CIRP Journal of Manufacturing Science and Technology 53 (2024) 128–138

Contents lists available at ScienceDirect

CIRP Journal of Manufacturing Science and Technology

journal homepage: www.elsevier.com/locate/cirpj

Full-length article

View planning in the visual inspection for remanufacturing using supervised- and reinforcement learning approaches

Jan-Philipp Kaiser [a,*], Dominik Koch [a], Jonas Gäbele [a], Marvin Carl May [a], Gisela Lanza [a,b]

[a] wbk Institute for Production Science, Karlsruhe Institute of Technology (KIT), Kaiserstraße 12, Karlsruhe, 76131, Germany
[b] Global Advanced Manufacturing Institute (GAMI), KIT China Branch, Suzhou, 215123, China

ARTICLE INFO

Keywords:
Remanufacturing
View planning
Supervised learning
Reinforcement learning

ABSTRACT

Visual inspection in remanufacturing, despite technological progress, is still mainly performed by humans. A rough assessment of the product's general condition and the dedicated inspection of individual product features or defects is necessary to identify the typically unknown product variant and assess the reusability of a used product and its components. Therefore, a system for automated visual inspection must be flexible and runtime-adaptive, as defects to be inspected in detail may occur anywhere on the product. In the present work, this problem is framed as a view planning problem solved by means of supervised learning and reinforcement learning using a specially developed simulation environment. Three variants of neural networks (PointNet, PointNet++, and Point Completion Network) are compared in the supervised learning case, whereas a deep learning SAC algorithm using the Point Completion Network as network structure is evaluated in the reinforcement learning case. Considering the specific boundary conditions prevailing in remanufacturing, the results are obtained from the use case of electric starter motor remanufacturing. The results show that supervised learning and reinforcement learning are suitable for determining the poses of an acquisition system at system runtime to react to an initially unknown inspection task. Our proposed framework is available open source under the following: https://github.com/Jarrypho/View-Planning-Simulation.

1. Introduction

Artificial intelligence is becoming increasingly relevant in production-related applications [1,2]. These include applications in logistics and the supply chain, maintenance management, production planning, and control, as well as quality management [3]. Research in the latter, for example, in robotics, presents an additional possibility for creating productivity in the context of production technology [4]. To realize this, robots' flexibility and versatility can be combined with intelligent approaches to enable self-learning and autonomous behavior [5]. Thus, the increase of automation of tasks originally performed by humans is enabled. This allows for counteracting the constantly growing shortage of skilled workers, the associated cost pressure in high-wage countries, and the subsequent migration of production capacities to low-wage countries. Furthermore, the automation of human visual inspection, which is error-prone, can reduce the number of misclassifications [6].

Such autonomous systems are particularly relevant in applications where information is incomplete and an incomplete information basis prevails. An example is remanufacturing, where uncertainty exists about the quality state and type of returned used products. This is why many remanufacturing processes, such as disassembly, reconditioning, and reassembly, are carried out manually. This also applies to the visual inspection process that precedes the actual remanufacturing [7].

The in-depth visual inspection takes place at a core dealer, an actor preceding the actual remanufacturer in the reverse value chain [8] or at the inbound acceptance in the remanufacturing plant [7]. Automating these inspection steps is still difficult due to the prevailing uncertainty. A reason for this is their highly variable quality condition and the fact that it is difficult to predict the type, amount, location, and severity of defects that will occur. Furthermore, additional information accompanying the product, such as its three-dimensional product model, is often unavailable [9]. This applies to the core dealers, and the remanufactures themselves. Independent remanufacturers that are not Original Equipment Manufacturers (OEM) often only have product information that was acquired through reverse engineering [10]. But even within the inbound acceptance in the remanufacturing plant of an OEM, when potentially using automated visual inspection, product identification would have to be carried out as the first step to decide on the specific inspection routines for the product at hand. This, in turn, is

* Corresponding author.
E-mail address: jan-philipp.kaiser@kit.edu (J.-P. Kaiser).

https://doi.org/10.1016/j.cirpj.2024.07.006
Received 26 April 2023; Received in revised form 23 January 2024; Accepted 15 July 2024
Available online 31 July 2024
1755-5817/© 2024 The Authors. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

a research subject currently being investigated, as a distinction usually has to be made between several hundred different variants [11].

Due to the introduced challenges, it is, therefore, challenging to plan inspection routines in advance [9]. In contrast to existing approaches in the state of research, inspection routines must be defined at system runtime using the adaptive capabilities of robot-based visual inspection systems in combination with intelligent planning techniques [12,13].

The present work addresses the acquisition planning of a robot-guided inspection system to enable a decision regarding the remanufacturability of a returning used product. This acquisition planning must be realized under the initial uncertainty of the existing three-dimensional product model and the occurring defects. Within this work, no evaluation of the image and geometry data collected during the acquisition is intended. Instead, this work lays the foundation for solving the relevant acquisition planning problems so that further work can be devoted to the evaluation.

The work is structured as follows. First, the foundations relevant to this work are presented, and the research deficit concerning adaptive visual acquisition planning is derived in Section 2.

In Section 3, a formalization of the human visual inspection process of returning used products is conducted. Afterward, the framework presented in this work for solving visual acquisition problems with approaches based on machine learning for autonomous and adaptive visual inspection is described. Results of the work are presented in Section 4.

2. State-of-the-art

Two basic problems of visual planning exist in the literature. An overview of existing work in visual planning is given in the recently published article by [14]. First, the Next-Best-View problem (NBV) is the task of determining the next viewpoint that provides the highest possible information gain with respect to the given task (e.g., the complete surface coverage of an object) [15]. Secondly, the View Planning Problem (VPP) aims to find a minimum sequence of viewpoints that covers the entire surface of an object to be acquired [16].

Using machine learning, the NBV is mostly formulated and solved as a supervised learning (SL) problem in the literature (cf. [17–22]). The goal of SL is to approximate a function $g : X \rightarrow Y$, where X is the input space and Y is the output space, given a set of N training examples $\{(x_1, y_1), \ldots, (x_N, y_N)\}$ [23]. A model trained by SL thereby predicts $\hat{y}_i$ given an input x_i, where, ideally, the prediction $\hat{y}_i$ equals the true label y_i ($\hat{y}_i = y_i$). In existing work dealing with the NBV, the input space is mostly represented by some form of geometric object information. The function to be learned then corresponds to mapping the input space to the output space, which is formed from a continuous or discrete set of poses to be taken by the acquisition system. In most works using SL, the true label y_i given an input x_i is thereby determined by choosing the pose as label y_i from the set of all possible poses that maximizes the information gain regarding the objective of the task.

When using machine learning methods, the VPP is mostly modeled as a sequential pose estimation problem and solved with reinforcement learning (RL) (cf. [24–28]). In contrast to SL, RL is concerned with finding a problem-solving strategy π to solve a sequential problem [29]. An agent acts at time t with an action a_t in an environment represented through state s_t. This action triggers the environment to transform into a subsequent state s_{t+1} based on its environment dynamics. The agent receives a reward r_t based on the quality of the action in fulfilling a predefined goal. The goal of RL is then to learn the optimal strategy π^*. This strategy maximizes the Return G_t, which is the estimation of the sum of all future rewards received in a state through optimal action selection, and, thus, enables the fulfillment of the predefined goal in the best possible way [29]. In works to solve the VPP using RL, the action space is mostly modeled similarly to the SL problem. In contrast, the state space definition offers more degrees of freedom. Like in SL, geometric object information can be passed as state s_t to determine the next pose of the acquisition system as action a_t. However, passing information of past states $s_0, \ldots, s_{t-1}$ or previous poses of the acquisition system $a_0 \ldots a_{t-1}$ of a state–action sequence is possible.

Analyzing the state-of-the-art shows that works addressing the NBV problem are simplified with respect to the input, e.g., by using voxelization (see [17–20]) or the output, e.g., in form of classification on predefined viewpoints (see [17,19–22]). A general problem solution where point clouds as the direct unprocessed output of a 3D sensor are directly mapped to a sensor pose by machine learning does not exist in the literature so far. The use of simplifications of the input space, such as a voxelization or of the output space, by discretizing the continuous set of possible poses of the acquisition system, allows a proof-of-concept of the presented approaches but offers only approximate solutions for real problems.

Deficits can also be identified with regard to the specific requirements in remanufacturing. Existing approaches for determining sequential poses by means of RL currently only address the view-planning problem with explicit knowledge of the object model (e.g., triangle mesh, see [24,25,28]). However, knowledge of an object model cannot be assumed in remanufacturing since remanufacturers are often not manufacturers of the original product and, thus, have no or only difficult access to such information. Therefore, deriving the sequential order of poses must not depend on a three-dimensional object model at runtime. Similarly, no approach currently allows the sequential determination of poses for inspecting defects or flaws that are variably located on an object. To summarize, no work currently provides a general approach to solving the problem of visual acquisition planning problems, especially considering the specific requirements in remanufacturing.

The research questions that have to be answered are as follows:

1. How can the visual inspection of the worker be characterized? How can the characteristics of a worker's visual inspection without prior knowledge of the product to be inspected be formalized in a sequence of visual planning tasks?
2. How can artificial intelligence approaches be used and modeled to solve a formalized sequence of visual planning problems as generically as possible without simplifying and automating an initial visual inspection?

The research questions are answered using the proposed methodology using a real remanufacturing product, the starter motor. The visual planning problem in the initial visual inspection of a returning used product in remanufacturing is considered for the first time. The novelty of this work is based on the general formalization and solution of the inspection problem in remanufacturing by methods of supervised and reinforcement learning without an available geometry model of the inspection object. The advantage of choosing a starter motor is that it represents a real remanufacturing product and is often available in a wide range of variants and sizes. Nevertheless, the basic structure of the different variants is similar, which initially makes it somewhat easier to evaluate the intelligent visual planning approaches developed. Therefore, the starter motor is the ideal demonstrator product for applying intelligent visual planning processes.

The insights gained from the general formalization of the problem can also be transferred to other problems in industrial production.

3. Methodology - Modeling of the view planning simulation framework

In the following, the challenges of inspection planning in remanufacturing are transferred to known problems of vision planning (see Section 3.1). This is followed by a detailed description of a simple, generic, and modular simulation environment that can be used to solve the derived problems of vision planning using both supervised and RL-based approaches (see Section 3.2 and Sections 3.3 and 3.4 for details).

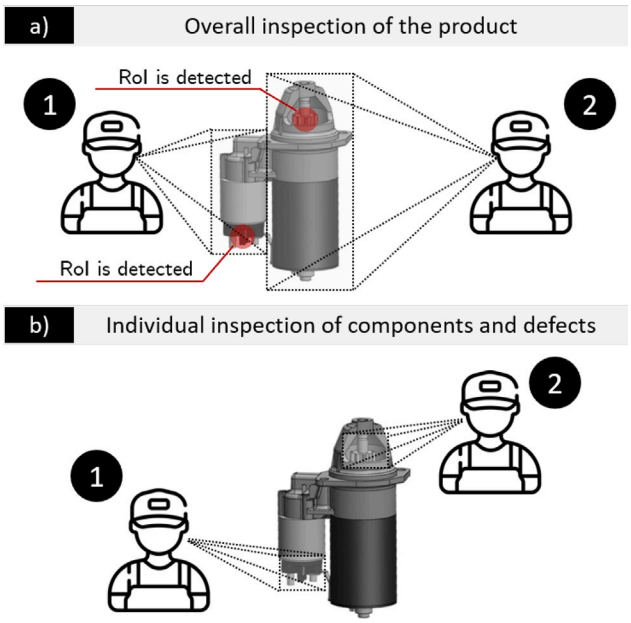

Fig. 1. Overview of the two distinct inspection problems addressed in this work. (a) Overall inspection of a product. (b) Individual inspection of components and defects.

3.1. Derivation of vision planning problems based on human inspection

The visual planning tasks relevant to the automation of visual inspection in remanufacturing can be derived from manual inspection by a worker. The so-called core acceptance criteria provide guidance on the relevant quality criteria to which particular attention is paid during the inspection [30]. Two types of occurring defects can be distinguished. First, the core acceptance criteria include defects that can be identified by overall surface inspection. Thereby, an object's whole surface must be examined to find all defects on the product. These include excessive corrosion and mechanical defects such as dents or broken and missing parts. Secondly, defects that require a detailed inspection are listed. These include overheated drive shafts, corroded pulleys, or burnt-out electrical connections and are only found on certain components. Based on these acceptance criteria, an inspector decides on the suitability of a product with regard to its reusability in a second life cycle after remanufacturing. The two types of inspection problems can be seen in Fig. 1.

Therefore, two fundamentally different visual planning problems can be derived from the inspection task of humans. First, (a) overall inspection of the product with full surface coverage is necessary to detect all possible defects on the entire product. In addition, an (b) individual inspection of components and defects, which represent Regions of Interest (RoI) that have been determined to be further examined by surface inspection, may be necessary.

The problem of overall surface inspection, as outlined in subproblem (a), can be modeled as a classical VPP. In this model, the solution comprises a sequence of viewpoints selected to enable the complete acquisition of an object's surface by an acquisition system. This process must satisfy certain constraints, such as minimizing the number of acquisitions and reducing the length of the robot's trajectory. This problem can be formulated as a sequential pose estimation problem and can be solved with RL.

The problem of individual inspection of single components or defects (subproblem (b)) can be solved using SL or RL. The problem to be solved is to find a pose of an acquisition system to inspect a certain RoI. In the case of several RoIs on one object, this task can again be converted into a sequential pose estimation problem, which can be solved with RL. Another possibility is to solve the entire problem by combining the solution of individual solutions. For each RoI to be inspected, a SL problem can be formulated, where the set of estimated poses covering all RoIs must be transformed into a trajectory of the robot to obtain the overall solution to cover all RoIs.

Considering both sub-problems, an inspection sequence looks as follows. First, subproblem (a) is solved. Assuming a missing geometry model of the inspection object (starter motor), the acquisition system's poses must be selected to cover the inspection object's surface completely. A suitable acquisition system is an RGBD system that acquires image data as well as geometry data (point clouds) of the inspection object. Defects that occur (corrosion, mechanical defects, etc.) can be evaluated in the image data, but components of the inspection object (e.g., pinion) can also be detected. Individual defects or components may need to be specifically inspected. Subproblem (b) must be solved for this. A geometric model of the inspection object on which the object areas to be inspected are mapped can be assumed as the starting point. Such a model can be generated with the help of semantic mapping methods [31] using the information generated in subproblem (a). A semantic map augments spatial information by additional information about entities that are located in space [31]. This is out of scope for this work but will be briefly explained for the sake of understanding.

The geometry data (point clouds) captured in subproblem (a) can be used to build a three-dimensional object model, as the entire surface of the inspection object has been covered. Furthermore, the captured image data can be analyzed with (intelligent) image processing algorithms to detect RoI and map it to the geometry model of the object (e.g., semantic segmentation or object detection methods). It is important to mention that these intelligent detection algorithms for specifying the RoI must be designed in advance, and therefore, a process engineer must also determine which components or defects always require detailed inspection. This can be done, e.g., based on prior knowledge or core acceptance criteria.

In summary, the solution of subproblem (a) thus provides an initial information basis (semantic geometry model) for subproblem (b), since in (a) defects or product features (RoI) to be inspected more closely can be identified and by means of (b) a dedicated inspection fine planning of the poses of the inspection system is enabled to inspect these RoI.

3.2. Overview of the proposed view planning framework

The framework presented in this paper consists of a scan simulation environment with uniform interfaces to which exchangeable modules can be attached. A simplified overview can be seen in Fig. 2. A pose p_t is passed to the scan simulation environment. The scan simulation environment then performs a virtual acquisition and passes an observation o_t to the module currently in use. In the scope of this work, the RL agent module and the SL framework module are presented. However, integrating other modules, such as a module for solving the derived problems with heuristics or analytical methods, is possible due to the uniform interfaces of the scan simulation environment.

3.3. Scan simulation environment

The scan simulation environment consists of a scan module into which a sensor and object model is loaded at the beginning of each acquisition task. The sensor model is used to replicate a real existing optical acquisition system. The relevant parameters for this are

- the resolution of the camera (acquisition system resolution),
- the aperture angles of the frustum (field of view) and
- the near and far bounds of the frustum (operating range of the acquisition system),

which are freely configurable and visualized in Fig. 3. For the opening angle, 27° and 25° in x and y axis are chosen while the resolution is set to 430 × 300 pixels. These parameters were deduced according to an existing acquisition system, Zivid One+ S. Using the mentioned

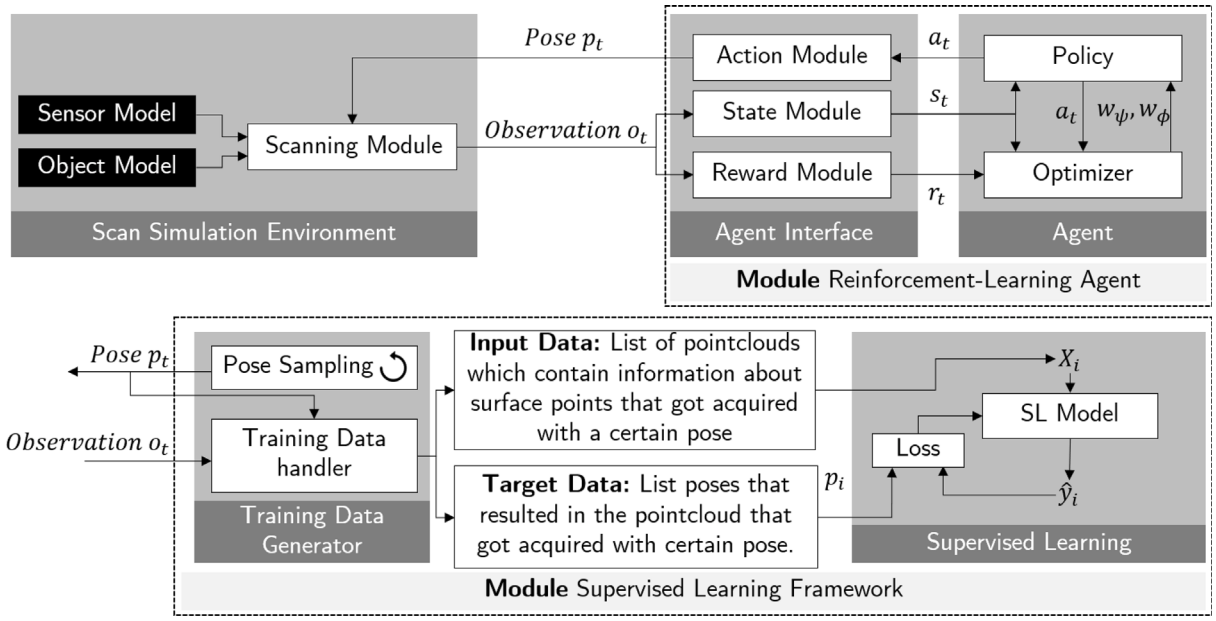

Fig. 2. Structure of the view planning simulation framework.

resolution, a good compromise between the computing time required to process the point cloud and the level of detail of the captured surface can be obtained. The near bound is set to 30 cm while the far bound is set to 50 cm. The object model is specified via the STL (stereolithography) format. Using said data format, the object is represented only by its surface in the form of triangular facets.

The scanning module uses the sensor and object models to simulate a three-dimensional acquisition process of the object using an optical coordinate acquisition system. The principle of ray tracing is used for this purpose. Depending on the camera parameters' aperture angle and resolution, rays emanating from the focus point, which is defined as the pose p_t given to the scan simulation environment, are simulated. The first intersection of each ray with a triangle of the object model defines the coordinates of a point on the object that may be captured. The (x, y, z)-coordinates are calculated for each intersection point in the global coordinate system. Additionally, the distance between the camera origin and the point at which the ray hits the object is determined for each intersection point in order to take into account the permissible working range of the 3D camera. This point is classified as not acquired if this distance is greater or smaller than the permissible working range, defined through the near and far bounds of the frustum.

The point cloud P_t acquired at the acquisition step t consists of all intersection points in the acquisition system's permissible working range. P_t is then used for an observation update from the last acquisition at step $t-1$ to step t. The total point cloud $P_{cov,t-1}$ is then calculated as the sum of all acquired point clouds $P_1, \dots, P_{t-1}$ up until step, $t-1$ with the point cloud P_t acquired at step t. Since the scan simulation directly calculates points in the global coordinate system, no registration algorithm is needed to perform the merging of the separate point clouds. In addition to updating the already acquired point cloud, further information is extracted and saved during the observation update. These serve as an information basis that can be utilized to extract training data for SL algorithms and state representation and reward calculation of RL agents. These are as follows:

1. Object model (triangle mesh) of the current episode.
2. Ground truth point cloud P_{GT} with predefined number of points. P_{GT} is generated by evenly sampling n_{GT} points on the object model using a voxel grid approach.
3. Number of acquisitions t in the current episode.

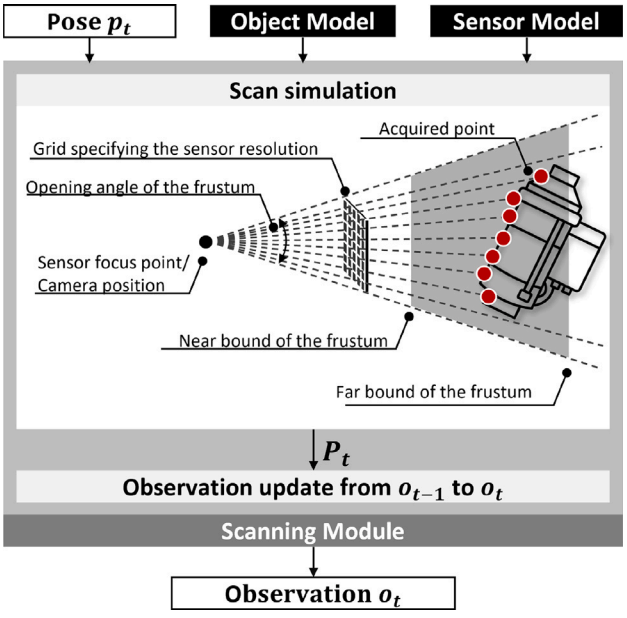

Fig. 3. Visual representation of the processes within the scan simulation environment.

4. List of camera poses $p_1, \dots, p_t$ of the current episode as well as the corresponding acquired point clouds $P_1, \dots, P_t$.
5. Total point cloud $P_{cov,t}$ captured in an episode up to interaction step t. This point cloud is obtained by registering the single point clouds $P_1, \dots, P_t$ and subsequent voxel downsampling to obtain a uniformly distributed point cloud P_t.
6. Total point cloud $P_{cov,t-1}$ acquired up to interaction step $t-1$.
7. Inverted point cloud $P_{inv,t-1}$ at interaction step $t-1$ which is calculated based on the ground truth point cloud P_G subtracting the point cloud $P_{cov,t-1}$ acquired up until interaction step $t-1$ (see Fig. 3).

Table 1
Values of d when generating state $s_{t,V2}$.

Description *"Point on ground truth point cloud ..."*	Value of d
... does not need to be inspected	0
... needs to be inspected but has not been seen	−1
... needs to be inspected and has been seen	1

3.4. Modeling and implementation of the supervised learning framework and RL agent

In the following Section, the modules considered in this work and shown in Fig. 2, the SL Framework and the RL Agent, are discussed in more detail.

3.4.1. Reinforcement learning agent

State modeling

The state module is used to encode an acquired point cloud in a way that the RL agent can process. In the presented work, two state encodings are implemented. In the first variant, in the following referred to as $s_{t,V1}$, the agent is given the combined total point cloud $P_{cov,t}$ from all acquisitions until step t after each acquisition. This state encoding thus represents the information state of a worker for the solution of the overall inspection (subproblem (a)), in which the worker must inspect the overall surface of the product and choose its next acquisition to inspect additional surfaces that have not been observed so far. Since the RL agent's policy is approximated by a neural network (see 3.4.3) and therefore requires a fixed number of input variables, the point cloud is reduced to a constant number of points m using voxel downsampling. In our preliminary tests, $m = 2048$ has proven to be an adequate dimension of m. With this number of points, a suitable compromise is obtained that accounts for both the level of detail with which an object is represented and the size of the point cloud which directly correlates to the computational effort needed.

$$s_{t,V1} = \begin{bmatrix} x_{t,1} & y_{t,1} & z_{t,1} \\ \dots & \dots & \dots \\ x_{t,2048} & y_{t,2048} & z_{t,2048} \end{bmatrix} \tag{1}$$

To solve the problem of individual inspection of RoIs, an additional dimension d is introduced in the modeling of the state. This encodes whether a point on the object should be inspected more closely and whether it has already been seen in a previous acquisition step. The logic behind the encoding is depicted in Table 1. At the beginning of the acquisition process, points that lie within the RoI are encoded with the value −1 in column d. Points that are not to be acquired are initialized with 0. If a point to be acquired within the RoI is detected during the acquisition process, its coding changes to 1 for the following acquisitions. Thus, the agent's goal is to set all points originally encoded with −1 to the value 1 by appropriately choosing the poses of the 3D camera system. The coordinates of these m points with the additional column d are stored in the matrix $s_{t,V2}$.

$$s_{t,V2} = \begin{bmatrix} x_{t,1} & y_{t,1} & z_{t,1} & d_{t,1} \\ \dots & \dots & \dots & \dots \\ x_{t,2048} & y_{t,2048} & z_{t,2048} & d_{t,2048} \end{bmatrix} \tag{2}$$

The matrix S_{V2} contains m points on the object which have been uniformly sampled on the object surface. These points are points of the ground truth point cloud P_{GT}. To determine which of the m points were acquired during an acquisition, the acquired point cloud P_t is compared with the ground truth point cloud P_{GT}. In this process, a point of P_{GT} is registered as seen if the Euclidean distance of this point to its neighbor in the acquired point cloud P_t is smaller than a defined threshold value.

Action module

The action module handles the actions specified by the agent. The agent outputs an action vector whose values are constrained in the interval of $[-1, 1]$. Each entry corresponds to a parameter for determining the position of the 3D camera for the next acquisition step. These entries represent the parameters ϕ, θ, and r in spherical coordinates. The three remaining Euler angles α, β and γ define the rotational degrees of freedom. However, they are not output by the agent but are determined with a heuristic that automatically orientates the 3D camera system to the object's center. The Euler angle γ is set to the value 0 in all cases due to its low influence. This is the reason why the degrees of freedom of the pose of the 3D camera system in space are reduced from six to three. Since the output of the RL agent is limited to an interval of $[-1, 1]$ based on tangens hyperbolicus chosen as the activation function of the last layers of the neural networks used (see Section 3.4.3), the action is mapped to parameters' real interval boundaries (e.g., $[0, 2\pi]$ for angle θ). This can be achieved by mapping the agent's output range onto the real interval boundaries. The action encoding can then be used to calculate the pose of the 3D camera for the next acquisition.

Reward module

A mixture of so-called dense and sparse rewards is used for the reward function. A dense reward is a reward that is given to the agent after each individual action. A sparse reward, on the other hand, is received by the agent only at the end of an episode. In the reward function (3) used, the dense reward R_1 is given at each time step, which is dependent on $\Delta\Psi$. If a target state s_G is reached, the agent receives the sparse reward $R2$.

$$R_t = R_1 + R_2$$

$$R_1 = \begin{cases} a, & \Delta\Psi > 0 \\ b, & \text{else} \end{cases} \tag{3}$$

$$R_2 = \begin{cases} E, & s = s_G \\ 0, & \text{else} \end{cases}$$

Ψ corresponds to the surface area of the object in percent that has been acquired so far. $\Delta\Psi$ corresponds to the newly acquired surface area in percent between the previous time step $t-1$ and t. If $\Delta\Psi = 0$, the negative reward b can also motivate the agent's learning behavior to position itself to increase surface coverage. Furthermore, when a target state s_G is reached, the agent receives a fixed sparse reward E that is chosen, so it is greater than the sum of all possible positive dense rewards it receives when reaching full object coverage ($\Psi = 100\%$). s_G is reached as soon as $\Psi \geq \Psi^*$, where Ψ^* corresponds to a threshold value of the desired surface coverage in percent. To reduce the process time, the acquisition task should be performed with the minimum possible number of acquisitions. However, the reward function does not explicitly consider minimizing the number of acquisitions. Instead, by choosing the discount factor $\gamma < 1$, the agent is implicitly motivated to reduce the number of steps until a target state s_G is reached. Discounting reduces the estimated rewards at later time steps. Since the agent maximizes the return, reaching states with high rewards becomes more profitable after fewer steps.

Agent

Since the environment used in this work uses continuous action (pose) and observation space (point clouds), the Proximal Policy Optimization (PPO) and Soft Actor-Critic (SAC) learning algorithms are used. These are implemented in the RL library stable-baselines by [32] and offer easy compatibility with OpenAI Gym environments. Both algorithms use an actor-critic policy. The actor is responsible for action selection, while the critic estimates the state value function. In the SAC algorithm, a target critic is also used, updated asynchronously to the critic, and stabilizes the training process. The network architectures evaluated in this work are described in detail in Section 3.4.3, since they are also used for SL.

3.4.2. Supervised learning framework

SL has already been applied successfully to the VPP. Thereby, only the NBV problem was addressed. To solve subproblem (b) of individual inspection, however, the task to be solved is to estimate the pose of a 3D camera which led to the acquisition of a specific set of points on the object's surface.

Dataset generation

A data set consists of the data x and the labels y. To obtain x_i, a randomly generated pose is used to acquire a point cloud in the scan simulation environment. The 2048 × 4 coded point cloud returned as an observation corresponds to x_i, the associated pose corresponds to y_i. In this case, the fourth column d is binary coded. Points acquired based on the pose y_i are coded 1, and other points are coded 0. The value range of the output of the neural network lies in the interval $[-1, 1]$ and is mapped to the value ranges of the representation of the sampled poses (value ranges of spherical coordinates and Euler angles) for comparison with the sampled pose y_i in analogy to the procedure in the action module of the RL agent.

Evaluation metrics

In order to compare the pose $\hat{y}_i$ predicted by the network with the true pose y_i of the dataset, comparison metrics are needed. A pose y_i consists of the position T and the rotation matrix R, which is built of the Euler angles α, β, γ. A pose can be transformed into a homogeneous transformation matrix $\mathcal{T}$ using these parameters.

$$\mathcal{T} = \begin{bmatrix} T_r^{3x3} & T_t^{1x3} \\ 0 & 1 \end{bmatrix} \tag{4}$$

Based on this transformation matrix, quantitative comparison metrics can be derived to compare the pose $\hat{y}_i$ determined by the network with the true value y_i. These are used to provide a quantified measure of the network's performance.

The error of the position e_t is described by the Euclidean distance of the translation vectors, where $\hat{t}_k$ corresponds to the network output and t_k to the ground-truth translation vector.

$$e_t = \sqrt{\sum_{n=1}^{3} (\hat{t}_k - t_k)^2} \tag{5}$$

One approach to compare the rotation of two camera positions, similar to translation, is to determine the Euclidean distance of the Euler angles. However, the rotations are interdependent and performed sequentially, so this metric is not meaningful. In contrast, as described by Huynh, a rotation matrix $T_{r,1\to2}$ can be calculated, which determines the necessary rotation that aligns a rotation matrix $T_{r,1}$ to $T_{r,2}$ [33]. This can be calculated using Eq. (6). To derive a general metric describing the rotation, the angle of the necessary rotation $T_{r,1\to2}$ is determined as error metric e_ξ according to Eq. (7). The smaller the angle ξ, the more similar the rotation matrices are, and the deviation of the predicted pose from the ground truth is correspondingly smaller.

$$T_{r,1\to2} = T_{r,1} * T_{r,2}^T \tag{6}$$

$$e_\xi = cos^{-1}\left(\frac{spur(T_{r,1\to2}) - 1}{2}\right) \tag{7}$$

3.4.3. Network architectures

Complex neural networks are needed to process point clouds since point clouds are usually unordered and spatial relationships must be detected in them. The state encodings $s_{t,V1}$ or $s_{t,V2}$ serve as inputs for the neural network architectures using RL investigated in this work. In the case of SL, the modified version of $s_{t,V2}$ with binary encoding is used. Three different approaches to feature extraction from the input data are evaluated and described briefly in the following. Please refer to the original work for more detailed information regarding the internal structures.

On the one hand, PointNet by [34] is used, which enabled point cloud processing for the first time. The layer structure of the encoder of the PointNet for feature extraction used in this work includes an input transformation layer, a Multi-Layer Perceptron (MLP) and a max-pooling layer for the final output of the feature vector. The input transformation layer takes the raw point cloud data and applies pose normalization by multiplying the original point cloud by a 3 × 3 transformation matrix, which is output by a subnetwork called T-net. The MLP processes each point of these transformed points independently to learn local feature representations. Followed by a feature transformation of the local feature representations by another T-Net, the max-pooling layer then aggregates the transformed local features of all points into a global feature vector.

Furthermore, the Point Completion Network (PCN) encoder developed by [35] is used as feature extractor. The PCN is a PointNet variant designed specifically for completing partial point clouds. The encoder of the PCN extends the PointNet architecture by a multi-scale feature fusion, combining local and global point cloud features. The layer structure involves feeding the input point cloud through the MLP to extract local point features. A pooling layer follows this to obtain a global feature vector. The multi-scale feature fusion step combines the local and global features across multiple scales to capture both fine-grained and coarse-grained information.

In addition, a more sophisticated network was implemented, the PointNet++, which was developed by [36]. This further develops the PointNet model and improves the processing of 3D point clouds through hierarchical data processing. It enables more accurate capture of local and global information and is more versatile in its application to different 3D datasets. Overall, it provides a more effective method for processing 3D point clouds for segmentation and object recognition tasks.

Due to the more complex structure and the higher number of layers, the number of parameters of the PointNet and PointNet++ used is higher than the PCN. The respective network architectures for feature extraction are extended by adding an MLP to obtain the complete networks. The output dimension of the MLP corresponds to the number of free parameters, which always output values in the interval $[-1, 1]$. In the case of RL, three free parameters are considered (θ, γ, r), whereas in the case of SL the number of free parameters is five (θ, γ, r, α, β). Note that in the case of SL, only one neural network is required, while the RL approach (SAC) requires three different networks (actor, critic, and target critic).

4. Results

In the following, the results of the modeling alternatives proposed in this paper for solving the two subproblems (a) and (b) derived in Section 3.1 are presented. After giving an overview of the dataset and standard agent configurations, the solution and results of the VPP using RL are discussed (Section 4.3). Subsequently, Section 4.4 assesses how SL can be used to regress the pose given an acquired point cloud. Finally, Section 4.5 presents the results where RL is used to determine sequential poses of the acquisition system to acquire RoIs located arbitrarily on the object.

4.1. Used dataset

Our study uses a dataset of starter engines that are automatically generated based on examining real remanufacturing starter engines, thereby addressing the properties of the remanufacturing use case under consideration. The synthetically generated geometric motor models, for which the RL-agent generates poses of the acquisition system, exhibit varying geometric properties. This is due to the vast number of different starter motor variants from diverse manufacturers in the field and the possibility of damaged or missing parts, leading to significant variations. The approach utilized in this study is based on an existing pipeline approach from [37], which has been tailored to generate starter engines.

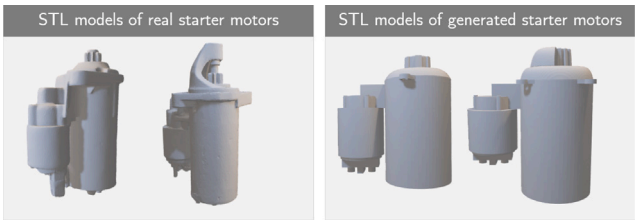

Fig. 4. Illustration of STL models of real existing starter motors (Figure left) and synthetically generated STL models based on the real starter motors (Figure right).

Table 2
Parameters and chosen default values for the sensor model.

Sensor model parameters	Values
Near and far bounds	[30, 50] cm
Resolution	$(430 \times 300)_{xy}$
Opening angles	$(27°, 25°)_{xy}$

Table 3
Parameters and chosen default values for the SL model.

Supervised model parameters	Chosen setup
Number of epochs	150
Batch size	32
Loss function $\mathcal{L}$	Huber loss
Learning rate η	0.001[a]
Optimizer	Adam
MLP structure	256-128-64-Output

[a] Linear decay with a final value of 0.

The starter engines consist of nine randomly generated components, such as a solenoid, gear housing, connector, and flange, with a total of 28 different parameters, including height and diameter. An illustration of the starter engines is shown in Fig. 4. The parameters are subject to parameter limits and clearly defined relationships with each other to ensure that they remain within a realistic range. This approach leads to a diverse and realistic dataset. For the present study, 100 motors with random parameters were generated and saved in STL format for training the agents.

4.2. Default agent configuration

Unless otherwise specified, fixed parameters are used for the following evaluations. These concern the parameters of the sensor model (see Table 2), those of the SL approach (see Table 3), as well as those of the RL approach (see Table 4).

The parameters for the sensor system used were derived from the data sheets of a recording system available in the lab. The number of epochs of the SL approach was estimated so that convergence is guaranteed. A batch size of 32 and a learning rate of 0.001 were chosen for stable learning behavior. In addition, a linear decay of the learning rate was introduced. The Huber loss, a modified variant of the mean squared error, was used as the loss function. Preliminary tests have shown the Adam Optimizer to be the most promising candidate.

The default values for the termination of an episode of the RL agent were defined on the basis of the surface coverage achieved or the exceeding of a maximum number of acquisitions. For both the overall inspection and the specific inspection of individual RoIs, a surface coverage of 90% was selected. Likewise, an episode is terminated if the agent has taken ten scans or the termination criterion of surface coverage has been met. The length of the entire training cycle was set to 75 000 steps, i.e., interactions of the agent with the simulation environment. The value 0.9 was chosen as the discount factor, while the learning rate was 0.000078 and was determined by preliminary tests. SAC was set as the default agent, while PCN was chosen as the feature extractor. Unless otherwise specified, $s_{t,V2}$ was chosen as the state encoding.

Table 4
Parameters and configuration, as well as the respective default values and default modeling choices of the RL agent.

Training cycle design	Values
Required object coverage	90%
Maximum number of acquisitions	10
Number of steps	75 000
Learning parameters	**Values**
Learning rate	0.000078[a]
Discount rate	0.9
Modeling configuration	**Chosen setup**
Algorithm	SAC
Encoder	PCN
MLP structure	256-128-64-Output
State	$s_{t,V2}$

[a] Linear decay with final value of 0.

4.3. Results for the solution of the view planning problem with RL

Through the interaction of the agent with the simulation framework, the agent has to learn a strategy that sequentially outputs poses of the acquisition system that maximize Ψ. An initial acquisition with a constant pose is carried out. The observation of this pose is used to deduce the initial state for the further selection of poses by the agent. The values $a = \frac{\Delta\Psi}{100}$ and $b = 0$ as well as $E = 1$ were determined for the reward function defined in Section 3.4.1 to work best with the present planning problem. Punishing the agent via negative reward $b < 0$ did not result in a different agent behavior. This can be attributed to the fact that in this case, the agent quickly learned to perform only acquisitions that entailed at least a minimal gain of not yet covered surface, and thus, the case of punishment by a negative reward never occurred. Fig. 5 shows the comparison of agents with state encodings $s_{t,V1}$ or $s_{t,V2}$ compared to a benchmark agent. The benchmark agent, which chooses random actions, achieves $\Psi = 60\%$ on average. The initial acquisition is included in the total number of acquisitions. According to the workspace of the acquisition system, it is set 47 cm away from the origin and, on average, acquires 46% of the total surface area for the starter motors considered.

Results show that both agents using SAC agents and one of the respective state encodings converge to strategies that reach a target state of $\Psi \geq \Psi^* = 90\%$ after only 4–6 acquisitions and 2000 epochs. The state encoding $s_{t,V2}$ performed better than $s_{t,V1}$ when considering the number of acquisitions needed to reach the required surface coverage for ending an episode. Additionally, the learning behavior of the agent with state encoding $s_{t,V2}$ shows higher stability with regard to fluctuation in the required number of acquisitions as well as the collected reward. In-depth analyses have shown that since the reward at each step directly depends on $\Delta\Psi$, the agent tries to exploit this and successively increases the radius r of the chosen pose to position itself further away from the object. This leads to the frustum capturing much more of the surface of the objects to be inspected, and thus more reward. Thereby, the agents also learn not to exceed the cutoff distance of 50 cm where the far bound of the frustum ends, and thus, no point cloud would be acquired, leading to no collection of reward.

Fig. 6 shows the visualization of the poses issued by an agent with the state coding $s_{t,V1}$ to solve the VPP over several episodes. It is evident that there are clusters of certain positions in space that the agent frequently outputs. Brighter areas show positions in space that the agent issues more frequently. More detailed analyses have shown that individual, distinct bright areas correspond to individual poses issued by the agent in the first, second, or third acquisition. Later acquisitions (fourth, fifth or sixth) are much less clustered in space and correspond to the rather dark point clusters in Fig. 6. These results suggest that the agents output specific positions in space in the early stages of the acquisition process, while in later stages of the acquisition

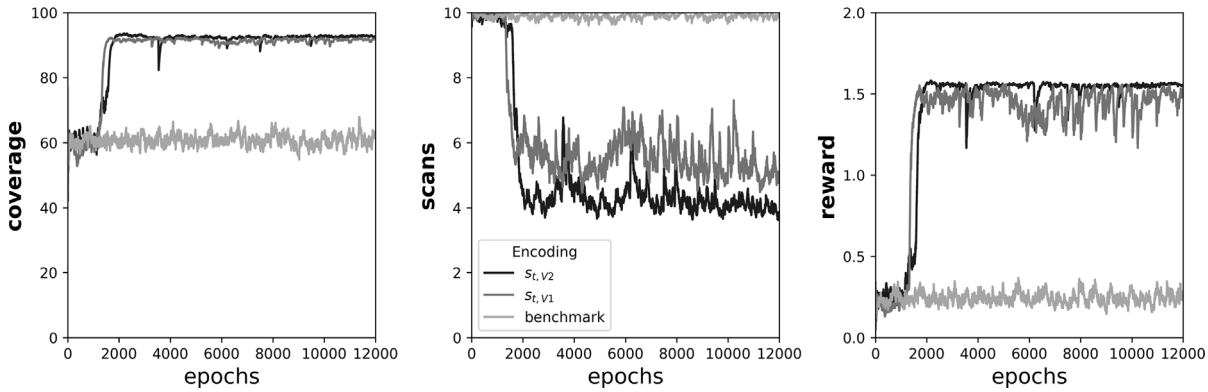

Fig. 5. Comparison of agents with the different state encodings $s_{t,V1}$ and $s_{t,V2}$ using coverage, the required number of acquisitions, and obtained reward.

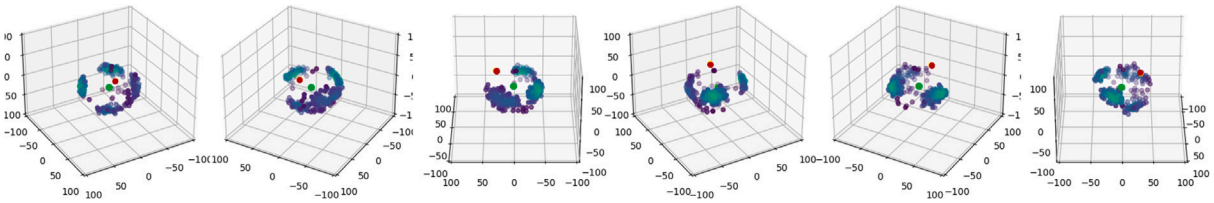

Fig. 6. Visualization of the poses output by the agent with state encoding $s_{t,V1}$ over the course of several episodes for the VPP after the training process. The position of the initial acquisition is shown with a red marker. The inspection object is marked with a green marker. The six subplots each show the accumulation of poses output by the agent from different perspectives.

Table 5
Tabular representation of the position and angle deviations for the investigated feature extractors.

Feature extractor	e_t	e_ξ	t_{train} [s]
PCN	μ = 17.203 σ = 0.845	μ = 25.452 σ = 0.601	μ = 381.41 σ = 6.494
PointNet	μ = 17.173 σ = 0.408	μ = 25.050 σ = 0.914	μ = 585.17 σ = 1.012
PointNet++	μ = 21.978 σ = 2.042	μ = 31.891 σ = 3.068	μ = 6342.8 σ = 7.745

process, poses are chosen to acquire the then remaining surface of the inspection object. It can also be observed that the agent learns to choose poses that are always distant from the initial acquisition pose (red). This is due to the fact that poses close to the initial acquisition pose would most likely provide little or no additional coverage of the surface of the inspection object.

In summary, the suitability of the chosen reward function can be derived from the agent being able to increase Ψ and learn a strategy to fulfill the inspection goal. Additionally, the number of exposures can be implicitly minimized by setting the discount rate γ to 0.9. The reason is that a relatively low discount rate pushes the agent to learn a strategy that acts rather greedily. The agent has to output a short pose sequence that achieves a high surface coverage Ψ since longer sequences lead to a lower estimated return for the agent since γ is chosen relatively low. Additionally, longer acquisition sequences increase the chance of the agent not acquiring additional surface areas, which in return leads to negative rewards. Subsequently, a separate parameter to control the number of acquisitions is unnecessary for the reward function.

4.4. Results for supervised learning of acquisition system poses

The results for training using SL are shown in Table 5. The metrics of position deviation e_t, angular deviation e_ξ and the training duration for using the networks with PCN, PointNet and Pointnet++ as feature extractors, are shown.

Three training runs were conducted for each feature extractor. From these runs, the mean value μ and the standard deviation σ are determined for the metrics used e_t and e_ξ, as well as for the training duration t_{train}. The learning curve, which is shown in Fig. 7, depicts the progression of the model error over the number of epochs performed. Due to the convergence of the training curve, it can be concluded that regressive learning of the sensor pose given an input point cloud augmented with the encoding of the object surface to be detected in the fourth column is possible. Furthermore, the results from Table 5 show that the PCN achieves almost the same position prediction accuracy as the PointNet. The results of both networks are also the same for the deviation of the rotation e_ξ. Pointnet++, on the other hand, shows a significantly higher deviation in the prediction, a 28.8% higher position accuracy, and 25.3% for the angular deviation. A major difference can be seen in the training duration of the networks used. While the duration of a training run with the PCN as a feature extractor is 381 s, the PointNet requires almost 53% more time at 585 s. This can be explained by the number of weights, which is 3 times higher with the Pointnet than with the PCN. However, Pointnet++, which has fewer weights than the original Pointnet, takes almost 10 times as long, which is significantly longer due to complex functions (e.g., point aggregation). Therefore, we use PCN as the feature extractor for the remaining RL experiments, as the calculation time is significantly shorter.

However, detailed analyses have shown that the position and angle deviations should not be used as the sole metrics for evaluating the regression quality. This is because an acquired point cloud is not always acquirable by only one viewpoint in the pose of an acquisition system. Fig. 8 shows a visual example of this. For instance, when introducing a translation Δt of the viewpoint in space, finding a rotation $\Delta\xi$ may be possible so that a similar or, in extreme cases, the same object surface is acquired with these two different viewpoints. This may result in the trained network still having positional and angular deviations in the form of the metrics e_t and e_ξ in the learned state but still acquiring the same or a similar point cloud since the network learned to average over the many examples provided in the learning phase. In further work, these results will be verified with the introduction of a metric that compares the ground truth point cloud and the point cloud

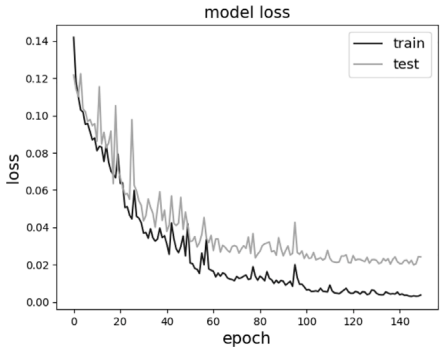

Fig. 7. Learning curve of the Huber loss of the model over the number of epochs of the SL approach.

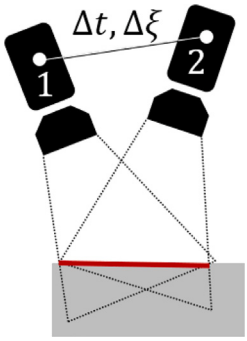

Fig. 8. Problem of SL regarding the ambiguity of an acquired point cloud with respect to the origin pose. Both viewpoint 1 and viewpoint 2 acquire the same object surface (marked with the red line) which lies in the field of view of the acquisition system (dashed line), although viewpoint 1 has a translational Δt and rotational $\Delta\xi$ deviation from viewpoint 1.

acquired with the estimated viewpoint of the trained network given the ground truth point cloud (e.g., using the Chamfer distance of both point clouds).

4.5. Results for the individual inspection with reinforcement learning

The parameters of the reward function used for the individual inspection deviate from the ones used for the overall inspection. The parameters are set to $+a = (\Delta\Psi)^3$ and $b = -0.25$ as well as $E = 5$. In this case, preliminary experiments showed that penalizing by negative reward $b < 0$ for not providing additional surface coverage after a view was helpful to quickly teach the agent to perform only acquisitions that cover a RoI. In addition, the dense reward was extended by a potentiating factor, which also seems to have a positive effect on the agent quickly learning to choose views that fully or almost fully cover the RoI. In Section 4.3, it was shown that for a camera working range of [30, 50] cm, depending on the underlying object model, three to five acquisitions can be sufficient to capture 90% of the entire object surface. For the evaluations of individual inspection capabilities of RL agents, the working area was reduced to make the problem more difficult to solve. The smaller the near and far bounds as well as opening angles of the sensor model, the lower the maximum possible percentage surface gain per acquisition. At the same time, this also corresponds to a human's intuitive approach to closely examine a detected defect. Positioning an acquisition system closer to the object makes the detected surface or image section smaller, but a greater resolution of this RoI can be achieved. This is advantageous for detecting and evaluating small defects and flaws, especially in remanufacturing. Thus, the working range is reduced to $[15, 30]$ cm for evaluating the problem of individual inspection. Furthermore, to reduce the exploration time of the agent, the interval boundaries of the coordinates the agents' action outputs are mapped to are adapted to the reduced working range. In particular, the radius is limited to $R = [0, 50]$ for spherical coordinates. A RoI is generated by randomly selecting a surface segment consisting of 150 points of the ground truth point cloud P_{GT} on the object. The maximum episode length is set to 6 acquisitions. This is to observe to what extent the agent is able to minimize the number of acquisitions to reach the inspection goal.

The following experiments evaluate the influence of the design of the RoIs to be inspected. For this purpose, the number of generated RoI on the surface of the object and the number of points per RoI is varied. To examine the influence of the number of RoIs, a training run is performed in which n RoIs are generated on the object surface each episode. Each RoI consists of 150 points and agents were trained to either inspect one, two or three RoI (see Fig. 9). For $n = 1$ RoIs, the episode length is limited to 4 acquisitions. For each additional RoI generated, 2 additional acquisitions are allowed. In the course of the training runs shown in Fig. 9, it can be shown that the agent constantly reaches a target state regardless of the number of RoIs and is able to reduce the number of acquisitions required to fulfill the inspection goal. Noticeably, the agent's strategy converges later, the lower the RoI number n is. This can be attributed to the fewer points of the total surface being encoded as RoIs when $n = 1$. Therefore, the problem is more complex, and exploration is more costly than when the number of RoIs is larger. Furthermore, the maximum exposure allowed depends on the RoI number n. For $n = 1$, only half the number of exposures is allowed in an epoch as for $n = 3$. Accordingly, the agent has fewer opportunities for exploration for the same number of epochs. The number of exposures required to reach a target state, on the other hand, corresponds to the expectations. The lower n, the lower the number of shots required. For $n = 1$, the agent requires an average of 1.6 exposures. For $n = 2$, 2.8 acquisitions are needed on average. A third RoI with $n = 3$ needs 3.3 acquisitions on average. Since at $n = 3$ already 20% of the total surface has to be inspected, the probability of two RoIs overlapping during generation increases. Therefore, the difference from $n = 2$ to $n = 3$ is smaller than to $n = 1$.

In a further experiment, the size of the RoIs varied. Two RoIs with one of $m = 50, 150, 250$ points each are generated in each training run. The fraction of the total surface coded as RoI corresponds to 4.8% for $M = 50$, 14.6% for $M = 150$, and 24.4% for $M = 250$. The results of this comparison are shown in Fig. 10. It can be concluded that the influence of the size of the RoIs is negligible. Only for $M = 50$ does the agent need 0.4 acquisitions less on average to reach a target state.

5. Conclusion

This paper presents a modular simulation framework to solve acquisition planning problems using SL and RL. Relevant problems of acquisition planning are derived from the use case of remanufacturing. Standardized interfaces of the simulation environment and the modules connected to it allow for the exchange and evaluation of the modules.

The results have shown that RL can be applied to the task of general inspection of returning used products. The agent configurations considered solved the problem of view planning and reduced the number of acquisitions needed to achieve the required surface coverage by sequential acquisitions of the object. Furthermore, it could be shown that SL approaches are able to learn which pose of an acquisition system has to be chosen in order to detect specific surface areas of a product. Such trained models can be used in the future to plan poses of an acquisition system in such a way that specific features or defects on the product surface, so-called RoI, can be inspected in a targeted manner. Sequential planning of poses of the acquisition system for inspecting several RoI of different sizes could also be accomplished using RL.

Even though the present approaches have consistently shown good results in terms of applicability in the simulation, they must be tested in a real inspection environment in further work. This requires further

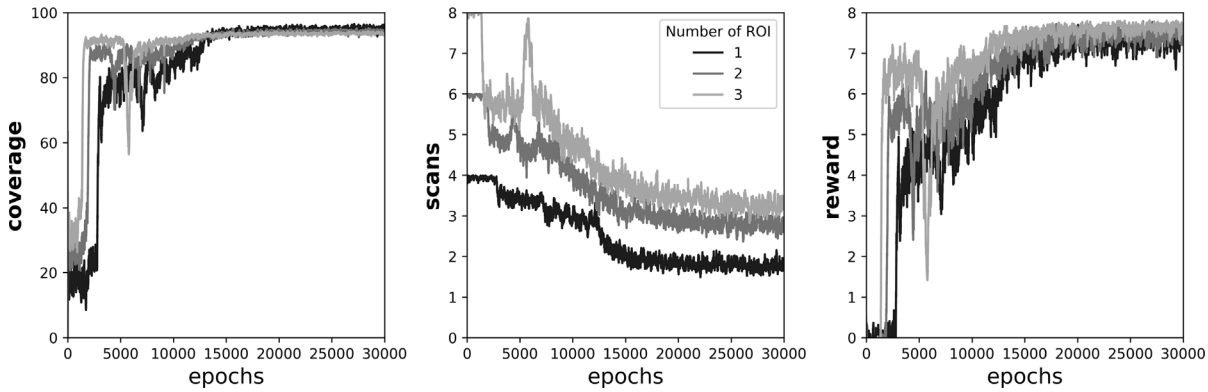

Fig. 9. Comparison of coverage, the required number of acquisitions and obtained reward when varying the number of RoI to be inspected on the product.

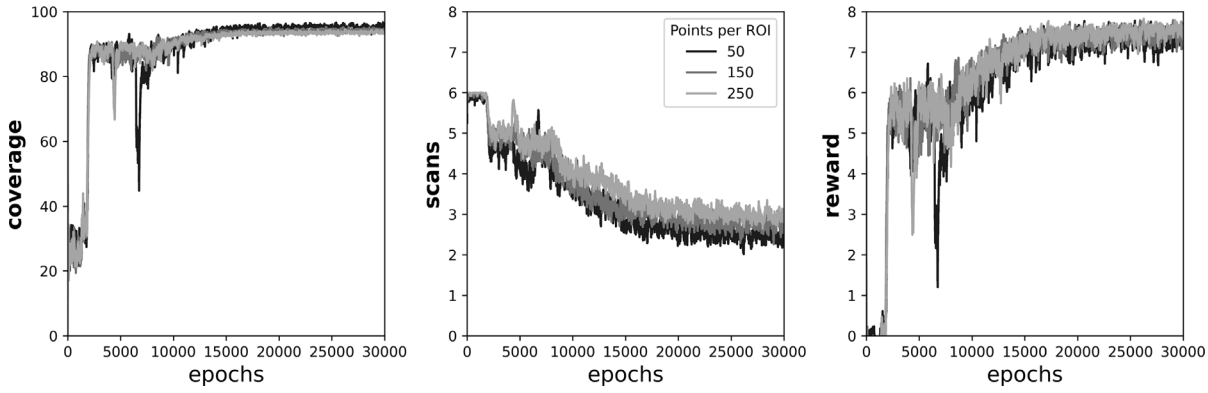

Fig. 10. Comparison of coverage, the required number of acquisitions and obtained reward when varying the size of two RoI to be inspected.

steps that exceed the scope of this work. On the one hand, detection algorithms have to be implemented, which, in a preliminary step, detect the RoI to be inspected more closely on the returned used product. Possible approaches are segmentation algorithms (e.g., variants of U-Nets) or object detectors (e.g., Mask-RCNN). For application in a real inspection environment, further constraints should also be considered and integrated into the proposed simulation framework in order to increase the realism of this framework. This includes, among others, the reachability analysis of the poses proposed by the machine learning methods to verify whether they can be approached by a robot-guided visual acquisition system at all.

It is also necessary for further work to deal with the interplay between the depth of inspection (and the associated inspection time) and the profitability of the entire remanufacturing process. As developed by Ridley et al. [38], there is a correlation between the economic profitability of the entire remanufacturing process and the time spent on the initial inspection, which must also be considered if the process is successful in the field.

CRediT authorship contribution statement

Jan-Philipp Kaiser: Conceptualization, Methodology, Validation (lead), Investigation (lead), Formal analysis (lead), Writing – draft (lead), Software. **Dominik Koch:** Software (lead), Investigation, Validation, Formal analysis, Writing – draft. **Jonas Gäbele:** Visualization, Data curation. **Marvin Carl May:** Project management, Resources, Writing – editing. **Gisela Lanza:** Funding acquisition, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was done in the AgiProbot project. AgiProbot is funded by the Carl Zeiss Foundation.

References

[1] T. Wuest, D. Weimer, C. Irgens, K.-D. Thoben, Machine learning in manufacturing: advantages, challenges, and applications, Prod Manuf Res 4 (1) (2016) 23–45, http://dx.doi.org/10.1080/21693277.2016.1192517.
[2] D. Weichert, P. Link, A. Stoll, S. Rüping, S. Ihlenfeldt, S. Wrobel, A review of machine learning for the optimization of production processes, Int J Adv Manuf Technol 104 (5) (2019) 1889–1902.
[3] M. Bertolini, D. Mezzogori, M. Neroni, F. Zammori, Machine learning for industrial applications: A comprehensive literature review, Expert Syst Appl 175 (2021) 114820.
[4] M.A.K. Bahrin, M.F. Othman, N.H.N. Azli, M.F. Talib, Industry 4.0: A review on industrial automation and robotic, J Tek 78 (6–13) (2016).
[5] C. Salkin, M. Oner, A. Ustundag, E. Cevikcan, A conceptual framework for industry 4.0, in: Industry 4.0: managing the digital transformation, Springer, 2018, pp. 3–23.
[6] J.E. See, C.G. Drury, A. Speed, A. Williams, N. Khalandi, The role of visual inspection in the 21st century, in: Proceedings of the human factors and ergonomics society annual meeting, vol. 61, SAGE Publications Sage CA: Los Angeles, CA, 2017, pp. 262–266.
[7] M. Errington, S.J. Childe, A business process model of inspection in remanufacturing, J Remanufacturing 3 (1) (2013) 1–22.
[8] M.A. Seitz, K. Peattie, Meeting the closed-loop challenge: the case of remanufacturing, Calif Manage Rev 46 (2) (2004) 74–89.
[9] A. Khan, C. Mineo, G. Dobie, C. Macleod, G. Pierce, Vision guided robotic inspection for parts in manufacturing and remanufacturing industry, J Remanufacturing 11 (1) (2021) 49–70.
[10] E. Sundin, O. Dunbäck, Reverse logistics challenges in remanufacturing of automotive mechatronic devices, J Remanufacturing 3 (2013) 1–8.
[11] E. Sundin, Product and process design for successful remanufacturing (Ph.D. thesis), Linköping University Electronic Press, 2004.

[12] M.U. Siddiqi, W.L. Ijomah, G.I. Dobie, M. Hafeez, S. Gareth Pierce, W. Ion, C. Mineo, C.N. MacLeod, Low cost three-dimensional virtual model construction for remanufacturing industry, J Remanufacturing 9 (2) (2019) 129–139.
[13] J.-P. Kaiser, S. Lang, M. Wurster, G. Lanza, A concept for autonomous quality control for core inspection in remanufacturing, Procedia CIRP 105 (2022) 374–379.
[14] M. Peuzin-Jubert, A. Polette, D. Nozais, J.-L. Mari, J.-P. Pernot, Survey on the view planning problem for reverse engineering and automated control applications, Comput Aided Des 141 (2021) 103094.
[15] J.E. Banta, Y. Zhien, X.Z. Wang, G. Zhang, M.T. Smith, M.A. Abidi, Best-next-view algorithm for three-dimensional scene reconstruction using range images, in: Intelligent robots and computer vision xiv: algorithms, techniques, active vision, and materials handling, vol. 2588, 1995, pp. 418–429.
[16] W.R. Scott, Model-based view planning, Mach Vis Appl 20 (1) (2009) 47–69, http://dx.doi.org/10.1007/s00138-007-0110-2.
[17] M. Mendoza, J.I. Vasquez-Gomez, H. Taud, L.E. Sucar, C. Reta, Supervised learning of the next-best-view for 3d object reconstruction, Pattern Recognit Lett 133 (2020) 224–231, http://dx.doi.org/10.1016/j.patrec.2020.02.024.
[18] J.I. Vasquez-Gomez, D. Troncoso, I. Becerra, E. Sucar, R. Murrieta-Cid, Next-best-view regression using a 3D convolutional neural network, Mach Vis Appl 32 (2) (2021) http://dx.doi.org/10.1007/s00138-020-01166-2.
[19] R. Monica, J. Aleotti, A probabilistic next best view planner for depth cameras based on deep learning, IEEE Robot Autom Lett 6 (2) (2021) 3529–3536, http://dx.doi.org/10.1109/LRA.2021.3064298.
[20] S. Pan, H. Hu, H. Wei, SCVP: Learning one-shot view planning via set covering for unknown object reconstruction, IEEE Robot Autom Lett 7 (2) (2022) 1463–1470, http://dx.doi.org/10.1109/LRA.2022.3140449.
[21] R. Zeng, W. Zhao, Y.J. Liu, PC-NBV: A point cloud based deep network for efficient next best view planning, in: 2020 IEEE/RSJ international conference on intelligent robots and systems, IROS, 2020, pp. 7050–7057, http://dx.doi.org/10.1109/IROS45743.2020.9340916.
[22] Kumar Ashutosh, Saurabh Kumar, Subhasis Chaudhuri, 3D-NVS: A 3D supervision approach for next view selection, 2020.
[23] S.J. Russell, P. Norvig, Artificial intelligence - a modern approach, Pearson, 2020.
[24] F. Deinzer, C. Derichs, H. Niemann, J. Denzler, A framework for actively selecting viewpoints in object recognition, 2009, undefined.
[25] Christian Korbach, M.D. Solbach, Raphael Memmesheimer, Dietrich Paulus, J.K. Tsotsos, Next-best-view estimation based on deep reinforcement learning for active object classification, 2021.
[26] Mustafa Devrim Kaba, Mustafa Gokhan Uzunbas, Ser Nam Lim, A reinforcement learning approach to the view planning problem, 2017, pp. 5094–5102, http://dx.doi.org/10.1109/CVPR.2017.541.
[27] S. Potapova, A. Artemov, S. Sviridov, D.A. Musatkina, D. Zorin, Evgeny Burnaev, Next best view planning via reinforcement learning for scanning of arbitrary 3D shapes, 2020, undefined.
[28] C. Landgraf, B. Meese, M. Pabst, G. Martius, M.F. Huber, A reinforcement learning approach to view planning for automated inspection tasks, Sens (Basel, Switz) 21 (6) (2021) 2030.
[29] R.S. Sutton, A.G. Barto, Reinforcement learning: An introduction, MIT Press, 2018.
[30] CoremanNet, Bosch Core acceptance criteria for starter motors, 2022, https://www.coremannet.com/assets/docs/return-criteria/new-2019/Starter.pdf. [Accessed 13 November 2022].
[31] A. Nüchter, J. Hertzberg, Towards semantic maps for mobile robots, Robot Auton Syst 56 (11) (2008) 915–926.
[32] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, N. Dormann, Stable-Baselines3: Reliable reinforcement learning implementations, J Mach Learn Res 22 (268) (2021) 1–8.
[33] D.Q. Huynh, Metrics for 3D rotations: Comparison and analysis, J Math Imaging Vision 35 (2009) 155–164.
[34] R.Q. Charles, H. Su, M. Kaichun, L.J. Guibas, PointNet: Deep learning on point sets for 3D classification and segmentation, in: 2017 IEEE conference on computer vision and pattern recognition, CVPR, 2017, pp. 77–85, http://dx.doi.org/10.1109/CVPR.2017.16.
[35] W. Yuan, T. Khot, D. Held, C. Mertz, M. Hebert, PCN: Point completion network, in: 2018 international conference on 3D vision (3DV), 2018, pp. 728–737, http://dx.doi.org/10.1109/3DV.2018.00088.
[36] C.R. Qi, L. Yi, H. Su, L.J. Guibas, PointNet++: Deep hierarchical feature learning on point sets in a metric space, 2017, http://dx.doi.org/10.48550/ARXIV.1706.02413.
[37] C. Wu, K. Zhou, J.-P. Kaiser, N. Mitschke, J.-F. Klein, J. Pfrommer, J. Beyerer, G. Lanza, M. Heizmann, K. Furmans, MotorFactory: A blender add-on for large dataset generation of small electric motors, Procedia CIRP 106 (2022) 138–143.
[38] S. Ridley, W. Ijomah, Pre-processing inspection–a worthwhile activity for re-manufacturers, in: International conference on remanufacturing, iCoR 2015, 2015.