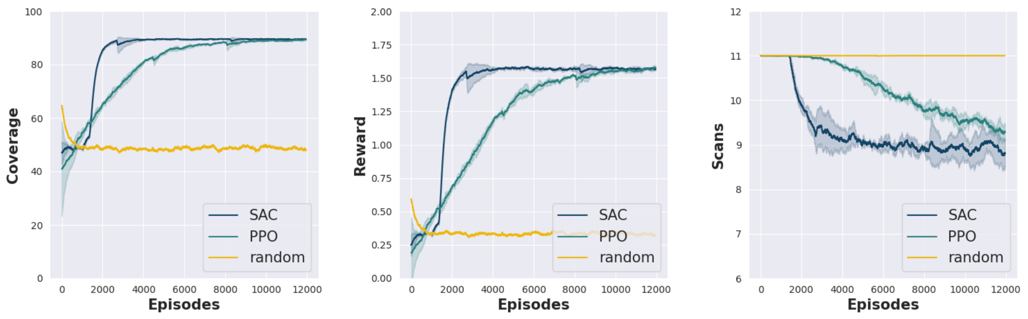

Fig. 3. Comparison of coverage, reward and number of acquisition (scans) for different learning algorithms and a random benchmark

sequentially selecting them based on their coverage of the object's surface to obtain a view plan. Due to the random behavior of an RL agent, three training runs were performed for each agent and averaged. For the training, the parameters depicted in Table 1 were used. The training results are shown in Figure 3. The following performance metrics are introduced to evaluate the success of the training. Firstly, the cumulative reward of the agent over an episode is used. This allows a statement to be made as to whether the agent learns to optimize the strategy. Coverage is also considered. This is the agent's main objective. The total surface area should be maximized, and a target state is only reached when at least $\Psi = 90\%$ total coverage is achieved. The number of scans required for this is also considered. In order to optimize in the context of the VPP and to increase efficiency, these should be minimized as far as possible while achieving the surface coverage of $\Psi = 90\%$. These performance metrics are plotted against the number of episodes in Figure 3. An episode comprises the acquisition process of an object model by the agent. Since several agents were trained for each agent configuration, the diagrams in Figure 3 contain the average progression of the performance metrics in a bold line. In addition, the minimum and maximum values of all trained agents for each episode are shown.

4.2. Comparison of action modeling alternatives

The action variants described in section 3.4 are compared below. The results of the training are depicted in Table 2. The last 500 episodes in the converged state are averaged over 3 runs for each action variant. 3T1R achieves a higher average coverage and reward than 3T3R. By achieving the target of $\Psi >$ 90% more reliable, 3T1R requires an average of 1.5 scans less than 3T3R. In average 3T3R chooses one pose per episode that is deemed non-reachable by *MoveIT*. This can be attributed to the additional variation of α and β.

Table 2. Training results for different reward and action types (mean values)

Metrics	Reward Types		Action Types	
	Dense	Sparse	3T1R	3T3R
Coverage	89.62	42.54	89.62	87.88
Reward	1.57	-10.0	1.57	1.50
Scans	8.60	11.0	8.60	10.17

4.3. Comparison of reward modeling alternatives

For reward comparison, the averaged results of these training runs are shown in Table 2. If the agent is rewarded by a dense reward function based on the acquired object surface of a step, the performance indicators used can be optimized. The required target of $\Psi > 90\%$ can be reliably achieved and the number of scans required can be reduced accordingly. In contrast, a reward that is only distributed at the end of an episode and is only positive when a target state $\Psi > 90\%$ is reached is not effective. In this scenario, reaching a target state is too difficult to achieve without an accompanying dense reward.

5. Validation on the real inspection station

Finally, the methodology is also validated at the real inspection station with used starter methods. For this purpose, two agents, A_{Sparse} and A_{Dense}, are trained in the simulation environment and then transferred to the real inspection station. Although 3T1R performed better in the simulation, 3T3R was chosen for real-world validation to verify if the simulation results hold in a more complex, realistic scenario. Since the use of the sparse reward, as shown in Table 2, does not lead to any learning success, some adjustments were made for A_{Sparse} for use at the real station. On the one hand, the sparse reward is always assigned at the end of an episode, even if the target state $\Psi > 90\%$ is not reached but terminated after 10 scans. Furthermore, the action space was adapted to 3T1R, since the additional rotation made reward assignment more difficult. Both agents are benchmarked against a random strategy and a heuristic. In the heuristic, a fixed viewpoint is set above the rotary table, with the acquisition system aligned horizontally. The motor rotates 60° per acquisition, totaling six. Results shown in Figure4 indicate that both A_{Sparse} and A_{Dense}, trained in a virtual environment, successfully transfer to the real inspection station, outperforming the random benchmark. Within five acquisitions, both agents achieve 80–90% surface coverage. A_{Sparse}, trained with a sparse reward, reaches comparable coverage to A_{Dense} with fewer acquisitions, confirming that simulation find-

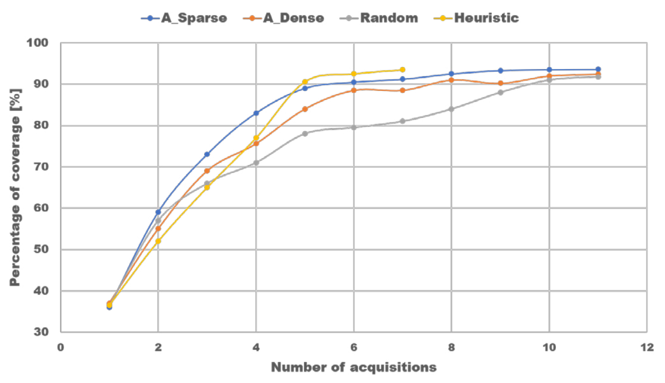

Fig. 4. Comparison of coverage and number of acquisition (scans) for different learning algorithms and benchmarks

ings translate well to real conditions. Notably, both agents surpass the heuristic benchmark in early acquisitions (1–4), as the heuristic requires multiple rotations (60° increments) for full coverage. The RL agents' strategies prioritize maximizing coverage in each acquisition, though potentially impacting later stages. This shows the relevance of reward design in reinforcement learning, which is essential to achieving the goal. Overall, simulation results align with real-world outcomes, with the heuristic slightly outperforming; this gap may be reduced by fine-tuning the RL agents for the real application.

6. Conclusion

In conclusion, this work developed and evaluated a Reinforcement Learning (RL) model to maximize the coverage in the View Planning Problem (VPP) in robotic inspection. RL algorithms show promising performance, consistently achieving over 90% surface coverage in virtual tests. Furthermore, real validation tests have shown that the strategies learned in the simulation can also be transferred to the real use case. A key advancement of this work is the integration of robot simulation and trajectory planning, allowing the RL agent to consider kinematic constraints and physical reachability for greater adaptability. This enabled the methodology to be validated on a real use case and the framework to be confirmed as a promising methodology for solving the VPP. This enables efficient, adaptive view planning for returned products, supporting inspections to inform further processing.In future work, we will enhance the framework by incorporating the total path length, measurement uncertainty estimation and improving sample efficiency.

Acknowledgements

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - SFB 1574 - Project number 471687386

References

[1] Parker, D., Riley, K., Robinson, S., Symington, H., Tewson, J., Jansson, K., Ramkumar, S. & Peck, D. Remanufacturing market study. (European Remanufacturing Network (ERN), 2015)

[2] Stamer, Florian and Fabig, Franziska. "Analysis and evaluation of adaptive remanufacturing strategies for mechanical products" at - Automatisierungstechnik, vol. 72, no. 9, 2024, pp. 884-892.

[3] Khan, A., Mineo, C., Dobie, G., Macleod, C. & Pierce, G. Vision guided robotic inspection for parts in manufacturing and remanufacturing industry. *Journal Of Remanufacturing*. **11**, 49-70 (2021)

[4] Chrysostomou, D., Kyriakoulis, N. & Gasteratos, A. Multi-camera 3D scene reconstruction from vanishing points. *2010 IEEE International Conference On Imaging Systems And Techniques*. pp. 343-348 (2010)

[5] Stark, R., Grosser, H. & Müller, P. Product analysis automation for digital MRO based on intelligent 3D data acquisition. *CIRP Annals*. **62**, 123-126 (2013)

[6] Kaiser, J., Gäbele, J., Koch, D., Schmid, J., Stamer, F. & Lanza, G. Adaptive acquisition planning for visual inspection in remanufacturing using reinforcement learning. *Journal Of Intelligent Manufacturing*. (2024)

[7] Kaiser, J., Koch, D., Gäbele, J., May, M. & Lanza, G. View planning in the visual inspection for remanufacturing using supervised- and reinforcement learning approaches. *CIRP Journal Of Manufacturing Science And Technology*. **53** pp. 128 - 138 (2024)

[8] Scott, W., Roth, G. & Rivest, J. View planning for automated three-dimensional object reconstruction and inspection. *ACM Computing Surveys (CSUR)*. **35** pp. 64 - 96 (2003)

[9] Sutton, R. & Barto, A. Reinforcement Learning: An Introduction. (MIT Press,1998)

[10] Panzer, M. & Bender, B. Deep reinforcement learning in production systems: a systematic literature review. *International Journal Of Production Research*. **60**, 4316-4341 (2022)

[11] Kuhnle, A., Schäfer, L., Stricker, N. & Lanza, G. Design, Implementation and Evaluation of Reinforcement Learning for an Adaptive Order Dispatching in Job Shop Manufacturing Systems. *Procedia CIRP*. **81** pp. 234-239 (2019), 52nd CIRP Conference on Manufacturing Systems (CMS), Ljubljana, Slovenia, June 12-14, 2019

[12] Kober, J. & Peters, J. Reinforcement Learning in Robotics: A Survey. *Reinforcement Learning: State-of-the-Art*. pp. 579-610 (2012)

[13] Deinzer, F., Derichs, C., Niemann, H. & Denzler, J. A Framework for Actively Selecting Viewpoints in Object Recognition. *Int. J. Pattern Recognit. Artif. Intell.*. **23** pp. 765-799 (2009)

[14] Korbach, C., Solbach, M., Memmesheimer, R., Paulus, D. & Tsotsos, J. Next-Best-View Estimation based on Deep Reinforcement Learning for Active Object Classification. *ArXiv*. **abs/2110.06766** (2021)

[15] Kaba, M., Uzunbas, M. & Lim, S. A Reinforcement Learning Approach to the View Planning Problem. *2017 IEEE Conference On Computer Vision And Pattern Recognition (CVPR)*. pp. 5094-5102 (2017)

[16] Potapova, S., Artemov, A., Sviridov, S., Musatkina, D., Zorin, D. & Burnaev, E. Next Best View Planning via Reinforcement Learning for Scanning of Arbitrary 3D Shapes. *Journal Of Communications Technology And Electronics* (2020).

[17] Landgraf, C., Meese, B., Pabst, M., Martius, G. & Huber, M. A Reinforcement Learning Approach to View Planning for Automated Inspection Tasks. *Sensors (Basel, Switzerland)*. **21** (2021)

[18] Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M. & Dormann, N. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal Of Machine Learning Research*. **22**, 1-8 (2021),

[19] Yuan, W., Khot, T., Held, D., Mertz, C. & Hebert, M. PCN: Point Completion Network. *2018 International Conference On 3D Vision (3DV)*. pp. 728-737 (2018)

[20] Wu, C., Zhou, K., Kaiser, J., Mitschke, N., Klein, J., Pfrommer, J., Beyerer, J., Lanza, G., Heizmann, M. & Furmans, K. MotorFactory: A Blender Add-on for Large Dataset Generation of Small Electric Motors. *Procedia CIRP*. **106** pp. 138-143 (2022)