wbk
Institut für Produktionstechnik

Forschungsberichte aus dem
wbk Institut für Produktionstechnik
Karlsruher Institut für Technologie (KIT)

Jan-Philipp Kaiser

Autonomous View Planning using Reinforcement Learning

Modeling and Application for Visual Inspection in Remanufacturing

Band 294

Forschungsberichte aus dem
wbk Institut für Produktionstechnik
Karlsruher Institut für Technologie (KIT)

Hrsg.: Prof. Dr.-Ing. Jürgen Fleischer
Prof. Dr.-Ing. Gisela Lanza
Prof. Dr.-Ing. habil. Volker Schulze
Prof. Dr.-Ing. Frederik Zanger

Jan-Philipp Kaiser

Autonomous View Planning using Reinforcement Learning

Modeling and Application for Visual Inspection in Remanufacturing

Band 294

Karlsruher Institut für Technologie

wbk
Institut für Produktionstechnik

Autonomous View Planning using Reinforcement Learning
Modeling and Application for Visual Inspection in Remanufacturing

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

von der KIT-Fakultät für Maschinenbau des

Karlsruher Instituts für Technologie (KIT)

angenommene

Dissertation

von

Jan-Philipp Kaiser, M.Sc.

Tag der mündlichen Prüfung: 18.09.2024

Hauptreferentin: Prof. Dr.-Ing. Gisela Lanza

Korreferent: Prof. Dr.-Ing. Jörg Krüger

KIT
Karlsruher Institut für Technologie

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the internet at http://dnb.d-nb.de.

Zugl.: Karlsruhe, Karlsruher Institut für Technologie, Diss., 2024

Print-ISBN 978-3-8440-9794-8
PDF-ISBN 978-3-8440-9888-4
ISSN 2944-6430
eISSN 2944-6449
https://doi.org/10.2370/9783844098884

Shaker Verlag GmbH • Am Langen Graben 15a • 52353 Düren
Phone: 0049/2421/99011-0 • Telefax: 0049/2421/99011-9
Internet: www.shaker.de • e-mail: info@shaker.de

wbk
Institut für Produktionstechnik

Vorwort der Herausgeber

Aktuelle Version des Vorworts der Herausgeber Die schnelle und effiziente Umsetzung innovativer Technologien wird vor dem Hintergrund der Globalisierung der Wirtschaft der entscheidende Wirtschaftsfaktor für produzierende Unternehmen. Universitäten können als “Wertschöpfungspartner” einen wesentlichen Beitrag zur Wettbewerbsfähigkeit der Industrie leisten, indem sie wissenschaftliche Grundlagen sowie neue Methoden und Technologien erarbeiten und aktiv den Umsetzungsprozess in die praktische Anwendung unterstützen.

Vor diesem Hintergrund wird im Rahmen dieser Schriftenreihe über aktuelle Forschungsergebnisse des Instituts für Produktionstechnik (wbk) am Karlsruher Institut für Technologie (KIT) berichtet. Unsere Forschungsarbeiten beschäftigen sich sowohl mit der Leistungssteigerung von additiven und subtraktiven Fertigungsverfahren, den Produktionsanlagen und der Prozessautomatisierung sowie mit der ganzheitlichen Betrachtung und Optimierung der Produktionssysteme und -netzwerke. Hierbei werden jeweils technologische wie auch organisatorische Aspekte betrachtet.

Prof. Dr.-Ing. Jürgen Fleischer
Prof. Dr.-Ing. Gisela Lanza
Prof. Dr.-Ing. habil. Volker Schulze
Prof. Dr.-Ing. Frederik Zanger

KIT
Karlsruher Institut für Technologie

wbk Institut für Produktionstechnik

Vorwort des Verfassers

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am wbk Institut für Produktionstechnik des Karlsruher Instituts für Technologie (KIT) im Rahmen des von der Carl-Zeiss-Stiftung geförderten Forschungsprojekts ’AgiProBot - Agiles Produktionssystem mittels mobiler, lernender Roboter mit Multisensorik bei ungewissen Produktspezifikationen’.

Mein besonderer Dank gilt Frau Prof. Dr.-Ing. Gisela Lanza als Hauptreferentin für ihr Vertrauen und ihre fachliche und persönliche Unterstützung während der gesamten Zeit. Weiterhin bedanke ich mich bei Herrn Prof. Dr.-Ing. Jörg Krüger für die Übernahme des Korreferats und das Interesse an meiner Arbeit sowie Herrn Prof. Dr.-Ing. Dr. h. c. Albert Albers für den Prüfungsvorsitz.

Dem Karlsruhe House of Young Scientists (KHYS) des KIT danke ich für die Förderung meines Forschungsaufenthaltes an der KU Leuven in Leuven, Belgien. Ebenso gilt mein Dank Prof. Jef Peeters der KU Leuven für die herzliche Gastfreundschaft und die wissenschaftlichen Diskussionen.

Für die gute Atmosphäre am und außerhalb des Instituts danke ich allen Kolleginnen und Kollegen des wbk, insbesondere im Bereich Produktionssysteme. Für die Zusammenarbeit und entstandenen Freundschaften bedanke ich mich bei Patrizia, Julia, Marco, Felix, Magnus, Finn und Edwin. Darüber hinaus gilt mein Dank besonders Dominik, Ali und Jork für das Korrekturlesen dieser Arbeit und Jonas Gäbele, Jonas Schmid sowie Verena Hollinger, die mit ihren Abschlussarbeiten und großem Engagement zum Gelingen der Arbeit beigetragen haben.

Letztlich möchte ich besonders meinen Eltern Elke und Martin sowie Stiefeltern Birgit und Klaus für ihre kontinuierliche Unterstützung und ihren steten Rückhalt danken. Abschließend gilt mein Dank Jana für ihr Vertrauen, ihre Energie und ihr Verständnis in den letzten Zügen meiner Arbeit.

Karlsruhe, 18.09.2024

Jan-Philipp Kaiser

KIT Karlsruher Institut für Technologie

Abstract

Companies will have to establish circular production patterns in the future. Remanufacturing is currently a niche process in which used products are remanufactured by first dismantling them, reworking or replacing components and reassembling them into a finished product. Visual inspection is one of the first steps in the entire remanufacturing process, in which a used product is assessed based on its external defects to determine whether it is suitable for remanufacturing. In most cases, visual inspection is manual and therefore expensive due to the flexibility and adaptability required when assessing used products, which hinders establishing remanufacturing in high-wage countries.

The adaptivity of artificial intelligence methods in combination with the flexibility of robotic production resources can be used to automate the visual inspection process in remanufacturing. To date, most authors in the field of remanufacturing, and more specifically visual inspection, have only dealt with product identification, defect localization and defect classification and not with how visual acquisition systems can acquire the data to be evaluated in a goal-oriented manner. The view planning problem is the problem of selecting viewpoints of a visual acquisition system in such a way that a given inspection target is met with the minimum number of acquisitions required. Particularly in remanufacturing, prior knowledge of the product variant to be inspected or its geometry model is oftentimes not available. Analytical approaches to solving the visual planning problem are, therefore, only applicable to a limited extent. The present work closes this research gap by using reinforcement learning (RL) methods to learn an inspection strategy by an RL agent independently of an existing geometry model.

The presented solution approach and methodological procedure provide a simulation framework with which differently modeled RL agents can be trained to solve different visual planning problems specific to the derived requirements of remanufacturing. The results show that the trained RL agents have comparable performance compared to a benchmark heuristic, but poorer performance compared to a near-optimal view plan obtained with an analytical solution method.

The present work represents a significant contribution towards autonomous visual inspection systems using RL algorithms to solve the view planning problem when prior knowledge necessary for selecting or computing the analytical solution is not available.

Kurzfassung

Unternehmen müssen zukünftig zirkuläre Produktionsmuster etablieren. Das Remanufacturing stellt derzeit einen Nischenprozess dar, bei dem Gebrauchtprodukte aufgearbeitet werden, indem diese zunächst demontiert werden, Komponenten aufgearbeitet oder ausgetauscht werden und diese zu einem Gesamtprodukt reassembliert werden. Die visuelle Inspektion stellt einen der ersten Schritte im gesamten Prozess des Remanufacturing dar, bei dem für ein Gebrauchtprodukt basierend auf dessen äußerlichen Mängeln beurteilt wird, ob es sich für das Remanufacturing eignet. Zumeist ist die visuelle Inspektion aufgrund der erforderlichen Flexibilität und Adaptivität bei der Bewertung der Gebrauchtprodukte manuell und somit teuer, was die Etablierung des Remanufacturing in Hochlohnländern behindert.

Die Adaptivität von Verfahren der künstlichen Intelligenz im Zusammenspiel mit der Flexibilität robotischer Produktionsressourcen kann genutzt werden, um den visuellen Inspektionsprozess im Remanufacturing zu automatisieren. Bislang haben sich im Umfeld des Remanufacturing und spezieller der visuellen Inspektion die meisten Autoren lediglich mit der Produktidentifikation, Defektlokalisierung und Defektklassifizierung befasst und nicht damit, wie visuelle Erfassungssysteme die auszuwertenden Daten zielgerichtet erfassen können. Das Sichtplanungsproblem stellt das Problem dar Ansichtspunkte eines visuellen Erfassungssystems so auszuwählen, dass ein vorgegebenes Inspektionsziel mit minimaler Anzahl der hierfür erforderlichen Erfassungen erfüllt wird. Insbesondere im Remanufacturing sind Vorwissen hinsichtlich der zu inspizierenden Produktvariante oder deren Geometriemodell nicht vorhanden. Analytische Lösungsansätze zur Lösung des Sichtplanungsproblem sind daher nur begrenzt anwendbar. Die vorliegende Arbeit schließt diese Forschungslücke, indem Verfahren des Reinforcement Learning (RL) genutzt werden, um eine Inspektionsstrategie durch einen RL Agenten unabhängig von einem vorliegenden Geometriemodell zu lernen.

Die vorgestellte Lösungsansatz und das methodische Vorgehen liefert ein Simulationsframework, mit dem verschieden modellierte RL Agenten zur Lösung verschiedener Sichtplanungsprobleme, spezifisch für die abgeleiteten Anforderungen des Remanufacturing, trainiert werden können. Die Ergebnisse zeigen, dass die trainierten RL Agenten vergleichbare Performanz im Vergleich zu einer Benchmarkheuristik aufweisen, jedoch eine schlechtere Performanz im Vergleich zu einem mit einem analytischen Lösungsverfahren berechneten nahezu optimalen Sichtplan aufweisen.

Die vorliegende Arbeit stellt insbesondere dann einen wesentlichen Beitrag in Richtung autonomer visueller Inspektionssysteme unter Zuhilfenahme von RL Algorithmen zur Lösung des Sichtplanungsproblems dar, wenn Vorwissen, das für eine Auswahl oder Berechnung der analytische Lösung notwendig ist, nicht vorliegt.

Contents

Abbreviations

Abbreviation	Description
BB	Bounding Box
BGR	Blue Green Red
B-Rep	Boundary Representation
CAD	Computer Aided Design
CBR	Convolution, Batch-Normalization and ReLu activation
CLAHE	Contrast Limited Adaptive Histogram Equalization
CM	Coverage Maximization
CNN	Convolutional Neural Network
DoF	Degree of Freedom
FDR	False Discovery Rate
FPR	False Positive Rate
GUM	Guide to the Expression of Uncertainty in Measurement
HDR	High Dynamic Range
ICP	Iterative Closes Point
IoU	Intersection over Union
IPP	Inspection Planning Problem
MLP	Multi-Layer Perceptron
NN	Neural Network
OEM	Original Equipment Manufacturer
OWD	Optimal Working Distance
PCN	Point Completion Network
PN	PointNet
PPO	Proximal Policy Optimization
RCNN	Region Convolutional Neural Network
ReLu	Rectified Linear Unit
RGB	Red Green Blue
RGB-D	Red Green Blue-Depth
RL	Reinforcement Learning
ROI	Region of Interest
ROS	Robot Operating System
RPN	Region Proposal Network
RPP	Reconstruction Planning Problem
SAC	Soft-Actor-Critic
SCP	Set Covering Problem

Abbreviation	Description
SGD	Stochastic Gradient Descend
SL	Supervised Learning
STL	Stereolithography
SRDF	Semantic Robot Description Format
tanh	Tangens Hyperbolicus
T-Net	Transformation Network used in PointNet
URDF	Universal Robot Description Format
USL	Unsupervised Learning
VP	Viewpoint
VPP	View Planning Problem
NBV	Next-Best-View
NURBS	Non-Uniform Rational B-Spline
YOLO	You Only Look Once (neural network architecture)

Symbols

The symbols are sorted by chapter and occurrence. In the case of a double assignment of symbols over the course of this thesis, the description listed in the list of symbols under the respective section is relevant.

Symbol	Description
Section 2.2	View planning for visual inspection systems
S	Object surface space
I	Imaging workspace
V	Viewpoint space
V^S	Generalized viewpoint
$\mathbf{p}^{A/Ref}$	Pose of acquisition system A in relation to reference frame Ref
λ_A	Acquisition system parameters
$\mathbf{p}^{O/Ref}$	Pose of inspection object O in relation to reference frame Ref
x, y, z	Cartesian coordinates
C_A	Coordinate system of an acquisition system A
C_R	Robot base coordinate system
$\mathbf{p}^{A/R}$	Pose of acquisition system A in relation to robot coordinate system C_R
$R^{A/R}$	Rotation matrix of acquisition system coordinate system C_A in relation to robot coordinate system C_R
$T^{A/R}$	Translation vector of acquisition system coordinate system C_A in relation to robot coordinate system C_R
$\mathbf{T}^{A/R}$	Homogeneous transformation matrix of acquisition system coordinate system C_A in relation to robot coordinate system C_R
$\mathbf{P}^A$	Point represented in acquisition coordinate system C_A
$\mathbf{P}^R$	Point represented in robot coordinate system C_R
X^A, Y^A, Z^A	Cartesian coordinates represented in coordinate system C_A
X^R, Y^R, Z^R	Cartesian coordinates represented in coordinate system C_R
Section 2.3	Machine learning and deep learning
$f(\cdot)$	Symbol denoting a function or functional mapping
X	Input space of a function
Y	Output space of a function
N	Number of training examples
$x_{f,i}$	Input training example with $i \in 1, ..., N$

Symbol	**Description**
$y_{f,i}$	Output training example (true label) with $i \in 1, ..., N$
$\hat{y}_{f,i}$	Predicted output for $x_{f,i}$ with $i \in 1, ..., N$
$h(\cdot)$	Approximation of the true function $f(\cdot)$
$h(x_{f,N+1} \vert x_{f,1}, ...x_{f,N})$	Probabilistic hypothesis function for new input $x_{f,N+1}$ given previous inputs $x_{f,1}, ...x_{f,N}$
π	Strategy of an RL agent
t	Time step
a_t, a_{t+1}	Actions of an RL agent at timesteps t and $t+1$
s_t, s_{t+1}	State of an environment the RL agent interacts with at timesteps t and $t+1$
S_E	Space of states of the environment
A_{RL}	Space of actions of the RL agent
r_t, r_{t+1}	Reward given to the agents at timesteps t and $t+1$
π^*	Optimal strategy of the RL agent
G_t	Sum of discounted future rewards, also called long-term reward
γ_{RL}	Discount factor
$\mathbf{w}$	Weight vector of a NN
$h(x \vert \mathbf{w})$	Hypothesis learned by a NN
n_f	Size of one dimension of the quadratic filter matrix of a convolution
m_I	Size of the input matrix of a convolution
$x_f^{i,j}$	Image input of a function. The indices i and j denote the pixel location with $i \in 1, ..., W$ and $j \in 1, ..., H$
W	Image width
H	Image height
N_p	Number of pixels in an image ($N_p = H \times H$)
L	Label space for pixel classification in semantic segmentation
l_i	Specific class in the label space with $i \in 0, ..., k$ and k+1 disjunct classes with class l_0 being the background
$l^{i,j}$	Ground truth classification (label) of a pixel. The indices i and j denote the pixel location with $i \in 1, ..., W$ and $j \in 1, ..., H$
$\hat{y}_f^{i,j}, \hat{l}^{i,j}$	Predicted label of pixel $x_f^{i,j}$ using the semantic segmentation network
$\hat{y}_{f,B}^{UL,i}$	Upper left corner position of BB i with $i \in 1, ..., m$ for m BB
$\hat{y}_{f,B}^{UR,i}$	Upper right corner position of BB i with $i \in 1, ..., m$ for m BB
$\hat{y}_{f,B}^{LL,i}$	Lower left corner position of BB i with $i \in 1, ..., m$ for m BB
$\hat{y}_{f,B}^{LR,i}$	Upper right corner position of BB i with $i \in 1, ..., m$ for m BB
$\hat{y}_{f,B}^{C,i}$	Classification of the object in BB i with $i \in 1, ..., m$ for m BB

Symbol	Description
$\hat{y}_{f,B}^{SM,i}$	Segmentation mask of the object in BB i with $i \in 1, ..., m$ for m BB
$V_\pi(s_t)$	State-value function
$Q_\pi(s_t, a_t)$	Action-value function
$\hat{V}_\pi(s_t)$	Estimate of the state-value function
$\hat{Q}_\pi(s_t, a_t)$	Estimate of the action-value function
$\pi(a_t \| s_t, \mathbf{w}_a)$	Parameterized policy
$\mathbf{w}_a$	Parameter vector of a parameterized policy (actor)
$J(\mathbf{w}_a)$	Performance measure of a policy parameterized with a parameter vector $\mathbf{w}_a$
$\Delta\mathbf{w}_a$	Weight update of the parameter vector of a parameterized policy (actor)
η	Learning rate of a learning algorithm
$\nabla_{\mathbf{w}_a} J(\mathbf{w}_a)$	Gradient of the performance measure of a policy parameterized with a parameter vector $\mathbf{w}_a$
$\hat{J}(\mathbf{w}_a)$	Estimate of the performance measure of a policy parameterized with a parameter vector $\mathbf{w}_a$
$\nabla\pi(a_t \| s_t, \mathbf{w}_a)$	Gradient of the policy $\pi(a_t \| s_t, \mathbf{w}_a)$
$\mathbf{w}_c$	Parameter vector of a state-value approximator (critic)
$\hat{V}(s_t \| \mathbf{w}_c), \hat{V}(s_{t+1} \| \mathbf{w}_c)$	Estimate of the state-value function at time steps t and $t+1$ using a state-value approximator with parameters $\mathbf{w}_c$
$\mathcal{L} = f_L(\cdot)$	Mean square error loss function for the state-value approximation
Section 5.1	Setup, modeling, and control of the inspection station and 3D reconstruction of the inspection object
$\mathbf{P}_{t,i}^{A}$	Point i of a point cloud acquired with an acquisition system A at time step t
$\mathbf{PC}_t^A$	RGB-D point cloud acquired with an acquisition system A at time step t
$\mathbf{X}_t^A$	Vector of x-axis coordinates of the points in $\mathbf{PC}_t^A$
$\mathbf{Y}_t^A$	Vector of y-axis coordinates of the points in $\mathbf{PC}_t^A$
$\mathbf{Z}_t^A$	Vector of z-axis coordinates of the points in $\mathbf{PC}_t^A$
$\mathbf{R}_t^A$	Vector of red channel values of the points in $\mathbf{PC}_t^A$
$\mathbf{G}_t^A$	Vector of green channel values of the points in $\mathbf{PC}_t^A$
$\mathbf{B}_t^A$	Vector of blue channel values of the points in $\mathbf{PC}_t^A$

Symbol	Description
$X_{t,i}^{A}$	Value of x-axis coordinates of the point i in $\mathbf{PC}_t^A$
$Y_{t,i}^{A}$	Value of y-axis coordinates of the point i in $\mathbf{PC}_t^A$
$Z_{t,i}^{A}$	Value of z-axis coordinates of the point i in $\mathbf{PC}_t^A$
$\mathbf{C}_{t,i}^{A}$	Color information of point i of a point cloud acquired with an acquisition system A at time step t
$R_{t,i}^{A}$	Color information of the red channel of the point i in $\mathbf{PC}_t^A$
$G_{t,i}^{A}$	Color information of the green channel of the point i in $\mathbf{PC}_t^A$
$B_{t,i}^{A}$	Color information of the blue channel of the point i in $\mathbf{PC}_t^A$
$\mathbf{P}_{t}^{A}$	Set of three-dimensional points in $\mathbf{PC}_t^A$
$I_{C,t}$	Color image acquired by an acquisition system at time step t
$I_{D,t}$	Depth image acquired by an acquisition system at time step t
$\mathbf{p}_{t}^{A/R}$	Pose of the acquisition system with coordinate system C_A in relation to robot coordinate system C_R at time step t
C_E	Coordinate system of the end-effector of a robot
C_{RT}	Static coordinate system of the rotary table
φ_{RT}	Absolute rotation angle of the rotary table
$C_{RT'}$	Co-rotating coordinate system of the rotary table
$\mathbf{T}^{R/E}$	Homogeneous transformation matrix of robot coordinate system C_R in relation to robot end-effector coordinate system C_E
$\mathbf{T}^{A/R}$	Homogeneous transformation matrix of acquisition system coordinate system C_A in relation to robot coordinate system C_R
$\mathbf{T}^{E/A}$	Homogeneous transformation matrix of end-effector coordinate system C_E in relation to acquisition system coordinate system C_A
$\mathbf{T}^{R/RT}$	Homogeneous transformation matrix of robot coordinate system C_R in relation to rotary table coordinate system C_{RT}
$\mathbf{T}^{RT/RT'}, \mathbf{T}^{RT'/RT}$	Homogeneous transformation matrix of rotary table coordinate system C_{RT} in relation to co-rotating rotary table coordinate system $C_{RT'}$ and the inverse rotation ($\mathbf{T}^{RT'/RT} = (\mathbf{T}^{RT/RT'})^{-1}$)
$\mathbf{T}^{A/RT'}$	Homogeneous transformation matrix of acquisition system coordinate system C_A in relation to co-rotating rotary table coordinate system $C_{RT'}$
$\mathbf{P}_{t,i}^{RT'}$	Point i of a point cloud represented in co-rotating rotary table coordinate system $C_{RT'}$, acquired at time step t
$X_{t,i}^{RT'}$	Value of x-axis coordinates of the point i in $\mathbf{PC}_t^{RT'}$
$Y_{t,i}^{RT'}$	Value of y-axis coordinates of the point i in $\mathbf{PC}_t^{RT'}$
$Z_{t,i}^{RT'}$	Value of z-axis coordinates of the point i in $\mathbf{PC}_t^{RT'}$

Symbol	Description
$\mathbf{PC}_t^{RT'}$	RGB-D point cloud acquired at time step t represented in the co-rotating rotary table coordinate system $C_{RT'}$
$\mathbf{X}_t^{RT'}$	Vector of x-axis coordinates of the points in $\mathbf{PC}_t^{RT'}$
$\mathbf{Y}_t^{RT'}$	Vector of y-axis coordinates of the points in $\mathbf{PC}_t^{RT'}$
$\mathbf{Z}_t^{RT'}$	Vector of z-axis coordinates of the points in $\mathbf{PC}_t^{RT'}$
$\mathbf{C}_t^{A}$	Color information acquired with an acquisition system at time step t represented in coordinate system C_A
$\mathbf{R}_t^{RT'}$	Vector of red channel values of the points in $\mathbf{PC}_t^{RT'}$
$\mathbf{G}_t^{RT'}$	Vector of green channel values of the points in $\mathbf{PC}_t^{RT'}$
$\mathbf{B}_t^{RT'}$	Vector of blue channel values of the points in $\mathbf{PC}_t^{RT'}$
$\mathbf{C}_t^{RT'}$	Color information acquired with an acquisition system at time step t represented in coordinate system $C_{RT'}$
$\Delta\varphi_{RT,t}$	Relative rotation angle of the rotary table at time step t
$\mathbf{PC}_{res,t-1}^{RT',t-1}$	Point cloud model of the ongoing reconstruction process at time step $t-1$ represented in the co-rotating rotary table coordinate system $C_{RT'}$ of the reconstruction step $t-1$
$C_{RT,t-1}$	Coordinate system configuration at time step $t-1$
$\mathbf{PC}_{res,t-1}^{RT'}$	Point cloud model of the ongoing reconstruction process at time step $t-1$ represented in the co-rotating rotary table coordinate system $C_{RT'}$ of the reconstruction step t
$\mathbf{T}(R_z(\Delta\varphi_{RT,t}))^{RT'_{t-1}/RT'_t}$	Transformation matrix resulting from the rotation of the rotary table with $\Delta\varphi_{RT,t}$
$\mathbf{PC}_{t,icp}^{RT'}$	Point cloud model of the ongoing reconstruction process at time step t represented in the co-rotating rotary table coordinate system $C_{RT'}$ of the reconstruction step t and registered using ICP algorithm
$\mathbf{PC}_f^{R} = \mathbf{PC}_{res,t}^{R}$	Final point cloud model of the reconstruction process represented in the coordinate system C_R
Section 5.2	Approach for semantic 3D Reconstruction of the inspection object
I_C	Color image acquired by an acquisition system
I_D	Depth image acquired by an acquisition system
φ_A	Azimuth angle for Spherical coordinates of the acquisition system position
θ_A	Polar angle for Spherical coordinates of the acquisition system position
r_A	Distance for Spherical coordinates of the acquisition system position

Symbol	**Description**
x_A, y_A, z_A	Value of x,y and z-axis coordinates of the position of the acquisition system C_A in cartesian coordinates
$\alpha_A, \beta_A, \gamma_A$	Euler angles specifying the orientation of the acquisition system
$\Delta\alpha_A, \Delta\beta_A, \Delta\gamma_A$	Offsets of the Euler angles specifying the acquisition systems orientation
C_{GT}	Segmentation label mask
$H \times W \times C$	Size of the label segmentation mask with image height H, image width W and number of classes C
I_C^*	Preprocessed RGB image
I_D^*	Preprocessed depth image
C_{GT}^*	Preprocessed label segmentation mask
$I_C^{'}$	Preprocessed and augmented RGB image
$I_D^{'}$	Preprocessed and augmented depth image
$C_{GT}^{'}$	Preprocessed and augmented label segmentation mask
C_P	Predicted label segmentation mask of the segmentation NN
U	Expanded uncertainty
k	Expansion factor
u_C	Combined standard uncertainty
$\hat{y}$	Output estimate of a measurement function
$f(x_1, ..., x_N)$	Measurement function
$u(x_i)$	Standard uncertainty of a input quantity
x_i	Input quantity of a measurement function
c_i	Sensitivity coefficient
y_i	Output of a simulated measurement
M_i	ith model of a NN ensemble
$D_{T,i}$	ith data tuple of the test dataset
$\hat{y}(D_{T,i})_{uv}$	Pixel-wise prediction of a NN model ensemble
$\hat{y}_{T,i,j}$	Model prediction of the jth model and the ith data tuple
$\hat{\sigma}(D_{T,i})_{uv}$	Pixel-wise predictive uncertainty
$D_{T,ij}^{'}$	jth augmentation of ith data tuple
$(\hat{y}_{T,ij,k})_{uv}$	Pixel-wise prediction of the jth augmentation of the ith data tuple for model k
$\mathbf{SPC}_t^A$	Semantic point cloud
$\mathbf{K}_t$	Vector containing the component classification results of all points in $\mathbf{SPC}_t^A$
K_i	Classification of point i

Symbol	**Description**
$\mathbf{D}_t$	Vector containing the defect classification results of all points in $\mathbf{SPC}_t^A$
Section 5.3	View planning using reinforcement learning
S_{Sim}	Simulation environment
I_{Agent}	Agent interface
A_{Plan}	Planning Agent
Ψ	Percentual surface coverage
$\mathbf{V}_t^S$	Viewpoint at time step t
$\mathbf{o}_t$	Observation at time step t
$\mathbf{a}_t$	Action at time step t
$\mathbf{s}_t$	State at time step t
$\mathbf{w}$	Parameters of the policy of the agent
$\mathbf{P}_t^A$	Point cloud acquired at time step t in reference coordinate system C_A
$\mathbf{p}_t^{A/RT}$	Pose of the acquisition system A in relation to the static rotary table coordinate system C_{RT} at time step t
$\mathbf{p}_t^{O/RT}$	Pose of the inspection object O in relation to the static rotary table coordinate system C_{RT} at time step t
$\mathbf{p}_{t-1}^{A/RT}$	Pose of the acquisition system A in relation to C_{RT} at time step $t-1$
C_O	Inspection object coordinate system
$X_t^{A/RT}, Y_t^{A/RT}, Z_t^{A/RT}$	Value of the x,y and z-axis coordinates of the pose $\mathbf{p}_t^{A/RT}$
$\alpha_t^{A/RT}, \beta_t^{A/RT}, \gamma_t^{A/RT}$	Euler angles of the orientation of the acquisition system A in relation to the static rotary table coordinate system C_{RT} at time step t
$\mathbf{T}_{p,t}^{A/RT}$	Homogeneous transformation matrix of the pose $\mathbf{p}_t^{A/RT}$
$\mathbf{T}_{p,t}^{A/R}$	Homogeneous transformation matrix of the pose $\mathbf{p}_t^{A/R}$
R_{uv}	Resolution of the acquisition system
$\boldsymbol{\vartheta}_{A,uv}$	Aperture angles of the acquisition system
b_{near}	Near bound of the acquisition system frustum
b_{far}	Far bound of the acquisition system frustum
$\vartheta_{A,C}$	Cut-off angle of the acquisition system
$\mathbf{P}_{GT}^{RT'}$	Ground truth point cloud represented in the co-rotating rotary table coordinate system $C_{RT'}$
n_{GT}	Number of points in $\mathbf{P}_{GT}^{RT'}$
$\mathbf{P}_G^{RT'}$	Goal point cloud represented in the co-rotating rotary table coordinate system $C_{RT'}$
$\mathbf{P}_t^{RT'}$	Point cloud acquired at time step t represented in $C_{RT'}$

Symbol	Description
$\mathbf{P}_{cov,t}^{RT'}$	Total point cloud of all acquired point clouds until current time step t represented in $\mathbf{p}_t^{A/RT}$
$\varphi_{RT,t}$	Absolute rotary table angle at time step t
s_G	Goal state
Ψ_t	Percentual surface coverage at time step t
Ψ_T	Percentual surface coverage threshold
s_T	Terminal state
Ψ_E	Percentual surface coverage at the end of an episode
n_E	Number of acquisitions needed to end an episode
$\mathbf{P}_{t,m}^{RT'}$	mth point of the point cloud $\mathbf{P}_t^{RT'}$
$X_{t,m}^{RT'}, Y_{t,m}^{RT'}, Z_{t,m}^{RT'}$	Value of the x,y and z-axis coordinates of the mth point of the point cloud $\mathbf{P}_t^{RT'}$
$n_{r,t,m}$	Radius of the normal of the mth point of the point cloud $\mathbf{P}_t^{RT'}$
$n_{\varphi,t,m}$	Azimuth angle of the normal of the mth point of the point cloud $\mathbf{P}_t^{RT'}$
$n_{\theta,t,m}$	Polar angle of the normal of the mth point of the point cloud $\mathbf{P}_t^{RT'}$
$\Delta_{t,m}$	Binary coding whether the mth point of the point cloud $\mathbf{P}_t^{RT'}$ has been acquired with the last acquisition
$\mathbf{s}_{t,modf,\mathbf{PC}}$	Model-free state containing point cloud information
$\mathbf{s}_{t,modf,\mathbf{p}}$	Model-free state containing pose information
$\mathbf{s}_{t,modb,\mathbf{PC}}$	Model-based state containing point cloud information
$\mathbf{s}_{t,modb,\mathbf{p}}$	Model-based state containing pose information
$n_{r,t,m}$	Length of a normal vector in spherical coordinates
$b_{t,m}$	Binary/ternary encoding of the acquisition state
S_X	State modeling alternative X with $X \in 1, ..., 6$
$a_{t,i}$	ith component of the action vector $\mathbf{a}_t$
$\varphi_t^{A/RT}$	Azimuth angle of the pose of the acquisition A in relation to coordinate system C_{RT} at time step t
$\theta_t^{A/RT}$	Polar angle of the pose of the acquisition A in relation to coordinate system C_{RT} at time step t
$r_t^{A/RT}$	Radius of the pose of the acquisition A in relation to coordinate system C_{RT} at time step t
$\Delta\varphi_{RT,t}$	Relative rotation angle of the rotary table based on the output of the RL agent
$\Delta\mathbf{p}_t^{O/RT}$	Relative pose based on the relative rotation angle $\Delta\varphi_{RT,t}$
M_X	Abbreviation of the integration of prior knowledge X into the action mapping

Symbol	Description
$\Delta\alpha_t^{A/RT}, \Delta\beta_t^{A/RT}$	Angle-based deviations of the Euler angles α_t^{RT} and β_t^{RT} determined by the automated orientation of the acquisition system A towards the center of the coordinate system C_{RT} at time step t
A_X	Abbreviation of the action mapping alternative X
d_E	Overall length of travel distances of the agent in one episode
R	Reward signal
$f(\Psi_E, n_E, d_E)$	General formulation of the reward signal as a function of the percentage surface coverage, number of acquisitions and overall length of travel distances
R_{dense}	Dense reward function
R_{sparse}	Sparse reward function
$\Delta\Psi_t$	Percentage of additionally acquired surface points that have not been acquired by previous acquisitions at time step t
d_t	Travel path length between the pose $\mathbf{p}_{t-1}^{A/RT}$ of time step $t-1$ and the pose $\mathbf{p}_t^{A/RT}$ output by the agent at time step t
$n_{cov,t}$	Number of points acquired on the ground truth point cloud $\mathbf{P}_{GT}^{RT'}$ up until time step t
ϵ	Threshold distance determining if a point on the ground truth point cloud $\mathbf{P}_{GT}^{RT'}$ has been acquired or not
$\Delta n_{cov,t}$	Number of surface points newly acquired in the acquisition at time step t
$\Psi_{t,rem}$	Percentage of remaining object surface to be covered at time step t
$\mathbf{p}_S^{RT}$	Auxiliary point for travel length calculation
$d_{t,lin}$	Linear travel path for travel length calculation
$d_{t,sph}$	Spherical travel path for travel length calculation
R_X	Reward modeling alternative X
$\mathbf{w}$	Parameter vector of the parameterized policy (actor) and value function approximator (critic)
$\mathbf{w}_\Phi$	Parameters of the value function approximator (critic)
$\mathbf{w}_\Psi$	Parameter vector of the parameterized policy (actor)
$\mathbf{v}_{g,PN}$	Global feature vector of the PN
$\mathbf{v}_{g,PCN}$	Global feature vector of the PCN
$\mathbf{s}_{t,\mathbf{PC}}$	Point cloud based state being either $\mathbf{s}_{t,modf,\mathbf{PC}}$ or $\mathbf{s}_{t,modb,\mathbf{PC}}$
$\mathbf{s}_{t,\mathbf{p}}$	Point cloud based state being either $\mathbf{s}_{t,modf,\mathbf{p}}$ or $\mathbf{s}_{t,modb,\mathbf{p}}$

Symbol	**Description**
Section 6.1	Results for the approaches of 3D reconstruction and semantic 3D reconstruction
$HD_{95}^{C\to L}$	95th percentile of the Hausdorff distance
$ME^{C\to L}$	Mean Error
Section 6.2	RL for the solution of VPPs in remanufacturing
$\mathbf{SCP}_{\boldsymbol{\Psi}}$	Set covering benchmark algorithm focusing solely on surface coverage maximization
$\mathbf{SCP}_{\boldsymbol{\Psi},\mathbf{d}}$	Set covering benchmark algorithm focusing on joint maximization of surface coverage and traveling path length
$\mathbf{SCP}_{\boldsymbol{\Psi}}^{\mathbf{D1,cov}}$	Set covering benchmark algorithm focusing solely on surface coverage maximization using dataset D1
$\mathbf{SCP}_{\boldsymbol{\Psi},\mathbf{d}}^{\mathbf{D1,cov}}$	Set covering benchmark algorithm focusing on joint optimization of surface coverage and traveling path length using dataset D1
$\mathbf{SCP}_{\boldsymbol{\Psi}}^{\mathbf{D2,cov}}$	Set covering benchmark algorithm focusing solely on surface coverage maximization using dataset D2
$\mathbf{SCP}_{\boldsymbol{\Psi},\mathbf{d}}^{\mathbf{D2,cov}}$	Set covering benchmark algorithm focusing on joint optimization of surface coverage and traveling path length using dataset D2
$\mathbf{SCP}_{\boldsymbol{\Psi}}^{\mathbf{D1,Ins}}$	Set covering benchmark algorithm for the IPP problem for dataset D1
$\mathbf{SCP}_{\boldsymbol{\Psi}}^{\mathbf{D2,Ins}}$	Set covering benchmark algorithm for the IPP problem for dataset D2
$\mathbf{SCP}_{\boldsymbol{\Psi}}^{\mathbf{D3,Ins}}$	Set covering benchmark algorithm for the IPP problem for dataset D3
$F1$	$F1$ is the harmonic meean between R and P
FDR	False discovery rate
FPR	False positive rate
IoU_{filt}	Filtered IoU based on pixel uncertainty $\hat{\sigma}(D_{T,i})_{uv}$
$F_{1,filt}$	Filtered IoU based on pixel uncertainty $\hat{\sigma}(D_{T,i})_{uv}$
FDR_{filt}	Filtered FDR based on pixel uncertainty $\hat{\sigma}(D_{T,i})_{uv}$
FPR_{filt}	Filtered FPR based on pixel uncertainty $\hat{\sigma}(D_{T,i})_{uv}$
Appendix A1	General solution approach for SCP, CM and NBV
U	Set of surface elements to be acquired of an inspection object
u_i	ith surface element containing surface sections of the overall inspection object to be acquired
G	Set of all combinations of surface sections seen from all VP j ($j \in [1, ..., m]$)
g_j	Surface sections acquirable from the jth VP

Symbol	Description
g^i	Acquired surface sections for acquisition i via NBV planning
$g^{R,i}$	Estimated remaining unknown surface sections after acquisition i
g^i_o	Subset of estimated acquirable surface sections after ith acquisition with VP o
Appendix A2	In-depth basics of coordinate transformations
α, β, γ	Euler angles of a coordinate system in relation to another
X, Y', Z''	Principal axes for subsequent rotation with Caradan angles
$R_x(\alpha)$	Rotation matrix for Euler angle α around X-axis
$R_{y'}(\beta)$	Rotation matrix for Euler angle β around Y'-axis
$R_{z''}(\gamma)$	Rotation matrix for Euler angle γ around Z"-axis
R	Combined rotation matrix based on $R_x(\alpha)$, $R_{y'}(\beta)$ and $R_{z''}(\beta)$
r	Radial distance of spherical coordinates
θ	Polar angle in spherical coordinates
φ	Azimuth angle in spherical coordinates
Appendix A3	Basics of the perceptron, multi-layer perceptron, and backpropagation algorithm
$\mathbf{x_f}$	Input feature vector of a perceptron
y_f	Output value of the perceptron
$\mathbf{w}$	Weight vector of the perceptron
$Z(\mathbf{x_f}, \mathbf{w})$	Weighted sum calculated inside the perceptron based on $\mathbf{x_f}$ and $\mathbf{w}$
$a(Z(\mathbf{x_f}, \mathbf{w}))$	Activation function of the perceptron
w_i	ith weight of the weight vector $\mathbf{w}$
$x_{f,i}$	ith feature of the feature vector $\mathbf{x_f}$ with $x_{f,0}$ = 1
$a_i(Z(\mathbf{x_f}, \mathbf{w}))$	Activation value of the ith neuron in a layer of a neural network (used, e.g., to compute
$w^l_{i,j}$	weight of a neuron in layer l, which forms the connection with neuron i in the previous layer and neuron j in layer l
$\hat{y}_{f,i}$	Prediction of a NN for an input $\mathbf{x_{f,i}}$ and weight vector $\mathbf{w}$
$y_{f,i}$	Actual label for the output of input $\mathbf{x_{f,i}}$
$\mathcal{L}$	Loss function
$f_L(\hat{y}_{f,i}, \mathbf{w}, x_{f,i})$	Loss function expressed as a function of input, predicted output and weight vector of the NN
w_n	Value of weight n of the weight vector $\mathbf{w}$
Δw_n	Change of weight between iteration steps of the backpropagation algorithm

Symbol	Description
η	Learning rate
$\nabla_{w_n}\mathcal{L} = -\eta\frac{\partial\mathcal{L}}{w_n}$	Partial derivative of the loss function $\mathcal{L}$ with respect to weight w_n
Appendix A4	Details on point cloud transformation using $\mathbf{T}^{A/RT'}$ and determination of $\mathbf{T}^{E/A}$ and $\mathbf{T}^{R/RT}$
$\mathbf{T}^{R/RT'}$	Homogeneous transformation matrix of robot coordinate system C_R in relation to the co-rotating rotary table coordinate system $C_{RT'}$
$\mathbf{T}^{RT'/R}$	Homogeneous transformation matrix of the co-rotating rotary table coordinate system $C_{RT'}$ in relation to the robot coordinate system C_R
$\mathbf{T}^{RT/R}$	Homogeneous transformation matrix of the rotary table coordinate system C_R in relation to the robot coordinate system C_R
$\mathbf{T}^{CAL/R}$	Homogeneous transformation matrix of the calibration pattern coordinate system C_{CAL} in relation to the robot coordinate system C_R
$\mathbf{T}^{CAL/A}$	Homogeneous transformation matrix of the calibration pattern coordinate system C_{CAL} in relation to the acquisition system coordinate system C_A
$\varphi_{RT,i}$	ith of n initial absolute rotation angles calculated for rotary table calibration
$\varphi_{RT,j}$	jth of n secondary absolute rotation angles calculated for rotary table calibration
$I_{D,i}$	Color image for the ith initial rotation of the rotary table for rotary table calibration
$I_{D,j}$	Color image for the jth secondary rotation of the rotary table for rotary table calibration
$\mathbf{P}^{R}_{C,i}$	Keypoints detected in the ith color image $I_{D,i}$ expressed in coordinate system of the robot C_R
$\mathbf{P}^{R}_{C,j}$	Keypoints detected in the jth color image $I_{D,j}$ expressed in coordinate system of the robot C_R
$\mathbf{P}^{A}_{C,i}$	Keypoints detected in the ith color image $I_{D,i}$ expressed in coordinate system of the acquisition system C_A
$\mathbf{P}^{A}_{C,j}$	Keypoints detected in the ith color image $I_{D,i}$ expressed in coordinate system of the acquisition system C_A
Appendix A5	In-depth details and foundations on the segmentation model architectures and training procedure
L_{Dice}	Dice loss
$p_c(u,v)$	Prediction of class membership of pixel (u,v) to class c

Symbol	**Description**
$g_c(u,v)$	Ground truth class membership of pixel (u,v) to class c
L_{Focal}	Focal loss
C	Number of classes to be segmented
α_c	Class dependent balancing factor when using focal loss
$p_t(u,v)$	Quantity derived from $p_c(u,v)$ and $g_c(u,v)$
γ_f	Modulating factor when using the focal loss
L_{M-RCNN}	Overall loss function for the M-Nets and MD-Nets
L_{box}	BB regression loss of the M-Nets and MD-Nets
L_{cls}	Classification loss of the M-Nets and MD-Nets
L_{mask}	Segmentation loss of the M-Nets and MD-Nets
Appendix A7	Evaluation metrics for the 3D reconstruction approach
X	Metric space
C	Set (in this thesis points) in a metric space X
L	Set (in this thesis points) in a metric space X
$d_{95}(C,L)$	95th percentile of the distances between all points in point cloud C to all points in point cloud L
$d_{95}(L,C)$	95th percentile of the distances between all points in point cloud L to all points in point cloud C
n	Number of points in point cloud C
c_i	Point in point cloud C
$d(c_i,l)$	Distance of point c_i in point cloud C to point l in point cloud L
Appendix A9	Evaluation metrics for the semantic segmentation approach using uncertainty-based filtering
P	Precision
R	Recall
TP	Number of true positive predictions
FP	Number of false positive predictions
FN	Number of false negative predictions
FDR	False discovery rate
FPR	False positive rate
Appendix A10	Details on the SCP benchmark algorithms used in this work
d_{VP}	Travel distance from the last selected VP to the current VP
d_{max}	Maximum travel distance of the last VP to any VP in the set of all VP evaluated in the set covering problem at hand

1 Introduction

The increase in resource consumption and the scarcity of key raw materials are forcing companies to adapt their mostly linear production strategies (Kara & Hauschild et al. 2022, P. 505). Companies will have to deal more intensively with the conflict between economic activity and implementing sustainable production patterns in the future (Corvellec & Böhm et al. 2020, P. 100). The circular economy represents an economic system that can address this conflict. It replaces the 'end-of-life' concept with the reduction or the reuse, recycling, or recovery of materials in production, distribution, or consumption processes (Kirchherr & Reike et al. 2017, P. 229). Remanufacturing is a circular process that is a key factor in the realization of the circular economy paradigm through disassembling a used product, reprocessing its components, and reassembling it (Tolio & Bernard et al. 2017, P. 585). Remanufacturing is considered the circular process with the highest standard in terms of quality and warranty of remanufactured products compared to alternative methods and is also the only circular process that can compete with a new product in these aspects (Matsumoto & Ijomah 2013, P. 392). Despite high potential savings in terms of costs, energy, and material consumption, as well as CO_2 emissions, remanufacturing remains a niche market (Parker & Riley et al. 2015, P. 1).

1.1 Problem statement

One of the reasons for the low intensity of remanufacturing (ratio of remanufacturing to new manufacturing) is that remanufacturing involves a large amount of manual labor across all processes (Parker & Riley et al. 2015, P. 65). As a result, even if a company successfully implements remanufacturing processes, they are often carried out in low-wage countries (Seitz & Peattie 2004, P. 83). A highly relevant but not directly value-adding step is the selection of returning used products, including the initial visual inspection by a worker, allowing a technological and economic decision regarding the remanufacturability of the returning used products (Schlüter & Lickert et al. 2021, P. 300). The collection of further information, such as the external appearance, also enables the planning of subsequent process steps, such as dismantling (Vongbunyong & Chen et al. 2015, P. 56). Automating the initial visual inspection offers, as for automated disassembly (Tolio & Bernard et al. 2017, P. 592), a reduction of process costs, making remanufacturing more profitable overall, especially in high-wage countries. However, the high degree of uncertainty in remanufacturing, based on the large number of variants (Kurilova-Palisaitiene & Sundin et al. 2018, P. 3229) of returning products, coupled with small batch sizes (Lundmark & Sundin et al. 2009, P. 135) with varying quality state of the returning products (Errington & Childe 2013, P. 19), still requires human flexibility and adaptability (Seitz & Peattie 2004, P. 83).

1.2 Research objective

An example of the needed flexibility and adaptivity is given by the challenges in inspection resulting from the varying inspection depth due to uncertainty regarding the product's condition. Some defects are immediately apparent, while individual components must be inspected in more detail to identify certain defects. Given the large number of product variants, the returning product and its components must be visually inspected, whereas the quality flaws, defects, and their combinations to be evaluated can occur anywhere on the product with varying extent and composition (Kin & Ong et al. 2014, P. 191 ff.). An inspection process is, therefore, unique for every used product, and humans can adapt to the variability in the inspection process.

Using artificial intelligence, current research is already addressing challenges regarding the multitude of possible defects and flaws to be detected and evaluated based on image data in remanufacturing (Saiz & Alfaro et al. 2021,Nwankpa & Ijomah et al. 2021). The advantage of artificial intelligence methods is that they enable problems to be solved by implicitly extracting knowledge from data and are therefore independent of an explicit definition of human decision rules (Domingos 2012, P. 78). This allows problems that are difficult for humans to formalize to be solved. Alongside automated defect detection and evaluation, each product's unique inspection process introduces challenges in automated data collection to detect and evaluate these defects. An inspection plan must be adapted reactively to each product since quality flaws to be inspected in depth may occur anywhere on the returning product. Therefore, planning an existing product's automated visual inspection process in advance is impossible. This thesis, therefore, deals with the overarching question of how data acquisition can be achieved without specific pre-planning of the inspection process in remanufacturing.

To meet these requirements, the adaptivity of artificial intelligence methods can be combined with robots as flexible production resources to enable autonomous system behavior (Salkin & Oner et al. 2018, P. 7). Autonomy of a resource refers to the accomplishment of a goal without external control by executing actions of an action plan generated by sensory information (Beer & Fisk et al. 2014, P. 76). Autonomous systems, therefore, differ from rule-based automated systems in their ability to reactively adapt their system behavior to a given situation (Endsley 2017, P. 6). This way, it is possible to autonomize the data acquisition for the initial visual inspection by reactively solving the visual planning problem resulting from the prevailing uncertainty at system runtime. The fundamental research objective of this dissertation is therefore:

Enabling an autonomous robotic visual acquisition system to conduct data acquisition for the initial visual inspection in remanufacturing under prevailing uncertainty

1.3 Structuring questions

Based on the overarching research objective in remanufacturing, the following three structuring questions can be derived to structure the fundamentals and systematically deduce the research deficit:

1. **Problem formalization:** How can the problem of autonomous data acquisition be formalized?
2. **Solution approach:** Which group of solution methods exists for solving the formalized data acquisition problem?
3. **Adaptivity:** How can the adaptivity of such solution methods be ensured to deal with the inability to plan the inspection process in advance?

1.4 Structure of the thesis

The contents of chapter 2 introducing the fundamentals are structured based on the structuring questions. This section discusses remanufacturing and the specific challenges of initial visual inspection in more detail. The basics of acquisition planning are introduced, and the basics of artificial intelligence methods are examined in more detail. Answering the respective structuring questions within the sub-chapters makes the specific discussion of the state of research in chapter 3 possible. From this, a clear research deficit can be formulated for the present use case. Based on the deduced research deficit, the solution approach of this thesis is presented in chapter 4. The methodological approach to resolve the derived research deficit and to investigate autonomous robotic visual acquisition systems for visual inspection in remanufacturing is then presented in chapter 5. The findings based on the methodological approach are presented in chapter 6. Chapter 7 summarizes the findings with a discussion of these and an outlook for future research work before this thesis concludes with a summary in chapter 8.

2 Fundamentals

The following chapter presents the required fundamentals of this thesis. Chapter 2.1 first dives into the fundamentals of remanufacturing and the challenges associated with the automation of the initial visual inspection. Chapter 2.2 introduces and formalizes the basics of visual planning problems. Finally, chapter 2.3 discusses the basics of machine learning in more detail.

2.1 Visual inspection in remanufacturing

An overview of remanufacturing and the processes involved is given in the section 2.1.1. Section 2.1.2 then examines the procedure and general boundary conditions for the initial visual inspection of used products in more detail. A short overview of automated visual inspection is given in section 2.1.3. Finally, section 2.1.4 discusses the challenges of automating the initial visual inspection of used products.

2.1.1 The remanufacturing process

Remanufacturing is defined as a process in which a product or a subsystem is rebuilt to be used in a new lifecycle (Seaver 1994, P. 24). Remanufacturing differs from recycling in the way how the function of the product, subsystems, or components are retained, while recycling is merely a form of material recovery (Tolio & Bernard et al. 2017, P. 3). Thus, remanufacturing maintains the value added to a product at a high level by recovering a substantial proportion of the resources incorporated in a used product in its first manufacturing (Ijomah & Childe et al. 2004, P. 1). Remanufacturing thus distinguishes itself from repair or reconditioning, as it is the only process that can compete with a new product in terms of quality (Matsumoto & Ijomah 2013, P. 390). Depending on the author, further aspects are added to the definition of remanufacturing. Ijomah & Childe et al. (2004) add restoring the functionality of the used product to the level of the new product with a matching warranty (Ijomah & Childe et al. 2004, P. 1). Khor & Udin (2012) even mention restoring "at least" the functionality of the new product (Khor & Udin 2012, P. 7) while others explicitly mention the possibility of functional upgrades (Tolio & Bernard et al. 2017, P. 3, Lund & Hauser 2010, P. 1, Thierry & Salomon et al. 1995, P. 119, Haynsworth & Lyons 1987, P. 24). The implementation of remanufacturing within the scope of an industrial process also plays a key role when differentiating it from other processes of the circular economy, such as repair (Tolio & Bernard et al. 2017, P. 2, Haynsworth & Lyons 1987, P. 24). National and international institutions have also taken up and adopted these definitions in the past (cf. Lange 2017, P. 11, BS8887-2:2009 2009, ANSI RIC001.2-2021 2021), including the most recent one from the German Institute for Standardization (DIN SPEC 91472:2023-06 2023).

The type and sequence of processes involved in remanufacturing can vary to meet the constraints and requirements of the product and the industry (Sundin & Bras 2005, P. 917). An overview of remanufacturing process sequences for products from different industries is, for example, given in the studies of Andrew-Munot & Ibrahim et al. (2015), Matsumoto & Umeda (2011) and Saavedra & Barquet et al. (2013) (Andrew-Munot & Ibrahim et al. 2015, P. 19, Matsumoto & Umeda 2011, P. 3 ff., Saavedra & Barquet et al. 2013, P. 271 ff.). According to Bras & Hammond et al. (1996), at a more general level, the four process categories (1) cleaning, (2) damage repair, (3) quality control, and (4) assembly and disassembly activities can be distinguished (Bras & Hammond et al. 1996, P. 5). Often, these general categories are further subdivided into sub-processes in the literature. Following the categorization of Nwankpa & Ijomah et al. (2021), an overview and description of the most relevant sub-processes is given below (Nwankpa & Ijomah et al. 2021, P. 2):

- **Identification:** Usually, one of the first process steps is to identify the type of product and year of production (Lund 1984, P. 4). This is done by manually comparing products against approved examples (Errington & Childe 2013, P. 7) or by reading out identification labels such as Barcodes, RFID tags, or QR-Codes (Kerin & Pham 2020, P. 1212, Schlüter & Lickert et al. 2021, P. 304). Identification of the present type of product offers significant advantages. It allows to quickly determine the most suitable place to carry out remanufacturing (Kerin & Pham 2020, P. 1213), for example, when the storage is not the actual facility of remanufacturing. Additionally, this enables optimized inventory management (Errington & Childe 2013, P. 13). Information regarding the estimated quantity of defective and non-defective components per batch and, related to this, the type and number of necessary new components to be supplied can be deduced.
- **Inspection:** Along the whole remanufacturing process, products, as well as subassemblies and components, are inspected to identify visible defects or wear (Lund 1984, P. 4). This is to ensure sorting out products, subassemblies, or non-remanufacturable components. In contrast to sampling-based methods in linear manufacturing, inspection in remanufacturing entails 100% inspection of all components during the process (Steinhilper 1998, P. 57) to ensure that remanufactured products meet the required specifications (Rickli & Dasgupta et al. 2014, P. 215). According to Matsumoto & Ijomah (2013), during the initial inspection of the whole product, these are grouped into three categories that are (1) reusable, (2) remanufacturable and (3) non-remanufacturable (Matsumoto & Ijomah 2013, P. 400). Other authors state that products suitable for remanufacturing may also be further split into quality classes to optimize process planning (cf. Aras & Boyaci et al. 2004 and Colledani & Battaïa 2016). Inspectors may use various non-destructive techniques and equipment (e.g., ultrasonic testing or X-Ray for internal

defects) to assess the condition of the product, its subassemblies, and its components (Tant & Mulholland et al. 2019, P. 61, Rickli & Dasgupta et al. 2014, P. 212).

- **Disassembly:** Once products are identified and inspected, they are disassembled into their subassemblies and components. This involves general-purpose tools (Andrew-Munot & Ibrahim et al. 2015, P. 18) such as power drills, but may also include destructive methods due to product design (Hjorth & Chrysostomou 2022, P. 9) and product condition (Gungor & Gupta 1998, P. 162). The varying condition of the products (Andrew-Munot & Ibrahim et al. 2015, P. 15), as well as the quality of the execution of the disassembly process (Ferrer 2001, P. 375) also influences the percentage of disassembly yield in a batch of products to be disassembled, as components may be damaged in the process.

- **Cleaning:** The components are thoroughly cleaned after disassembly. Cleaning includes de-greasing, de-oiling, de-rusting, and freeing the parts from old paint (Matsumoto & Ijomah 2013, P. 398). A variety of mechanical (e.g., sandblasting) or chemical (e.g., petrol cleaning) processes and tools may be employed for cleaning (Matsumoto & Ijomah 2013, P. 398). Based on their research, Hammond & Amezquita et al. (1998) found that cleaning is among the costliest processes of remanufacturing (Hammond & Amezquita et al. 1998, P. 10), which may be due to the extensive know-how needed (Matsumoto & Umeda 2011, P. 7) to select and configure suitable processes.

- **Component remanufacturing or replacing:** Component remanufacturing refers to all processes required to restore the specifications of a new product (Ijomah & Bennett et al. 1999, P. 194). Parts that are deemed unable to be remanufactured are replaced by spare parts (Matsumoto & Ijomah 2013, P. 401). According to Kin & Ong et al. (2014), the types of reprocessing processes that restore the condition of a defect-free component can be classified into the categories (1) removing surface and shape defects, (2) material addition or surface replacement, (3) restoring material properties, (4) intermediate assembly and fastening of subassemblies and (5) surface finishing (Kin & Ong et al. 2014, P. 190 f.). Often, even for the same product type, the type and sequence of needed processes vary based on the product condition (Andrew-Munot & Ibrahim et al. 2015, P. 18).

- **Reassembly:** This step involves reassembling remanufactured components, spare parts, and subassemblies to a remanufactured product. Often, the same tools used in disassembling the product are used to assemble the new product (Steinhilper 1998, P. 56).

- **Functional testing:** After or during assembly, each of the remanufactured products or their subassemblies are tested to verify their functionality. Functional tests can be mechanical or electrical (Ijomah & Bennett et al. 1999, P. 194). In contrast to inspection activities, functional testing is a quantitative method since the product or its subassemblies have to perform a specific function within specified tolerance limits (Bras & Hammond et al. 1996, P. 6). The methods for testing thereby used are usually the same or similar to those in manufacturing a new product of the same type (Haynsworth & Lyons 1987, P. 4).

2.1.2 Challenges for initial visual inspection

A challenging factor in visual inspection is the product variety (Tolio & Bernard et al. 2017, P. 5), primarily when differing product types (e.g., starter motors, generators, or turbines), their variants, and generations are jointly inspected and sorted. While in manufacturing, the ***Original Equipment Manufacturer*** (OEM) is mainly concerned with the production of one or a few product generations, remanufacturers have to cope with a range of product generations (Sundin 2004, P. 34). The problem of product diversity may be even more evident for independent remanufacturers who do not produce the new products themselves. They can choose to have products from different OEMs in their portfolio. In combination with the uncertain timing of the returning products, this causes heterogeneity of the inputs to the remanufacturing system, resulting in small batch sizes (Liu & Zhu et al. 2019, P. 4801) and hindering the adoption of automated process execution (Tolio & Bernard et al. 2017, P. 592). Considering initial inspection, where product identification has not been conducted, product information deficits represent a severe barrier to the automated execution of processes. Even if a product identification is conducted before the inspection of the used product, independent remanufacturers often do not have access to product information (Parker & Riley et al. 2015, P.1) such as ***Computer Aided Design*** (CAD) models (Khan & Mineo et al. 2021, P. 67). However, when considering the automated execution of initial inspection, such information can be relevant for planning the core inspection process to detect quality defects. This information deficit is, among other reasons explained below, why the majority of inspections in remanufactured are carried out visually by a human today (Steinhilper 1998, P. 50).

When performing an initial visual inspection of the returned product, an operator is assessing the reusability of the returned product. Based on his visual inspection and, if necessary, a haptic inspection of the product, acceptance or even quality grading of the product can be achieved. According to DIN 31051:2019-06 (2019) and DIN EN 13306:2018-02 (2018), inspection is defined as the testing of the conformity of the relevant characteristics of an object by measurement, observation, or functional testing (DIN 31051:2019-06 2019, P. 5, DIN EN 13306:2018-02 2018, P. 41). The initial visual inspection can thus be categorized as the testing of the conformity of relevant characteristics by observation of a human being. In a metrological sense, visual inspection, as performed along the process of remanufacturing, is a non-dimensional inspection method (Pfeifer & Schmitt 2010, P. 27). According to Pfeifer & Schmitt (2010), dimensional inspection methods use gauges or quantitative measuring equipment to obtain an objective measurement result (Pfeifer & Schmitt 2010, P. 27). In contrast, non-dimensional inspection methods are characterized by human sensory perception and provide a subjective inspection result.

The usage phase and handling of the product in the reverse supply chain before arriving at the remanufacturing site is unique for every used product. Thus, in contrast to linear production, it is impossible to accurately predict where product defects will occur (Kurilova-Palisaitiene & Sundin et al. 2018, P. 3229). Simultaneously, the possible occurring defects are manifold and vary from product to product (Robotis & Boyaci et al. 2012, P. 385) and are often difficult to quantify. The uncertainty in the condition of the returned products is, therefore, a major complexity driver in remanufacturing in comparison to linear manufacturing (Liu & Zhu et al. 2019, P. 4798). The uncertainty in the core condition is reflected in varying composition and extent of damage, wear, and contamination of the core. An overview of possible quality defects is given in Figure 2.1[1], in which a starter motor is used as an example to illustrate exemplary quality conditions of returning cores that must be identified and evaluated concerning reusability and quality grades of the core.

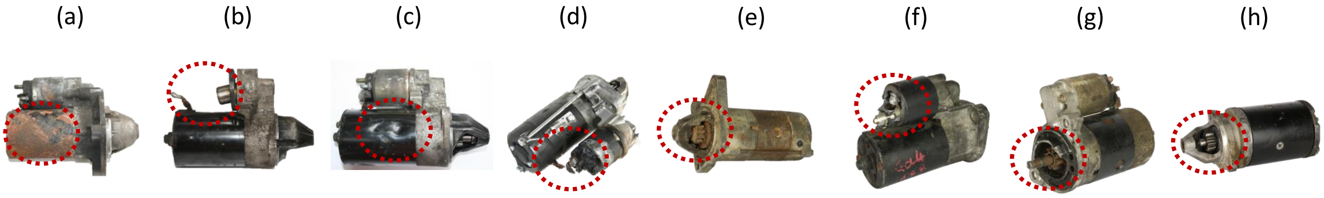


Figure 2.1: Illustration of exemplary quality conditions of returning cores that prevent reusability. (a) Light corrosion on the main frame. (b) Missing parts. (c) Damage on the main body of the core. (d) Burnt electrical connection. (e) Damaged teeth due to corrosion. (f) Damaged solenoid. (g) Broken housing. (h) Damaged shaft.

The nature and extent of the quality defects of the returning product can be defined concerning a (1) lack of predictability, (2) multidimensionality of the possible wear, (3) a lack of localizability, (4) difficult quantifiability and (5) varying scales of size.

The condition of the individual products before arrival at the remanufacturing plant cannot be predicted (1) and is characterized by a multitude of wear, contamination, and defects that may occur (2). Furthermore, these quality defects, such as corrosion, contamination, or defects

[1] Images are extracted of:
https://www.partinfo.co.uk/files/Remy%20Rotating%20Core%20Criteria%20MASTER%20QU-017.pdf
https://www.elstock.com/en/core-acceptance-criteria/starters/
accessed: 11.08.2023

including deformation, can appear anywhere on the product (3). In addition, degradation such as corrosion are difficult to quantify, as the question of how severe the extent also depends on the components on which it is occurring (4). Last, these defects and quality flaws appear on varying scales of size, and missing major components are easier to identify than, for example, broken gear teeth (5).

Remanufacturers often specify core acceptance criteria (e.g., CoremanNet (2022)) for used products to support initial visual inspection. The operator can use these to check whether the returned product is remanufacturable. The core acceptance criteria vary depending on the remanufacturer and product type. They provide information on permissible mechanical and electrical defects, acceptable corrosion, and the permitted degree of disassembly or missing components. However, the core acceptance criteria are qualitative, and the operators mostly decide subjectively (Schlüter & Lickert et al. 2021, P. 302). The missing formalization of the knowledge and expertise of operators inspecting returning products daily represents a significant complexity factor. Experiences from previous inspections and information aggregated through knowledge sharing with colleagues are implicitly used. Auxiliary resources for comparison, such as a reference list (Schlüter & Niebuhr et al. 2018, P. 384), may also be employed. However, direct quantification of the decisions is not possible, and thus, a formalization of the decision problem is also complicated, which results in difficulties for tasks to be automated.

2.1.3 Automated visual inspection

Manual visual inspection by a worker is prone to errors and can reach error rates of up to 30% (See & Drury et al. 2017, P. 262). In the last few decades, automated systems for visual inspection have therefore been increasingly researched (Newman & Jain 1995, P. 231). In the simplest case, such automated systems for visual inspection consist of a light source, an acquisition system, and a processing unit (Ren & Fang et al. 2022, P. 662). In the most straightforward case, the light source serves as a passive unit for optimal illumination of the inspection object to ensure that the incident light intensity on the photo sensor is optimally converted into high-contrast digital signals. In other cases, the generation of digital signals is only possible by actively controlling the light source, such as, for example, in structured light projection (Ren & Fang et al. 2022, P. 664). Newman & Jain (1995) distinguish between four different types of digital signals that are provided by an acquisition system and can be evaluated by the processing unit:

1. **Binary images:** Binary images in which each pixel is either black (0) or white (1). They are often used to check the presence of objects or represent an inspection object's contours.

2. **Gray level images:** Grayscale images use different brightness levels between black and white to display image information. Each pixel has a gray value that indicates the brightness of the pixel. For example, they can be used for texture inspection of surfaces if color information is not required.

3. **Color images:** Color images contain information about the color values in each pixel. They consist of three color channels (red, green, and blue) with which a large color palette can be displayed. They are used if, for example, color differences indicate defects.

4. **Depth images and point clouds:** Depth images or point clouds are 3D representations of scenes in which each point contains information about the distance or depth to the acquisition system. They are used if geometric information of the inspection object is required for inspection decision-making. Methods such as structured light projection obtain three-dimensional information by projecting light patterns onto a surface (Ren & Fang et al. 2022, P. 664). The acquisition system captures this pattern to obtain 3D surface structure and shape information.

The acquired data in digital form provides the starting point for further processing. An overview of processing methods in defect detection is provided by Ren & Fang et al. (2022). Early methods worked with simple image processing methods and template matching in the spatial domain or Fourier transformation in combination with wavelet transformation in the frequency domain. More recent methods use feature extraction in combination with machine learning up to deep learning using ***Neural Networks*** (NN).

2.1.4 Summary and conclusion: View planning tasks for the initial visual inspection

When a human performs the inspection, the core acceptance criteria provide guidance on the relevant quality criteria to which particular attention is paid during the inspection (e.g., CoremanNet (2022)). Due to the uncertainty regarding the quality state of the returning products derived in chapter 2.1.2, these are inspected to detect defects on the entire product. Furthermore, individual components or specific defects are inspected in detail. This is necessary, for example, if a particular defect occurs frequently on a component or if a defect is detected during the general inspection, but a more detailed inspection is required for the final acceptance decision. In conclusion, two types of inspection processes performed by the human can be distinguished. These can be seen in Figure 2.2.

First, to detect defects visible by general inspection, **(a) overall inspection of the returning product with full surface coverage** is necessary to detect all possible defects that may occur on the returning product. In addition, an **(b) individual inspection of components**

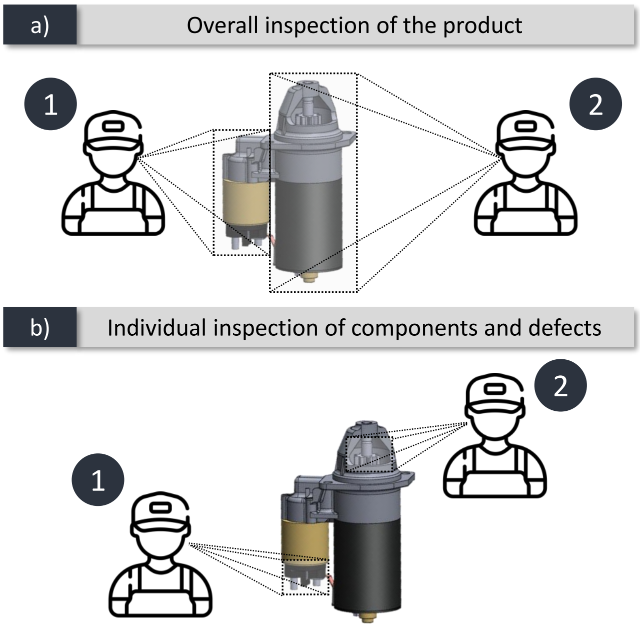

a)
Overall inspection of the product
1
2
b)
Individual inspection of components and defects
2
1

Figure 2.2: Overview of the two distinct inspection problems addressed in this thesis. a) Overall inspection of a product. b) Individual inspection of components and defects.

and defects, which represent ***Regions of Interest*** (ROI) that have been determined to be further examined by general inspection, may be necessary.

Although both subproblems a) and b) pursue two different objectives (general inspection and individual inspection of ROI), they can be formalized in the same way, which allows answering the first structuring question of section 1.3:

Problem formalization: How can the problem of autonomous data acquisition in remanufacturing be formalized?

The answer is based on the fact that both inspection processes describe a procedure where the worker implicitly solves the so-called ***view planning problem*** (VPP). The worker manually manipulates the used product to be inspected and changes the movements of his eyes and gaze so that the product can be inspected thoroughly (subproblem a)) or individual components or defects can be inspected in detail (subproblem b)).

2.2 View planning for visual inspection systems

View planning refers to the problem of selecting the optimal acquisition system poses for the targeted acquisition of a 3D scene or object (Peuzin-Jubert & Polette et al. 2021, P. 1). View planning is a subfield of robot active vision and is, therefore, a research area at the intersection of robotics and computer vision (Zeng & Wen et al. 2020, P. 225). This chapter introduces the basics of robot active vision and view planning in chapter 2.2.1 first. Different alternatives for three-dimensional object representation to solve the VPP are presented in chapter 2.2.2. Based on that, approaches to solving the VPP are introduced in chapter 2.2.3. The acquisition goal, performance metrics, and algorithmic requirements are then presented (section 2.2.4), followed by a section providing an overview of fundamentals of coordinate systems and pose transformations (section 2.2.5). The section concludes with a summary concerning view planning in remanufacturing (section 2.2.6), summarizing insights obtained from sections 2.1 and 2.2.

2.2.1 Robot active vision and the view planning problem (VPP)

Robot active vision refers to the concept of actively adjusting a robot's visual acquisition system to obtain information for various tasks (Zeng & Wen et al. 2020, P. 225). Use cases range from navigation, tracking, and mapping to site exploration and modeling up to object reconstruction and inspection (Chen & Li et al. 2011, P. 1344). Zeng & Wen et al. (2020) mention robots and visual acquisition systems as core components for an active robot vision application. The authors further distinguish between robotic arms and mobile robots that can freely explore their environment (Zeng & Wen et al. 2020, P. 226). Stationary robots, thereby, often have an acquisition system attached to their end-effector. The joints then determine the degrees of freedom of the acquisition system system. In contrast to stationary robot arms, mobile robots can acquire scenes or larger objects. Due to occlusions and limited working distance and field of view of visual acquisition systems, more than one view of an acquisition system is often required. Thus, for a variety of vision tasks, planning multiple views is necessary to achieve the overall goal (Chen & Li et al. 2011, P. 1343).

Definition of the VPP

Scott & Roth et al. (2003) define the VPP as the problem of finding the shortest view plan consisting of a minimal number of views, which fulfills a specified reconstruction target within an acceptable computation time for a given environment and object. To formalize the VPP, Scott & Roth et al. (2003) introduce the imaging environment. This includes the object itself, an acquisition system to capture the object's shape, a system to fixate the object, and a positioning system to move the acquisition system or the object relative to each other. In addition, three sets are introduced (Scott & Roth et al. 2003, P. 65ff). These are the object surface space S, the imaging workspace I, and the viewpoint space V. The object surface space S is the set of 3D surface points sampled on the object. The imaging workspace is the 3D region capable of being viewed by the acquisition system over the full range of the positioning system. Finally, the viewpoint space is the set of generalized viewpoints, which was first introduced in Tarabanis & Tsai et al. (1995, P. 73) and adopted in the definition of Scott & Roth et al. (2003). A generalized ***Viewpoint*** (VP) V^S of a system consists of the pose $\mathbf{p}^{A/Ref}$

of an acquisition system A in relation to a reference Ref and a set of controllable acquisition system parameters λ_A. This results in the tuple $V^S = (\mathbf{p}^{A/Ref}, \lambda_A)$. The acquisition systems pose $\mathbf{p}^{A/Ref}$ can be expressed in six-dimensional space by its position and orientation relative to its reference Ref.

In this thesis, the above definition is extended further. The definition of Scott & Roth et al. (2003) already mentions a positioning system that can reposition the object O between successive views. Consequently, in addition to the acquisition system poses, the object poses $\mathbf{p}^{O/Ref}$ must also be included in the VP space V since it may change during the acquisition cycle. A VP V^S of a system then consists of $\mathbf{p}^{O/Ref}$, $\mathbf{p}^{A/Ref}$ and λ_A ($V^S = (\mathbf{p}^{O/Ref}, \mathbf{p}^{A/Ref}, \lambda_A)$). Furthermore, some acquisition systems move on trajectories to acquire measurement data (Peuzin-Jubert & Polette et al. 2021, P. 1). In this case, the definition of a VP over a single pose of the acquisition system is insufficient. Instead, the trajectory must be represented by a list of acquisition system poses. In this case, a trajectory to be followed by the acquisition system is represented by a list of multiple individual poses of the acquisition system and the object and the acquisition system parameters. With the introduced extensions, the overall goal remains, as stated earlier, to find a suitable short view plan (Scott 2009, P. 48) to achieve the task at hand.

Subcategories of VPP

According to Scott & Roth et al. (2003), finding a solution of the VPP involves reasoning about the surface space S of the object, as well as the imaging workspace I and the space V of all possible VPs. However, Scott & Roth et al. (2003) describes only reconstruction as the overall goal of view planning (Scott & Roth et al. 2003, P. 68). The planning of a single VP, i.e., the ***Next-Best-View*** (NBV) to obtain information about an unknown scene, can also be considered a VPP. Also, inspection tasks with the overall goal of only acquiring specific object surfaces to be inspected (ROI) can be formalized as VPP. An extension of the definition is consequently given in Scott (2009), which also explicitly addresses the solution of an inspection task as the goal of VPP (Scott 2009, P. 48). For this reason, three subcategories of VPP based on their respective planning objective are distinguished in this thesis:

1. **NBV:** Banta & Zhien et al. (1995) define the ***Next-Best-View*** (NBV) as the next VP that will extract the greatest amount of unknown scene information. One of the earliest mentions of the term "next-best-view" is from Connolly (1985), who focuses on three-dimensional scene modeling (Banta & Zhien et al. 1995, P. 418, Connolly 1985, P. 432).

2. **RPP:** In the context of this thesis, the ***Reconstruction Planning Problem*** (RPP) refers to the problem of finding a minimum number of VP so that the entire surface of the inspection object can be acquired and thus reconstructed.

3. **IPP:** The ***Inspection Planning Problem*** (IPP) is equivalent to the RPP within the context of this thesis, but only a partial area of the inspection object (the ROI) must be acquired. Consequently, in contrast to RPP, a minimum number of VP must be found so that the ROI of the inspection object are covered.

An essential difference to NBV is that a view plan, which represents a solution of RPP, does not only contain VP that represent the next best VP concerning the previous VP. It may well be that, as shown in Figure 2.3, the choice of NBV alters the problem in such a way that the optimal solution of VPP cannot be found. In this Figure, the optimal view plan consists of the acquisition indicated by numbers one, two, and three. By choosing the acquisition "NBV", which represents the NBV after performing acquisition one, still, two acquisitions would be needed to fully cover the surface of the triangular object. This view plan would consist of four (in contrast to three for the optimal view plan) acquisitions. This proves that a view plan consisting of a sequence of NBV does not always correspond to the optimal view plan regarding the required number of acquisitions, and the consecutive solution of the NBV problem is not equal to the solution of the RPP.

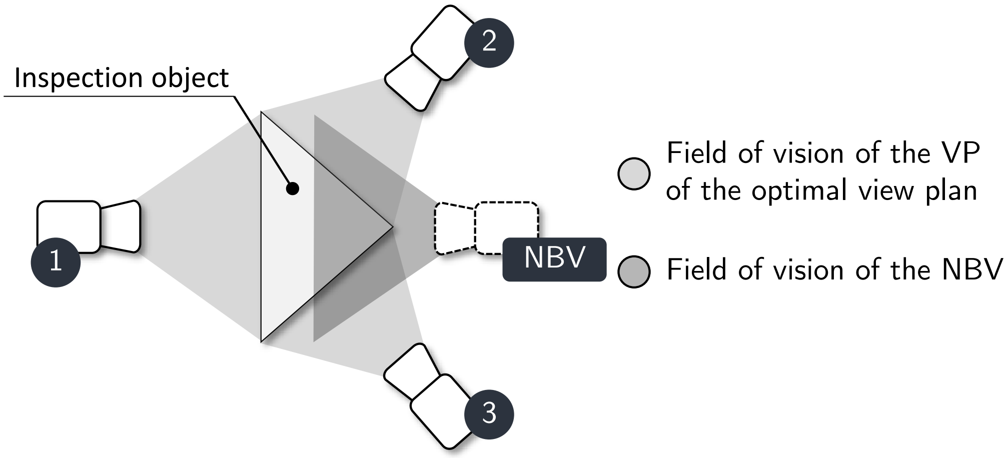

Inspection object
2
1
NBV
3
Field of vision of the VP of the optimal view plan
Field of vision of the NBV

Figure 2.3: Visualization of the choice of the next VP on the performance of the solution to the VPP.

Acquisition system selection and solution complexity

Depending on the application, different acquisition systems can be used to acquire the object's surface. Peuzin-Jubert & Polette et al. (2021) mention laser scanners and structured light systems, especially for the use case of object reconstruction (Peuzin-Jubert & Polette et al. 2021, P. 3 f.). According to Zeng & Wen et al. (2020), additional acquisition system principles such as stereo vision and time of flight can be used (Zeng & Wen et al. 2020, P. 226 f.). The selection of an acquisition system is always closely coupled with the application and the associated requirements. Zeng & Wen et al. (2020) propose criteria such as acquisition distance, acquisition accuracy, depth image, and color image resolution as well as costs and robustness against environmental influences (e.g., changing light conditions) to compare different types of acquisition systems (Zeng & Wen et al. 2020, P. 227).

The complexity of finding a solution of the VPP based on the system used differs. Generally, the solution of solving the VPP when considering trajectories of the acquisition system is more complex. Using the extended definition, the solution may additionally include an altered pose of the object relative to the previous acquisition, regardless of the acquisition system used. In the past, it has been shown by Tarbox & Gottschlich (1995) that the view planning problem is NP-complete (Tarbox & Gottschlich 1995, P. 90). Therefore, finding a globally optimal solution is usually not feasible. Instead, approximate solutions that can be found with acceptable time and computational effort are often desired (Zeng & Wen et al. 2020, P. 228).

Reconstruction cycle

Planning the optimal view plan in advance is only possible in a model-based setting if a model of the object of interest is available. Scott & Roth et al. (2003) therefore distinguishes problems where an object model is available (model-based) and those for which this is not the case (model-free) (Scott & Roth et al. 2003, P. 65). Given the case where no model of the object is available, the next VP must be calculated based on the current incomplete model available after each step. The solution of the VPP is then obtained by sequentially executing the planning cycle, while the next VP is calculated iteratively in each iteration.

Scott & Roth et al. (2003) defines the classical acquisition cycle and divides its execution into four steps (Scott & Roth et al. 2003, P. 67). These are (1) planning, (2) scanning, (3) registration, and (4) integration. Given the case where a view plan as the solution of the VPP is computed in advance, the sequence of VP is available before execution. These individual VP must be approached sequentially to perform the scans. After each scan, the current scan must be registered to all previous scans to compensate for errors in the positioning system. Then, all previous scans are integrated into a single, non-redundant model until a termination criterion is met.

2.2.2 Object representation alternatives for the solution of the VPP

Both for model-free methods, which compute the NBV, and for model-based solution methods, which directly calculate the solution of the VPP, a suitable representation of the object in the form of an object model is necessary. In a recent review article, Peuzin-Jubert & Polette et al. (2021) analyzed VPP solution methods and the different forms of representation of the object model commonly used. In the following, the most important ones in the scope of this thesis are introduced and explained in detail, using recent review articles of Gezawa & Zhang et al. (2020), Ahmed & Saint et al. (2018), and Fahim & Amin et al. (2021) as references. Other representation alternatives are discussed briefly. A rough distinction of 3D representations is made by Ahmed & Saint et al. (2018) and Bronstein & Bruna et al. (2017) into Euclidean and non-Euclidean structured representations. Fahim & Amin et al. (2021) follow this classification and discuss representation forms that do not fall into the first two categories. The categorization and exemplary representation alternatives for objects can be found in Figure 2.4, in accordance to Fahim & Amin et al. (2021) and Peuzin-Jubert & Polette et al. (2021), and are explained in the following.

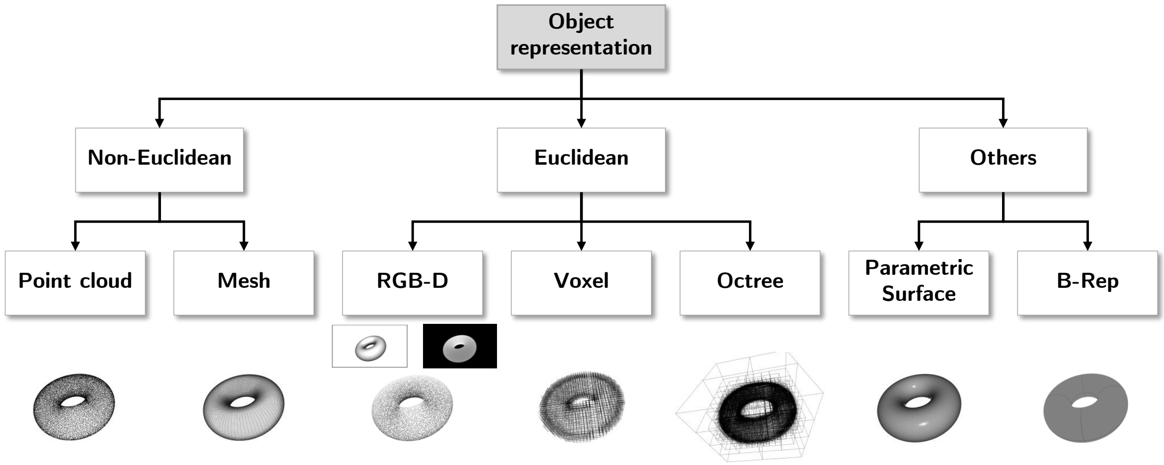


Figure 2.4: Categorization of exemplary representation alternatives of object models relevant to VPP in accordance to Fahim & Amin et al. (2021) and Peuzin-Jubert & Polette et al. (2021)

Point cloud representation

The primary type of non-Euclidean data is point clouds (Ahmed & Saint et al. 2018, P. 5). Point clouds are raw data that can be collected by most existing depth acquisition systems today (Gezawa & Zhang et al. 2020, P. 57567). A Point cloud captures the three-dimensional object as a set of unstructured individual data points and allows it to be represented as a set of 3D points in space (Peuzin-Jubert & Polette et al. 2021, P. 3, Gezawa & Zhang et al. 2020, P. 57567). Each point represents a specific spatial location on the object's surface (i.e., x, y, and z coordinates) (Zeng & Wen et al. 2020, P. 228). Point clouds are acquired through laser scanning, structured light projection, and many other three-dimensional measurement principles (Gezawa & Zhang et al. 2020, P. 57567), making them useful for use cases such as 3D reconstruction, object recognition, and vehicle detection (Ahmed & Saint et al. 2018, P. 6). Point clouds are simple and memory-efficient. However, processing them can be challenging since they lack connectivity information (Ahmed & Saint et al. 2018, P. 5 f., Gezawa & Zhang et al. 2020, P. 57568, Fahim & Amin et al. 2021, P. 170). Additionally, they may require further processing due to incompleteness, noise, or missing data after the acquisition process (Gezawa & Zhang et al. 2020, P. 57568). Lastly, they are not able to describe unknown or empty spaces (Zeng & Wen et al. 2020, P. 228).

Mesh representation

A mesh can represent an object's outer surface (Peuzin-Jubert & Polette et al. 2021, P. 3). A mesh is composed of vertices, edges, and faces (Gezawa & Zhang et al. 2020, P. 57568) to form a polynomial network. Vertices are the three-dimensional points in space that define points on the object's surface. A point cloud can, therefore, serve as a starting point to create an object mesh with suitable algorithms. Each vertice contains a connectivity list that describes how edges connect the vertices (Gezawa & Zhang et al. 2020, P. 57568). Edges represent the straight lines between pairs of vertices. They contribute to the mesh's overall shape by defining the faces' boundaries. Faces are formed by connecting three (triangle) or more vertices with edges. All faces of the mesh define the structure of the 3D object and its surface appearance. Meshes are versatile and can approximate complex shapes easily. However, the level of detail in a mesh is determined by the number of vertices, making it memory and computationally intensive to represent volumetric data or smooth surfaces.

RGB-D representation

RGB-D data are a representation form of a 3D object by 2.5D data by providing ***Depth*** (D) as well as color information in ***Red, Green and Blue*** (RGB) domain (Gezawa & Zhang et al. 2020, P. 57568) in the form of images (Ioannidou & Chatzilari et al. 2017, P. 1). Often, RGB-D data are directly output by an acquisition system, allowing for the representation of the object surface with the said depth map while additionally providing color information (Zeng & Wen et al. 2020, P. 226, 229). Spatial information is contained in a grid-like manner by the image, which is why RGB-D data are categorized by Ahmed & Saint et al. (2018) as Euclidean data (Ahmed & Saint et al. 2018, P. 4). With suitable calibration, direct relationships between the acquired color image data and the depth image can be established with such systems (Lacher & Vasconcelos et al. 2019, P. 13). The result is a color-coded point cloud. Each pixel of the image can then be assigned a spatial coordinate. This offers the advantage of applying highly performant image-based machine learning techniques to the image data that has been acquired and fusing the results with the three-dimensional acquisition results as done by Martín & González et al. (2021)(Martín & González et al. 2021, P. 1, 2).

Volumetric representation

Voxel grids and octrees are Euclidean forms of three-dimensional volumetric data representation of objects (Ahmed & Saint et al. 2018, P. 4, 5). Voxel grids are used to represent an object in space by a three-dimensional regular grid dividing the object's volume into small cubic elements called voxels (Peuzin-Jubert & Polette et al. 2021, P. 3). In its simplest form, each voxel can be encoded binary, whether it is inside the object or outside (Fahim & Amin et al. 2021, P. 165). For view planning, voxels that belong to the object can, for example, further be classified whether they are visible (on the surface of the object), not visible (inside the object), occluded or self-occluded (Gezawa & Zhang et al. 2020, P. 57568). However, voxel grids can be memory and computationally expensive, especially for high-resolution object representation (Fahim & Amin et al. 2021, P. 169, Ahmed & Saint et al. 2018, P. 4). The drawback of memory and computational intensity is improved with octrees (Fahim & Amin et al. 2021, P. 169). An octree is a voxel grid with voxels of varying sizes, which is built by recursively subdividing root voxels (Ahmed & Saint et al. 2018, P. 4) until a termination criterion is met. Despite advances in memory and computational efficiency, voxel grids, as well as octrees, are still considered to have limitations concerning the preservation of the geometry of the object regarding the shape and smoothness of the surface (Ahmed & Saint et al. 2018, P. 5, Gezawa & Zhang et al. 2020, P. 57568, Fahim & Amin et al. 2021, P. 169).

Other representation alternatives
According to Peuzin-Jubert & Polette et al. (2021), parametric surfaces and ***Boundary Representation*** (B-Rep) are further variants for 3D object representation, relevant for VPP. Parametric surfaces such as ***Non-Uniform Rational B-Spline*** (NURBS) represent complex shapes using control points, knot sequences, and weights. Parametric surfaces are beneficial for generating smooth surfaces. However, defining complex surfaces with parametric equations can be challenging and computationally expensive. B-Rep are used to describe solids such as CAD models and are therefore widely used in the industrial domain. An object represented with B-Rep comprises several elements that form the object's skin. Faces of said model can be represented using analytical descriptions of a surface, such as NURBS or B-Splines. These faces are connected by wires composed of edges and vertices. (Peuzin-Jubert & Polette et al. 2021, P. 3)

2.2.3 Analytical solution methods for VPP

According to Scott & Roth et al. (2003), methods for the solution of the VPP can be divided into model-based and model-free approaches (Scott & Roth et al. 2003, P. 65). Model-based methods are based on a priori knowledge of the object's geometry using a 3D object representation. This enables the solution of the VPP by planning a suitable view plan even before execution. When no a priori knowledge is available, e.g., in reverse engineering, model-free methods that utilize an iterative approach are used (Peuzin-Jubert & Polette et al. 2021, P. 2). In each iteration of the planning cycle introduced in chapter 2.2.1, the next optimal VP is determined until a termination criterion is met.

Model-based solution approaches

An early categorization of model-based algorithms for solving the VPP exists by Scott & Roth et al. (2003). The authors distinguish between solution algorithms from the domains of Set Theory, Graph Theory, and Computational Geometry (Scott & Roth et al. 2003, P. 74). A later definition by Peuzin-Jubert & Polette et al. (2021) generalizes this classification. According to Peuzin-Jubert & Polette et al. (2021), the algorithms designed to solve the model-based VPP can be divided into only two groups. The first group is just like in the categorization of Scott & Roth et al. (2003) from the domain of set theory, which transforms the VPP into a so-called ***Set Covering Problem*** (SCP) and solves it using optimization algorithms. Solving the view planning problem (VPP) using approaches to solve SCP involves discretizing and evaluating multiple poses of the acquisition system and then sequentially selecting the smallest possible set of poses to obtain a view plan. The number of poses of the detection system must be selected so that the coverage of the object surface to be detected is maximized and the VPP is solved accordingly.

The second group combines the categories of graph theory and computational geometry of Scott & Roth et al. (2003), resulting in the solution approach via ***Coverage Maximization*** (CM). The authors generally define these methods as approaches that do not solve the VPP via an optimization algorithm but directly pursue the goal of maximizing the surface coverage. Using CM methods, the surface of the inspection object is first divided into sections. Acquisition poses that aim to maximize coverage of the surface clusters are then determined. An exemplary grouping into sections can take place with the help of spatial proximity and orientation of sampled points located on the surface of the inspection object. The number of acquisitions required to achieve that goal is only considered a secondary objective, in contrast to the solution of SCP.

Regardless of the approach used, the view plan can be generated offline, while in the use phase, it only needs to be executed. An adapted version of the general workflow of the acquisition cycle introduced by Scott & Roth et al. (2003) for model-based solution approaches can be seen in Figure 2.5.

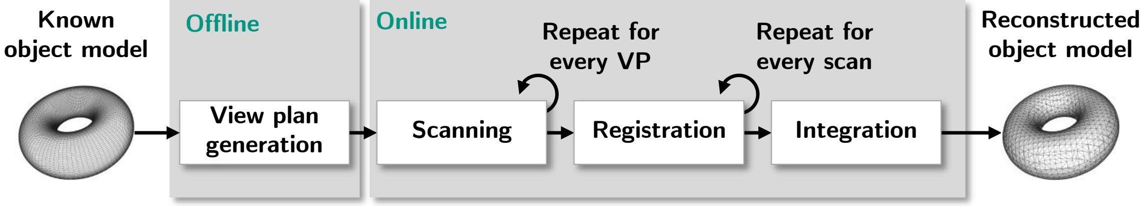


Figure 2.5: Adapted acquisition cycle for model-based solution approaches in accordance to Scott & Roth et al. (2003).

Model-free solution approaches

In contrast to model-based solution approaches, model-free approaches cannot rely on prior knowledge about the object's geometry. Instead, these methods attempt to iteratively determine the next VP with the highest information gain, the NBV, at system runtime after an acquisition. In contrast to model-based solution approaches, no offline planning takes place. The adapted version of the acquisition cycle introduced by Scott & Roth et al. (2003) for model-free solution approaches can be seen in Figure 2.6.

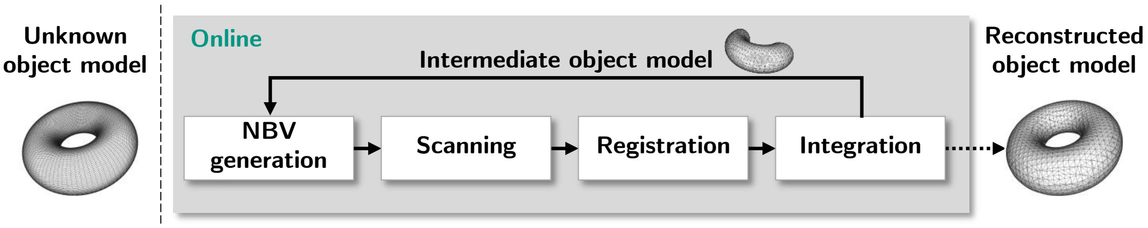


Figure 2.6: Adapted acquisition cycle for model-free solution approaches in accordance to Scott & Roth et al. (2003).

Peuzin-Jubert & Polette et al. (2021) define the procedure as follows. At the beginning of the procedure, an initial acquisition is performed with a VP. After the initial acquisition, the acquired surface sections are deduced, and the remaining unknown surface sections are estimated. Based on the estimated remaining surface sections and their spatial localization, new VPs are generated. Each of these VP gets assigned a subset of the estimated remaining unknown surface sections that can be acquired with said VP. The NBV is then chosen as that VP that maximizes the coverage of the estimated remaining surface sections.

Summary of analytical solution methods to the VPP

Please refer to Appendix A1 for a more detailed mathematical formulation of the analytical approaches presented for solving VPP. Figure 2.7 shows a schematic overview of the three solution approaches using SCP (subplot a) and CM (subplot), both of which are model-based, as well as the model-free approach for determining the NBV (subplot c).

2.2.4 Algorithmic requirements, acquisition goal, and evaluation metrics

Depending on the application, a solution method for the VPP must fulfill different acquisition goals and requirements. The review articles of Scott & Roth et al. (2003), Scott (2009), and Peuzin-Jubert & Polette et al. (2021) give an overview of existing performance characteristics and general algorithmic requirements. These are outlined and summarized in the following.

Regarding the general algorithmic requirements, an algorithm solving the VPP must be independent of the object or acquisition system. This implies that the VP of the solution

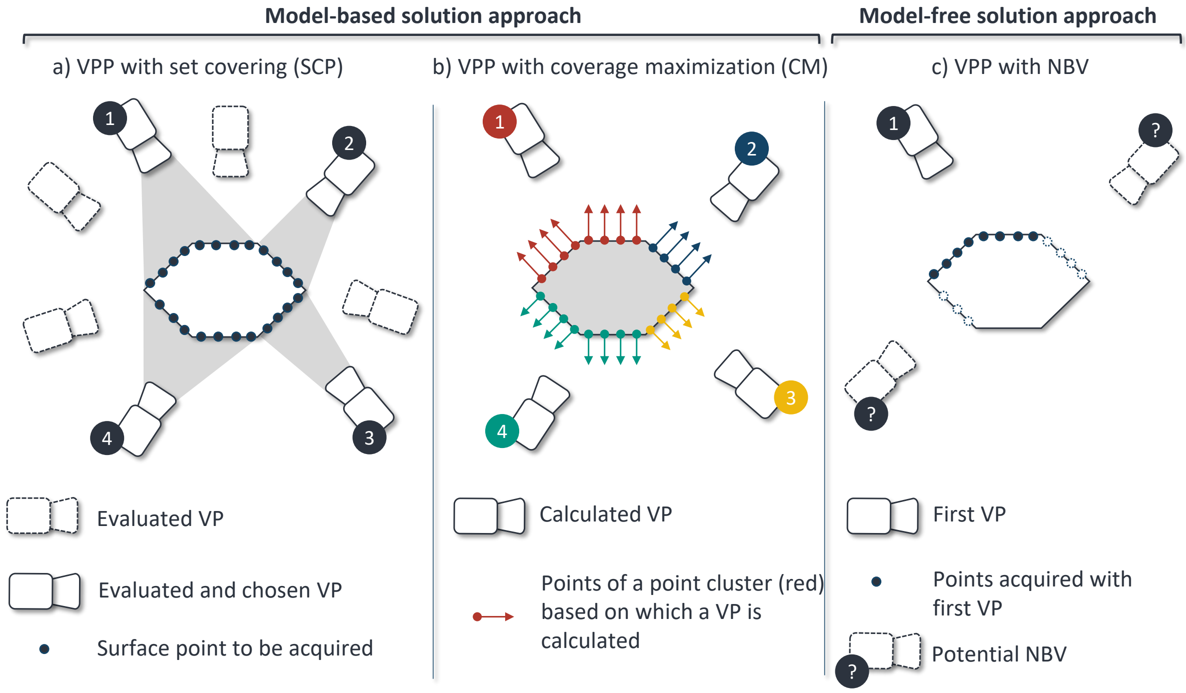

Model-based solution approach
Model-free solution approach
a) VPP with set covering (SCP)
b) VPP with coverage maximization (CM)
c) VPP with NBV
1
2
3
4
?
Evaluated VP
Evaluated and chosen VP
Surface point to be acquired
Calculated VP
Points of a point cluster (red) based on which a VP is calculated
First VP
Points acquired with first VP
Potential NBV

Figure 2.7: Schematic overview of VPP solution approaches using SCP (subplot a), using CM (subplot b) and calculating the NBV (subplot c).

algorithm are generalized and defined in such a way that they can be used independently of a present acquisition system or measurement principle (Scott & Roth et al. 2003, P. 71, Peuzin-Jubert & Polette et al. 2021, P. 2 f.). Ideally, the definition of VP then follows the definition presented in chapter 2.2.1 and only the parameter vector λ_A is adapted to the acquisition system at hand (Scott & Roth et al. 2003, P. 71). Consequently, the object pose $\mathbf{p}^{O/Ref}$ and the acquisition system pose $\mathbf{p}^{A/Ref}$ are always defined the same regardless of the acquisition system and object used. Additionally, algorithms should be generalizable, i.e., widely applicable (Scott & Roth et al. 2003, P. 71). Peuzin-Jubert & Polette et al. (2021) further detail this aspect. The authors state that an algorithm must take into account a variety of constraints, such as diverse objects in size and shape, constraints on VP generation due to the design of the imaging environment consisting of the acquisition system, positioning system, and object as well as acquisition goals (Peuzin-Jubert & Polette et al. 2021, P. 2 f.).

Depending on the application, different evaluation metrics exist. A solution algorithm must align its solution of the VPP based on these evaluation metrics. Based on the definition of VPP, the main goal is first to maximize the surface area of interest (ROI) (Scott 2009, P. 48). According to the introduced nomenclature of this thesis, this can be the entire object surface (NBV and RPP) or one or more specific parts of the object surface (IPP) for inspection tasks. Aspects that may also play a key role are the precision of the individual point clouds acquired in each acquisition as well as the required density (Scott & Roth et al. 2003, P. 49). For reconstruction tasks, the execution time of the vision plan is of minor importance, especially for industrial applications, but the execution time of the vision plan has to be considered. This is associated with the requirements for the generation of a view plan, which minimizes the number of necessary views (Scott & Roth et al. 2003, P. 72, Peuzin-Jubert & Polette et al. 2021, P. 3). Furthermore, especially for model-free solution methods, the computation time for calculating the NBV is an important performance characteristic. Only in this way can the dead time between individual views be kept small and the inspection time be minimized (Peuzin-Jubert & Polette et al. 2021, P. 3). Within this scope, an important aspect that will play a significant role in future research, both for model-based and model-free solution approaches, is solving the coupled problem of VPP and minimizing the travel distances between VPP to reduce the cycle time even more (Jing & Polden et al. 2017, P. 1294). This can be done by, for example, solving the extended SCP, where the costs of each VP correspond to the travel costs between the current VP and the VP to be evaluated.

2.2.5 Fundamentals of coordinate systems and pose transformations

In the following, the basics of coordinate systems and pose transformations are briefly explained using the example of a robot-guided acquisition system. For more in-depth explanations, please refer to the available fundamental literature, such as Mareczek (2020, P. 54 ff.).

To move the acquisition system to the VPs determined by an algorithm that provides a solution for the VPP, geometric descriptions of the system are necessary. This is achieved by introducing coordinate systems and describing their geometric relationships. An example is an acquisition system with a coordinate system C_A, which is mounted on a robot with a static base coordinate system C_R. The pose $\mathbf{p}^{A/R}$ of the acquisition system, as a component of one VP in the overall VPP solution, corresponds to the translation and orientation of the coordinate system C_A with respect to C_R.

The geometrical relationships between C_A and C_R can then be established using homogeneous coordinates. The pose of a coordinate system is represented in homogeneous coordinates by a 4×4 transformation matrix, including rotational and translational components. In the example of an acquisition system mounted on a robot, the 3×3 rotation matrix $R^{A/R}$ and 3×1 translation vector $T^{A/R}$ represent the acquisition system's orientation and translation of the acquisition systems coordinate system C_A relative to the robot base coordinate system C_R.

The homogeneous transformation matrix $\mathbf{T}^{A/R}$ is formulated as:

$$\mathbf{T}^{A/R} = \begin{bmatrix} R^{A/R} & T^{A/R} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{2.1}$$

Usually, translational components in Cartesian space and Euler angles to represent the orientation in space are used to represent poses, e.g., $\mathbf{p}^{A/R}$. In this case, the translational components correspond directly to the translation vector $T^{A/R}$, whereby the rotation matrix $R^{A/R}$ can be calculated using the Euler angles. An exemplary rotation matrix calculation based on given Euler angles can be found in Appendix A2. In some cases, the translational components of a pose are also specified in non-Cartesian coordinate systems (e.g., Spherical coordinates). In this case, it is also necessary to convert the translational components of the pose $\mathbf{p}^{A/R}$ from the non-Cartesian coordinate system to a Cartesian representation to calculate the translation vector $T^{A/R}$. A mathematical example of this can also be found in Appendix A2. The description of the geometric relationships of the system then allows, for example, the calculation of the inverse kinematics, i.e., the robot's joint angles required to move the acquisition system to a particular pose $\mathbf{p}^{A/R}$.

In addition, the description of the geometric relationships also enables the conversion of points represented in C_A into the representation in the coordinate system C_R. The point $\mathbf{P}^A$ in C_A, is initially represented by coordinates X^A, Y^A and Z^A. To transform $\mathbf{P}^A$ to C_R, the homogeneous transformation matrix $\mathbf{T}^{A/R}$ is used as follows:

$$\begin{bmatrix} \mathbf{P}^R \\ 1 \end{bmatrix} = \mathbf{T}^{A/R} \cdot \begin{bmatrix} \mathbf{P}^A \\ 1 \end{bmatrix} = \begin{bmatrix} R^{A/R} & T^{A/R} \\ \mathbf{0}^T & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{P}^A \\ 1 \end{bmatrix} = \begin{bmatrix} R^{A/R}\,\mathbf{P}^A + T^{A/R} \\ 1 \end{bmatrix} \tag{2.2}$$

The result of this calculation is the point $\mathbf{P}^R = [X^R, Y^R, Z^R]$, represented in C_R. For the calculation, the vectors of the points $\mathbf{P}^R$ and $\mathbf{P}^A$ are each extended by one dimension with the value 1 to enable multiplication with the 4×4 homogeneous transformation matrix $\mathbf{T}^{A/R}$ and to ensure the point $\mathbf{P}^A$ is rotated by $R^{A/R}$ and translated by $T^{A/R}$ to be correctly represented in C_R.

2.2.6 Summary and conclusion: View planning in Remanufacturing

The solution of a VPP consists of finding a minimum number of VP that denote poses or trajectories $\mathbf{p}^{A/Ref}$ of the acquisition system and, if necessary, $\mathbf{p}^{O/Ref}$ of the object in order to fulfill the planning goal at hand. In this section, the three sub-problems of the VPP, namely the NBV problem, the RPP and the IPP have been defined. The problem formulation of autonomous data acquisition in remanufacturing, from which the subproblems of general inspection and individual inspection of ROI resulted, can be mapped directly to the subproblems of VPP. This makes it possible to answer the structuring question raised in section 1.3:

Solution approach: Which group of solution methods exists for solving the formalized data acquisition problem in remanufacturing?

The problem of general inspection (subproblem a)) can thereby be directly mapped to the problem formalization of RPP, while the problem of individual inspection of ROI (subproblem b)) can be mapped to the problem formalization of IPP. Both problem types (RPP and IPP) can be solved model-based or model-free depending on the availability of forms of object representation of the inspection object.

Regardless of the solution approach (model-based or model-free), both approaches require the solution of an optimization problem. The goal is to optimize the target variables (e.g., maximizing the surface coverage while minimizing the required number of VP). Machine learning methods as a subfield of artificial intelligence have proven successful in solving optimization problems, including the VPP (Peuzin-Jubert & Polette et al. 2021, P. 19). Machine learning methods are capable of discovering patterns within data. This ability enables machine learning models to generate effective solutions to the VPP that traditional algorithms (such as, e.g., analytical solution methods) may not be able to address efficiently due to their high complexity (Peuzin-Jubert & Polette et al. 2021, P. 19).

2.3 Machine learning and deep learning

The following chapter first gives a general overview of the three categories of machine learning (section 2.3.1). Based on this, neural networks as universal function approximators are discussed (section 2.3.2). Section 2.3.3 overviews advanced methods based on deep learning, such as segmentation, semantic mapping, learning using point clouds, and actor-critic reinforcement learning. The chapter closes with a summary on the application of machine learning in view planning (section 2.3.4).

2.3.1 Overview of machine learning categories

Three types of learning can be distinguished based on their type of feedback (Russell & Norvig 2010, P. 694). These are ***Supervised Learning*** (SL), ***Unsupervised Learning*** (USL) and ***Reinforcement Learning*** (RL).

According to Russell & Norvig (2010), the goal of supervised learning is to approximate a function $f : X \rightarrow Y$, where X is the input space and Y is the output space, given a set of N

training examples $(x_{f,1}, y_{f,1}), ..., (x_{f,N}, y_{f,N})$ (Russell & Norvig 2010, P. 695). A model trained by SL thereby predicts $\hat{y}_{f,i}$ given an input $x_{f,i}$, where, ideally, the prediction $\hat{y}_{f,i}$ equals the true label $y_{f,i}$ ($\hat{y}_{f,i} = y_{f,i}$). The approximation of the function $f(\cdot)$ that generated the training samples is called the hypothesis and is denoted by $h(\cdot)$. In this case, learning is concerned with searching through the space of all candidate hypothesis functions to find the one that best suits the unknown function $f(\cdot)$. The accuracy of a given hypothesis is measured by its accuracy of predicting $y_{f,i}$ for a given $x_{f,i}$, which was not included in the training examples (Russell & Norvig 2010, P. 696). A distinction between SL is made between classification, where the output is one or more discrete classes, and regression, where the output is one or more continuous variables (Bishop & Nasrabadi 2006, P. 3).

Unsupervised learning is concerned with reasoning about the input space without a corresponding output space (Bishop & Nasrabadi 2006, P. 3). Based on an input space X and a set of N training examples $(x_{f,1}, ..., x_{f,N})$, the goal is to extract knowledge of the properties of the probabilistic function $f(\cdot)$ that generated those training examples (Hastie & Tibshirani et al. 2009, P. 486). The learning is concerned with finding a probabilistic hypothesis function $h(x_{f,N+1}|x_{f,1}, ...x_{f,N})$ which models the probability distribution of a new input $x_{f,N+1}$ given the training examples (Ghahramani 2003, P. 74). The determination of the probabilistic distribution of data within the input space is known as density estimation and can be used for dimensionality reduction (Bishop & Nasrabadi 2006, P. 3). Another approach is to find patterns inside said distribution, which is called clustering (Ghahramani 2003, P. 73).

In contrast to SL and USL, reinforcement learning is concerned with training an agent to solve a sequential decision-making problem. The agent interacts with an environment through its strategy π, which determines actions a_t given a state s_t of the environment at the time t. The strategy π consequently is a function $\pi : S_E \rightarrow A_{RL}$ that maps the space of states S_E of the environment to the space of actions A_{RL} of the RL agent (Sutton & Barto 2018, P. 58). After interacting with the environment, the agent receives a reward r_{t+1} and the new state s_{t+1}. The agent must then find an optimal strategy π^* that maximizes the sum of all discounted future rewards G_t (see equation 2.3), also called long-term reward, through continuous interaction with its environment. The discount factor $\gamma_{RL} \in [0, 1]$ represents a parameter that indicates the farsightedness of the strategy learned by the agent. By choosing $\gamma_{RL} = 0$, the agent only learns a policy π that chooses actions that lead to an immediate reward. When setting $\gamma_{RL} = 1$, future rewards are not discounted, and the agent learns a farsighted strategy. (Sutton & Barto 2018). RL methods thereby specify how the policy π is changed as a result of this interaction to best approximate π^* (Sutton & Barto 2018, P. 58).

$$G_t = r_{t+1} + \gamma_{RL} r_{t+1} + \gamma_{RL}^2 r_{t+2} + ... = \sum_{k=0}^{\infty} \gamma_{RL}^k r_{t+k+1} \qquad 2.3$$

According to Zhang & Yu (2020), model-based and model-free approaches can be distinguished. Although the designation of model-free and model-based RL is synonymous with the designation in VPP, they differ concerning the type of model used. Model-based approaches aim to predict the new state s_{+1} and the reward r_{t+1} given a state s_t and action a_t, i.e., learning the environment model while interacting with the environment. Then, planning methods can be used directly to deduce an optimal strategy π^*. In contrast, model-free approaches directly attempt to learn the optimal policy π^*. By interacting with the environment, the agent tries to maximize the sum of all collected future rewards based on the previously known samples by continuously adjusting its strategy π. In the model-free case, the agent does not learn the environment dynamics but aims to find an optimal policy π^* by trial and error. Compared to model-based approaches, model-free algorithms are easier to implement. Moreover, learning an environment model can be very difficult in some circumstances. In the remainder of this thesis, model-free approaches are focused. According to Zhang & Yu (2020), model-free approaches are divided into value-based and policy-based approaches. These are explained in detail in chapter 2.3.3.

As seen by the definition of SL, USL, and RL, all learning tasks involve learning a functional mapping or functional description of the data in some sort. Often, a good hypothesis of the model to approximate must be found without knowledge about special properties of the data or the application (Ertel 2018, P. 194). Excellent results have been achieved with NNs, which are universal function approximators (Arena & Fortuna et al. 1998, P. 7). NNs learn the parameters $\mathbf{w}$ of a hypothesis $y = h(x|\mathbf{w})$ to best estimate the function $y = f(\cdot)$ (Goodfellow & Bengio et al. 2016, P. 164). NNs are introduced in the following.

2.3.2 Neural networks (NNs) as universal function approximators

NNs are strongly oriented by the structure of the human brain, which processes and transmits information via a network of neurons (Russell & Norvig 2010, P. 727). The neurons receive an input signal from other neurons via synapses and are either activated or disabled (Russell & Norvig 2010, P. 11). As soon as a neuron's stimulus exceeds a threshold value, the neuron activates and transmits this information as a potential pulse via its output to the following neurons. In a network of neurons, the strength of the influence of the individual neurons on each other is responsible for the overall output of the following neurons and consequently the overall output of the network (Russell & Norvig 2010, P. 11).

The Perceptron is the basic building block of a simple NN and the mathematical approximation of a human neuron. A perceptron is a simple form of a NN consisting of one neuron (Russell & Norvig 2010, P. 22). This neuron calculates a weighted sum of its inputs, applies an activation function to it, and outputs the result. A multilayer perceptron (MLP) extends the perceptron by consisting of multiple layers of neurons: an input layer, at least one or more hidden layers, and an output layer. Each neuron in one layer is connected to all neurons in the next layer, each having a weight. The weights of a NN then essentially represent the parameters $\mathbf{w}$ of the hypothesis $y = h(x|\mathbf{w})$ of the NN. Nowadays, NNs are trained using backpropagation, minimizing the error between the actual and predicted outputs of the network. During training, the weights and biases are continuously adjusted by backpropagation and optimization methods such as gradient descent to minimize the prediction error of the NN (Russell & Norvig 2010, P. 733). Please refer to the Appendix A3 for a detailed introduction to the basic mathematical concepts of the perceptron, MLPs, and the backpropagation algorithm.

Another variant of NNs is ***Convolutional Neuronal Networks*** (CNNs). They have become established for processing image data. The development of CNNs was inspired by visual perception (Gu & Wang et al. 2018, P. 354). To recognize an object in an image, the neural network must be able to do so independently of translation, (small) rotations, and scaling (different sizes) of the object (Bishop & Nasrabadi 2006, P. 267). Additionally, nearby pixels form an object in images, and inputs should not be processed independently. MLP can only meet these requirements to a limited extent due to fully connected input and hidden layers. Each pixel would be processed in each input neuron independently, and the output is fed to all neurons in the subsequent layer, representing a rather global than local data processing. Therefore, unlike MLP, CNNs use (1) local connections, (2) weight sharing, and (3) subsampling (Bishop & Nasrabadi 2006, P. 268). One or more convolutional kernels, also called a filter, are applied over the entire input matrix. In doing so, a feature representation of the input is learned. A filter represents the local connections (1) and is defined as an $n_f \times n_f$ matrix whose individual values are the weights to be learned. The filter runs stepwise along the input matrix. In most cases, a quadratic image is fed into the CNN, which makes all the input matrixes quadratic. A quadratic input matrix with size $m_I \times m_I$ can be the image itself when the convolution is the first of the NN, or the input matrix can be the output of a filtering step of another convolution. It computes the output by multiplying the associated values, then summing the products and applying an activation function, just like in traditional MLP. Since the filter runs stepwise over the entire input matrix, weights are shared across the whole input (2) using the same filter values, and significantly fewer weights have to be learned than in a comparable MLP network. The input dimension can be increased by defining multiple independent filters applied to the same input matrix. Pooling layers reduce the dimension of the learned feature representations by subsampling (3). In doing so, the dominant features are retained. For this purpose, a rectangular window is defined in analogy to convolution, which gradually slides over the feature matrix after a convolution. According to the pooling operation, the value is written into the new smaller feature matrix. Popular pooling operations are max-pooling, min-pooling, or average-pooling. Several convolutional and pooling layers can successively be added until a MLP is stacked onto the convolutional part of the neural network to generate the network output. A highly simplified example of this abstraction concept can be found in Figure 2.8 using the example of binary classification. Several layers of convolutional layers and pooling layers are connected one after the other to continuously extract features at an increasing level of abstraction from the initial image. These extracted features can then be used in a downstream MLP to generate the actual network output (in this case, binary classification on whether a starter is present in the image or not).

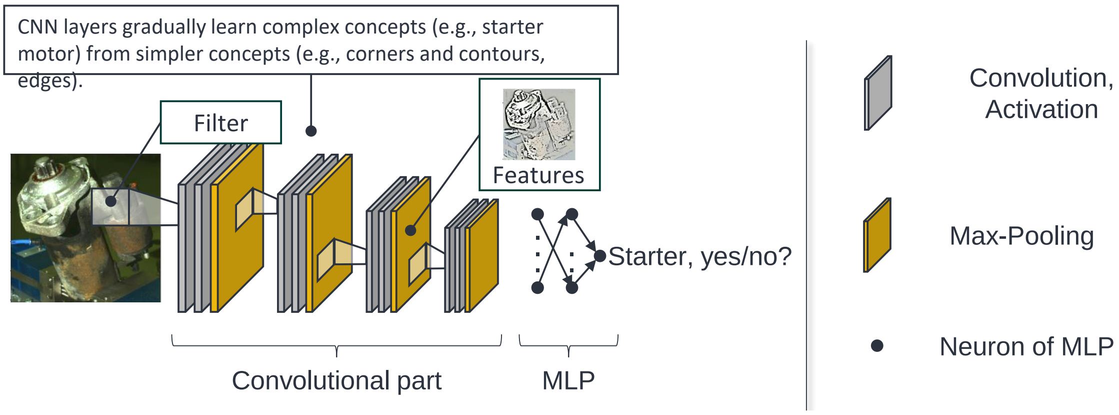

CNN layers gradually learn complex concepts (e.g., starter motor) from simpler concepts (e.g., corners and contours, edges).
Filter
Features
Starter, yes/no?
Convolutional part
MLP
Convolution, Activation
Max-Pooling
Neuron of MLP

Figure 2.8: Highly simplified example of the inner workings and abstraction concept of a CNN.

CNNs have recently coined the term deep learning in the context of intelligent image processing. Many artificial intelligence tasks can be solved by designing the correct set of features to extract for that task, using them in simple machine learning algorithms (Goodfellow & Bengio et al. 2016, P. 3). When done manually, this step of designing a feature extractor requires careful engineering and considerable domain expertise (LeCun & Bengio et al. 2015, P. 436). Often, these feature extractors would fail to generalize. For example, when feature extractors are parameterized to segment cars in daylight, these would almost certainly fail at night due to changing light conditions. One solution is to let the machine learning algorithm learn relevant features for the task to solve itself, which is known as representation learning (Goodfellow & Bengio et al. 2016, P. 4). Deep learning methods are automatic representation learning methods that transform the representation of the input data into a representation at a higher level needed to solve the given task (LeCun & Bengio et al. 2015, P. 436). Deep learning enables the computer to build complex concepts (concept of a person) by combining more straightforward concepts (corners and contours), which are again based on more straightforward concepts such as edges (Goodfellow & Bengio et al. 2016, P. 5). Goodfellow & Bengio et al. (2016) argue that the quintessential example of a deep learning model is the MLP. The authors state that each application of a mathematical function in a neuron provides a new representation of the input. For them, deep learning is "the study of models that involve a greater amount of composition of either learned functions or learned concepts than traditional machine learning does" (Goodfellow & Bengio et al. 2016, P. 5). This is often associated with more hidden layers in the neural network architecture. The main distinction is that the parameterization of these layers is learned from data using a general-purpose learning procedure (LeCun & Bengio et al. 2015, P. 236).

2.3.3 Advanced machine learning using deep learning

This chapter discusses the application of deep learning to problems relevant in this thesis. These are semantic image segmentation and 3D semantic mapping, learning on point clouds, and deep RL with actor-critic approaches.

Image segmentation with deep learning and 3D semantic mapping
In image segmentation, the primary task is to assign each pixel of an image to a specific class. There are two segmentation problems to be distinguished here. The first is semantic segmentation, and the second is instance segmentation. Semantic segmentation has the task of segmenting objects independently. In contrast, instance segmentation must distinguish between several objects of the same class in the image (Minaee & Boykov et al. 2021, P. 3523).

Following the nomenclature introduced in section 2.3.1, according to Garcia-Garcia & Orts-Escolano et al. (2018), deep semantic segmentation is defined as the assigning of a label to each variable of an input $(x_f^{1,1}, x_f^{1,2}, ..., x_f^{W,H})$, which usually is a 2D image with $W \times H = N_p$ pixels. Each label L in the label space $L = (l_0, l_1, l_2, ..., l_k)$ represents a different class or object, where $k + 1$ is the size of the label space with l_0 representing the background (Garcia-Garcia & Orts-Escolano et al. 2018, P. 43). The output of the a deep semantic segmentation NN is then given by $(\hat{y}_f^{1,1}, \hat{y}_f^{1,2}, ..., \hat{y}_f^{W,H})$, where ideally, the assigned label of each pixel $\hat{y}_f^{i,j} = \hat{l}^{i,j}$ equals the true label $l^{i,j}$ ($\hat{y}_f^{i,j} = \hat{l}^{i,j} = l^{i,j}$). Deep semantic segmentation is very similar to the above example of deep binary classification. For deep semantic segmentation, the class membership of each of the N_p pixels has to be predicted, and the number of neurons in the last layer increases solely to enable a classification for each pixel.

In contrast to semantic segmentation, deep instance segmentation is concerned with detecting and segmenting instances of objects in an image. The critical task is not to segment every pixel in an image but to correctly detect, classify, and segment relevant objects in an image. This is often done by predicting a ***bounding box*** (BB) containing an object in an image, classifying which object is inside the BB, and simultaneously performing a segmentation of the object. In accordance to the above definition, instance segmentation first deals with predicting a certain amount m of object BB positions $[(\hat{y}_{f,B}^{UL,1}, \hat{y}_{f,B}^{UR,1}, \hat{y}_{f,B}^{LL,1}, \hat{y}_{f,B}^{LR,1}), ..., (\hat{y}_{f,B}^{UL,m}, \hat{y}_{f,B}^{UR,m}, \hat{y}_{f,B}^{LL,m}, \hat{y}_{f,B}^{LR,m})]$ with UL, UR, LL, LR denoting the upper left, upper right, lower lef and lower right corners of the BBs. For each proposed BB, a predicted classification of the object in it $\hat{y}_{f,B}^{C,i}$ is performed, resulting in the classification vector [$\hat{y}_{f,B}^{C,1}$, ..., $\hat{y}_{f,B}^{C,m}$]. Additionally, for each predicted BB, a binary segmentation is performed on whether a pixel inside a BB contains the object or not, resulting in the segmentation masks [$\hat{y}_{f,B}^{SM,1}$, ..., $\hat{y}_{f,B}^{SM,m}$] with the size of the masks being the size of the predicted object BBs.

Based on the different tasks, different NN architectures have been used to solve these problems. However, all of these architectures are based on an intelligent combination of convolutional layers to process the images as input and individual, problem-specific adaptations of the network architecture to achieve the objective. Well-known representatives of network architectures for semantic segmentation use encoder-decoder architectures (e.g., U-Net), dilated/atrous convolution (e.g., DeepLab architectures), or multi-scale feature extraction or pyramid architectures (e.g., PSPNet) for goal achievement (Emek Soylu & Guzel et al. 2023, P. 4, ff.). Popular NN architectures, for instance segmentation, are the Mask-RCNN (He & Gkioxari et al. 2017) and versions of the YOLO (you only look once) architecture (Redmon & Divvala et al. 2016). The two popular variants for each category, the U-Net and the Mask-RCNN are described in detail below and are visualized in Figure 2.9.

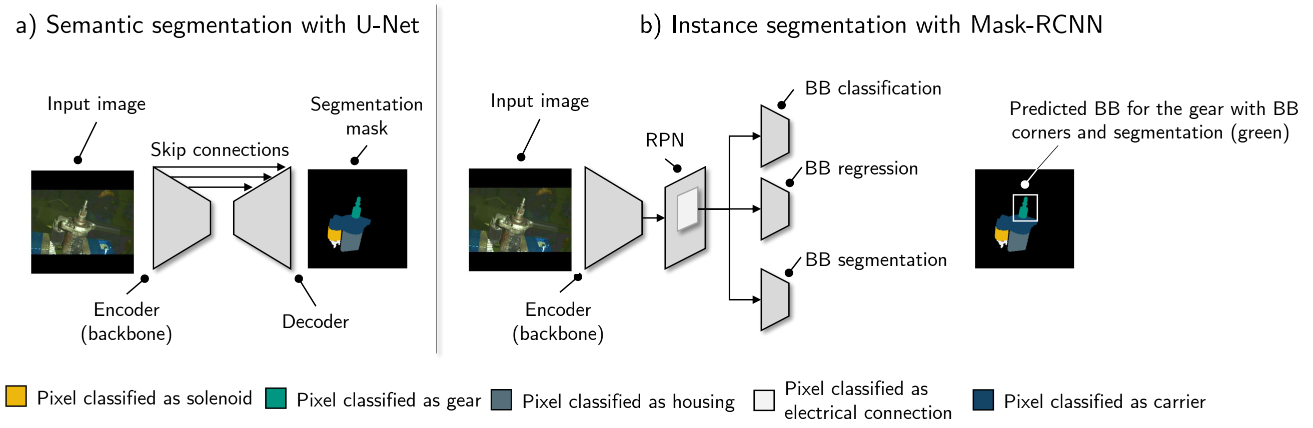

a) Semantic segmentation with U-Net
Input image
Skip connections
Segmentation mask
Encoder (backbone)
Decoder
b) Instance segmentation with Mask-RCNN
Input image
RPN
BB classification
BB regression
BB segmentation
Predicted BB for the gear with BB corners and segmentation (green)
Encoder (backbone)
🟨 Pixel classified as solenoid
🟩 Pixel classified as gear
⬛ Pixel classified as housing
□ Pixel classified as electrical connection
🟦 Pixel classified as carrier

Figure 2.9: Visualization of the segmentation procedure for the U-Net (semantic segmentation) and Mask-RCNN (instance segmentation) based on the example of a starter motor.

The U-Net by Ronneberger & Fischer et al. (2015) employs an encoder-decoder structure consisting of stacked layers of convolution blocks and upsampling blocks. The backbone, also called the encoder, reduces the spatial dimensions of the input image using convolutional blocks while extracting relevant image features. The decoding path of the U-Net is the counterpart to the encoding path. It consists of upsampling blocks using a series of deconvolution and convolutional layers that increase the spatial dimensions to yield an output with the same size (image height and width) as the input image after the decoder. Deconvolution layers perform said upsampling of the input feature space by learning the weights of the upsampling operation instead of upsampling strategies with predefined filters. Between the encoder and decoder, there are skip connections. These allow the network to directly integrate features from the encoder to the decoder. These connections help recover fine-grained spatial information lost while extracting high-level features during encoding and deconvolution. Skip connections enable the fusion of global context information from the encoder using skip connections with local details from the decoding path. One advantage of this model is that good performance can often be achieved even with little training data using suitable image augmentation (Ronneberger & Fischer et al. 2015, P. 240).

The Mask R-CNN, developed by He & Gkioxari et al. (2017), integrates stacked network architectures to achieve object detection and instance segmentation. The architecture includes a convolutional neural network as a backbone, which extracts hierarchical features from input images. This is followed by a Region Proposal Network (RPN). The RPN determines potential object locations as candidate object BBs in the image. The extracted features of the candidate object BBs are then fed to neural network branches for BB classification, BB regression, and binary object segmentation. The BB classification takes features of the BB and performs object classification for each candidate BB. The BB regression branch predicts the offset of the BB coordinates surrounding the object proposed by RPN to improve object detection accuracy. The segmentation head predicts a per-pixel binary segmentation mask for each candidate BB, indicating whether a pixel inside the mask can be associated with the object.

As stated in section 2.2.2, applying image-based machine learning techniques to the RGB or depth image of RGB-D data enables gathering additional information about the acquired object or scene. An application case is provided by the concept of three-dimensional semantic mapping (3D semantic mapping). A semantic map augments spatial information by additional information about entities that are located in space (Nüchter & Hertzberg 2008, P. 915). Currently, these approaches are mainly used for mobile robots. An overview of existing applications is given in Kostavelis & Gasteratos (2015). Techniques for constructing semantic three-dimensional environment representations using deep learning approaches are at the core of recent research. NN are used to enable the segmentation of objects in image data (RGB image and $\setminus$ or depth image) or the segmentation of objects in the acquired point clouds (Qi & Su et al. 2017) and the transfer of the segmentation results to the three-dimensional model (see e.g. works of McCormac & Handa et al. 2017, Sünderhauf & Pham et al. 2017, Martín & González et al. 2021). While 3D semantic mapping using images to segment the 3D scene can be approached by semantic or instance segmentation in images using deep learning, working with point clouds, e.g. for segmentation, using deep learning makes other types of NN architectures necessary.

Deep learning on point clouds
In contrast to data such as color and depth images (RGB-D data) or occupancy grids using voxels, processing point clouds is efficient since only information on the location of the points needs to be stored and processed. However, point clouds are non-Euclidean data that networks cannot process using convolutional layers since these require input in a grid-like structure. Additionally, point clouds have unique properties that pose a challenge to potential network architectures (Qi & Su et al. 2017, P. 254):

- Point clouds are unordered. The order of points at the network input may vary. Therefore, a network must be invariant to permutations of the input order.
- Invariance to transformations. The representations of the point cloud learned by the network must be preserved when geometric transformations such as rotation and translation are applied.
- Relationships of points to each other. The points have a neighborhood relationship, defined by distance in space. The network must be able to extract features from local structures draw conclusions concerning the global point cloud structure.

The *Pointnet*, presented by Qi & Su et al. (2017), is one of the first networks to apply deep learning based on point clouds to classify and segment them. The basic idea of this network architecture is to learn local and global features extracted from point clouds. To solve the problem of invariance to transformations, a subnetwork called ***Transformation Network*** (T-Net) is used. The T-Net takes the raw point cloud data and performs pose alignment by multiplying the original point cloud by a learned 3×3 transformation matrix. Features are extracted in the aligned form using a MLP. These features are aligned in feature space, using another T-Net to obtain the local features. Local features are then processed by another MLP, and a max pooling layer then aggregates the transformed local features of all points into a global feature vector. The local and global features can then be used for classification, regression, or semantic segmentation tasks by stacking further layers on top of the feature extractor to obtain a suitable output.

Deep Actor-critic RL
Modern RL almost exclusively relies on neural networks. Successful applications of deep RL are found in the field of games (Mnih & Kavukcuoglu et al. 2013; Silver & Schrittwieser

et al. 2017; Berner & Brockman et al. 2019), where superhuman performance could be achieved. Other application areas include robotics (Kober & Bagnell et al. 2013) or production system control (Panzer & Bender 2021). Actor-critic RL is the basis of most modern RL algorithms and combines value- and policy-based approaches. The following derivation of the foundations of actor-critic RL is based on the work of Sutton & Barto (2018), which the author refers interested readers to for further details.

Value-based RL approaches aim to estimate the state-value function $V_\pi(s_t)$ or action-value function $Q_\pi(s_t, a_t)$. The state-value function $V_\pi(s_t)$ represents the expected return G_t when starting in s_t and following the policy π. Similarly, the action-value function $Q_\pi(s_t, a_t)$ represents the expected return G_t when starting in state s_t, choosing an action a_t, and then following the policy π. A policy π can be deduced from value-based approaches indirectly. For approaches that estimate $V_\pi(s_t)$, an action is chosen that leads to the state s_{t+1} with the highest state value $V_\pi(s_{t+1})$. When estimating $Q_\pi(s_t, a_t)$, the action a_t with the highest action value is chosen out of all possible actions in state s_t. The rewards r_{t+1} collected during these interactions with the environment are continuously used to update the estimates $\hat{V}_\pi(s_t)$ and $\hat{Q}_\pi(s_t, a_t)$ which also continuously updates the policy π indirectly.

In contrast, policy-based approaches directly optimize a parameterized policy $\pi(a_t|s_t, \mathbf{w}_a)$ with a parameter vector $\mathbf{w}_a$ that directly outputs the probability of selecting action a_t given state s_t. The goal to maximize the expected return can be expressed as an optimization problem concerning a specific performance measure $J(\mathbf{w}_a)$ of the policy $\pi(a_t|s_t, \mathbf{w}_a)$ and its parameters $\mathbf{w}_a$. The optimization problem can then be expressed as in equation 2.4 with the learning rate η.

$$\Delta \mathbf{w}_a = \eta \nabla_{\mathbf{w}_a} J(\mathbf{w}_a) \tag{2.4}$$

It can be shown that the weight update rule can be expressed as seen in equation 2.5 by estimating the performance measure $\hat{J}(\mathbf{w}_a)$ through taking action a_t in state s_t and observing the collected rewards, which constitute to G_t (Sutton & Barto 2018, P. 327):

$$\Delta \mathbf{w}_a = \eta \nabla_{\mathbf{w}_a} \hat{J}(\mathbf{w}_a) = \eta G_t \frac{\nabla \pi(a_t|s_t, \mathbf{w}_a)}{\pi(a_t|s_t, \mathbf{w}_a)} \tag{2.5}$$

Intuitively, $\nabla \pi(a_t|s_t, \mathbf{w}_a)$ indicates the influence of individual weights on the policy's action choice. The weight adjustment is proportional to the influence of the weights on the action choice and the obtained return G_t, i.e., the performance of the policy. At the same time, it is inversely proportional to the probability $\pi(a_t|s_t, \mathbf{w}_a)$ of action choice a_t, so weights that influence actions that are not yet chosen very often, are adjusted more strongly.

A disadvantage of this estimation is that a weight update of $\mathbf{w}_a$ is only possible after observing a significant number of subsequent rewards and the associated estimate of the expected return after taking action a_t in state s_t. Therefore, an alternative form of estimation exists using equation 2.6 (Sutton & Barto 2018, P. 331).

$$\Delta \mathbf{w}_a = \eta(r_{t+1} + \gamma_{RL}\hat{V}(s_{t+1}|\mathbf{w}_c) - \hat{V}(s_t|\mathbf{w}_c))\frac{\nabla\pi(a_t|s_t, \mathbf{w_a})}{\pi(a_t|s_t, \mathbf{w}_a)} \tag{2.6}$$

In this estimation, the expected return is expressed by the so-called Advantage, which indicates how good it is in a state s_t to perform an action a_t resulting in a state s_{t+1}. An estimated state value function $\hat{V}(\cdot)$ is also needed. This is implemented by another parameterized function with the parameters $\mathbf{w}_c$. The parameterized function, which is optimized using equation 2.6, is called Actor, while the parameterized function, which estimates $\hat{V}(\cdot)$, is called Critic. The optimization of Critic is performed by using the mean square error between the estimate of the state value at time t and the successor state at time $t + 1$ via equation 2.7.

$$\mathcal{L} = f_L(\hat{V}(s_{t+1}|\mathbf{w}_c), \hat{V}(s_t|\mathbf{w}_c)) = (r_{t+1} + \gamma_{RL}\hat{V}(s_{t+1}|\mathbf{w}_c) - \hat{V}(s_t|\mathbf{w}_c))^2 \tag{2.7}$$

This utilizes the so-called Bellmann equation, which states that the value of a state is equal to the discounted value of a successor state and the reward obtained directly after the action a_t that is performed to get from s_t to s_{t+1}. The procedure is called Actor-Critic since the Actor first takes action a_t, and the action selection policy is optimized via the Advantage estimation with the help of the Critic.

2.3.4 Summary and conclusion: View planning enabled by machine learning

The solution methods for VPP aim to calculate a view plan with minimal length to fulfill a planning goal. Analytical methods enable the efficient definition of successive acquisitions at predefined VP for a pre-defined planning task. However, the planning task for a returning used product in remanufacturing is not predefined and must therefore be performed reactively. The insights into machine learning methods in this section enable the question raised in section 1.3 to be answered:

Adaptivity: How can the adaptivity of such solution methods be ensured to deal with the inability to plan the inspection process in advance?

Due to the possibility of formalizing VPP as an optimization problem, machine learning approaches can also be applied to VPP, as they are predestined to solve optimization problems. On the one hand, SL is suitable for this, provided a suitable training data set can be constructed. But RL is also suitable, as the solution of a VPP represents a sequence of VP, which can be determined using RL as a solution method for sequential decision problems. Furthermore, deep learning approaches, such as semantic segmentation, enable the additional integration of planning knowledge by understanding and interpreting the planning scene at hand. Integrating such approaches may, thus, enable a more intelligent and adaptive selection of VP than analytical methods under the prevailing planning uncertainty in remanufacturing.

3 State-of-the-art literature review

Considering the research objective, the state of research of this thesis is presented below. First, a rough overview of existing work in the field of image-based inspection in remanufacturing is provided (section 3.1). Based on that, the overarching delimitation criteria for differentiating existing research work (section 3.2) relevant to this thesis are deduced. The existing works are briefly presented in the following, focusing on the delimitation criteria introduced. Within these sections, a brief presentation of the industrial application of analytical approaches for solving the VPP (section 3.3) and a detailed discussion of approaches based on machine learning (section 3.4) takes place. The degree of fulfillment of the delimitation criteria of differentiation of existing works is then used to derive the research deficit of this work (section 3.5).

3.1 Approaches for visual inspection in remanufacturing

This section first analyses existing work in the field of image-based defect detection in remanufacturing. Based on this, an interim conclusion is formulated.

3.1.1 Works using image-based defect detection in remanufacturing

Globisch & Thäter et al. (2019) presents a test setup for acquiring data from surfaces and determining the type and extent of surface contamination of objects based on image data. In remanufacturing, this can be used to determine the degree of contamination of an object before the cleaning process. Similarly, it can be used after a cleaning process to verify the success of the process. The system works without intelligent evaluation software based on classic image processing methods. Depending on the size of the object to be inspected, the camera system can be moved linearly using a motor to ensure the optimum working distance between the object and the acquisition system for varying inspection objects. The present experimental setup is only suitable to detect contaminations, but not to subdivide them into different types of contaminations.

An approach that is limited solely to the detection of corrosion as part of a remanufacturing process is that of Gibbons & Pierce et al. (2018). For this purpose, the image data is first displayed in the LAB color space, and texture features are extracted. With the help of principal component analysis, these features were reduced to the essential features describing the texture, and a Gaussian Mixture Model was used to detect pixels that represent corrosion.

Nwankpa & Ijomah et al. (2021) consider an inspection process in which classification is to take place based on image data of surfaces with and without defects. They use video data and carry out manual pre-processing to have images with and without defects that are cleanly aligned. The authors use a NN architecture with a pre-trained ResNet18 backbone as an algorithm to distinguish between surfaces with and without defects and to classify individual defects (e.g. pitting, corrosion, cracks, ...). They demonstrate that a classification of predefined defects with pre-recorded and pre-processed image data is possible with high accuracy.

In contrast to Nwankpa & Ijomah et al. (2021), Zheng (2021) does not use a NN for defect classification, but for defect localization for remanufacturing and repair tasks. Zheng (2021) also use a pre-trained ResNet backbone, in this case the ResNet101 backbone, which has the same architectural structure as the ResNet18 structure used by Nwankpa & Ijomah et al. (2021), but uses more layers. This backbone is used in a Mask-RCNN that is especially suitable for localization tasks. The authors only distinguish between damaged and undamaged object surfaces and not between different defects. They can show that high detection accuracy of damaged object areas can be achieved and demonstrate this on a specially constructed data set of damaged plastic pipes.

An approach that takes up and combines the ideas of the work of Globisch & Thäter et al. (2019), Nwankpa & Ijomah et al. (2021), and Zheng (2021) is that of Saiz & Alfaro et al. (2021). A setup is presented that makes it possible to acquire image data that, with an ensemble consisting of a semantic segmentation network and object detection algorithm as an evaluation method, enables both the localization of defects and the classification into defect-free, recyclable, and reject. This approach is evaluated using the use case of joint cages. The approach shows better localization and classification accuracies compared to traditional machine learning methods with manual feature extraction.

An approach to product identification is presented in the two consecutive works Schlüter & Niebuhr et al. (2018) and Schlüter & Lickert et al. (2021). While in Schlüter & Niebuhr et al. (2018) the product identity is determined using inherent product characteristics (weight, size, volume, barcodes, and visual features) and used for sorting in remanufacturing, in Schlüter & Lickert et al. (2021) only several images of the product from different perspectives are used for identification. The work utilizes specially developed test setups that use sensor technology (e.g., camera systems and scales) to acquire the relevant data for identification. In particular, NN are used in both works. While in Schlüter & Niebuhr et al. (2018) these are used to extract the visual product features, a representative of the ResNet architecture (ResNet50) is used in the work Schlüter & Lickert et al. (2021) to directly output the object class. In Schlüter & Lickert et al. (2021), a high level of accuracy was achieved in the identification of various starter motors, which represent real remanufacturing products, only using image data.

In addition to pure product identification, the localization of defects, or their evaluation, image processing systems can also be used to support remanufacturing. One example is the work of Khan & Mineo et al. (2021). Here, a camera system is first used to create a 3D model with Structure from Motion. To do this, this camera system is attached to a robot arm and a specific robot trajectory is followed around an object of interest. The 3D model generated from the image data can then be used to create a trajectory for a robot-guided ultrasonic inspection system. The rationale behind this approach is that the geometry of a product that is already in the usage phase can deviate from the standard geometry of a new product and therefore the trajectory of a robot-guided ultrasonic inspection system must be planned adaptively. This is the only way to accurately test relevant product features using a non-contact, non-destructive testing method.

3.1.2 Interim conclusion to existing works on image-based defect detection in remanfuacturing

The analysis of existing work in the field of visual inspection in remanufacturing shows the range of research activities that have taken place and are currently taking place. Ranging from the determination of the general degree of contamination (Globisch & Thäter et al. 2019) to the detection of individual types of contamination (Gibbons & Pierce et al. 2018) and the differentiation of several different surface defects (Nwankpa & Ijomah et al. 2021). In addition, there are also works focus on the localization of defects on product surfaces for targeted reprocessing (Zheng 2021; Saiz & Alfaro et al. 2021), and product identification (Schlüter & Niebuhr et al. 2018; Schlüter & Lickert et al. 2021).

The analysis of these works shows that they mainly focus on the processing of image data but not on the planning of a setup that generates this image data. For example, Nwankpa & Ijomah et al. (2021) assumes that defects in a video stream can be detected but does not further specify how this video stream is generated. Globisch & Thäter et al. (2019), Zheng (2021), Saiz & Alfaro et al. (2021), Schlüter & Niebuhr et al. (2018) and Schlüter & Lickert et al. (2021) also present experimental setups that use one or more camera systems positioned rigidly in space. Thus, a major limitation of these works is the flexibility that can be achieved by moving the camera system in relation to the object to inspect or evaluate defects on the entire object surface from different perspectives. Only the work of Khan & Mineo et al. (2021) uses a robot-guided camera system to first generate a 3D model of the inspection object and thus enable the detailed planning of the trajectory of an ultrasonic inspection system, which is also robot-guided. However, the robot's initial trajectory to create the 3D model is fixed and not specifically adapted to the respective inspection object.

A comprehensive consideration of the inspection problem in remanufacturing, therefore, requires not only the correct evaluation of the measurement and image data but also, under certain circumstances, targeted guidance of a flexible measuring system to be able to detect the defects and flaws on the entire object surface or individual ROI. The present thesis deals with the latter problem. In the following section, the relevant delimitation criteria are defined, and the specific state of research is presented.

3.2 Derivation of delimitation criteria

The delimitation criteria of this thesis are categorized according to the structuring questions introduced in chapter 1.3 and are used to derive the research deficits on which this thesis is based.

1. **Problem formalization:** How can the problem of autonomous data acquisition be formalized?

    a) **Use case**
    Most existing work can be divided into one of the application use cases remanufacturing, metrology, object reconstruction, or active object recognition. In such applications, a static, non-moving robotic system is used to solve the VPP. The comparison with existing work in the field of moving robotic systems (e.g., using driverless transport systems) is omitted. This is because the boundary conditions are different despite the problem to be solved (VPP) being the same. In most cases, the solution of such systems describes trajectories in unknown terrain, whereby subproblems (e.g., simultaneous localization and mapping) that do not play a role in the application under consideration have to be solved.

    b) **Inspection problem**
    A key differentiation criterion is which inspection problems a solution approach solves. In particular, whether the solution approach is only evaluated for one subproblem (e.g., NBV, RPP or IPP) or whether several inspection problems are solved with the approach presented in the respective works.

2. **Solution approach:** Which group of solution methods exists for solving the formalized data acquisition problem?

    a) **Solution algorithm**
    A distinction is made between existing work based on the solution algorithm used for these application use cases. A distinction is made between methods that take an analytical approach to the problem and those with a learning component.

Algorithms aiming for an analytical solution are mainly those presented in section 3.3. In contrast, approaches relying on learning-based algorithms use machine learning to solve the VPP and are presented in section 3.4.

b) **Use of a geometry model**
Regardless of whether it is an analytical or learning approach, a distinction can be made as to whether this approach depends on a geometry model or is only applicable when the product to be investigated does not change its shape. A distinction is, therefore, made as to whether the approach in question is model-free or model-based or partly model-based. In a partly model-based setting, no object geometry is available but only one object shape is investigated. In the case of the model-based approach, a distinction can also be made as to whether a geometry model of the inspection object is used to plan the inspection process in advance (model-based-A) or whether a geometry model is first generated at system runtime but before the actual inspection (model-based-G) is carried out and then the inspection is completed.

c) **Multicriteria optimization**
The overall objective of every VPP is to acquire the surface of the inspection object required to achieve the inspection objective with a minimum number of necessary acquisitions. However, there are also secondary objectives (e.g., minimizing the travel path length of the inspection system or maximizing the accuracy of the acquired surface points for reconstructing the inspection object). A delimitation criterion is, therefore, which and how many secondary objectives the respective solution approach considers.

3. **Adaptivity:** How can the adaptivity of such solution methods be ensured to deal with the inability to plan the inspection process in advance?

a) **Integration of prior knowledge**
If used, a geometry model is central prior knowledge for solving the VPP. However, further problem-specific prior knowledge can also be used to exclude suboptimal solutions of the VPP from the start. This concerns, for example, the restriction of the value range of optimization variables of VPP or the analytical determination of such variables.

b) **Adaption at system runtime**
Adaptivity is a fundamental prerequisite for autonomous system behavior. This adaptivity describes a reactive adjustment of the system behavior (e.g., adjustment of a view plan) at system runtime so that it can react to changing conditions (e.g.,

varying ROI to be inspected). In this work, the term adaptivity at system runtime describes the ability of a system to adapt its behavior without the system coming to a standstill. The adaptation of the system behavior must be within the lower single-digit second range.

The derived criteria for comparison are addressed in the following descriptions of the works in the state of research wherever possible.

3.3 Approaches using analytical methods

In the industrial field, the majority of works solving the VPP using analytical approaches deal with the RPP or IPP. A detailed summary of existing works is given by Peuzin-Jubert & Polette et al. (2021). Due to the large number of works in the industrial environment and the rough restriction of alternative solutions already carried out in the fundamentals, the derivation of the research deficit is conducted only considering works using robot-guided acquisition systems with static data acquisition. These works are presented briefly below.

In a series of papers, Raffaeli, Germani and Mengoni et al. present their approach to reconstruction planning (Germani & Mengoni et al. 2009; Germani & Mandorli et al. 2010; Raffaeli & Mengoni et al. 2013b; Raffaeli & Mengoni et al. 2013a) called *Intelliscan* in a metrological application. An automated system is presented to optimize the process of verifying geometric tolerances for mechanical components. The system includes the planning, simulation, and control of the inspection process using a 3D acquisition system guided by a robotic arm and the presented software tool for solving the VPP given a CAD model of the inspection object. After determining them, a secondary goal is minimizing travel path lengths between successive VP. However, this minimization is not part of the actual VP planning process. The solution algorithm presented in Germani & Mengoni et al. (2009) is based on the solution method of CM using surface normals and visibility maps, whereby a human operator is nevertheless responsible for the final selection of the VP. A continuation of this work is that of Germani & Mandorli et al. (2010). This work describes the underlying software product and how, compared to the work Germani & Mengoni et al. (2009), further planning approaches have been built into a knowledge base for optimal VP determination. A further extension of the planning approach with additional sampling strategies for object geometries and rules for determining the VP was then presented in the works Raffaeli & Mengoni et al. (2013b) and Raffaeli & Mengoni et al. (2013a). The experimental results have shown the system's robustness with respect to the algorithms for position planning and the possibilities for time and cost efficiency in pre-planning the inspection plan for checking geometric tolerances. However, the authors also discuss in detail the problems such as occlusions, the complexity of the planning algorithm due to a large number of parameters, and the realism of the simulated observations.

To enhance the realism between simulated acquisitions and real system acquisitions, Koutecký & Paloušek et al. (2016) incorporate prior knowledge in the form of a reflectance model into their planning approach. Utilizing a provided CAD model of the inspection object, the authors employ the concepts of CM and visibility maps to determine the optimal combination of VP for solving the RPP in a metrological use case. Emphasizing complete surface coverage (RPP) of the object, the primary focus of the study lies in fine-tuning the parameters of the acquisition system, particularly the exposure time. This optimization minimizes reflectance issues and maximizes surface coverage for each individual VP. The authors show that their approach increases the realism in predicting object coverage while solving the planning problem.

An approach tailored to large-scale planar objects given a CAD model of the inspection object is presented by Wu & Lu et al. (2015). As the previous works, it uses CM for VP generation and evaluation. To achieve this, the mesh's vertices of the approximately planar inspection object are projected onto a 2D plane. This planar projection subdivides the mesh into individual scan regions in the subsequent step. When calculating the view plan, the authors not only consider the field of view of the acquisition system but the authors also consider the need for overlaps between the individual acquisitions so that the data of the 3D data acquisition at individual VP can be registered and full surface coverage can be achieved. The experiments show the proposed method's applicability to the use case of an airplane wing model to solve the RPP for a metrological use case. However, a major drawback of the proposed approach is that it is only effective for planar inspection objects based on the assumptions used to deduce the acquisition strategy.

To account for pose variations of inspection objects on a conveyor belt, Jing & Goh et al. (2018) use a Monte Carlo tree search approach to solve the RPP in a metrological application. Based on a CAD model, an offline generation of VP takes place using prior knowledge, enabling random sampling in the space around the CAD model. The visibility of the VP is evaluated, and for all VP, the robot trajectories are calculated in relation to each other. During the system runtime, the optimal inspection strategy can be learned using the robot trajectories and the information about which object areas can be acquired via which VP and the Monte Carlo tree search algorithm. In this work, a paradigm shift takes place for the first time. Although a model of the inspection object is available, the view plan is generated at runtime since the exact position and orientation of the object are unknown, enabling a view plan adaption. Furthermore, the approach directly takes into account the travel lengths between chosen VP to minimize them in conjunction to the surface coverage. The proposed approach is virtually validated, demonstrating improved performance in handling online coverage planning with pose variation compared to NBV methods.

A similar metrological approach is taken by Wu & Lin et al. (2020). The authors freely place known inspection objects on a conveyor belt. A robot-mounted acquisition system is used to recognize objects on the conveyor belt. Thereafter, the pose and orientation of the inspection objects are determined by the acquired 3D data by registering these to the known inspection object models. This enables view plan adaption by calculating the desired end effector positions for inline inspection of the inspection object's predefined ROI. Like the work of Jing & Goh et al. (2018), this work also enables a certain adaptivity in the system behavior to solve the present IPP, but depends on a geometric model for registration of the acquired point clouds to the available model to readjust the pose of the acquisition system.

3.4 Approaches using machine learning

The following section presents methods for solving the VPP using machine learning methods. In addition to the previously introduced delimitation criteria, three further delimitation criteria are introduced and defined as relevant for the following approaches based on machine learning methods.

1. **Input space**
   A distinction is made as to the extent to which the information relevant for predicting the next VP is represented by the machine learning method. In the current state of research, the object representation forms in the form of point clouds, occupancy/voxel grids, or RGB-D data are relevant for this purpose. Additionally, the robot's current state, represented by the end effector's pose or joint angles, can be utilized to plan the subsequent VP.

2. **Ouput space**
   Analogous to the input space, a distinction can be made as to how the output space of the machine learning process is defined, from which a VP can ultimately be derived using a suitable mapping. A distinction can be made if one of several predefined VP is selected by the machine learning procedure or if the pose of the VP in the continuous space (possibly including restrictions) is regressed independently by the machine learning procedure.

3. **Object**
   Finally, it is relevant whether the machine learning method is used to provide a strategy for determining the next VP for only one object or for several different objects. Specifically, the question is whether only data from one object was used to train the machine learning

procedure. In this case, the corresponding procedure has been fine-tuned to this object. If multiple objects are used, the goal of the procedure is for the respective learning procedure to learn to make good predictions for more than just one object to determine the VP.

3.4.1 Approaches using supervised learning

Based on image data, Ashutosh & Kumar et al. (2022) present an approach that calculates the NBV for surface coverage maximization to enable the best possible object reconstruction. Based on an input image, a CNN selects the NBV as one of 11 possible predefined further acquisition system poses on a sphere (prior knowledge). The proposed method addresses the challenge of selecting the NBV for 3D reconstruction without using preprocessed ground truth next views and a needed object model at runtime, thus being a model-free approach. The novelty lies in extracting a supervisory signal from the reconstruction process to drive the training of the CNN to classify the next acquisition system pose. To this end, the acquired image from the NBV and the image of the previous acquisition systems pose are input to a reconstruction model to generate a 3D shape represented in voxel grids with binary occupancy. The loss for the neural network is then calculated based on the reconstruction quality compared to the ground truth 3D shapes. This joint classifier-reconstructor NN model is trained end-to-end using the loss function based on the reconstruction error, enabling the training process without needing preprocessed ground truth NBV. The proposed method is evaluated on both synthetic and real data. Additionally, the authors present a qualitative analysis of the selected NBV to gain insights into the process and their dependence on object categories and shapes. The approach is suitable for training one model to handle multiple objects simulatenousely.

In the work of Monica & Aleotti (2021), a method for planning the NBV of a depth camera for scene exploration and scene reconstruction is proposed, leveraging a CNN in conjunction with a probabilistic occupancy map of the environment. The proposed method is a hybrid approach that combines the CNN for object completion and a probabilistic NBV planner to evaluate the information gain of possible acquisition system poses. The CNN is trained to predict environment priors and generate a probabilistic occupancy map of the environment (thus being model-free), which is then used in the probabilistic NBV planner to compute the optimal acquisition system pose. Therefore, a voxel grid is encoded with known, unknown, or empty. This partially known representation of the environment is fed into an encoder-decoder CNN to obtain a probabilistic map. This map contains occupancy probability values for each cell in the environment, indicating the likelihood of the cell being occupied. Multiple acquisition system poses with different orientations are sampled in each empty cell in the environment representation. These candidate poses represent potential locations for the acquisition system to move to to gather more environmental information. A probabilistic ray-casting method determines the information gain of each candidate acquisition system pose. This involves evaluating the potential increase in information (surface coverage) of the voxels that have not yet been acquired, allowing the system to select the optimal sensor pose for the NBV. The experimental results indicate that the proposed method achieves results comparable to the state-of-the-art method. Additionally, the authors present results from experiments conducted on a real robot, demonstrating the practical applicability of their approach. As with the work of Ashutosh & Kumar et al. (2022), the approach is suitable for training one model to handle multiple objects simulatenousely.

An approach to defining the NBV for full object reconstruction using SL is proposed in the joint work of Mendoza & Vasquez-Gomez et al. (2020) and Vasquez-Gomez & Troncoso et al. (2021). In the first work, the authors train a 3D CNN that learns to predict the NBV in each acquisition step using a previously generated dataset with annotation of the optimal VP (Mendoza & Vasquez-Gomez et al. 2020). The generated dataset consists of voxel grids of multiple objects as input for which acquisitions have already been performed. Information about previously acquired voxels and voxels yet to be acquired are encoded accordingly. From a predefined set of 14 acquisition poses, the network must then classify the NBV that will achieve the highest possible increase in information (surface coverage) of the voxels that have not yet been acquired. In a subsequent paper, the authors generalize the problem. In contrast to the classification of the acquisition pose, they formulate the problem as a regression problem (Vasquez-Gomez & Troncoso et al. 2021). In contrast to the classification of a fixed acquisition pose, the 3D CNN used must now output a desired position of the acquisition system in continuous space. The acquisition system's orientation is calculated so that the acquisition system points to the object using prior knowledge of the origin. The loss of the 3D CNN is calculated from the error of the predicted position of the next best acquisition pose and the best position from the 14 poses contained in the data set with multiple objects. The authors compare different neural network architectures and can show that the regression approach performs better than the classification approach first proposed.

In the work of Pan & Hu et al. (2022), a method for estimating multiple VP of an acquisition system on a robot for object reconstruction tasks is proposed. The approach uses a 3D CNN to learn prior knowledge given an object and the optimal solution of a set-covering algorithm. The 3D CNN takes the volumetric occupancy grid as input and directly predicts all acquisition system poses of a finite set of predefined VP needed to cover all of the object's surface in a one-shot manner (solution of the RPP). This is achieved by transforming the view planning problem into a set covering optimization problem, allowing the 3D CNN to efficiently predict the subset of views that minimizes movement cost while covering all surface areas.

The 3D CNN training dataset is automatically labeled using the set covering algorithm (prior knowledge). The loss function is designed to achieve high surface coverage of the objects while simultaneously minimizing travel costs between acquisition positions. The proposed method addresses the limitations of traditional NBV methods by enabling global path planning for object reconstruction tasks and a one-shot solution to the VPP. It reduces inference time and movement cost compared to search-based and data-driven RPP methods. The authors perform comparative experiments, benchmarking the proposed approach to analytical and learning-based algorithms such as the NBV-Net proposed by Mendoza & Vasquez-Gomez et al. (2020) and Vasquez-Gomez & Troncoso et al. (2021).

An approach to solving the NBV problem for robotic reconstruction tasks based on point clouds is introduced in the work Zeng & Zhao et al. (2020). The proposed method aims to determine the NBV a robot system sensor after each acquisition, enabling solving the VPP for 3D object reconstruction. To this end, the authors propose the PC-NBV network, a NN that can process point clouds and determine one of 33 predefined candidate VP sampled on a sphere (prior knowledge) that it deems will have the highest information gain (surface coverage) concerning the reconstruction task. The PC-NBV network takes raw point cloud data of the currently scanned object surface, and the view selection states it as input and predicts the information gained for each candidate view. The network is trained using simulated reconstruction processes on synthetic 3D object models (ShapeNet dataset) to build effective training supervision. The PC-NBV network demonstrates improved efficiency and outperforms state-of-the-art NBV methods using SL, also including the NBV-Net by Mendoza & Vasquez-Gomez et al. (2020).

3.4.2 Approaches using reinforcement learning

Deinzer & Denzler et al. (2003) carried out early work on the optimal choice of the NBV for object recognition. Based on a series of acquired images and camera movements, the authors attempt to estimate the present object class and its pose. Only one object is used (a cup) where different labels ('A' or 'B') on the object denote different object classes. As actions, the RL agent outputs a relative movement of the camera (action a_t) based on the current estimation of the object class and acquisition system pose (used as the state s_t). The object class and object pose are then estimated using all the acquired images and the camera pose in the current recognition process after moving the camera and taking an image. In addition, a reward r_t is returned to the agent. This reward is calculated based on the quality of the selected camera pose expressed as the informational content of the state after performing an action. Using collected state-space-reward transitions, the authors use Monte Carlo learning to estimate the action-value function $Q_\pi(s_t, a_t)$ for action selection. The experimental results show that the presented approach can learn an optimal strategy for viewpoint selection that generates only the minimal number of images required for reliable object recognition. The authors do not use an object model for state deduction, action selection, and reward calculation. However, the authors only evaluate object recognition in a simple case of one product, whose associated class depends on numbers printed on it. Prior knowledge is integrated by restricting the camera to move on a circular path around the object.

An approach to determine the pose of the NBV by manipulating an object using a robotic arm for object recognition is presented by Korbach & Solbach et al. (2021). In contrast to Deinzer & Denzler et al. (2003), in this case, the camera does not move around the object but is actively manipulated by a robotic arm. The authors use image data from the camera and a CNN for classification. A distinction is made between 48 different objects, for which no object model is required. The aim of the presented algorithm is to select the poses of the object in such a way that the confidence of the classifier is maximized for the correct object class and is evenly distributed for all other object classes. As the state s_t, the authors pass the pose of the end effector and the confidence difference for classifying the object with respect to all other objects in the data set to the agent. The agent outputs a pose of the agent consisting of a Cartesian coordinate and orientation as a quaternion as action a_t. The agent receives a reward r_t based on the confidence difference of the classification. The authors choose the SAC algorithm as a solution approach. The results of the presented approach show that learning of NBV to maximize the confidence difference of the classification of present objects with respect to other objects can be learned by RL using a SAC algorithm.

One of the first works to solve VPP using RL is that of Devrim Kaba & Gokhan Uzunbas et al. (2017). The authors train one Reinforcement Learning (RL) agent for the sequential positioning of an acquisition system for the complete acquisition of the surface for each of the inspection objects evaluated. The authors define a fixed number of possible poses of the acquisition system in space. The action value function $Q_\pi(s_t, a_t)$ is estimated to determine the next pose of the acquisition system. In this case, an action describes the agent's selection of one of the possible poses of the acquisition system. To select the next pose of the acquisition system, the RL agent receives a binary coding of all previously performed actions and a binary coding of the action (pose of the acquisition system) for which the action value is to be estimated as the state s_t. A MLP is used for estimation of the action-value function $Q_\pi(s_t, a_t)$. As RL algorithms, the authors evaluate the SARSA, TD, and a so-called Watkins-Q algorithm. As a reward signal, the authors choose a simple reward r_t of -1 per additional acquisition performed to detect an inspection object as quickly as possible (with as few acquisitions as possible). The approach is evaluated on 20 different objects. The evaluated algorithms are trained separately for each object and optimized to its geometric structure. For this purpose, the authors do not explicitly use the geometry model of the object to deduce state information

s_t provided to the agent but keep the boundary conditions (one object and its positioning of the object) fixed to learn an acquisition strategy for solving the VPP by the agents. However, since the agent is only trained on one object and predefined VP as well as the object surface covered by said VP is used to determine termination criteria of episodes, it is essentially model-based. The results show that with sufficient exploration, the algorithms perform at least as well as a greedy algorithm in terms of surface coverage achieved.

An approach for solving the VPP for arbitrary objects is presented by Potapova & Artemov et al. (2020). The authors use the geometry model of the objects under consideration and use it to plan the next acquisitions to be carried out until the required surface coverage of the entire object is achieved. The RL agent used in the work selects as action a_t one of 100 predefined VP on a spherical surface around the inspection object. To determine the action, the DDQN learning algorithm and a CNN as function approximator are used, which processes the previously voxelized geometry model of the object to be reconstructed as state s_t to determine the action. The reward signal r_t used consists of several components and rewards the agent based on the surface coverage gained per acquisition, the reconstruction error, an uncertainty measure, and a negative term. This constant negative term is intended to incentivize the agent not to perform infinite actions if there is no benefit from additional surface coverage or a more accurate reconstruction. The authors compare agents trained with different weightings of the individual reward components with heuristic benchmark algorithms. Although the authors propose an optimization of several target variables (including surface coverage), they only evaluate the reconstruction error based on the Hausdorff distance.

The work of Landgraf & Meese et al. (2021) couples a RL agent for solving the VPP with a robot simulation for the first time. Like Korbach & Solbach et al. (2021), the authors use the robot's end effector pose as the agent's state. The authors also use a predefined sampling procedure to define the action space. The authors evaluate two possibilities of defining the action space. In the discrete case, the authors create a grid with a fixed grid spacing in the robots' working space, whereby intersection points are possible positions of the acquisition systems' poses. A certain number of orientations are then heuristically defined for each position. In the continuous case, the RL agent has to output x,y, and z coordinates and angles specifying the end effector's orientation. In addition to this defined procedure for determining the action space of possible actions a_t, the authors also use a random sampling of possible positions and orientations of the end effector. After performing an action, the agent receives a reward based on the additional surface area of the inspection object covered. For action selection, the authors evaluate Q-learning, DQN, and PPO as learning algorithms and use a MLP for the latter two to process the state s_t and compute the action a_t. Similar to Devrim Kaba & Gokhan Uzunbas et al. (2017) and Potapova & Artemov et al. (2020), the authors learn one

RL agent for each object. A geometry model is not assumed, but the position and orientation of the individual objects are constant for each learning run so that the RL agents can adapt their actions to the object shape at hand. The results show that the proposed approach is basically suitable for training RL agents to solve VPP. However, the authors also discuss in depth the challenges associated with the modeling due to the lack of complete object coverage of the considered objects by the evaluated RL agents. In particular, the complexity resulting from a large and continuous action space and the associated optimization in local minima are discussed and related to possible future research.

Zeng & Zaenker et al. (2022) presents another method using a robot simulation to solve a visual planning problem. In contrast to the previous work, this work solves a search problem for the automatic exploration of an unknown environment. The use case is an agricultural problem in which the objects to be picked must first be located before a picking operation can be carried out. The authors use the recording system's history of acquisition poses and a so-called observation map as state s_t. The action selection takes place via a neural network with two heads (CNN for processing the observation map and MLP for processing the pose history of the acquisition system). As an action a_t, this network outputs how the acquisition system should move in space (left, right, up, and down) and whether it should reorient itself (pitch up/down and yaw left/right). The reward of the RL agent is based on the newly found ROI and newly found occupied and free voxels of the space to be explored. The DDQN is used as the learning algorithm. The results, evaluated in two different learning environments, show the applicability of the proposed approach and a better performance compared to a state-of-the-art method.

An approach for VPP solution based on RL taking into account illumination optimization is proposed by Wang & Peng et al. (2023). The approach addresses challenges in capturing high-quality images due to occlusions and lighting issues, particularly for complex products. For a fixed given inspection object, the authors perform offline evaluation of a multitude of virtual acquisitions with different lighting conditions. The RL algorithm used aims to optimize the selection of VP to ensure comprehensive coverage and visibility of product surfaces to acquire. This is realized by developing a visibility estimation method and applying the Asynchronous Advantage Actor-Critic (A3C) RL approach. The agent iteratively adds acquisition system positions and lighting source positions to the scene, until the inspection object is fully acquired. Similar to Devrim Kaba & Gokhan Uzunbas et al. (2017) the state is compromised of a binary vector, whether a specific acquisition system position and lighting source has been already placed. Similarily, the output is compromised of a binary vector, which acquisition system and lighting system position to take next. The method's effectiveness is validated through experiments with varying inspection object models, while an RL agent has to be trained for each inspection object separately (similar to e.g., Landgraf & Meese et al. (2021)). The experiments show, that the RL approach outperforms a particle swarm optimization (PSO) method. However, the approach is not benchmarked against a deterministic analytical solutions.

3.5 Derivation of the research deficit

Based on the existing work presented in the state-of-the-art, the deficits of analytical methods and methods based on machine learning can be derived. An overall research deficit for the autonomous view planning in remanufacturing is derived. Table 3.1 summarizes the approaches and shows the fulfillment of the delimination criteria for the solution method derived in section 3.2.

Deficit specific to analytical approaches
So far, current approaches to solving the VPP in the industrial environment have mainly focused on traditional, model-based methods. These use a previously known geometry model of the inspection object to calculate a view plan to solve the VPP. This calculation occurs upstream, whereby the resulting view plan is implemented in the production environment. The objects are always identical (based on the available geometry model) and always have a fixed orientation in space (e.g., via a zero-point clamping system). Adaptive behavior of such systems is not required or considered in the work that considers industrial use cases. The only exceptions are the works of Jing & Goh et al. (2018) and Wu & Lin et al. (2020), which consider a variable positioning of objects on conveyor belts but also assume the geometry models as given. Analytical planning approaches are therefore not applicable for two reasons. First, the availability of geometry models during inspection cannot be guaranteed, especially for independent remanufacturers. Second, the planning task in remanufacturing may vary so that variable ROI on the product has to be inspected. Analytical approaches have planning times in the seconds to minutes range, which leads to long cycle times in an industrial environment, limiting the overall system's profitability. This is since in most cases, multiple potential VP have to evaluated regarding their suitability to be selected as the next VP, which is computationally intensive. This is not the case for machine learning methods, as they implicitly learn to predict the next VP to be selected based on a given input. This calculation is usually in the sub-second range.

Deficit specific to SL approaches
In contrast to analytical methods, the analyzed machine learning methods offer the possibility of performing a real-time capable calculation of the next best VP even with varying planning tasks due to their short calculation time. This applies to the methods of both SL and RL. However, SL approaches have two major disadvantages. First, just like the analytical

approaches, most of the methods presented depend on the knowledge of a geometric model of the inspection object. Furthermore, these approaches are trained based on a training data set that must be generated in advance. In the application, such approaches therefore offer at most the possibility of performing just as well as the algorithm that generated the training pairs consisting of input data and the corresponding VP.

Deficit specific to RL approaches

In contrast, RL methods offer the possibility of achieving better performance compared to a human through self-learning behavior by trial and error. However, most of the approaches also assume knowledge of the inspection object and learn an inspection strategy specifically for each object category. Furthermore, a detailed analysis of existing works in the field of view planning using RL reveals that the approach for problem formulation and modeling varies greatly. Many RL solutions in existing works are strongly tailored to the specific problem. In particular, the modeling of the state and action space (input and output space) and the definition of the reward signal, which defines the agent's objective, vary and are widely limited. As in the case of SL, strong simplifications of the problem are made. The majority of existing works are simplified with respect to the input, e.g., by using voxelization (see Mendoza & Vasquez-Gomez et al. (2020), Vasquez-Gomez & Troncoso et al. (2021), Monica & Aleotti (2021), and Pan & Hu et al. (2022)) or the output, e.g., in the form of classification on predefined viewpoints (see Mendoza & Vasquez-Gomez et al. (2020), Monica & Aleotti (2021), Pan & Hu et al. (2022), and Zeng & Wen et al. (2020)). A general problem solution where point clouds as the direct unprocessed output of a 3D sensor are directly mapped to a sensor pose in continuous space without discretization by a learned RL inspection strategy 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.

General research deficit for autonomous view planning in remanufacturing

Overall, no approach to date addresses the specific challenges of automated initial visual inspection of used products in remanufacturing. As part of this thesis, the necessary inspection steps have been broken down into relevant sub-problems of VPP. It has been shown that due to the high variance in the used product condition of the returned used products and the associated uniqueness of the inspection process to be carried out, a high degree of flexibility of the inspection system and adaptivity of a planning algorithm to be used is required. For this task, RL offers a uniform methodological framework for solving the individual relevant sub-problems of VPP to realize an autonomous visual inspection in remanufacturing.

However, there are still deficits that need to be resolved, especially for the use case of remanufacturing.

1. **Research deficit 1:** *Missing evaluation of differently modeled RL agents*
   In the current state of research, there is no derivation and discussion of different modeling variants of the RL agent system, which consequently does not allow any statement to be made as to which modeling variants are best suited for the present subproblems of VPP.

2. **Research deficit 2:** *Dependence on available object geometries at system runtime*
   Existing approaches for determining sequential poses by means of RL currently only address the VPP with explicit knowledge of the object model (e.g., the triangle mesh, see Deinzer & Denzler et al. (2003), Korbach & Solbach et al. (2021), and Landgraf & Meese et al. (2021)). 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 an available three-dimensional object model or knowledge of the product type to specifically adapt the inspection routine to the present product at runtime.

3. **Research deficit 3:** *Missing investigation of varying planning tasks*
   The existing state-of-the-art approaches have not been evaluated with changing planning tasks (e.g., inspecting any ROI located on the object to be inspected or varying object categories). Overall, there is no systematic and general approach to the solution of the VPP (NBV, RPP and IPP) with consideration of the specific challenges of remanufacturing as a use case and RL as a solution method.

Table 3.1: State of research based on the derived elimination criteria

● Considered

◐ Partly considered

○ Not considered

			Problem formalization							Solution approach							Adaptivity												
			Use case				Inspection problem			Geometry model			Multicriteria optimization				Prior knowledge		Adap-tion		Input space				Output space			Object	
			Metrology	Object reconstruction	Object recognition	Remanufacturing	NBV	RPP	IPP	Model-free	Model-based-A	Model-based-G	Surface coverage	View plan length	Travel path length	Other	Integrated	Not integrated	Varying RoI	View plan adaptation	Sensor poses	Images (RGB and/or depth)	Occupancy/ Voxel grid	Point cloud	Predefined viewpoints	Continous	Continous with restrictions	One	Multiple
Solution approach - Solution algorithm		**Analytical approaches**																											
		Raffaeli, Mengoni & Germani et al. (2009-2013)	●	○	○	○	○	○	●	○	●	○	●	○	◐	○	●	○	○	○									
		Koutecký & Paloušek et al. (2016)	●	◐	○	○	○	●	○	○	●	○	●	○	◐	○	●	○	○	○									
		Wu & Lu et al. (2015)	●	◐	○	○	○	●	○	○	●	○	●	○	◐	○	●	○	○	○									
		Jing & Goh et al. (2018)	●	◐	○	○	○	●	○	○	●	○	●	●	●	○	●	○	○	●									
		Wu & Lin et al. (2020)	●	○	○	○	○	○	●	○	●	○	●	○	○	○	○	●	○	●									
		Machine learning approaches																											
	Supervised learning	Ashutosh & Kumar et al. (2022)	○	●	○	○	●	◐	○	●	○	○	●	○	○	○	●	○	○	○	○	●	○	○	●	○	○	○	●
		Monica & Aleotti et al. (2021)	○	●	○	○	●	◐	○	●	○	○	●	○	○	○	○	●	○	○	○	●	●	○	○	○	●	○	●
		Mendoza & Vasquez-Gomez et al. (2020)	○	●	○	○	●	◐	○	○	●	○	●	○	○	○	●	○	○	○	○	○	●	○	●	○	○	○	●
		Vasquez-Gomez & Troncoso et al. (2021)	○	●	○	○	●	◐	○	○	●	○	●	○	○	○	●	○	○	○	○	○	●	○	○	○	●	○	●
		Pan & Hu et al. (2022)	○	●	○	○	○	●	○	○	●	○	●	●	●	○	●	○	○	○	○	○	●	○	●	○	○	○	●
		Zeng & Zhao et al. (2020)	○	●	○	○	●	◐	○	●	○	○	●	○	○	○	●	○	○	○	●	○	○	●	●	○	○	○	●
	Reinforcement learning	Korbach & Solbach et al. (2021)	○	○	●	○	●	○	○	●	○	○	○	○	○	●	○	●	○	●	○	●	○	○	○	●	○	○	●
		Deinzer & Denzler et al. (2003)	○	○	●	○	●	○	○	●	○	○	○	○	○	●	●	○	○	●	●	○	○	○	○	○	●	●	○
		Devrim Kaba & Gokhan Uzunbas et al. (2017)	○	●	○	○	○	●	○	○	●	○	●	○	○	○	●	○	○	○	●	○	○	○	●	○	○	●	○
		Potapova & Artemov et al. (2020)	○	●	○	○	○	●	○	○	●	○	●	●	○	●	●	○	○	●	○	○	●	○	●	○	○	○	●
		Landgraf & Meese et al. (2020)	○	●	○	○	○	●	○	○	●	○	●	○	○	○	◐	◐	○	○	●	○	○	○	○	●	○	●	○
		Wang & Peng et al. (2023)	○	●	○	○	○	●	○	○	●	○	●	●	○	○	●	○	○	○	●	○	○	○	●	○	○	●	○
		Zeng & Zaenker et al. (2022)	○	●	○	○	○	●	○	●	○	○	●	○	○	○	○	●	○	●	●	○	○	○	○	○	●	●	○

4 Overview of the solution approach

Based on the research deficit derived in chapter 3, this chapter 4 outlines the approach developed in this thesis for the application of RL for autonomous view planning for visual inspection in remanufacturing.

Assumptions and boundary conditions
Before the solution approach is explicitly discussed in the following, the assumptions and boundary conditions made for this thesis are introduced and explained below.

1. **Product group under consideration:** The product group of starter motors will serve as an example in this thesis. The approach to be developed for autonomous view planning is to be applied to different variants of this product group and evaluated for these variants. Starter motors are real remanufacturing products that are available on the market in a large number of variants. The most common use case is considered where there is no prior knowledge of the product or its geometry model at the start of the inspection.

2. **Research setting:** An inspection station consisting of a robot-guided acquisition system and a clamping system mounted on a rotary table for clamping and subsequently inspecting a starter motor variant (inspection object) is considered the research environment. Upstream handling processes for clamping the inspection objects are not part of the research in this thesis. Similarly, the evaluation and decision-making process for accepting or rejecting the starter motors based on the data acquired at the inspection station is not the subject of this thesis. This is due to the fact that the task of detecting and evaluating defects is already a multi-layered and complex endeavor due to the variety of defects present in remanufacturing and requires independent research work.

3. **Subject of research:** Based on the fundamental research objective and the research deficits derived in section 1 and section 3 respectively, this thesis mainly focuses on the methodical development and implementation of a solution approach to realize autonomous robot-based data acquisition for visual inspection in remanufacturing. At the core of this research is the development, implementation, and evaluation of a software architecture for the control of the inspection system by a RL agent with a given hardware setup to solve the sub-problems of the VPP relevant for remanufacturing. It is assumed that the hardware structure fulfills the essential requirements for evaluating the approaches to be developed. Due to the prevailing procedure for training and evaluating approaches of RL in simulative environments, a detailed digital representation of the existing inspection station is set up in this thesis and used to train the RL agents.

Schematic explanation of the exemplary inspection procedure

An exemplary inspection process at the inspection station can proceed as follows. Initially, an arbitrary variant of a starter motor is clamped. First, the problem of the general inspection, i.e., the (a) overall inspection of the starter motor with full surface coverage, must be solved to identify all possible defects on the product. The RL agent must solve the RPP at inspection time without prior knowledge of the geometry of the starter motor variant in question. The RGB-D acquisition system mounted on the inspection system is used. The RGB and depth data captured with this acquisition system enable the detection and evaluation of defects on the starter motor (out of scope for this thesis and subject of further research). Furthermore, the data collected during the general inspection (RGB and depth data as well as point clouds) offer the possibility to create an information basis for the further planning procedure of the RL agent. On the one hand, this concerns the planning procedure within a problem-solving process (e.g., RPP or IPP). When considering the RPP as an inspection problem, the next VP can be selected based on the surface points detected so far in such a way that the surface of the starter motor not yet covered by the next VP is maximized (model-free inspection problem). On the other hand, it is also possible that the information acquired during the general inspection is processed to be used for the problem of the (b) individual inspection. A concrete example is creating a three-dimensional semantic model of the starter motor as an inspection object during the general inspection, which can then be used downstream for the individual inspection (model-based inspection problem). Not only does this three-dimensional semantic model contain geometry information of the starter motor, but it also contains information about where on this generated geometry model which ROI (components or defects to be inspected in more detail) are located that need to be inspected in more detail by means of the individual inspection.

Structure of the methodical solution approach of this thesis

Based on the schematic explanation of this thesis's solution approach, a methodical approach can be derived, which is presented in the following. The methodical solution approach is divided into three work packages to investigate the specific challenges of autonomous view planning for visual inspection in Remanufacturing. These three work packages correspond directly to the three main sections detailed in chapter 5. A visualization can be found in Figure 4.1. The main work packages are briefly explained below.

1. **Section 5.1:** In this section, the modeling of the inspection system is presented in detail. First, the process of generating a digital representation of the inspection station is discussed. Furthermore, the result of modeling the kinematic relationships at the inspection station is used to develop an approach for continuously reconstructing and updating a three-dimensional model of the inspection object using the three-dimensional

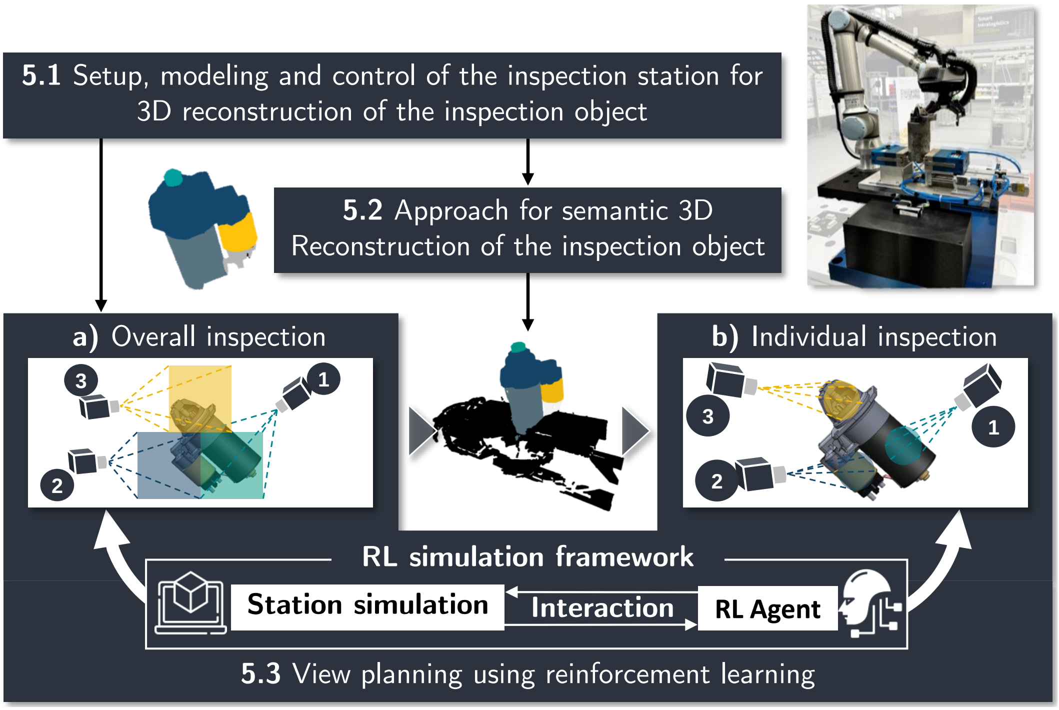

5.1 Setup, modeling and control of the inspection station for 3D reconstruction of the inspection object
5.2 Approach for semantic 3D Reconstruction of the inspection object
a) Overall inspection
b) Individual inspection
1
2
3
RL simulation framework
Station simulation
Interaction
RL Agent
5.3 View planning using reinforcement learning

Figure 4.1: Overview of the overall solution approach of this thesis.

data (point clouds) acquired at the inspection station during the acquisition process. This intermediate three-dimensional model can be used to further plan the current inspection process (e.g., **a)** the overall inspection) or as an input for a further inspection process (e.g. **b)** the individual inspection after the overall inspection).

2. **Section 5.2:** This section discusses how the approach developed in section 5.1 for the three-dimensional reconstruction of the inspection object can be extended to include semantic information. Specifically, a procedure is presented on reconstructing a semantic three-dimensional model with the help of semantic segmentation and/or instance segmentation methods using the superordinate approach of semantic mapping. In addition to geometric information, this model also contains information about the assignment of individual object surfaces to inspection-relevant product features as ROI (e.g., gear to be always inspected individually to detect broken gear teeth). These ROI might need to be inspected more closely, and the three-dimensional semantic model contains information on where on the inspection object they are located. This three-dimensional semantic model of the starter motor as an inspection object, generated for example in the overall inspection step, can then serve as input for the inspection process of the individual inspection.

3. **Section 5.3:** The concepts presented in section 5.3 form the core of this thesis. The results of the thesis presented in section 5.1 and section 5.2 are combined to en-

able and evaluate the RL based autonomous view planning for visual inspection in remanufacturing. This section’s center is the RL simulation framework, which has been conceptualized, implemented, and evaluated as part of this thesis. This framework uses the virtual representation of the inspection station generated in 5.1 to train RL agents to solve the subproblems of VPP considered in this thesis. The main research focus here is on the modularity of the RL simulation framework to generate different modeling variants of the RL agents and to evaluate their performance. The agent modeling must be designed in such a way that overall inspection, considered as RPP where initially no geometry information of the inspection object is available, as well as individual inspection, considered as IPP with the availability of a semantic three-dimensional model of the inspection object, can be handled equally.

5 Autonomous view planning for visual inspection in Remanufacturing

The individual sections already introduced in the chapter 4 on the methodical realization of the solution approach are discussed below in detail. A detailed explanation of the approach to solving the respective problems in the individual sections and the structuring of the approach can be found at the beginning of each section.

5.1 Setup, modeling, and control of the inspection station and 3D reconstruction of the inspection object

This section first addresses the hardware setup of the inspection station, the acquisition process, and the data obtained by an acquisition of the acquisition system (section 5.1.1). This is followed by a description of the virtual representation of the inspection station (section 5.1.2), including all kinematic relationships (section 5.1.3). Using the virtual model of the inspection station and the kinematic relationships, virtual trajectory planning of the robot and 3D reconstruction of a starter motor during an acquisition process becomes possible (section 5.1.4). Furthermore, the virtual representation of the inspection station is used in section 5.3 to build up the RL simulation framework. A visual overview is found in Figure 5.1

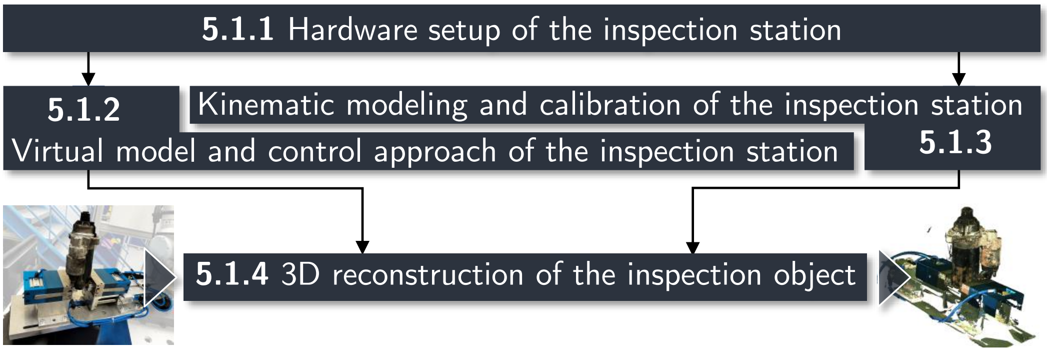

5.1.1 Hardware setup of the inspection station
5.1.2
Virtual model and control approach of the inspection station
Kinematic modeling and calibration of the inspection station
5.1.3
5.1.4 3D reconstruction of the inspection object

Figure 5.1: Overview of the approach to trajectory planning using a virtual station model and 3D reconstruction of the inspection object.

5.1.1 Hardware setup of the inspection station and acquisition process

Figure 5.2 depicts the hardware setup of the inspection station on the left side (subfigure a)). It comprises a collaborative *UR10e* robot from *Universal Robots* [1] with a gripper and an

[1] Link to the product website:
https://www.universal-robots.com/products/ur10-robot/
accessed: 04.06.2024

acquisition system on its end effector. The gripper handles starter motors, the exemplary products considered in this thesis. The gripper is the *2F140* from *Robotiq* [1]. The acquisition system is a *Zivid One+ S*. The technical data of the *Zivid One+ S* were extracted from the technical datasheet[2] and can be found in Table 5.1. A key advantage of this acquisition system is the high-resolution acquisition of color-coded point clouds.

Table 5.1: Technical data of the acquisition system Zivid One+ S extracted from the technical datasheet.

Characteristic	**Value or Description**
Resolution	1920 x 1200 Pixels
Output	3D (XYZ) & Color (RGB)
Optimal working distance (OWD)	500 mm
Point precision	25 μm
Field of view	350 mm x 220 mm at OWD
Spatial resolution	0.18 mm at OWD

Each point $\mathbf{P}_{t,i}^{A}$ of the acquired point cloud $\mathbf{PC}_{t}^{A} = [\mathbf{X}_{t}^{A}, \mathbf{Y}_{t}^{A}, \mathbf{Z}_{t}^{A}, \mathbf{R}_{t}^{A}, \mathbf{G}_{t}^{A}, \mathbf{B}_{t}^{A}]$ at time t of acquisition is represented by three-dimensional coordinates $\mathbf{P}_{t,i}^{A} = [X_{t,i}^{A}, Y_{t,i}^{A}, Z_{t,i}^{A}]$ (points in space) as well as color information $\mathbf{C}_{t,i}^{A} = [R_{t,i}^{A}, G_{t,i}^{A}, B_{t,i}^{A}]$. The set of three-dimensional points $\mathbf{P}_{t}^{A}$ contained in $\mathbf{PC}_{t}^{A}$ is initially represented in the acquisition systems' coordinate system C_A. $\mathbf{PC}_{t}^{A}$ is extracted from RGB-D data, i.e. an acquired color image $I_{C,t}$ and a depth image $I_{D,t}$. $I_{C,t}$, $I_{D,t}$ as well as $\mathbf{PC}_{t}^{A}$ are directly output by the software of the *Zivid One+ S*, where the indices of the color image directly correspond to the indices of $\mathbf{P}_{t}^{A}$. Thus, a three-dimensional coordinate can be assigned to each pixel in $I_{C,t}$. These relationships are also shown in Figure 5.2, where each point of $\mathbf{PC}_{t}^{A}$ also contains the color information of $I_{C,t}$.

To perform an acquisition, the acquisition system has to be parameterized concerning its exposure time, aperture, gain, and the illumination intensity of the projector. The predefined capture assistant of the *Zivid One+ S* software defines these parameters of the individual acquisitions. Given a maximum capture time, this capture assistant combines acquisitions of different parameters optimized by the system to provide a High Dynamic Range (HDR)

[1] Link to the product website:
https://robotiq.com/products/2f85-140-adaptive-robot-gripper
accessed: 04.06.2024

[2] Link of datasheet:
https://www.zivid.com/hubfs/Zivid%20One%20Plus%20Datasheet%20(1).pdf?hsCtaTracking=f6b119a4-7444-47ac-83a3-44d4840aa429%7Ccabb4557-32f2-4851-9612-99fe7552235b
accessed: 24.08.2023

image[1]. Ideally, all image pixels are exposed optimally in a HDR image. Thus, these HDR images cover a higher dynamic range than individual exposures (Reinhard & Heidrich et al. 2010, P. 148).

The starter motors are clamped by a specially developed clamping system for data acquisition. So-called needle pads from *Matrix GmbH*[2], which can be locked pneumatically, are used. This enables them to clamp any geometries and thus cover the wide range of variants and generations of starter motors. The clamping system is mounted on a rotary table to enable the rotation of the starter motor. The rotary table and the clamping system can be controlled via a serial interface using the high-level programming language Python. The programming of the functionalities (e.g., relative rotation of the rotary table by a certain angle or opening and closing of the clamping system) has been realized in this way. These functionalities are detailed in section 5.1.2. A separate control system has been developed for the clamping system with the aid of an Arduino microcontroller. Further information on the clamping system's design, structure, and functionality and the rotary table can be found in the master thesis of A_Schnaberich 2022, supervised by the author of this thesis. Using the proposed setup, the inspection system can perform multiple sequential acquisitions of the starter motor utilizing varying end-effector poses and, thereby, the acquisition system and varying rotations of the clamped starter motor.

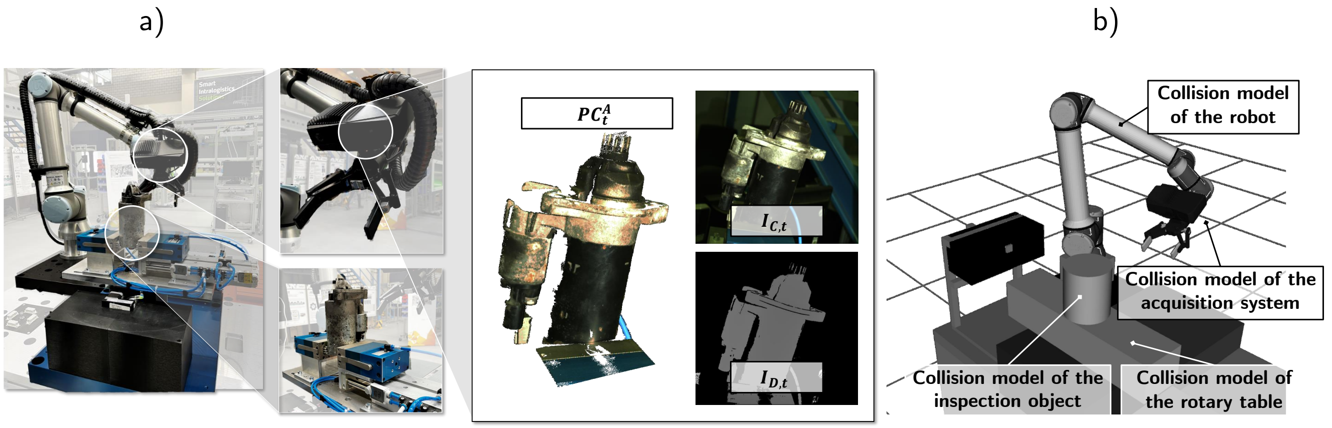

a)
b)
PC_t^A
$I_{C,t}$
$I_{D,t}$
Collision model of the robot
Collision model of the acquisition system
Collision model of the inspection object
Collision model of the rotary table

Figure 5.2: Illustration of the inspection station. a) shows the hardware structure of the inspection station, including exemplary output data of the acquisition system. b) shows the virtual station model/ planning scene of MoveIt, including all collision models.

[1] Link to documentation:
https://support.zivid.com/en/latest/academy/camera/capture-tutorial.html#capture-assistant
accessed: 28.08.2023

[2] Link to company website:
https://www.matrix-innovations.com/
accessed: 04.06.2024

5.1.2 Virtual model and control approach of the inspection station

The inspection station consisting of the robot, acquisition system, and rotary clamping system is controlled via the ROS (Robot Operating System) framework (Stanford Artificial Intelligence Laboratory et al. 2018). The robot's motion planning is realized with the help of the *MoveIt* motion planning framework (Coleman & Sucan et al. 2014).

ROS is an open-source software framework for building and controlling robotic systems. It follows a modular architecture, where software components, known as nodes, can communicate via data transfer on topics or provide specific services to other nodes. Nodes are individual software components in ROS, possibly written in multiple programming languages, that perform specific tasks. (Stanford Artificial Intelligence Laboratory et al. 2018)

MoveIt is a widely used motion planning framework built on ROS. It provides a set of libraries, tools, and interfaces to enable robot arm and manipulation planning, control, and execution. *MoveIt* simplifies planning and visualizing motion trajectories, considering specified constraints (e.g., joint constraints) and collision avoidance. (Coleman & Sucan et al. 2014)

The rotary table and clamping system functionalities are implemented in ROS nodes using Python and specific services. The advantage of ROS and its open-source policy is evident in the fact that packages containing nodes to operate their hardware are already provided by the manufacturers of and for the *Zivid One+ S* as well as the *UR10e* offers extensive software packages for its robots to integrate them into ROS but also to enable the interaction with *MoveIt*. These have also been used and integrated to set up the control system for the inspection station.

An overview of the nodes and services available and implemented for the control of the inspection station and a description of their functionality can be found in Table 5.2. The clamping system is generally controlled via opening and closing services, and services provide rotary table control for performing a reference drive and relative and absolute rotations. The Zivid node provides services for finding the optimal parameter configuration of a HDR acquisition and performing the acquisition. The acquired data ($I_{C,t}$, $I_{D,t}$, and $\mathbf{PC}_t^A$) are published on their respective topics and can be accessed by other code, e.g., self-written scripts. The *move_group* node of *MoveIt* enables the planning of robot trajectories and their execution with the respective services. Last, the *UR_driver* node provides information on the current pose $\mathbf{p}_t^{A/R}$ of the acquisition system in relation to the robot base via the *Robot Pose* topic.

A simplified visual overview of these relationships is shown in Figure 5.3. An inspection script can use the available services and topics to control the individual components of the inspection station. The Table also shows that all nodes are embedded in the so-called

Table 5.2: Overview of nodes and their provided services used in the control of the inspection station

Node: *Pneumatics*

Service or Topic	Description
Service *Open*	Request opening the clamping system.
Service *Close*	Request closing the clamping system.

Node: *Rotary Table*

Service or Topic	Description
Service *Reference Drive*	Request performing a reference drive where the goal state is the rotary table in zero position.
Service *Rotate Absolute*	Request a rotation to an absolute position expressed in degrees.
Service *Rotate Relative*	Request a rotation in degrees relative to the current rotation of the rotary table.

Node: *Zivid*

Service or Topic	Description
Service *Suggested Settings*	Request the optimal parameter configuration for HDR acquisition of a given scene (test acquisitions are performed by the *Zivid One+ S* to obtain the optimal parameter configuration.
Service *Capture*	Request an acquisition of the *Zivid One+ S* with specified parameter configuration.
Topic *RGB Image*	After an acquisition at timestep t, the Zivid node publishes the RGB image I_C on the *RGB Image* topic.
Topic *Depth Image*	After an acquisition at timestep t, the Zivid node publishes the depth image I_D on the *Depth Image* topic.
Topic *XYZRGB*	After an acquisition at timestep t, the Zivid node publishes the color-coded point cloud data $\mathbf{PC}_t^A$ on the *XYZRGB* topic.

Node: *MoveIt move_group*

Service or Topic	Description
Service *Plan Path*	Request the planning of a path of the specified robot from a given start state (robots' joint state) to a goal state (robots' joint state or pose $\mathbf{p}_t^{A/R}$ in space) with the possibility of including intermediate states.
Service *Execute Path*	Request executing a planned path on the specified robot.

Node: *UR_driver*

Service or Topic	Description
Topic *Robot Pose*	The node provided by *Universal Robots* for the used robot *UR10e* publishes the robots pose $\mathbf{p}_t^{A/R}$ on the topic *Robot Pose*.

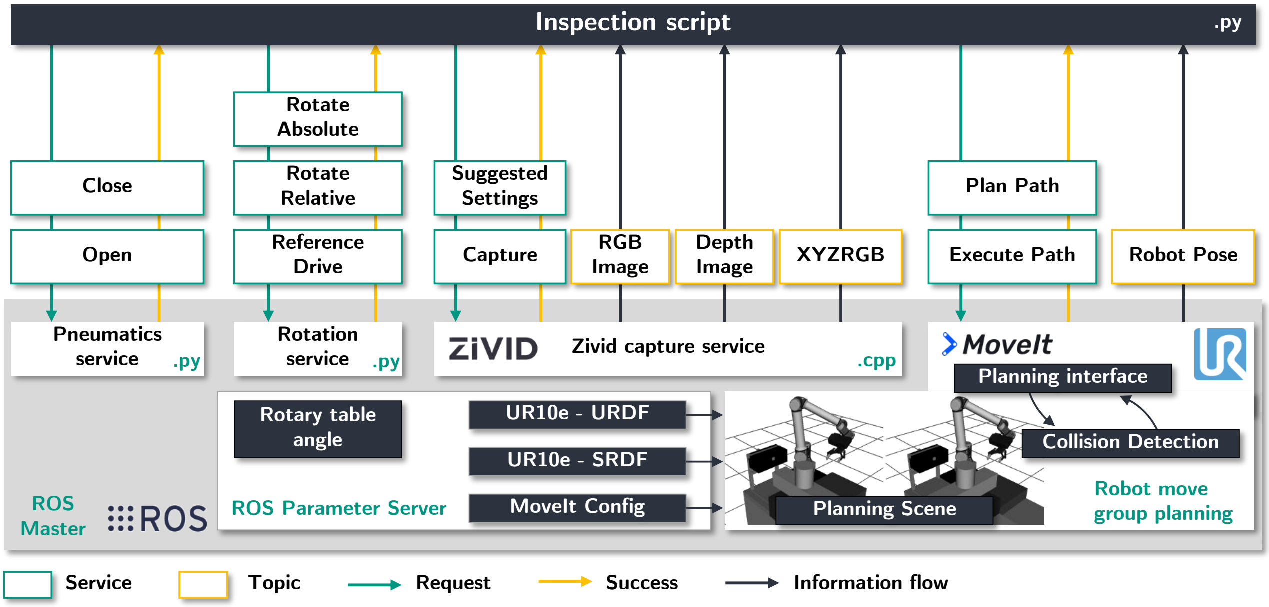


Figure 5.3: Simplified visualization of how an inspection script interacts with the nodes provided by the ROS master via services and topics and how planning tasks for trajectory planning are accomplished using MoveIt.

ROS master. This coordinates the communication between individual nodes and provides a parameter server in which a wide range of information, such as the current angle of the rotary table, but also descriptions of the robot used or configuration files are stored. This information is then available to the nodes and can be used by them. An example is *MoveIt*, which uses the *URDF* (Universal Robot Description Format) and *SRDF* (Semantic Robot Description Format) files of the *UR10e* used from the parameter server to carry out collision-free path planning. For collision-free path planning with the *Planning Interface*, *MoveIt* also uses the so-called *Planning Scene*. This virtual representation of the objects present in reality is defined in advance using CAD files and primitives (cuboids, spheres, etc.) to build up the planning scene. A visualization of the planning scene built in this thesis can be seen on the right side (b)) in Figure 5.2.

For further information on the software implementation for clamping the starter motor, controlling the robot, and acquiring RGB-D data at the station, please refer to A_Scheiger 2022, a master thesis supervised by the author of this thesis.

5.1.3 Kinematic modeling and calibration of the inspection station

Natively, *MoveIt* only offers the option of planning with defined coordinate systems (e.g., that of the end effector) of the robot used. Using an acquisition system mounted on a robot, however, it is essential to plan with the coordinate system of the acquisition system to align it with the object to be inspected. In addition, images from different poses of the acquisition system and the rotary table, and thus of the inspection object, must be correlated to each other to enable the three-dimensional modeling and, thus, reconstruction of the inspection object. A kinematic description of the entire system is therefore necessary.

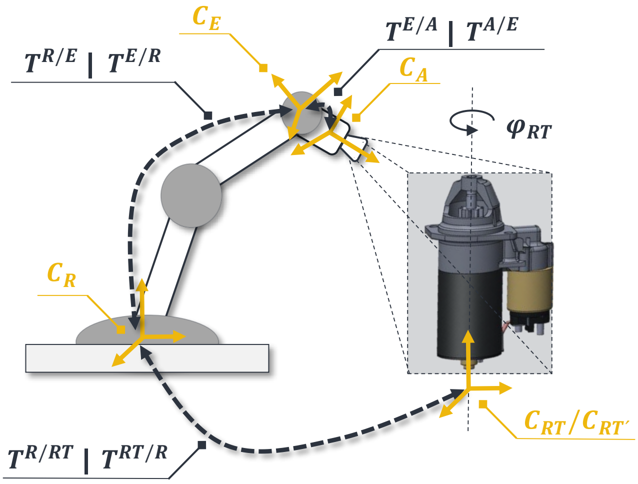


Figure 5.4: Simplified depiction of the inspection station and its kinematic relationships.

For the description of the kinematic relationships of the inspection station, the coordinate systems and transformation matrices are shown in Figure 5.4. First, all relevant coordinate systems and the transformation matrices that describe the relationships of the coordinate systems to one another are introduced. Second, the procedure for transforming point clouds between the introduced coordinate systems using the established kinematic relationships is detailed. Last, the procedure for determining the transformation matrices is laid out.

Coordinate systems

During an acquisition, the robot's end-effector is in a particular pose. The acquisition system with coordinate system C_A is mounted on the robot end-effector with the coordinate system C_E. The robot base coordinate system is denoted with C_R. C_R is a static coordinate system that never changes its translation or orientation in space. The rotary table has two coordinate systems. The first is C_{RT}, which is also static. The second is the rotated coordinate system $C_{RT'}$, which has the same origin as C_{RT} but is rotated in relation to C_{RT} with some absolute rotation angle φ_{RT}. To enable the three-dimensional reconstruction of the inspection object by continuously fusing acquired point clouds, each point cloud has to be expressed in the same reference coordinate system. $C_{RT'}$ is chosen as an intermediate reference system to registrate and fuse the acquired point clouds during acquisition process.

Transformation matrices between coordinate systems

The static global coordinate system C_R, which is located in the center of the robot's base, can be described in relation to the coordinate system C_E of the end-effector via the homogeneous transformation matrix $\mathbf{T}^{R/E}$. The kinematic relationship between the camera coordinate system C_A, which is defined by the focal point of the acquisition system's camera sensor, and the coordinate system of the end-effector can be described via the homogeneous transformation matrix $\mathbf{T}^{E/A}$. In addition to the kinematic relation of the acquisition system concerning the global coordinate system, the kinematic relation of the global coordinate system C_R to the coordinate system of the rotary table C_{RT} can be described via the homogeneous transformation matrix $\mathbf{T}^{R/RT}$. Additionally, the transformation between C_{RT} and $C_{RT'}$ due to rotation of the rotary table with absolute rotation angle φ_{RT} is described with the Matrix $\mathbf{T}^{RT/RT'}$. $\mathbf{T}^{RT/RT'}$ is a homogeneous transformation matrix with no translational and just rotational components since the rotary table only has one rotational degree of freedom.

Coordinate transformation of acquired point clouds using kinematic relationships
The rules for transforming the acquired point clouds between the mentioned coordinate systems can be deduced assuming the knowledge of all transformation matrices above. Each point $\mathbf{P}_{t,i}^{A} = [X_{t,i}^{A}, Y_{t,i}^{A}, Z_{t,i}^{A}]$ of an acquired point cloud $\mathbf{PC}_t^A$ is initially represented in the acquisition system's coordinate system C_A. For transforming the point cloud to the intermediate reference coordinate system $C_{RT'}$, the transformation homogeneous transformation matrix $\mathbf{T}^{A/RT'}$ is used. The computation of the coordinate transformation is performed for all points of the point cloud according to equation 5.1 to yield points in the reference coordinate system $\mathbf{P}_{t,i}^{RT'} = [X_{t,i}^{RT'}, Y_{t,i}^{RT'}, Z_{t,i}^{RT'}]$.

$$\mathbf{P}_{t,i}^{RT'} = \mathbf{T}^{A/RT'}\,\mathbf{P}_{t,i}^{A} \quad \forall\,\mathbf{P}_{t,i}^{A} \in \mathbf{PC}_t^{A} \tag{5.1}$$

This results in a transformed point cloud $\mathbf{PC}_t^{RT'} = [\mathbf{X}_t^{RT'}, \mathbf{Y}_t^{RT'}, \mathbf{Z}_t^{RT'}, \mathbf{R}_t^{RT'}, \mathbf{G}_t^{RT'}, \mathbf{B}_t^{RT'}]$ where now the points are represented in the rotating coordinate system of the rotary table. At the same time, the color values remain unchanged ($\mathbf{C}_t^{A} = \mathbf{C}_t^{RT'} = [\mathbf{R}_t^{RT'}, \mathbf{G}_t^{RT'}, \mathbf{B}_t^{RT'}]$). A detailed mathematical derivation of the components of the homogeneous transformation matrix $\mathbf{T}^{A/RT'}$ can be found in the Appendix A4.

5.1.4 3D reconstruction of the inspection object

Based on the kinematic relationships derived in section 5.1.3, a workflow can be defined that allows three-dimensional modeling of the inspection object. This workflow is based on the reconstruction cycle of Scott & Roth et al. (2003) and is shown in Figure 5.5. It is detailed in the following.

- **View generation (1) and scanning (2):** Regardless of whether the VPP solution approach is a model-based or non-model-based approach, the robot is moved with the pose of the acquisition system being $\mathbf{p}_t^{A/R}$. The rotary table is rotated with a relative rotation angle $\Delta\varphi_{RT,t}$ and the acquisition system acquires the clamped object with

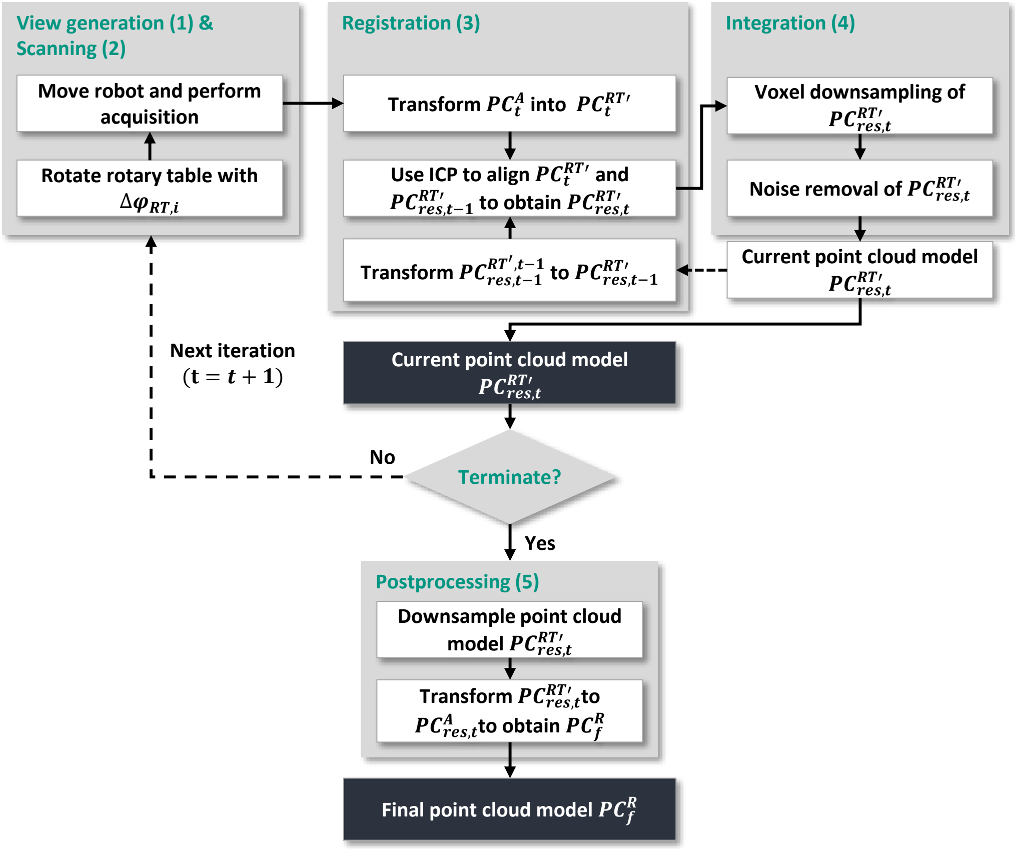

View generation (1) & Scanning (2)
Move robot and perform acquisition
Rotate rotary table with $\Delta\varphi_{RT,i}$
Registration (3)
Transform PC_t^A into $PC_t^{RT'}$
Use ICP to align $PC_t^{RT'}$ and $PC_{res,t-1}^{RT'}$ to obtain $PC_{res,t}^{RT'}$
Transform $PC_{res,t-1}^{RT',t-1}$ to $PC_{res,t-1}^{RT'}$
Integration (4)
Voxel downsampling of $PC_{res,t}^{RT'}$
Noise removal of $PC_{res,t}^{RT'}$
Current point cloud model $PC_{res,t}^{RT'}$
Next iteration $(t = t + 1)$
Current point cloud model $PC_{res,t}^{RT'}$
No
Terminate?
Yes
Postprocessing (5)
Downsample point cloud model $PC_{res,t}^{RT'}$
Transform $PC_{res,t}^{RT'}$ to $PC_{res,t}^{A}$ to obtain PC_f^R
Final point cloud model PC_f^R

Figure 5.5: Flowchart for 3D reconstruction at the inspection station, following the formalization of Scott & Roth et al. (2003).

suggested HDR settings. The result is a point cloud $\mathbf{PC}_t^A$ represented in the coordinate system C_A.

- **Registration (3):** The acquired point cloud $\mathbf{PC}_t^A$ is transformed into the coordinate system $C_{RT'}$ using equation 5.1 to yield $\mathbf{PC}_t^{RT'}$. The current point cloud model $\mathbf{PC}_{res,t-1}^{RT',t-1}$ which combines the point clouds from previous steps ($\mathbf{PC}_0^{RT'}, \ldots, \mathbf{PC}_{t-1}^{RT'}$) is still represented in the moving coordinate system $C_{RT',t-1}$ of the rotary table of the previous acquisition step. It is therefore also transformed into the moving coordinate system $C_{RT'}$ to account for the relative rotary table rotation $\Delta\varphi_{RT,t}$, yielding $\mathbf{PC}_{res,t-1}^{RT'}$. This is achieved by calculating an intermediary homogeneous transformation matrix $\mathbf{T}(R_z(\Delta\varphi_{RT,t}))^{RT'_{t-1}/RT'_t}$ around the z-axis of $C_{RT',t-1}$ that enables calculation of $\mathbf{PC}_{res,t-1}^{RT'} = \mathbf{T}(R_z(\Delta\varphi_{RT,t}))^{RT'_{t-1}/RT'_t}\ \mathbf{PC}_{res,t-1}^{RT',t-1}$. $\mathbf{PC}_t^{RT'}$ and $\mathbf{PC}_{res,t-1}^{RT'}$ are now expressed in the same coordinate system C_{RT} and can be registered (e.g., using an *iterative closest point* (ICP) algorithm). Registration algorithms iteratively estimate a transformation (translation and rotation) to optimally align $\mathbf{PC}_t^{RT'}$ with $\mathbf{PC}_{res,t-1}^{RT'}$. The ICP is used in this thesis to account for various geometric inaccuracies in the system, such as robot positioning, rotary table positioning, and calibration errors. By using the result of the ICP, the transformed point cloud $\mathbf{PC}_{t,icp}^{RT'}$ and $\mathbf{PC}_{res,t-1}^{RT'}$ are merged by stacking to obtain $\mathbf{PC}_{res,t}^{RT'}$.

- **Integration (4):** Voxel downsampling with predefined functions[1] provided by the python package *Open3D* are applied to the current point cloud model $\mathbf{PC}_{res,t}^{RT'}$. Voxel downsampling provides a point cloud with equal density distribution and enables reducing the size of a point cloud to a fixed number of points. The downsampled point cloud model is filtered to remove noise. Predefined filtering functions[2] from *Open3D* are used.

- **Postprocessing (5):** In postprocessing, the three-dimensional model is ultimately processed after acquisition. Although downsampling is already carried out during the continuous model generation of the three-dimensional inspection object, it may be necessary to repeat the downsampling to an even smaller point cloud size after the finalization of the acquisition process. Furthermore, the three-dimensional model $\mathbf{PC}_{res,t}^{RT'}$ is ultimately transformed into the static robot coordinate system C_R to obtain the final three-dimensional model of the inspection object $\mathbf{PC}_f^R = \mathbf{PC}_{res,t}^R$. An optional postprocessing step not conducted in this thesis is a surface reconstruction (e.g., meshing).

5.2 Approach for semantic 3D reconstruction of the inspection object

In the previous section 5.1, the structure and modeling of the inspection station and the approach to the three-dimensional reconstruction of the starter motor as an inspection object have been discussed in detail. This section now deals with how the derived three-dimensional reconstruction approach can be enriched with semantic information. This is done with the help of the concept of semantic mapping and the use of deep learning methods for semantic and/or instance segmentation. To realize a semantic 3D reconstruction approach using segmentation, the generation of the data set and its preprocessing are first discussed (section 5.2.1). This is followed by the description of segmentation approaches (section 5.2.2) and the derivation of an approach for determining the uncertainty of the segmentation result (section 5.2.3). The section concludes with the approach for 3D reconstruction of the inspection object, extended by the component of semantic segmentation (section 5.2.4). It should be noted that the approach to semantic 3D reconstruction in this work is carried out using the example of segmenting starter motor components. The methodology can be directly applied

[1] Link to voxel downsampling documentation in *Open3D*:
http://www.open3d.org/docs/0.6.0/python_api/open3d.geometry.voxel_down_sample.html
accessed: 28.08.2023

[2] Link to noise removal documentation in *Open3D:*
http://www.open3d.org/docs/latest/tutorial/Advanced/pointcloud_outlier_removal.html
accessed: 28.08.2023

to segmenting defects, whereby in this case individual defects that require more detailed inspection must be segmented instead of the motor components. A visual overview is found in Figure 5.6

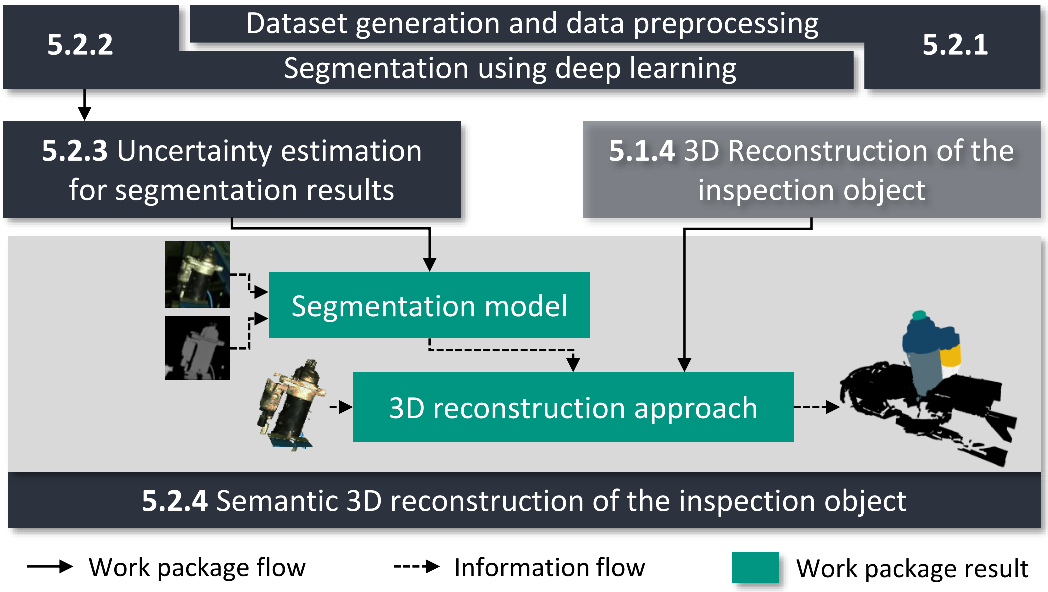

5.2.2
Dataset generation and data preprocessing
Segmentation using deep learning
5.2.1
5.2.3 Uncertainty estimation for segmentation results
5.1.4 3D Reconstruction of the inspection object
Segmentation model
3D reconstruction approach
5.2.4 Semantic 3D reconstruction of the inspection object
Work package flow
Information flow
Work package result

Figure 5.6: Overview of the approach for semantic 3D reconstruction of the inspection object

5.2.1 Dataset generation and data preprocessing

The following section details the process of dataset generation and data preprocessing. In the remainder of this section, the "time" index t for color images $I_{C,t}$ and depth images $I_{D,t}$ is omitted. This is because the acquisition time does not matter for training a segmentation model.

Data acquisition

A suitable dataset containing training and test data is necessary to train deep-learning semantic and instance segmentation models in a supervised manner. Most deep-learning semantic segmentation approaches use images as input to obtain semantic segmentation results. However, approaches using additional depth images exist. Therefore, a training dataset consisting of HDR color images I_C and depth images I_D are acquired at the inspection station. To acquire a suitable dataset for training and testing the deep learning semantic segmentation models, 42 different starter motors returning from a usage phase are used. These are fixed using the clamping system. For each starter motor, 15 poses of the acquisition system and, therefore, coordinate system C_A and varying rotary table rotation are used to acquire I_C and I_D. Due to the stochastic output of the adaptive view planning, random poses of the acquisition system and the rotary table are sampled in a permissible value range.

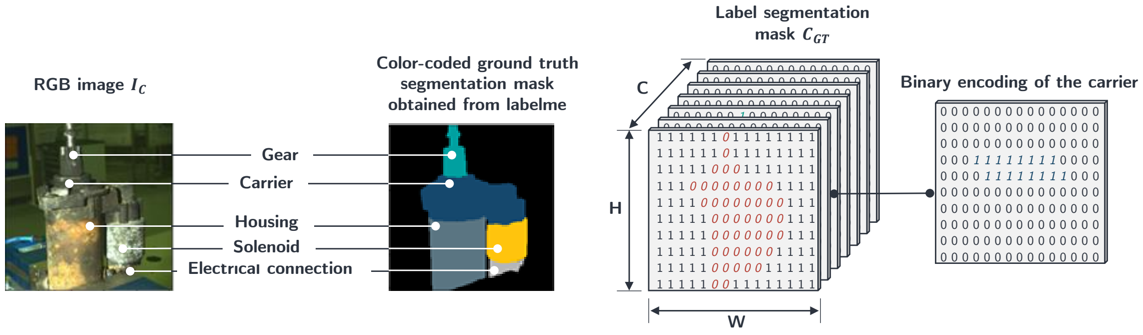
The positions of the acquisition system are sampled on a spherical surface in spherical coordinates with azimuth angle φ_A ($\varphi_A \in$ [$45°$, $90°$]) and polar angle θ_A ($\theta_A \in$ [$45°$, $90°$]) and varying distance r_A ($r_A \in$ [400 mm, 600 mm]) from the center of the sphere. The center of the sphere is the center of the coordinate system of the rotary table C_{RT}. The coordinates φ_A, θ_A and r_A are then transformed to cartesian coordinates x_A, y_A and z_A. The orientation of C_A is represented by Euler angles α_A, β_A, and γ_A and is calculated so that the acquisition system always faces the sphere's center. Offsets $\Delta\alpha_A$, $\Delta\beta_A$, $\Delta\gamma_A$ ($\Delta\alpha_A \in$ [$-10°$, $10°$], $\Delta\beta_A \in$ [$-10°$, $10°$], $\Delta\gamma_A \in$ [$-10°$, $10°$]) are added to $\alpha_A, \beta_A, \gamma_A$ respectively to increase image variety. Due to the pose sampling on a spherical surface and to increase the variety even further, the rotary table was rotated to a random absolute angle φ_{RT} ($\varphi_{RT} \in$ [$0°$, $360°$)) before each acquisition. This process ensures that the starter motors are acquired from different perspectives and under different lighting conditions due to differing reflections caused by varying poses of the acquisition and rotary system. After sorting out unusable RGB and depth images due to the incomplete acquisition of starter motors in some images, this resulted in 584 distinctive RGB and depth images. In addition to the RGB and depth images I_C and I_D, the point clouds $\mathbf{PC}_t^A$ acquired were saved. Furthermore, the poses $\mathbf{p}_t^{A/R}$ of the acquisition system and the homogeneous transformations $\mathbf{T}^{A/R}$ in relation to the robot base coordinate system as well as absolute rotation angles φ_{RT} of the rotary table were also saved for each acquisition.

Label mask generation

To enable supervised segmentation using Deep Learning, a segmentation label mask C_{GT} in the form of a labeled class image is necessary. Manual labeling of components, which are different classes in the images, is performed using the graphical annotation tool *labelme*[1]. According to Füvesi & Kovács et al. (2010), the structure of a starter motor consists, on a higher level, of the four main components: Housing, Solenoid, Carrier, and Gear (Füvesi & Kovács et al. 2010, P. 2). In this thesis, the electrical connection is additionally introduced as a class to be segmented. Using the tool *labelme*, different image segments are first encoded with different colors. This color-coded ground truth must be processed further depending on the segmentation method.

For semantic segmentation tasks, it is common to transform the color encoding into a so-called label segmentation mask C_{GT} with the dimensions $H \times W \times C$. Thereby, H corresponds to the image height, W to the image width, and C to the maximum number of possible classes, including the background. Each pixel is thus encoded one-hot, with the non-zero entry at the index to which class the pixel belongs. An exemplary color encoding of a starter motor with

[1] Link to tool:
https://github.com/wkentaro/labelme
accessed: 28.08.2023


RGB image I_C
Gear
Carrier
Housing
Solenoid
Electrical connection
Color-coded ground truth segmentation mask obtained from labelme
Label segmentation mask C_{GT}
C
H
W
Binary encoding of the carrier

```
1 1 1 1 1 1 0 1 1 1 1 1 1 1 1
1 1 1 1 1 1 0 1 1 1 1 1 1 1 1
1 1 1 1 1 0 0 0 1 1 1 1 1 1 1
1 1 1 0 0 0 0 0 0 0 0 1 1 1 1
1 1 1 1 0 0 0 0 0 0 0 0 1 1 1
1 1 1 1 1 0 0 0 0 0 0 0 1 1 1
1 1 1 1 1 0 0 0 0 0 0 0 1 1 1
1 1 1 1 1 0 0 0 0 0 0 1 1 1 1
1 1 1 1 1 0 0 0 0 0 1 1 1 1 1
1 1 1 1 1 0 0 1 1 1 1 1 1 1 1
```

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 1 1 1 1 1 1 1 0 0 0 0
0 0 0 0 1 1 1 1 1 1 1 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Figure 5.7: Visual representation of the starter motor and all components to be segmented and the label segmentation mask C_{GT} derived from the color-coded ground truth segmentation mask.

all relevant components, as well as the label segmentation mask of the image and a binary coding of the carrier, can be found in Figure 5.7.

The resulting label segmentation mask can be used directly as a ground truth label when using semantic segmentation methods. For use in instance segmentation methods, so-called *bounding boxes* (BB) and features describing these rectangular bounding boxes (corner positions and area) must also be extracted for each object present in the image. Furthermore, a binary segmentation mask is extracted based on the label segmentation mask and the knowledge of the position of the BB. An illustration can be found in Figure 5.8.

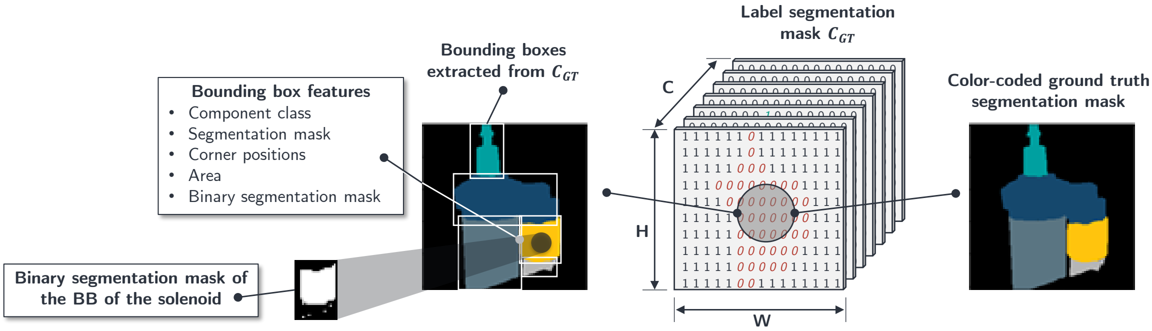

Bounding box features
• Component class
• Segmentation mask
• Corner positions
• Area
• Binary segmentation mask
Bounding boxes extracted from C_{GT}
Label segmentation mask C_{GT}
Color-coded ground truth segmentation mask
Binary segmentation mask of the BB of the solenoid
C
H
W

Figure 5.8: Visual representation of the BB features and segmentation mask of the individual components derived from the label segmentation mask C_{GT}

Image data preprocessing

Due to the specific acquisition properties of the acquisition system used, the resulting color images are underexposed. Preliminary experiments in a master thesis supervised by the author of this thesis have shown the benefit of brightness increase prior to segmentation (A_Hollinger 2022, P. 66). Therefore, brightness is increased by a fixed value for all color images I_C of the acquired data.

The acquired image data of the *Zivid One+ S* have a 1920×1200 pixels resolution. Segmentation algorithms usually operate on square image data. Thus, the aspect ratio is adjusted in the next step. Cropping is the first traditional approach to resize the image data without introducing distortion into the image (Vaquero & Turk et al. 2010, P. 5). Another way is padding the original image with black pixels to obtain the final output resolution (Vaquero & Turk et al. 2010, P. 5). Both of these methods are applied in this thesis. 50% of the image data are cropped and 50% are padded. For cropping the color and depth images, it is assumed that, due to the orientation of the acquisition system to the origin of the clamping system, the object is located approximately in the center of the acquired color and depth images. Therefore, a rectangular window is placed in the center of the original color and depth images to obtain square image data with size 1200×1200. For padding, the color and depth images are padded with black pixels of size 1920×360 at the top and bottom to obtain square image data with size 1920×1920. The resulting square image data can then be resized to arbitrary sizes using algorithms such as nearest neighbor interpolation. This results in the preprocessed dataset of images I_C^*, I_D^* and C_{GT}^*

5.2.2 Semantic and instance segmentation with deep learning

The following section introduces the data augmentation and standardization carried out during the training of the segmentation networks. The network architectures and the training procedure are presented, considering the integration of depth images. In addition, transfer learning and the backbones used for feature extraction of the network architectures used are discussed.

Data augmentation and data standardization
Due to the manual effort involved in labeling, the generated dataset is small compared to existing benchmark datasets for semantic segmentation. To artificially increase the size of the data set, increase the amount of information extracted from the dataset, and thus reduce overfitting, data augmentation is standard (Shorten & Khoshgoftaar 2019, P. 3). The successful use of data augmentation in a semantic segmentation with a small dataset has been shown, for example, by Ronneberger & Fischer et al. (2015) using a U-Net. In this thesis, augmentation techniques based on basic image manipulations, which are described in the work of Shorten & Khoshgoftaar (2019), are applied (Shorten & Khoshgoftaar 2019, P. 7, ff.). These are geometric transformations, color transformations, and filtering operations. Geometric transformations include operations such as flipping, rotation, scaling, translation, and cropping of the images. Blank spaces created by geometric transformations are filled with black pixels. Color transformations include changes in brightness, contrast, and color saturation as well as modifications of the images in various color domains (e.g., RGB color domain). In addition, other augmentation methods are often used to increase noise, enhance

contrast (for example, using *Contrast Limited Adaptive Histogram Equalization* (CLAHE)), or induce blurring.

Preliminary work has shown that the learning success and the resulting segmentation quality depend, among other things, on the choice of augmentation methods and the augmentation intensity (please refer to the master thesis of A_Hollinger 2022 supervised by the author of this thesis). The augmentation methods and their parameterizations used in this thesis have been determined based on the mentioned preliminary work and are considered fixed in the remainder of the present thesis. An overview can be seen in Table 5.3.

All transformations are applied to the color images I_C^*. Geometric transformations are additionally applied to depth images I_D^* and masks C_{GT}^* to prevent a discrepancy between color images I_C^* and depth images I_D^*, as well as segmentation masks C_{GT}^*. This results in the augmented color and depth images I'_C and I'_D and ground truth mask C'_{GT}. A visual illustration can be found in Figure 5.9.

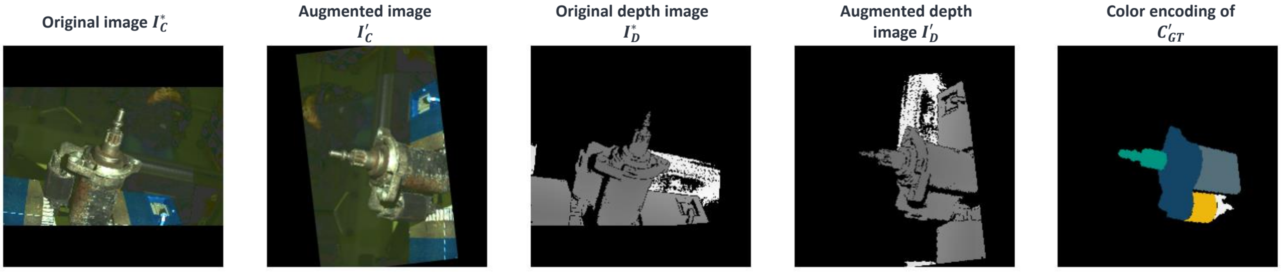

Original image I_C^*
Augmented image I'_C
Original depth image I_D^*
Augmented depth image I'_D
Color encoding of C'_{GT}

Figure 5.9: Overview of the data used for training. Original and augmented images (I_C^*, I_D^*, I'_C and I'_D) as well as ground truth color encoding (C'_{GT}) of a real motor with padding.

The images are augmented during the training and evaluation of the model. As a result, new or slightly modified images are continuously passed to the model. This online augmentation allows a significant saving of the required storage space compared to an offline augmentation (Shorten & Khoshgoftaar 2019, P. 40). All images are augmented with a probability of 50 %, so that half of the images remain unchanged. Each previously described augmentation method is applied with a defined probability on the other half of the images. Thus, the augmentation of the images occurs with a different, random combination of the previously described methods.

In most cases, data standardization is also performed in addition to data augmentation. Depending on the NN model used, standardization of the image data used for learning is helpful to increase the learning speed and performance of the model (García & Luengo et al. 2015, 46 f.). This includes the adjustment of the color channels (e.g., switch from RGB to BGR) and processing methods like zero-centering, scaling, and normalization of the data to

Table 5.3: Tabular representation of the augmentation methods used in this thesis. The operand indicates whether all transformations are carried out with a certain probability (AND) or whether one of the transformations (OR) is selected with a fixed probability.

Geometric transformations

Augmentation type	Operand	Probability	Description
Flip	AND	50%	Flip the input horizontally and vertically.
Perspective		20%	Random four-point perspective transform.
Shift		100%	Random translational shift.
Scale		100%	scaling, i.e., zooming in or out.
Rotate		100%	rotation of the input.

Color space transformations

Augmentation type	Operand	Probability	Description
Color jitter	AND	50%	Change brightness, contrast, saturation.
RGB shift		50%	Shift values for each RGB channel.
Grayscale conversion		25%	Convert the RGB image to grayscale.
HSV shift		50%	Change hue, saturation, and value.

Noise

Augmentation type	Operand	Probability	Description
ISO noise	OR	50%	Apply camera sensor noise.
Gaussian noise			Apply Gaussian noise.

Blurring using kernel operations

Augmentation type	Operand	Probability	Description
Random blur	OR	50%	Blur using a random kernel.
Gaussian blur			Blur using a random Gaussian kernel.

Contrast enhancement

Augmentation type	Operand	Probability	Description
CLAHE	OR	50%	Apply CLAHE.
Gamma			Apply gamma correction.

map them from one range of values (e.g., $[0, 255]$ in the case of RGB) to another (e.g., $[0, 1]$ or $[-1, 1]$).

Segmentation model architectures and transfer learning strategy

The following section provides a rough overview of the network architectures and transfer learning training strategy evaluated in this thesis. Please refer to Appendix A5 for a detailed description of the concepts used. For semantic segmentation and instance segmentation, this thesis compares two of the best-known basic architectures that have achieved good results in various use cases in the past. These are:

- The U-Net by Ronneberger & Fischer et al. 2015 is a commonly used base architecture for semantic segmentation.
- The Mask-RCNN (in the following called M-Net) by He & Gkioxari et al. (2017) is a commonly used base architecture for object detection and instance segmentation.

In this thesis, pretrained backbones are used when constructing the NN, the U-Net, and the M-Net. Two well-known pre-trained backbone NN representatives, the VGG16[1] and the ResNet34[2], are compared with each other. A recent overview of available pre-trained backbone architectures can be found in Elharrouss & Akbari et al. (2022) and a detailed introduction to both backbone variants can be found in Appendix A5. The choice of these architectures can be justified by their simplicity, widespread use, and availability of pre-trained weights. With these pre-trained weights, it is possible to considerably shorten the training time using transfer learning to achieve good results. Transfer learning involves using a model with pre-learned weights instead of initially randomizing and relearning the weights of a model. These pre-trained weights already enable good feature extraction from the input image data at the start of training and do not have to be learned from scratch. This feature extraction is then merely refined during training. Often, the weights of the backbone are also frozen in the early training phase of the NN so that initially, only the late layers of NN, whose weights have been randomly initialized, are trained.

Therefore, this thesis implements and compares three alternatives for the segmentation approach. Given a NN for semantic/ instance segmentation consisting of a backbone and additional layers,

[1] Link to online documentation:
https://pytorch.org/vision/main/models/generated/torchvision.models.vgg16.html
accessed: 15.06.2024

[2] Link to online documentation:
https://pytorch.org/vision/main/models/generated/torchvision.models.resnet34.html
accessed: 15.06.2024

1. no pre-trained backbone is used, i.e., the structure (e.g., a VGG-16 backbone) is used, but the weights are initialized randomly before training.
2. the pre-trained weights of the backbone used are first frozen, and the NN is trained with a comparably high learning rate. Only the segmentation model architecture weights that are not part of the backbone are adjusted. After a certain amount of training, the weights of the backbone are unfrozen, and all layers of the NN are trained.
3. no freezing of the pre-trained NN weights is applied and the NN is fine-tuned directly.

The integration of depth data is based on the current state of research. An overview of common information fusion methods in semantic segmentation can be found in Zhang & Sidibé et al. (2021). A rough distinction is made between early, intermediate, and late fusion methods, which are examined in this thesis. The U-Net and M-Net architectures integrating depth data are called UD-Net and MD-Net in the following. The fusion approaches are shortly summarized in the following. A visualization of the fusion approaches of the UD-nets and MD-nets examined in this thesis can be found in Figure 5.10.

- **Early fusion option:** RGB and depth data are combined before entering the respective network. This is done by stacking the individual channels (R, G, and B) of the color image with the depth channel of the depth image.
- **Intermediate fusion option:** The intermediate features of the individual layers of the backbones (VGG-16 or ResNet34) for RGB and depth data are first fused and then used as input to the subsequent layer of the backbone of RGB data is used. Two options for feature fusion are explored. With addition, the features of a layer of the backbones of RGB and depth data are added element-wise. The other option is a fusion NN using a convolutional layer to get features that can be used as input for the subsequent layer of the RGB backbone. See Appendix A5 for more information on the two alternatives.
- **Late fusion option:** For the MD-Net, separate feature extraction and subsequent fusion of RGB and depth data takes place. The UD-Net must use the skip connections to implement the late data fusion. For this purpose, the intermediate features of the individual layers are either added or processed by an intermediate network and fed into the decoder.

Segmentation model training and evaluation

With the knowledge of the available data pipeline (augmentation and standardization) and the network architectures used, the procedure for training and evaluating the segmentation networks is explained below.

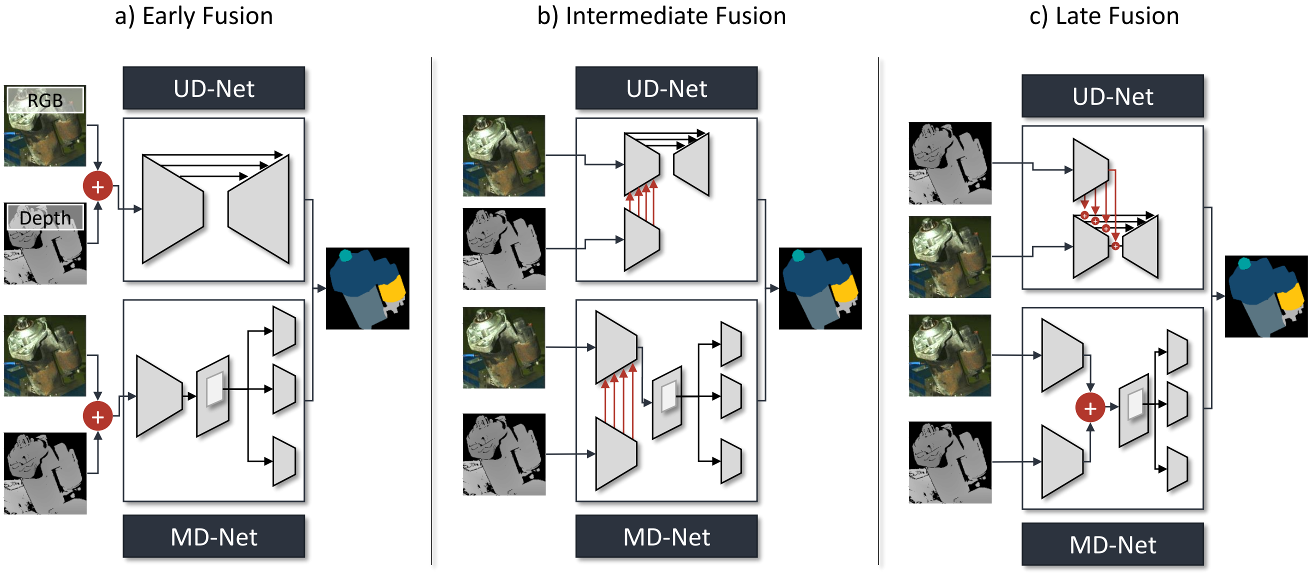


Figure 5.10: Overview of the fusion approaches for the UD-Nets and MD-Nets evaluated in this thesis.

The data set acquired at the real station with 584 RGB and depth images and label masks as ground truth segmentation is split into a training, validation and test data set. The split used for this is chosen according to common practice at 80%, 10%, and 10% for the training, validation, and test data, respectively.

The segmentation networks are trained in several epochs. Each epoch, the training data set is propagated through the respective network configuration and the weights are continuously adjusted based on the error of the segmentation of the segmentation network. The learning behavior is validated with the validation data set after each episode. This involves using data not used for training to check whether the respective segmentation network learns a generalizing segmentation and does not overfit. Finally, the trained segmentation network is tested using the test data set.

While training, a distinction of the loss functions used is made between loss functions for semantic segmentation (variants of U-Net and UD-Net) and the loss function used for instance segmentation (variants of M-Net and MD-Net). The loss function for training a semantic segmentation model is calculated using the augmented ground truth label segmentation mask C'_{GT} and the model's prediction C_P. In contrast, the loss function used for the instance segmentation model variants comprises multiple different loss functions. These are the loss functions for the (1) localization of the object in the image using the bounding box, (2) assignment (classification) of the object to a class, and (3) predicted segmented object mask.

Two standard loss functions for semantic segmentation are dice loss and focal loss. The dice loss is based on the overlap between the predicted and true masks of the classes in an image. It aims to optimize the similarity between these class masks by increasing the overlap of all class masks (Jadon 2020, P. 2). The focal loss is based on a pixel's prediction. It introduces a modulating factor that dynamically adjusts the loss contribution of each pixel. This factor decreases the contribution of well-classified pixels and increases the contribution of misclassified pixels. The focal loss enables the model to learn more effectively by putting more emphasis on challenging examples, thus performing well in tasks with class imbalance (Jadon 2020, P. 2) (see also Appendix A5).

After successful segmentation model training and while training in the segmentation model validation, evaluation metrics based on the classification quality of individual pixels or region-based metrics can be used to evaluate the segmentation quality of the approaches used. The pixel accuracy calculates the percentage of correctly classified pixels in the entire image to be segmented. It is a simple and intuitive measure of overall segmentation accuracy but does not account for class imbalances. Region-based metrics (e.g., *intersection over union* - IoU) consider the similarity of the resulting segmentation masks. The IoU calculates the ratio of the area of overlap between the predicted class mask and ground truth mask for a class to the area of their union. It is used in this thesis as a standard metric used in segmentation tasks to evaluate the accuracy of semantic segmentation and instance segmentation including object localization, class prediction and class segmentation.

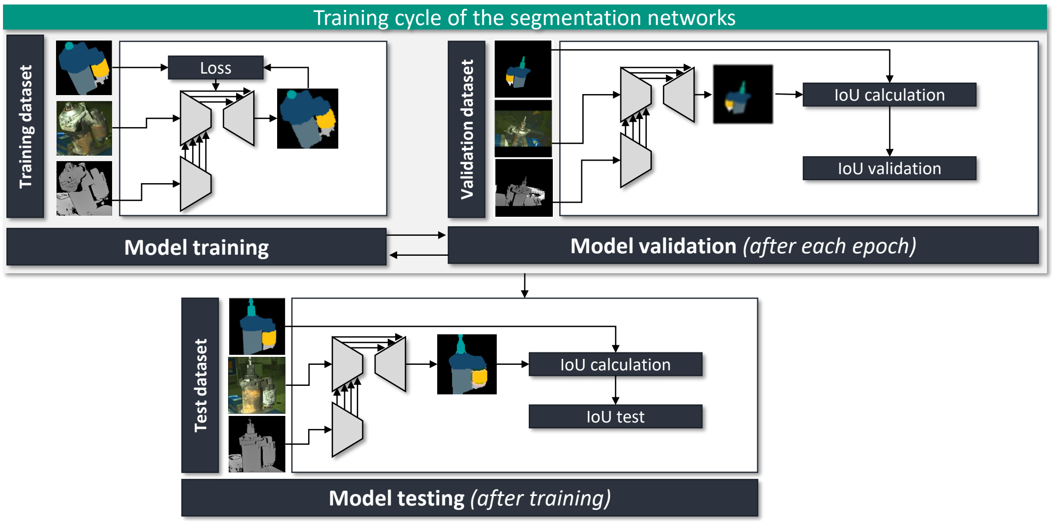


Figure 5.11: Overview of the training, validation and test procedure for the segmentation networks.

A visual representation of the training process can be found in Figure 5.11. The training or the weight updates of the respective NN (in this Figure, for example, a UD-Net with intermediate fusion of the depth images) takes place by means of the loss of the training data set. After each epoch, the IoU of the validation dataset is calculated using the validation dataset to evaluate the training success and progress. After successful training, the IoU is calculated on the test data using the training data set. It should be noted that the data is augmented for model training and validation, whereas the data is not augmented for the model test to test the performance of the network configurations with data that corresponds exactly to data that will be acquired in the real case.

5.2.3 Uncertainty estimation for segmentation results

Due to the stochasticity of the selected viewpoints, it is unavoidable that the same object areas are segmented in successive viewpoints. Due to the possibility of incorrect segmentation, a confidence or uncertainty measure regarding the segmentation result helps evaluate possible ambiguities in segmenting overlapping object areas. It improves the process of semantic modeling of the product.

In metrology, the framework of GUM (Guide to the expression of uncertainty in measurement) exists to express the uncertainty in measurements (ISO/IEC Guide 98-3 2008). The generalized approach of GUM is based on expressing the expanded uncertainty $U = k \cdot u_C(\hat{y})$, which describes an interval ±U around the output estimate $\hat{y}$ of a measurement function $\hat{y} = f(x_i, ..., x_N)$. The combined standard uncertainty $u_C(\hat{y})$ thereby results out of the standard uncertainties $u(x_i)$ of a certain number of N input quantities $x_i (i = 1, ..., N)$ by the formula $u_c(\hat{y}) = \sqrt{\sum_{i=1}^{n}(c_i u(x_i))^2}$. The sensitivity coefficients c_i specify how the variable $\hat{y}$ to be determined changes with small changes in the input variables x_i.

For measurement uncertainty analysis according to GUM, for which, for example, the sensitivity coefficient cannot be determined or can only be determined with difficulty, or for which the influence of input quantities on the output quantity is highly nonlinear, the GUM supplement 1 is available (ISO/IEC Guide 98-3-1 2008). This supplement describes the simulation-based determination of the expanded uncertainty U by simulated, random drawing of many concrete values of input quantities. The estimated value $\hat{y}$ of the target variable is then obtained by averaging using $\hat{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ of the n simulations with outputs $y_i (i = 1, ..., n)$. The standard uncertainty $u_C(\hat{y})$ is then obtained via $\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(y_i - \hat{y})^2}$.

Considering the general definition of machine learning procedures and procedures of segmentation as a functional mapping, it can be noted that they are similar to the general definition of GUM. In this case, the NN represents the highly nonlinear measurement function. Thus, it is possible to apply existing approaches for uncertainty estimation of NN, by the principles of the GUM supplement 1, to the segmentation approach presented in this thesis. However, a significant difference lies in the scope of consideration of the performed uncertainty analysis. According to GUM, measurement uncertainty analysis is traditionally performed for a given, fixed measurement system and measurement object or measurand by repetition of the measurement setup and measurement execution followed by statistical analysis of the measurement results according to GUM. The measurement uncertainty analysis is measurement system and measurement object dependend.

In the present thesis, however, no predefined measurement object or measurand exists. The segmentation mask as a measurement result is composed of the classification of all individual pixels to a component of a measurement object, which can vary with respect to its type. Since the pose of the inspection object can also change, a ground truth can never be assigned to a pixel. This depends on the object's type and the acquisition system's pose. Thus, a measurement uncertainty analysis of the segmentation has to be done for each individual image and pixel in the image. The measurement uncertainty analysis according to GUM pursued in this thesis must, therefore, be carried out in a data-bound manner.

An overview article on the basic approaches to determining uncertainty in deep learning methods is provided by Gawlikowski & Tassi et al. (2023). The authors distinguish between (1) deterministic methods, (2) Bayesian methods, (3) ensemble methods, and (4) test-time augmentation methods. All these approaches have in common that they enable a prediction $\hat{y}$ and, at the same time, provide a predictive uncertainty $\hat{\sigma}$ as a combination of model uncertainty (systemic) and data uncertainty (aleatory) (Gawlikowski & Tassi et al. 2023, P. 4, 10).

This approach is adopted in this thesis. Several models are trained as an ensemble to estimate the model and data uncertainty. In contrast to existing GUM-compliant approaches, estimating the probability distribution of the color values of individual pixels is not directly possible for a single image. However, the test-time augmentation approach proposed in Gawlikowski & Tassi et al. (2023) can be used to account for variance in the input data. Figure 5.12 shows a schematic illustration of the proposed method.

First, n models $M_1, ...M_n$ with varying compositions of the training dataset are trained to solve the segmentation task. In the context of this thesis, $n = 10$ is chosen due to the large time requirements for training the derived network architectures. Given a data tuple $D_{T,i}$ of the test dataset, the pixel-wise prediction $\hat{y}(D_{T,i})_{uv}$ is first derived by averaging all model predictions $\{\hat{y}_{T,i,1}, ..., \hat{y}_{T,i,n}\}_{uv}$ for the given data tuple (ensemble prediction).

The predictive uncertainty $\hat{\sigma}(D_{T,i})_{uv}$ is obtained by randomly augmenting the data tuple $D_{T,i}$ m times to obtain $D'_{T,i1}, ..., D'_{T,im}$ and then performing the pixel-wise prediction of all augmented data tuples with all trained models. The standard deviation for each pixel of the data tuple is then calculated based on the individual model predictions $\{\hat{y}_{T,i1,1}, ..., \hat{y}_{T,im,1}\}_{uv}$, $..., \{\hat{y}_{T,i1,n}, ..., \hat{y}_{T,im,n}\}_{uv}$ of the augmented data tuples. Using this uncertainty estimation, it

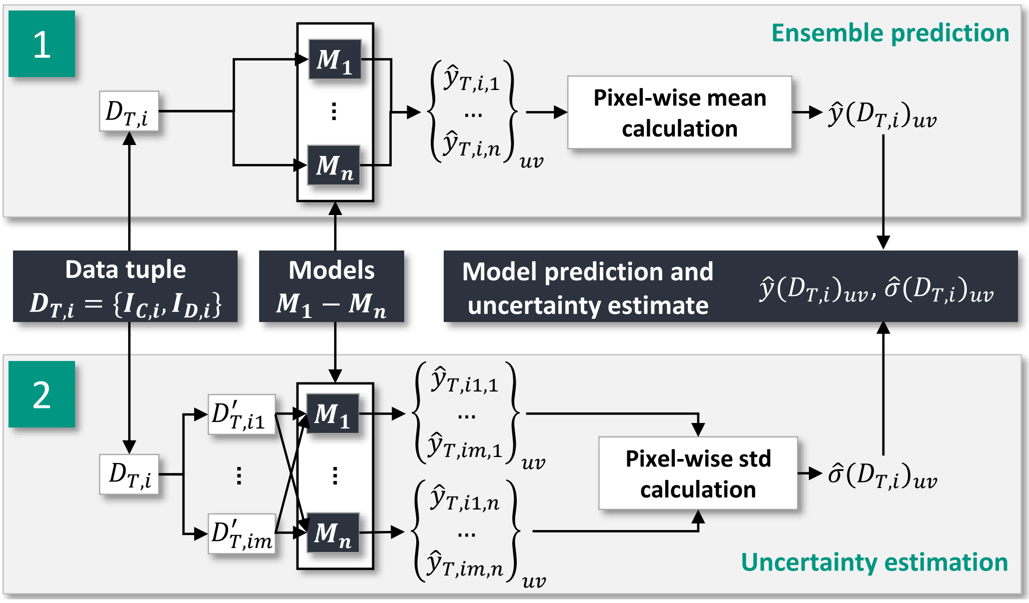

1
Ensemble prediction
$D_{T,i}$
M_1
M_n
$\hat{y}_{T,i,1}$ … $\hat{y}_{T,i,n}$ uv
Pixel-wise mean calculation
$\hat{y}(D_{T,i})_{uv}$
Data tuple $D_{T,i} = \{I_{C,i}, I_{D,i}\}$
Models $M_1 - M_n$
Model prediction and uncertainty estimate
$\hat{y}(D_{T,i})_{uv}, \hat{\sigma}(D_{T,i})_{uv}$
2
$D_{T,i}$
$D'_{T,i1}$
$D'_{T,im}$
$\hat{y}_{T,i1,1}$ … $\hat{y}_{T,im,1}$ uv
$\hat{y}_{T,i1,n}$ … $\hat{y}_{T,im,n}$ uv
Pixel-wise std calculation
$\hat{\sigma}(D_{T,i})_{uv}$
Uncertainty estimation

Figure 5.12: Overview of ensemble model prediction and uncertainty estimation procedure.

is possible to specify the pixel-wise prediction $\hat{y}(D_{T,i})_{uv}$ as well as a pixel-wise predictive uncertainty $\hat{\sigma}(D_{T,i})_{uv}$, which is a combination of systemic and aleatory uncertainty.

5.2.4 Semantic 3D reconstruction of the inspection object

With the additional use of segmentation, the workflow presented in section 5.13 can be extended. After the view generation and acquisition (1 & 2) and before the registration (3), the segmentation of the acquired point cloud $\mathbf{PC}_t^A$ takes place with the help of the color image I_C and depth image I_D. As described in section 5.1.1, a unique relationship between pixels of I_C and I_D and the spatial coordinates of these pixels is provided by the *Zivid One+ S* in the form of the point cloud $\mathbf{P}_t^A$. The semantic point cloud $\mathbf{SPC}_t^A$ can thus be generated from the results of the segmentation (segmentation map S_t), which assigns a class to each pixel (i.e., the assignment of a pixel to a motor component) in I_C and I_D. The semantic point cloud $\mathbf{SPC}_t^A = [\mathbf{X}_t^A, \mathbf{Y}_t^A, \mathbf{Z}_t^A, \mathbf{R}_t, \mathbf{G}_t, \mathbf{B}_t, \mathbf{K}_t]$ thus extends the point cloud $\mathbf{PC}_t^A$ with additional information about the class assignment K_i of the individual points in $\mathbf{PC}_t^A$.

It should be noted that several segmentation models can be used in parallel for the semantic 3D reconstruction of the inspection object. In addition to a segmentation model for component segmentation, a segmentation model can also be used for defects to be inspected in more detail, as mentioned at the beginning but not considered in this work. In this case, for example, $\mathbf{SPC}_t^A$ would be extended by a further column vector $\mathbf{D}_t$, which, in addition to assigning individual pixels to components, also encodes an assignment to defects located on the surface of the inspection object. In this case, $\mathbf{SPC}_t^A = [\mathbf{X}_t^A, \mathbf{Y}_t^A, \mathbf{Z}_t^A, \mathbf{R}_t, \mathbf{G}_t, \mathbf{B}_t, \mathbf{K}_t, \mathbf{D}_t]$ applies.

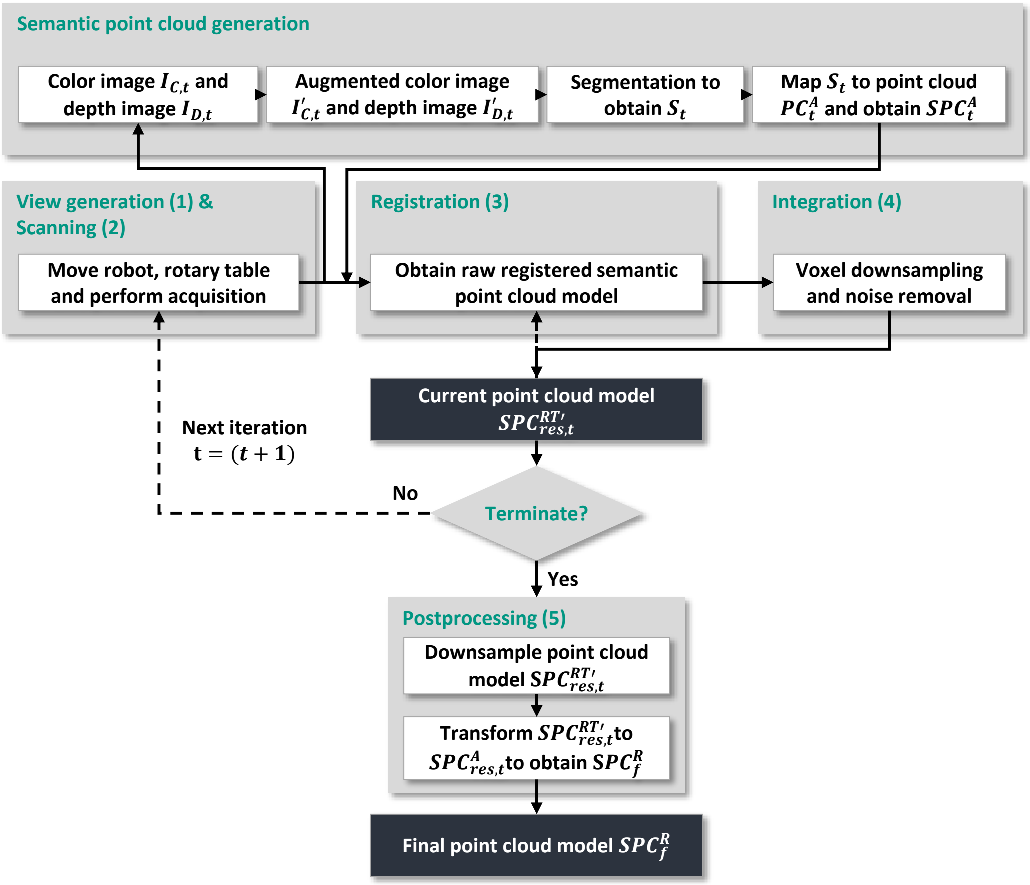

Semantic point cloud generation
Color image $I_{C,t}$ and depth image $I_{D,t}$
Augmented color image $I'_{C,t}$ and depth image $I'_{D,t}$
Segmentation to obtain S_t
Map S_t to point cloud PC_t^A and obtain SPC_t^A
View generation (1) & Scanning (2)
Move robot, rotary table and perform acquisition
Registration (3)
Obtain raw registered semantic point cloud model
Integration (4)
Voxel downsampling and noise removal
Current point cloud model $SPC_{res,t}^{RT'}$
Next iteration $t = (t+1)$
No
Terminate?
Yes
Postprocessing (5)
Downsample point cloud model $SPC_{res,t}^{RT'}$
Transform $SPC_{res,t}^{RT'}$ to $SPC_{res,t}^{A}$ to obtain SPC_f^R
Final point cloud model SPC_f^R

Figure 5.13: Flowchart for semantic 3D reconstruction of an inspection object at the inspection station using segmentation algorithms.

It should also be mentioned that the approach derived in Section 5.2.3 to determine the predictive uncertainty must be applied separately for all segmentation models used. This predictive uncertainty can then be used in the following to adjust the semantic 3D reconstruction of the inspection object in that, in the case of a high predictive uncertainty of individual pixels for certain segmentations, these pixels are not taken into account for the reconstruction.

5.3 View planning using reinforcement learning

The results obtained in sections 5.1 and 5.2 can be used to build a simulation framework as a learning environment for RL agents. The problems to be solved for RL agents are the VPP specific to remanufacturing. The main methodological approach for solving the VPP presented in this section has already been presented and used in previous publications Kaiser & Koch et al. (2024), Kaiser & Gäbele et al. (2024) and Koch & Kaiser et al. (2024) by the author of this thesis. This chapter combines the methodology for the RPP presented in Kaiser & Gäbele et al. (2024) with the methodology for the IPP presented in Kaiser & Koch et al. (2024). For the RPP, Koch & Kaiser et al. (2024) introduces the extension of the methodology to include robot-based view planning. In this section, the planning goals (NBV, RPP and IPP) are therefore first formalized for a RL agent (section 5.3.1). This is followed by an overview of the RL simulation framework (section 5.3.2). The individual components of this framework are then presented. These are the scan simulation environment S_{Sim} (section 5.3.3), the agent interface I_{Agent} (section 5.3.4) and the RL planning agents A_{Plan} used in this thesis (section 5.3.5). A visualization can be found in Figure 5.14.

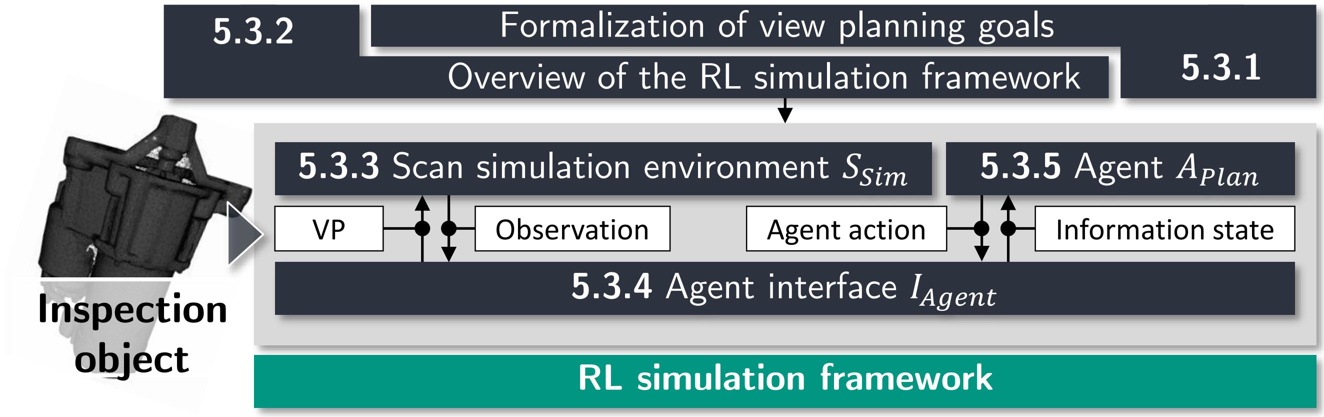

5.3.2
Formalization of view planning goals
Overview of the RL simulation framework
5.3.1
5.3.3 Scan simulation environment S_{Sim}
5.3.5 Agent A_{Plan}
VP
Observation
Agent action
Information state
5.3.4 Agent interface I_{Agent}
Inspection object
RL simulation framework

Figure 5.14: Overview of the approach for RL agent training for solving VPP in Remanufacturing

On a higher level, S_{Sim} handles all operations related to the virtual acquisition and processing of point clouds given an inspection object, sensor model and VP. The acquisition results are transferred to I_{Agent}. I_{Agent} serves as interface between S_{Sim} and A_{Plan} and handles all the necessary steps to convert the actions issued by A_{Plan} into a form that can be used by S_{Sim}. Additionally, it converts the current information status with regard to the current acquisition progress into a form that can be interpreted by A_{Plan}.

5.3.1 Formalization of view planning goals

The planning goal of the RL agents trained in this thesis is fulfilling the specified inspection task. These inspection tasks are either the overall inspection (subproblem a) of section 2.1.4) of the inspection object or the individual inspection of certain components or ROI (subproblem b) of 2.1.4). To do so, they must solve the respective view planning problem. As already introduced in 2.2.1, these differ in whether the entire surface (NBV, RPP) or only a subset of it (IPP) must be acquired. A visualization of this can be found in Figure 5.15.

The surfaces of the inspection object that are ideally covered by the acquisition process are shown in black. This corresponds to the entire object surface in NBV and RPP. As explained, for the IPP, this only corresponds to a subset of the entire object surface. These are the ROI. They can be located anywhere on the object. In remanufacturing, these are defects that were detected during the overall inspection and for which a closer inspection is required

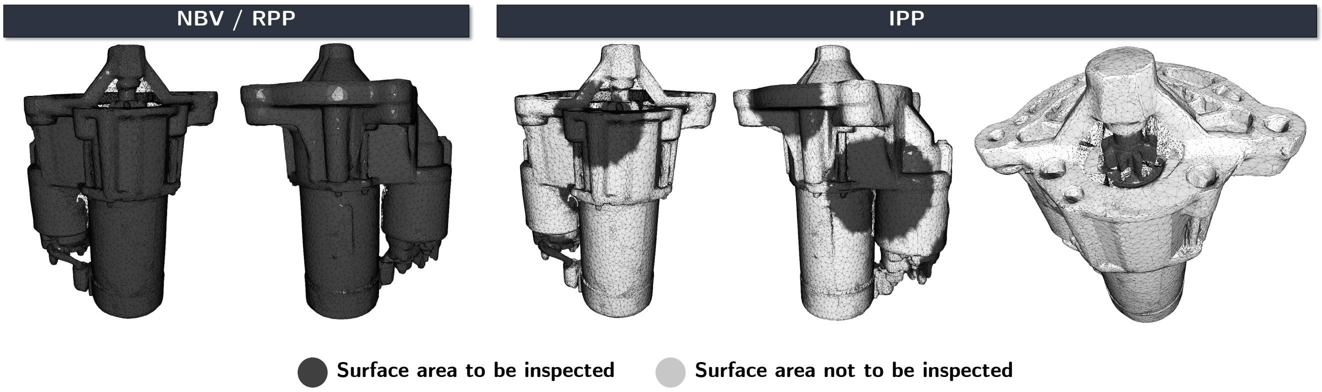

NBV / RPP
IPP
Surface area to be inspected
Surface area not to be inspected

Figure 5.15: Overview of the planning problems to be solved by the RL agent and the object surfaces to be inspected using the example of a starter motor.

to allow an inspection decision to be made. Another possibility is the specific inspection of individual components (e.g., the pinion of a starter motor in Figure 5.15 on the right side), which must always be inspected in order to assess the suitability of the starter motor for remanufacturing.

In terms of VPP, the planning task is to achieve the highest possible percentual surface coverage Ψ of the target surface of the inspection object for the respective planning problem in a minimum number of acquisitions.

5.3.2 Overview of the simulation framework using a reinforcement learning agent $A_{Plan,RL}$

Figure 5.16 depicts the modular simulation framework developed in this thesis. The framework is modularly structured so that different agent types can be easily integrated. This includes the use of RL agents (A_{Plan} then becomes $A_{Plan,RL}$) but also agents that use predefined rules and analytical solution methods for the considered problems of NBV, RPP, and IPP. In the following, the basic functions of the individual modules are explained.

The scan simulation environment S_{Sim} consists of two modules. The planning module and the scanning module. Before each acquisition cycle, an inspection object model is loaded as a stereolithography file (STL file). Multiple acquisitions are performed in an acquisition cycle using a parameterized sensor model. A VP $\mathbf{V}_t^S$ is passed to the planning module to realize one acquisition process. The planning module then performs a reachability check of the issued pose and subsequent trajectory planning if the pose is reachable. After that, an acquisition is simulated using the scanning module. The current information status of the current acquisition progress, which results from all previous acquisitions in an acquisition cycle, is saved in the observation $\mathbf{o}_t$. The observation $\mathbf{o}_t$ represents a processed form of the current and all previous observations.

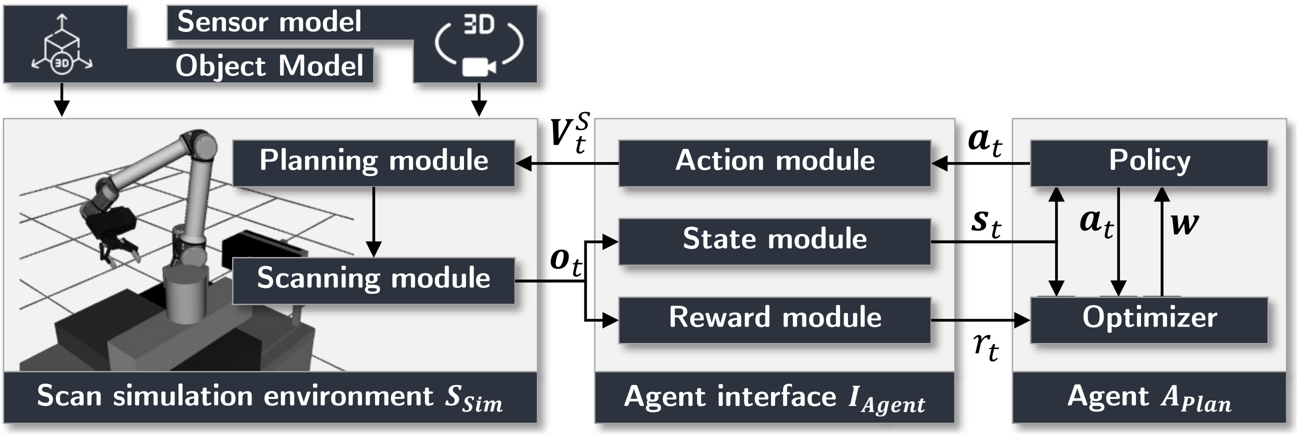


Figure 5.16: Overview of the RL simulation framework for RL agent training.

The observation $\mathbf{o}_t$ is processed by the agent interface I_{Agent}, which consists of three modules. The action module, the state module and the reward module. The action module maps actions $\mathbf{a}_t$, which the agent outputs, to the VP $\mathbf{V}_t^S$. The state module uses the observation $\mathbf{o}_t$ to create a state $\mathbf{s}_t$ that the agent can interpret. The reward module uses the observation $\mathbf{o}_t$ to derive a reward r_t for the agent $A_{Plan,RL}$ based on its action.

When using a RL agent, at a higher level, $A_{Plan,RL}$ consists of a policy and an optimizer. The policy defines the next action $\mathbf{a}_t$ to be performed, based on the current state $\mathbf{s}_t$. The optimizer then adjusts the parameters w of the policy given the tuple $(\mathbf{s}_t, \mathbf{a}_t, r_t)$ so that a strategy is learned that maximizes the long-term return (see 2.3.3).

The proposed and implemented structure allows the easy interchangeability of individual modules to analyze different modeling alternatives and the learned strategies of the defined RL agents to solve the visual acquisition planning problems. This is based on the fixed, standardized variables ($\mathbf{V}_t^S$, $\mathbf{o}_t$, $\mathbf{a}_t$, $\mathbf{s}_t$, and r_t) that are passed between the individual components of the simulation framework. As long as the interface definition between the individual components is adhered to with the standardized variables to be passed between each other, modules with varying internal functionality can be used. This also enables the straightforward replacement of the scanning module with the real acquisition system at a later stage when deployed on the real inspection station.

5.3.3 Modelling of the scan simulation environment S_{Sim}

In each acquisition step t of the agent with the environment, S_{Sim} receives a VP $\mathbf{V}_t^S$. After a reachability analysis and trajectory planning simulates an acquisition using the object and sensor model with the scanning module to obtain a point cloud $\mathbf{P}_t^A$. The acquisition results are aggregated and provided as an observation $\mathbf{o}_t$. An overview of the working principles of S_{Sim} can be seen in Figure 5.17.

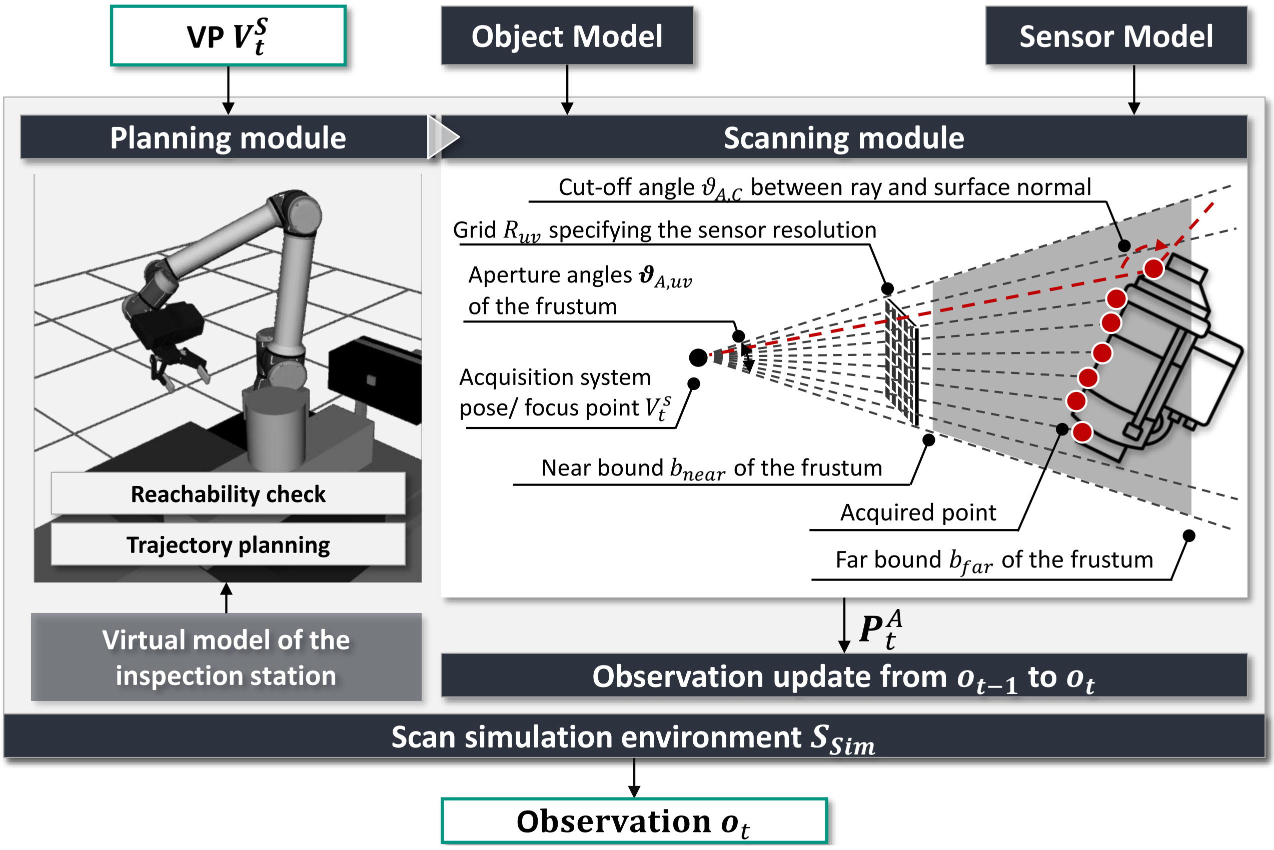

VP V_t^S
Object Model
Sensor Model
Planning module
Scanning module
Cut-off angle $\vartheta_{A,C}$ between ray and surface normal
Grid R_{uv} specifying the sensor resolution
Aperture angles $\boldsymbol{\vartheta}_{A,uv}$ of the frustum
Acquisition system pose/ focus point V_t^s
Near bound b_{near} of the frustum
Acquired point
Far bound b_{far} of the frustum
Reachability check
Trajectory planning
Virtual model of the inspection station
$\boldsymbol{P}_t^A$
Observation update from $\boldsymbol{o}_{t-1}$ to $\boldsymbol{o}_t$
Scan simulation environment $\boldsymbol{S}_{Sim}$
Observation $\boldsymbol{o}_t$

Figure 5.17: Visualization of the inner workings of the scan simulation environment S_{Sim}.

Planning process in the planning module

As introduced in section 2.2.1, $\mathbf{V}_t^S$ is a tuple. It consists of the poses $\mathbf{p}_t^{A/RT}$, $\mathbf{p}_t^{O/RT}$ and the acquisition parameters λ_A. It is possible that a VP, which is determined based on the action output by the agent, cannot be reached. This is because the acquisition system is mounted on a robot. Not every pose can be reached due to possible collisions of the robot with itself or its surroundings based on the trajectory from $\mathbf{p}_{t-1}^{A/RT}$ (current pose of the acquisition system) to $\mathbf{p}_t^{A/RT}$ (desired pose as the output of the agent). Furthermore, certain poses in working space can not be reached due to preset constraints, e.g., restrictions of joint angles.

Both poses, $\mathbf{p}_t^{A/RT}$ of the acquisition system and $\mathbf{p}_t^{O/RT}$ of the object with its coordinate system C_O are expressed in relation to the static rotary table coordinate system C_{RT}. This convention has been chosen, amongst other reasons, as the agent is allowed to influence the rotation angle φ_{RT} of the rotation table around the z-axis of C_{RT} and therefore C_{RT} is suitable as a static reference system. However, C_R is also a conceivable alternative, whereby in theory, all possible coordinate systems can be selected whose kinematic relationships (described via homogeneous transformation matrices) to the spatially fixed coordinate systems C_R or C_{RT} are known at all times.

The pose $\mathbf{p}_t^{O/RT}$ of the object in relation to C_{RT} is composed of a fixed, initial pose of the object expressed in relation to C_{RT} at the beginning of the acquisition process and the variable rotational part based on the rotation angle φ_{RT} due to rotations of the rotation table between successive acquisitions. The rotation of the rotary table takes place before the trajectory planning using *MoveIt* to update the planning scene with all collision models (see section 5.1.2).

The pose $\mathbf{p}_t^{A/RT} = [X_t^{A/RT}, Y_t^{A/RT}, Z_t^{A/RT}, \alpha_t^{A/RT}, \beta_t^{A/RT}, \gamma_t^{A/RT}]$ of the acquisition system is a vector. For $X_t^{A/RT}$, $Y_t^{A/RT}$, and $Z_t^{A/RT}$, Cartesian coordinates indicate the position of the acquisition systems coordinate system C_A in the rotary table coordinate system C_{RT}, while $\alpha_t^{A/RT}$, $\beta_t^{A/RT}$, and $\gamma_t^{A/RT}$ are Euler angles, representing the rotation of the frustum. For reachability analysis and trajectory planning, $\mathbf{p}_t^{A/RT}$ must first be expressed in the static robot coordinate system C_R. $\mathbf{p}_t^{A/RT}$ is first transformed to its equivalent homogeneous transformation matrix $\mathbf{T}_{p,t}^{A/RT}$ and then expressed in C_R ($\mathbf{T}_{p,t}^{A/R} = \mathbf{T}_{p,t}^{A/RT}\mathbf{T}^{RT/R}$). The reachability check, trajectory planning, and execution use the virtual station model (see section 5.1.1), including all deduced kinematic relationships. Based on the two poses $\mathbf{p}_{t-1}^{A/RT}$ and $\mathbf{p}_t^{A/RT}$, which represent the start and end pose of the desired trajectory, a virtual reachability analysis is carried out using *MoveIt*. This ensures that the calculated pose $\mathbf{p}_t^{A/RT}$ can be reached and the resulting trajectory is collision-free.

Sensor and object model

Using a real acquisition system, an acquisition with the acquisition parameters λ_A (e.g., defined via the service *Suggested Settings* of the *Zivid One+ S*, see section 5.2) would take place. In S_{Sim}, the sensor model reproduces the functionality and properties of such three-dimensional acquisition systems. The relevant configuration options of the simulated acquisition system result from modeling the field of view as a frustum and the specification of the resolution. In addition, a cut-off angle between the acquisition system and the normal of a surface point of the object must be defined, above which a point is not considered to be acquired. Therefore, the sensor model used in this thesis is formally described by the:

- Resolution of the acquisition system R_{uv} that determines the number of surface points acquired per acquisition.
- Aperture angles $\boldsymbol{\vartheta}_{A,uv}$ of the frustum (field of view)
- Near and far bounds b_{near} and b_{far} of the frustum (operating range of the acquisition system).

- cut-off angle $\vartheta_{A,C}$.

The object model used in the simulated environment is specified via the ***stereolithography*** (STL) format. Thereby, the object is represented only by its surface in triangular facets. The representation of the object with triangular facets enables a computationally efficient ray tracing procedure, which the scanning module uses. A visualization of the raytracing procedure, taking into account the characteristics of the sensor model, can be seen in Figure 5.17.

Acquisition process in the planning module

The scanning module connects the sensor model and the object model utilizing the Python package *trimesh* (Dawson-Haggerty et al. 2019). *Trimesh* is used to enable efficient ray tracing to emulate the three-dimensional acquisition of the object. The virtual acquisition system is positioned using the pose $\mathbf{p}_t^{A/RT}$ based on the output of the previous step of trajectory planning. Afterward, an acquisition is simulated. Depending on the resolution and aperture angles of the sensor model, rays are defined starting from the focal point. The coordinate of each first intersection point of the rays with the object defines a point in space in the static coordinate system C_{RT}. After simulating the acquisition of the object with the scanning module based on the acquisition system pose $\mathbf{p}_t^{A/RT}$, the point cloud $\mathbf{P}_t^A$, as a sum of all intersection points of the rays with the object, is returned. Given a pose $\mathbf{p}_t^{A/RT}$ where the object does not lie in the frustum of the acquisition system (i.e., the sensor faces away from the object), $\mathbf{P}_t^A$ is empty.

Observation update in the planning module

The scanning module bundles all information relevant to the agent through observation $\mathbf{o}_t$. $\mathbf{o}_t$ represents all information of the acquisition process up until the current interaction step t.

At the beginning of each acquisition process, the object model (STL model) to be inspected is fed into S_{Sim} and stored in the observation data. Additionally, at the beginning of an acquisition process, the *ground truth point cloud* $\mathbf{P}_{GT}^{RT'}$ is calculated and stored in the observation data as well. $\mathbf{P}_{GT}^{RT'}$ is generated by evenly sampling n_{GT} *ground truth points* on the object model using a uniform sampling[1] approach and transforming these points to $C_{RT'}$. On top of the ground truth point cloud, the target point cloud $\mathbf{P}_G^{RT'}$, which is the point cloud of points that must be inspected based on the inspection goal, is stored in $\mathbf{o}_t$. Depending on the inspection goal, $\mathbf{P}_G^{RT'}$ is either subset a of the object surface and thus of $\mathbf{P}_{GT}^{RT'}$ (for IPP) or equals $\mathbf{P}_{GT}^{RT'}$ (for NBV and RPP).

[1] Link to uniform sampling documentation in *Open3D*:
https://www.open3d.org/docs/0.7.0/python_api/open3d.geometry.sample_points_uniformly.html
accessed: 06.01.2024

During an acquisition process, the scanning module collects all information gathered by the simulated sequential acquisitions until interaction step t and stores them in the $\mathbf{o}_t$. Thereby, $\mathbf{P}_t^A$ is first transformed into the co-rotating rotary table reference system $C_{RT'}$ obtaining $\mathbf{P}_t^{RT'}$. The total point cloud $\mathbf{P}_{cov,t}^{RT'}$ is then calculated based on the point cloud $\mathbf{P}_t^{RT'}$ acquired at step t and all acquired point clouds $\mathbf{P}_1^{RT'}, ..., \mathbf{P}_{t-1}^{RT'}$ up until step $t-1$. Here, the approach for 3D reconstruction, introduced in section 5.1.4, is used to continuously update $\mathbf{P}_{cov,t}^{RT'}$ and keep it updated in the co-rotating rotary table coordinate system $C_{RT'}$. Furthermore, the ground truth point cloud $\mathbf{P}_{GT}^{RT'}$ and goal point cloud $\mathbf{P}_G^{RT'}$ are also continuously updated during the acquisition process to be represented in the same co-rotating rotary table coordinate system as $\mathbf{P}_{cov,t}^{RT'}$. This makes a comparison of the point clouds possible to check, e.g., coverage.

In summary, the following information given in the list below is computed and stored in the observation data $\mathbf{o}_t$ as well as updated after each acquisition:

- Object model (triangle mesh) of the object to be inspected in the current acquisition process.
- Ground truth point cloud $\mathbf{P}_{GT}^{RT'}$ with predefined number of points.
- Target point cloud $\mathbf{P}_G^{RT'}$ as a subset of $\mathbf{P}_{GT}^{RT'}$. The target point cloud contains all points of $\mathbf{P}_G^{RT'}$ that need to be inspected in the current acquisition cycle. For NBV and VPP, $\mathbf{P}_G^{RT'} = \mathbf{P}_{GT}^{RT'}$ applies, while for IPP $\mathbf{P}_G^{RT'} \subset \mathbf{P}_{GT}^{RT'}$ applies.
- Number of acquisitions t in the current acquisition process.
- List of acquisition system poses $\mathbf{p}_1^{A/RT}, ..., \mathbf{p}_t^{A/RT}$ of the current acquisition process and the corresponding acquired point clouds $\mathbf{P}_1^{RT'}, ..., \mathbf{P}_t^{RT'}$ as well as the absolute rotary table angles $\varphi_{RT,1}, ..., \varphi_{RT,t}$ for each acquisition.
- *Total point cloud* $\mathbf{P}_{cov,t}^{RT'}$ acquired in an acquisition process up to interaction step t. This point cloud is obtained by merging all the acquired point clouds $\mathbf{P}_1^{RT'}, ..., \mathbf{P}_t^{RT'}$ and performing subsequent uniform downsampling to obtain a uniformly distributed point cloud $\mathbf{P}_{cov,t}^{RT'}$.
- Total point cloud $\mathbf{P}_{cov,t-1}^{RT'}$ acquired up to interaction step $t-1$.

This observation is then used in the agent interface I_{Agent} to derive the state $\mathbf{s}_{t+1}$ given to the agent to select the following action $\mathbf{a}_{t+1}$ at the interaction step $t+1$. However, before the agent is triggered to select the next action $\mathbf{a}_{t+1}$ based on $\mathbf{s}_{t+1}$, it is checked whether the current acquisition process can be considered finished.

Episode design

The check on whether an acquisition process must be terminated is based on $\mathbf{o}_t$. The acquisition process in RL is called an episode. In RL, an episode is a state, action, and reward transition sequence. Ideally, an episode ends in a goal state s_G where the agent's task is considered completed. It is also possible to terminate an episode (i.e., an acquisition process) based on predefined rules (e.g., exceeding a maximum permissible amount of agent interactions with the environment).

A goal state s_G is reached when the percentage surface coverage Ψ_t in inspection step t exceeds a threshold value Ψ_T (e.g., 90%). The calculation rule for the percentage surface coverage Ψ_t based on the point clouds $\mathbf{P}_G^{RT'}$ and $\mathbf{P}_{cov,t}^{RT'}$ is presented in 5.3.4.3. If the agent does not reach s_G after a certain number of interactions with the environment (acquisitions), it enters a terminal state s_T. This convention is reasonable due to the industrial context since the agent should not be able to inspect an object indefinitely. For this reason, exceeding a maximum permitted number of 10 acquisitions forces the episode to end without reaching a s_G.

After the end of an episode, either by reaching s_G or s_T, the reached surface coverage Ψ_E and the required number of acquisitions n_E is known. A new episode is then started by resetting the complete environment (loading a new inspection object, resetting $\mathbf{o}_t$).

5.3.4 Modelling of the agent interface I_{Agent}

The agent interface transfers information provided by the scan simulation environment into a representation that can be interpreted by the agent (see section 5.3.4.1). Furthermore, it translates selected agent actions into an acquisition step interpretable for the scan simulation environment (see section 5.3.4.2). Besides, the agent interface also calculates the rewards for the learning agent (see section 5.3.4.3).

5.3.4.1 State module

In the state module, the information stored in the overall observation data $\mathbf{o}_t$ is used to deduce the state $\mathbf{s}_t$ given to the agent to determine the acquisition system's next pose $\mathbf{p}_{t+1}^{A/RT}$. To ensure the transferability of the simulated system to a real scenario where no object model is available, it is important that only data that is also available to the agent in the said scenario is integrated into the calculation of the state $\mathbf{s}_t$. Two base cases have to be distinguished.

1. **STL model of the inspection object is not available (model-free):**
   If the object model does not exist at the time of inspection, the state $\mathbf{s}_t$ must only contain information that has been collected up to time $t-1$. This includes all information stored in the observation $\mathbf{o}_{t-1}$. This implies that, for example, after an initial acquisition, only

the point cloud $\mathbf{P}_0^{RT'}$ or the pose $\mathbf{p}_0^{A/RT}$ of this acquisition may be used to deduce state $\mathbf{s}_1$ and passed to the agent. This effectively allows the agent to only be provided with information about object surfaces already acquired or poses that have already been approached. Further information that can be extracted from the object model a-priori (e.g., the shape) cannot be used to model the state.

2. **STL model of the inspection object is available (model-based):**
   If the model of the inspection object is known at the time of the inspection, it is permissible to model the state $\mathbf{s}_t$ in such a way that information about object surfaces that the agent has already inspected and additionally, those that still need to be inspected can be passed.

When modeling $\mathbf{s}_t$, special attention must therefore be paid to which information is passed to the agent in which form based on an inspection goal under given boundary conditions (model-based and model-free). Modeling the state vector $\mathbf{s}_t$ is thus a fundamental task to provide the agent with all the information needed to select the next action.

In this thesis, the components of the state vector $\mathbf{s}_t$ are defined as generally as possible. To realize this, we employ point clouds to represent the state. In contrast to low-resolution occupancy grids, point clouds can represent object geometries in greater detail. Regardless of whether a model-free or model-based inspection goal is being pursued, a combination of the following information can be passed to the agent.

- Information of the current acquisition state, i.e., the acquired surface points of the object (model-free) until interaction step t or information about acquired surface points until interaction step t and still to be acquired surface points of the object (model-based).
- Surface orientation (normal) of already acquired (model-free) or already acquired and still to be acquired surface points (model-based).
- The current pose $\mathbf{p}_t^{A/RT}$ of the acquisition system.
- Information about the surface points acquired with the last pose $\mathbf{p}_{t-1}^{A/RT}$ of the acquisition system.

The modeling approaches are explained below depending on the availability of the object model.

State modeling for model-free inspection problems

For model-free inspection, the agent is only provided with information based on the surface points of the object that acquisitions have already acquired until interaction step t.

Therefore, the model-free option for state coding presented in this thesis is based on the total point cloud $\mathbf{P}_{cov,t}^{RT'}$ acquired up to the current interaction step t of the agent with the environment. Thereby, $X_{t,m}^{RT'}$, $Y_{t,m}^{RT'}$ and $Z_{t,m}^{RT'}$ denote the coordinates of a point $\mathbf{P}_{t,m}^{RT'}$ of the point cloud $\mathbf{P}_{cov,t}^{RT'}$. The first three columns of the state $\mathbf{s}_{t,modf,\mathbf{PC}}$ (see equation 5.2) thus capture the available information about the objects surface points already acquired during previous acquisitions.

$$\mathbf{s}_{t,modf,\mathbf{PC}} = \begin{bmatrix} X_{t,1}^{RT'} & Y_{t,1}^{RT'} & Z_{t,1}^{RT'} & n_{\varphi,t,1} & n_{\theta,t,1} & \Delta_{t,1} \\ ... & ... & ... & ... & ... & ... \\ X_{t,2048}^{RT'} & Y_{t,2048}^{RT'} & Z_{t,2048}^{RT'} & n_{\varphi,t,2048} & n_{\theta,t,2048} & \Delta_{t,2048} \end{bmatrix} \quad \text{and} \quad \mathbf{s}_{t,modf,\mathbf{p}} = \mathbf{p}_t^{A/RT} \tag{5.2}$$

Since $\mathbf{P}_{cov,t}^{RT'}$ is generated based on point clouds of several consecutive acquisitions, a voxel downsampling is carried out before calculating $\mathbf{s}_{t,modf,\mathbf{PC}}$. This has two key advantages. First, voxel downsampling provides a point cloud $\mathbf{P}_{cov,t}^{RT'}$ with equal density distribution. Second, voxel downsampling enables reducing the size of the point cloud to a fixed value. This is necessary since the agents employ neural networks (see subsection 5.3.5) that require a fixed state size $\mathbf{s}_t$ as inputs. In the present thesis, $\mathbf{P}_{cov,t}^{RT'}$ is therefore reduced to a size of 2048 points. The number of points was chosen to correspond to a size that is also frequently chosen as a reference in other works, such as in, for example, Achlioptas & Diamanti et al. (2018), Wang & Ma et al. (2020) and Wen & Li et al. (2020). It has to be noted that there is no general rule in choosing the number of points in the point cloud. The number of points should be chosen as low as possible to reduce the calculation time of the agent but as high as necessary to preserve the object's shape.

Based on the high-resolution point cloud $\mathbf{P}_{cov,t}^{RT'}$, the normal vector can be calculated for each point. In contrast to the representation in Cartesian coordinates, the normal vectors of the surface points are encoded in $\mathbf{s}_{t,modf,\mathbf{PC}}$ in spherical coordinates, resulting in $n_{\varphi,t,m}$, $n_{\theta,t,m}$. This is because the required parameters expressing the points orientation can be reduced from three to two, when leaving out the third component of the spherical normal vector of a point $n_{r,t,m}$. This can be done since the radius $n_{r,t,m}$ as the length of the normal vector in spherical coordinates only plays a subordinate role for the actual orientation since it only serves as a normalizing factor to ensure the length of the normal vector is 1. The mathematical formulations for calculation of the normal vector in spherical coordinates given a point $\mathbf{P}_{t,m}^{RT'}$ can be found in Appendix A2.

Finally, the binary coding $\Delta_{t,m} \in [0, 1]$ can be used to specify whether this surface point was acquired with the last acquisition of the acquisition system with the pose $\mathbf{p}_t^{A/RT}$. Said pose is also passed to the agent with state by $\mathbf{s}_{t,modf,\mathbf{p}}$, making the final state a tuple

$\mathbf{s}_{t,modf}$=($\mathbf{s}_{t,modf,\mathbf{PC}}$,$\mathbf{s}_{t,modf,\mathbf{p}}$). Thus, in theory, it is possible for the agent to explicitly learn the relationships between the pose $\mathbf{p}_t^{A/RT}$ passed to it by $\mathbf{s}_{t,modf,\mathbf{p}}$ of the last acquisition and the surface points acquired with this pose that are coded accordingly in $\mathbf{s}_{t,modf,\mathbf{PC}}$.

State modeling for model-based inspection problems

For model-based inspection, the agent is provided with information based on which surface points have already been acquired until interaction step t and which surface points still need to be acquired in the rest of an episode to fulfill the inspection goal. As in the model-free case, the state $\mathbf{s}_{t,modb}$=($\mathbf{s}_{t,modb,\mathbf{PC}}$,$\mathbf{s}_{t,modb,\mathbf{p}}$) is a tuple. In contrast to model-free state modeling, in which the current state is based on the point cloud $\mathbf{P}_{cov,t}^{RT'}$, in the model-based case, the state is calculated based on the ground-truth point cloud $\mathbf{P}_{GT}^{RT'}$. The encoding of the current acquisition state can then be carried out using an additional column with the entries $b_{t,m}$. The normal vectors $n_{\varphi,t,m}$, $n_{\theta,t,m}$ and the binary coding $\Delta_{t,m} \in [0,1]$ are determined analogously to the model-free case. The complete state vector can be found in equation 5.3.

$$\mathbf{s}_{t,modb,\mathbf{PC}} = \begin{bmatrix} X_{t,1}^{RT'} & Y_{t,1}^{RT'} & Z_{t,1}^{RT'} & b_{t,1} & n_{\varphi,t,1} & n_{\theta,t,1} & \Delta_{t,1} \\ ... & ... & ... & ... & ... & ... & ... \\ X_{t,2048}^{RT'} & Y_{t,2048}^{RT'} & Z_{t,2048}^{RT'} & b_{t,2048} & n_{\varphi,t,2048} & n_{\theta,t,2048} & \Delta_{t,2048} \end{bmatrix}, \quad \mathbf{s}_{t,modb,\mathbf{p}} = \mathbf{p}_t^{A/RT} \tag{5.3}$$

Two cases must be distinguished in the case of VPP as an inspection target. Either the respective point $\mathbf{P}_{t,m}^{RT'}$ depicted in the state has already been acquired in the previous acquisition process and is therefore contained in $\mathbf{P}_{cov,t}^{RT'}$ or not. In the case of VPP as an inspection target, the following therefore applies with $m \in \{1, ...2048\}$:

$$b_{t,m} = \begin{cases} 1 & \mathbf{P}_{t,m}^{RT'} \in \mathbf{P}_{cov,t}^{RT'} \\ 0 & \mathbf{P}_{t,m}^{RT'} \notin \mathbf{P}_{cov,t}^{RT'} \end{cases} \tag{5.4}$$

For the inspection target IPP, only a subset of all surface points must be inspected ($\mathbf{P}_{G}^{RT'} \subset \mathbf{P}_{GT}^{RT'}$) and $\mathbf{P}_{G}^{RT'} \neq \mathbf{P}_{GT}^{RT'}$ applies. Therefore, in the case of IPP as an inspection target, in addition to checking whether a surface point has already been acquired or not ($\mathbf{P}_{t,m}^{RT'} \in \mathbf{P}_{cov,t}^{RT'}$), it must also be checked whether it should be acquired at all ($\mathbf{P}_{t,m}^{RT'} \in \mathbf{P}_{G}^{RT'}$) in order to fulfill the inspection target. In this case, the following applies with $m \in \{1, ...2048\}$:

$$b_{t,m} = \begin{cases} 1 & \mathbf{P}_{t,m}^{RT'} \in \mathbf{P}_{cov,t}^{RT'} \quad \text{and} \quad \mathbf{P}_{t,m}^{RT'} \in \mathbf{P}_{G} \\ 0 & \mathbf{P}_{t,m}^{RT'} \notin \mathbf{P}_{G} \\ -1 & \mathbf{P}_{t,m}^{RT'} \notin \mathbf{P}_{cov,t}^{RT'} \quad \text{and} \quad \mathbf{P}_{t,m}^{RT'} \in \mathbf{P}_{G} \end{cases} \tag{5.5}$$

Implemented state modeling alternatives

The presented model-based and model-free states correspond to the complete information available to the agent in each acquisition step. The variants of state modeling shown in Table 5.4 are implemented in the simulation framework to enable an evaluation of the extent to which state modeling offers a learning advantage.

Table 5.4: Overview of the model-free ($S_1 - S_3$) and model-based ($S_4 - S_6$) state modeling alternatives.

Model-free state modeling alternatives		
Abbreviation	$\mathbf{s}_{t,modf,\mathbf{PC}}$	$\mathbf{s}_{t,modf,\mathbf{p}}$
S_1	$\begin{bmatrix} X^{RT'}_{t,1} & Y^{RT'}_{t,1} & Z^{RT'}_{t,1} \\ ... & ... & ... \\ X^{RT'}_{t,2048} & Y^{RT'}_{t,2048} & Z^{RT'}_{t,2048} \end{bmatrix}_{cov}$	-
S_2	$\begin{bmatrix} X^{RT'}_{t,1} & Y^{RT'}_{t,1} & Z^{RT'}_{t,1} & n_{\varphi,t,1} & n_{\theta,t,1} \\ ... & ... & ... & ... & ... \\ X^{RT'}_{t,2048} & Y^{RT'}_{t,2048} & Z^{RT'}_{t,2048} & n^{RT'}_{\varphi,t,2048} & n^{RT'}_{\theta,t,2048} \end{bmatrix}_{cov}$	-
S_3	$\begin{bmatrix} X^{RT'}_{t,1} & Y^{RT'}_{t,1} & Z^{RT'}_{t,1} & n_{\varphi,t,1} & n_{\theta,t,1} & \Delta_{t,1} \\ ... & ... & ... & ... & ... & ... \\ X^{RT'}_{t,2048} & Y^{RT'}_{t,2048} & Z^{RT'}_{t,2048} & n_{\varphi,t,2048} & n_{\theta,t,2048} & \Delta_{t,2048} \end{bmatrix}_{cov}$	$\begin{bmatrix} X^{A/RT}_t \\ ... \\ \gamma^{A/RT}_t \end{bmatrix}$
Model-based state modeling alternatives		
Abbreviation	$\mathbf{s}_{t,modb,\mathbf{PC}}$	$\mathbf{s}_{t,modb,\mathbf{p}}$
S_4	$\begin{bmatrix} X^{RT'}_{t,1} & Y^{RT'}_{t,1} & Z^{RT'}_{t,1} & b_{t,1} \\ ... & ... & ... & ... \\ X^{RT'}_{t,2048} & Y^{RT'}_{t,2048} & Z^{RT'}_{t,2048} & b_{t,2048} \end{bmatrix}_{GT}$	-
S_5	$\begin{bmatrix} X^{RT'}_{t,1} & Y^{RT'}_{t,1} & Z^{RT'}_{t,1} & b_{t,1} & n_{\varphi,t,1} & n_{\theta,t,1} \\ ... & ... & ... & ... & ... & ... \\ X^{RT'}_{t,2048} & Y^{RT'}_{t,2048} & Z^{RT'}_{t,2048} & b_{t,2048} & n_{\varphi,t,2048} & n_{\theta,t,2048} \end{bmatrix}_{GT}$	-
S_6	$\begin{bmatrix} X^{RT'}_{t,1} & Y^{RT'}_{t,1} & Z^{RT'}_{t,1} & b_{t,1} & n_{\varphi,t,1} & n_{\theta,t,1} & \Delta_{t,1} \\ ... & ... & ... & ... & ... & ... & ... \\ X^{RT'}_{t,2048} & Y^{RT'}_{t,2048} & Z^{RT'}_{t,2048} & b_{t,2048} & n_{\varphi,t,2048} & n_{\theta,t,2048} & \Delta_{t,2048} \end{bmatrix}_{GT}$	$\begin{bmatrix} X^{A/RT}_t \\ ... \\ \gamma^{A/RT}_t \end{bmatrix}$

The modeling alternatives S_1 and S_4 represent the simplest model-free and model-based modeling variants. Here, only the point information $(X^{RT'}_{t,m}, Y^{RT'}_{t,m}, Z^{RT'}_{t,m})$ based on $\mathbf{P}^{RT'}_{cov,t}$ (model-free state) or $\mathbf{P}^{RT'}_{GT}$ (model-based state), as well as the coding $b_{t,m}$ in the model-based case are used in the respective state modelings. Further modeling variants include the integration of the normal vectors (S_2 and S_5) in the model-free and model-based case. Furthermore, S_3 and S_6 each represent the complete, already introduced, state modeling variants. In addition to the information $\Delta_{t,m}$, these modeling variants also contain the pose $\mathbf{s}_{t,modf,\mathbf{p}}$ (in the model-free case) or $\mathbf{s}_{t,modb,\mathbf{p}}$ (in the model-based case) as input. It should be noted that the poses in the model-free and model-based case do not differ ($\mathbf{s}_{t,modf,\mathbf{p}} = \mathbf{s}_{t,modb,\mathbf{p}}$)

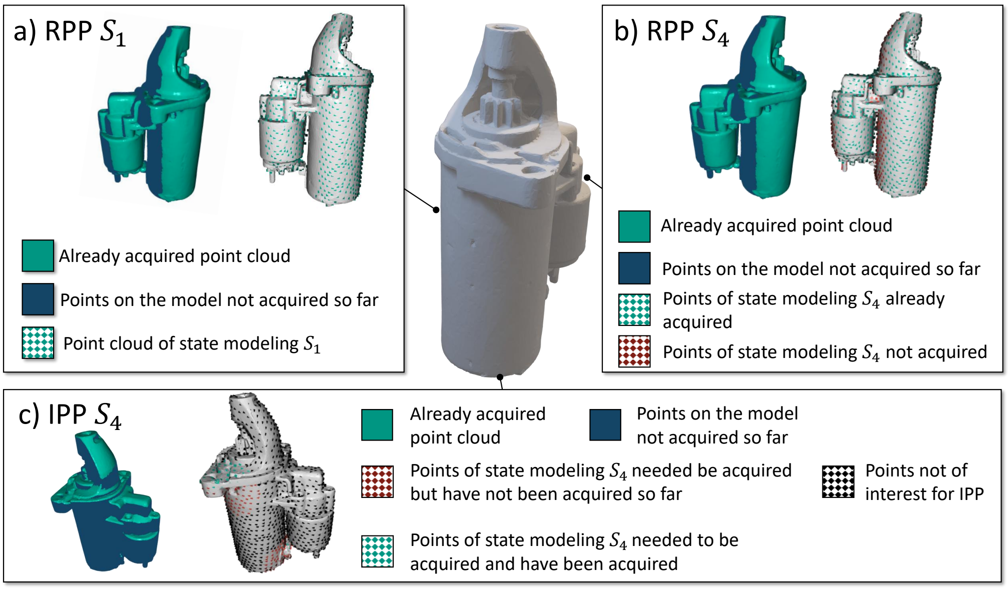

a) RPP S_1
🟩 Already acquired point cloud
🟦 Points on the model not acquired so far
▦ Point cloud of state modeling S_1
b) RPP S_4
🟩 Already acquired point cloud
🟦 Points on the model not acquired so far
▦ Points of state modeling S_4 already acquired
▦ Points of state modeling S_4 not acquired
c) IPP S_4
🟩 Already acquired point cloud
🟦 Points on the model not acquired so far
▦ Points of state modeling S_4 needed be acquired but have not been acquired so far
▦ Points not of interest for IPP
▦ Points of state modeling S_4 needed to be acquired and have been acquired

Figure 5.18: Visualisation of selected modelling variants of the state.

A visualization of the process of generating the state $\mathbf{s}_t$ can be found in Figure 5.18. The Figure shows how the state S_1 is derived in the case of RPP (model-free case in subfigure a) with knowledge of all points acquired so far. In this case, only all previously acquired points on the surface of the inspection object (green) are sampled down to a fixed size. In contrast, the state generation process is shown in the model-based case for the RPP with state definition S_4 (subfigure b). Since knowledge of the object model can be assumed in this case, information can also be included regarding which surface points have already been acquired (green) and which have not (red). This color coding corresponds to the variable $b_{t,m}$ introduced in the text. In the case of RPP, this assumes the values 1 (a surface point already acquired) and 0 (a surface point not yet acquired). Similarly, the state modeling S_4 is used in the model-based case for the IPP. In this case, only a third coding type ($b_{t,m} = 0$) is introduced for points that are not to be acquired and are shown in black in subfigure c of Figure 5.18.

5.3.4.2 Action module

Based on a state $\mathbf{s}_t \in [\mathbf{s}_{t,modf}, \mathbf{s}_{t,modb}]$, the action module generally receives an n-dimensional action vector $\mathbf{a}_t$ as the agent's output based on $\mathbf{s}_t$ in each interaction step with the environment. To ensure consistency of interaction between I_{Agent} and A_{Plan}, it is defined that each component of the action vector $\mathbf{a}_t$ in equation 5.6 contains values in the range between -1 and 1. This is due to the fact, that the NN, that will output the action vector $\mathbf{a}_t$, are limited in the value range due to the activation functions used in the last layer of the NN. For more information on the NN used in this thesis, see section 5.3.5 for an in-depth explanation.

$$\mathbf{a}_t = (a_{t,1}, ..., a_{t,n}) \quad a_{t,i} \in [-1, 1] \quad \text{with} \quad i = \{1, ...n\} \tag{5.6}$$

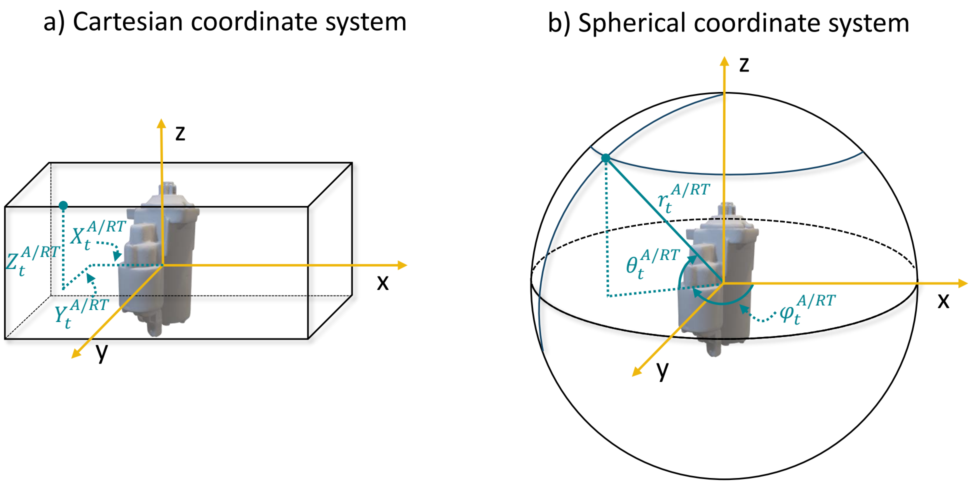
The action module then maps the action vector $\mathbf{a}_t$ to a VP $V_t^S = (\mathbf{p}_t^{A/RT}, \mathbf{p}_t^{O/RT}, \lambda_A)$ of the acquisition system and object. The need for an action module arises due to the following reasons:

- A mapping between output values of the action vector $\mathbf{a}_t$ and their relationship to the components of the poses $\mathbf{p}_t^{A/RT}$ and $\mathbf{p}_t^{O/RT}$ of the acquisition systems coordinate system C_A and objects coordinate system C_O must be established.
- Due to the restriction of the value ranges of the components $a_{t,i}$ (between -1 and 1) of the action vector $\mathbf{a}_t$, an adjustment of these value ranges to the real working range of the inspection station (robot and rotary table) has to be carried out.
- The action module allows for the integration of additional prior knowledge.

The tasks of the action module listed above are explained in detail below.

Coordinate system mapping
As explained in section 5.3.3, the pose $\mathbf{p}_t^{A/RT}$ is a vector with the components $X_t^{A/RT}$, $Y_t^{A/RT}$, $Z_t^{A/RT}$, $\alpha_t^{A/RT}$, $\beta_t^{A/RT}$ and $\gamma_t^{A/RT}$. Furthermore, the pose $\mathbf{p}_t^{O/RT}$ of the inspection object depends on the rotation angle $\varphi_{RT,t}$ of the rotary table. In the most straightforward case, the values of the action vector $\mathbf{a}_t = (a_{t,1}, ..., a_{t,n})$ are mapped directly to the components of the poses $\mathbf{p}_t^{A/RT}$ and $\mathbf{p}_t^{O/RT}$. This results in an action vector with seven entries. Three for the Cartesian coordinates $X_t^{A/RT}$, $Y_t^{A/RT}$, $Z_t^{A/RT}$, three for the Euler angles $\alpha_t^{A/RT}$, $\beta_t^{A/RT}$ and $\gamma_t^{A/RT}$ and one entry for the relative rotation angle $\Delta\varphi_{RT,t}$ of the rotary table. The agent can, therefore, not influence the absolute angle φ_{RT} of the rotary table but only the relative rotation based on the current absolute angle.

However, an intermediate step using spherical coordinates is also possible for mapping the position of the acquisition system. In this case, the first three components of the action vector initially represent the spherical coordinates azimuth angle $\varphi_t^{A/RT}$, polar angle $\theta_t^{A/RT}$, and the radius $r_t^{A/RT}$. As in the first case, the last three components correspond to the Euler angles $\alpha_t^{A/RT}$, $\beta_t^{A/RT}$ and $\gamma_t^{A/RT}$. Conversion to Cartesian coordinates is then carried out using the relationships explained in Appendix A2. A potential advantage of defining the components of the action vector in spherical coordinates lies in the dependence of the distance of the acquisition system from the rotary table and, therefore, the object to be inspected on only the parameter $r_t^{A/RT}$. This suggests that the RL agent can easily learn the positioning of the acquisition system from the object at an optimal working distance.


a) Cartesian coordinate system

b) Spherical coordinate system

Figure 5.19: Visualization of the mapping of the agent output to the position in space in Cartesian coordinates (left, subfigure a)) or spherical coordinates (right, subfigure b))

An exemplary visualization of the mapping of the agent output to the position of the next acquisition pose is shown in Figure 5.19.

Adjustment of value ranges

By restricting the values of the components $a_{t,i}$ in the range between -1 and 1, the value ranges must subsequently be adjusted, given an assignment of the components of the action vector to components of a coordinate system. Assuming a unique assignment of the components of the action vector $\mathbf{a}_t$ to Cartesian coordinates and a square workspace around the center of the inspection object, the first three components $a_{t,1}, a_{t,2}, a_{t,3}$ of $\mathbf{a}_t$ can be mapped, to the Cartesian coordinates $X_t^{A/RT}$, $Y_t^{A/RT}$ and $Z_t^{A/RT}$ in the value range $[-100, 100]$. In this case, a mathematical mapping $(a_{t,1}, a_{t,2}, a_{t,3}) \in [-1, 1] \to (X_t^{A/RT}, Y_t^{A/RT}, Z_t^{A/RT}) \in [-100, 100]$ takes place using *Min-Max scaling*. Similarly, using spherical coordinates, this implies that a mapping of $a_{t,i}$ with the value ranges of $[-1, 1]$ to $[0, 2\pi]$ for the azimuth angle $\varphi_t^{A/RT}$ ($a_{t,1} \in [-1, 1] \to \varphi_t^{A/RT} \in [0, 2\pi]$) as well as $[0, \pi]$ for the polar angle $\theta_t^{A/RT}$ ($a_{t,2} \in [-1, 1] \to \theta_t \in [0, \pi]$) and, for example, $[0, 100]$ for the radius $r_t^{A/RT}$ ($a_{t,3} \in [-1, 1] \to r_t \in [0, 100]$), is performed. Similarly, the angles $\alpha_t^{A/RT}$, $\beta_t^{A/RT}$ and $\gamma_t^{A/RT}$ as well as the relative rotation angle of the rotary table $\Delta\varphi_{RT,t}$ are scaled to valid value ranges. The explained relationships can be found in Table 5.5.

Based on these relationships, the poses $\mathbf{p}_t^{A/RT}$ and $\mathbf{p}_t^{O/RT}$ of the acquisition system and inspection object as part of the VP V_t^S can be determined based on the output $\mathbf{a}_t$ of the agent.

Table 5.5: Overview of the adjustment of the value ranges of the individual mapped components of the action vector to the components of the poses $\mathbf{p}_t^{A/RT}$ and $\mathbf{p}_t^{O/RT}$. (a) Value ranges when using Cartesian coordinates. (b) Value ranges when using spherical coordinates.

Parameter	Value range
$X_t^{RT}, Y_t^{RT}, Z_t^{RT}$	$[-100, 100]$
α_t^{RT}, γ_t^{RT}	$[-180°, 180°]$
β_t^{RT}	$[-\frac{180°}{2}, \frac{180°}{2}]$
$\Delta\varphi_{RT,t}$	$[0°, 360°]$

Parameter	Value range
φ_t^{RT}	$[0, 360°]$
θ_t^{RT}	$[0, \frac{180°}{2}]$
r_t^{RT}	$[0, 100]$
α_t^{RT}, γ_t^{RT}	$[-180°, 180°]$
β_t^{RT}	$[-\frac{180°}{2}, \frac{180°}{2}]$
$\Delta\varphi_{RT,t}$	$[0°, 360°]$

When calculating the pose $\mathbf{p}_t^{O/RT}$, however, it must be noted that the absolute angle of rotary table must be used to calculate the pose, which can be retrieved by I_{Agent} from the ROS parameter server (see 5.1.2). This is because the agent can only output one relative angle $\Delta\varphi_{RT,t}$ and can, therefore, only be used to determine the relative pose $\Delta\mathbf{p}_t^{O/RT}$.

Integration of prior knowledge

The modeling of the action vector presented so far assumes that the agent can and has to influence all degrees of freedom of the system. For the given problem of visual planning, however, prior knowledge can be used meaningfully to reduce the decision complexity of the agent and thus simplify the visual planning problem. This way, it is assumed that agents can be trained faster since the solution space is restricted and the exploration of non-optimal strategies is excluded from the beginning.

The variants of prior knowledge considered in this thesis are:

1. Reasonable restriction of the value ranges to which individual agent outputs are mapped.
2. Use of analytical determination of individual components of the poses $\mathbf{p}_t^{A/RT}$ and $\mathbf{p}_t^{O/RT}$ to simplify the learning task of the RL agent.

An overview of different possibilities of integrating prior knowledge for mapping the agent output to poses of the acquisition system is listed in Table 5.7.

In the case of spherical coordinates, limiting the radius $r_t^{A/RT}$ to a small and fixed range or even a fixed value, exploiting knowledge of the optimal working distance of the acquisition system, is possible ($M_{r,fix}$ and $M_{r,range}$). Concerning the orientation of the acquisition system, a heuristical approach can be applied to ensure that the acquisition system is always oriented toward the center of gravity of the inspection object ($M_{\alpha,\beta}$). The calculation rule is identical to the rule for data generation for segmentation already mentioned in section 5.2.1. In this case, the angle $\gamma_t^{A/RT}$, which corresponds to the rotation around the longitudinal axis of the acquisition system, is determined so that the y-axis of C_A is aligned coplanar to the x-y plane of the coordinate system C_{RT} of the rotary table $M_{\gamma,align}$. Thus, it is possible to have the RL agent learn the position of the acquisition systems pose $\mathbf{p}_t^{A/RT}$, while the orientation is always being specified heuristically. The number of entries of the action vector $\mathbf{a}_t$ as the output of the agent can thus be reduced to three parameters (representing $X_t^{A/RT}$, $Y_t^{A/RT}$ and $Z_t^{A/RT}$ when using Cartesian coordinates or $\varphi_t^{A/RT}$, $\theta_t^{A/RT}$ and $r_t^{A/RT}$ when using spherical coordinates). An alternative is $M_{\Delta\alpha,\Delta\beta}$, where $\alpha_t^{A/RT}$ and $\beta_t^{A/RT}$ are oriented towards C_{RT}, but the agent can specify relative angle deviations $\Delta\alpha_t^{A/RT}$, $\Delta\beta_t^{A/RT}$, as in the case of the rotation angle of the rotary table.

If the planning problem is initially to be solved without considering the restrictions on the accessibility of poses by the robot model, further simplifications can be integrated. This concerns, for example, the angle $\gamma_t^{A/RT}$. $\gamma_t^{A/RT}$, has almost no influence on the acquired points of the object surface due to its almost rotationally symmetric frustum but has a large influence on the joint angles of the robot required to reach a pose $\mathbf{p}_t^{A/RT}$. If the robot's trajectory planning is excluded, $\gamma_t^{A/RT}$ can be fixed, resulting in $M_{\gamma,fix}$ (e.g., $\gamma_t^{A/RT} = 0$). In this case, the planning problem can also be simplified to the extent that $\Delta\varphi_{RT,t}$ can be set to zero, since any pose $\mathbf{p}_t^{A/RT}$ in space can be reached by the acquisition system and rotation of the inspection object is not necessarily required to completely cover the surface of the inspection object. This leads to $M_{\Delta\varphi_{RT,t},fix}$ ($\Delta\varphi_{t,RT} = 0$).

If the robot's trajectory planning is considered, it is useful to directly limit unreachable poses in space. This applies in particular to poses of the acquisition system in space whose positions are too far away from the robot base C_R. In this case, the value ranges of the spherical coordinates $\varphi_t^{A/RT}$ and $\theta_t^{A/RT}$ can be restricted, which leads to $M_{\varphi,\theta,range}$. In this case, the ranges of $\varphi_t^{A/RT}$ and $\theta_t^{A/RT}$ are chosen, so that the resulting pose $\mathbf{p}_t^{A/R}$ of C_A in C_R (after coordinate transformation of $\mathbf{p}_t^{A/RT}$ to $\mathbf{p}_t^{A/R}$) is reachable with high probability.

Variants of implemented action alternatives

With the variants of coordinate system mapping introduced and the options for integrating prior knowledge presented, there are several alternative variants of possible action encoding for the agents. These variants vary depending on the degrees of freedom (DoF) the agent exhibits and the prior knowledge used to restrict the DoF of the agent. An excerpt of the implemented variants can be found in Table 5.8 for the Spherical action modeling variants

Table 5.7: Overview of valid possibilities to integrate prior knowledge into the action mapping.

Abbreviation	Parameter	Description
$M_{r,fix}$	$r_t^{A/RT}$	Fixed value of $r_t^{A/RT}$ ($r_t^{A/RT} = 470mm$).
$M_{r,range}$	$r_t^{A/RT}$	$r_t^{A/RT}$ is set in a small range around working distance of the acquisition system ($r_t^{A/RT} \in [450mm, 550mm]$).
$M_{\gamma,fix}$	$\gamma_t^{A/RT}$	Fixed value of $\gamma_t^{A/RT}$ ($\gamma_t^{A/RT} = 0°$).
$M_{\gamma,align}$	$\gamma_t^{A/RT}$	Choose $\gamma_t^{A/RT}$ to align the y-axis of C_A coplanar to the x-y plane of C_{RT}.
$M_{\alpha,\beta}$	$\alpha_t^{A/RT}$, $\beta_t^{A/RT}$	$\alpha_t^{A/RT}$ and $\beta_t^{A/RT}$ are determined analytically to face C_{RT}.
$M_{\Delta\alpha,\Delta\beta}$	$\alpha_t^{A/RT}$, $\beta_t^{A/RT}$	$\alpha_t^{A/RT}$ and $\beta_t^{A/RT}$ are determined analytically to face C_{RT}. Angular deviations $\Delta\alpha_t^{A/RT}, \Delta\beta_t^{A/RT} \in [-10°, 10°]$ from these heurstically calculated angles $\alpha_t^{A/RT}$ and $\beta_t^{A/RT}$ are determined by the agent.
$M_{\Delta\varphi_{RT,t},fix}$	$\Delta\varphi_{t,RT}$	$\Delta\varphi_{t,RT} = 0$ for the whole acquisition cycle.
$M_{\varphi,\theta,range}$	$\varphi_t^{A/RT}$, $\theta_t^{A/RT}$	$\varphi_t^{A/RT}$ and $\theta_t^{A/RT}$ are restricted in the ranges $\varphi_t^{A/RT} \in [45°, 135°]$ and $\theta_t^{A/RT} \in [22.5°, 90°]$.

and in Table 5.9 for the cartesian action modeling variants. These are explained briefly in the following.

The most basic modeling variant is the variant with the abbreviation $A_{2S,0R}$. In this case, spherical coordinates are used for the mapping. The agent has two degrees of freedom $\varphi_t^{A/RT}$ and $\theta_t^{A/RT}$. The distance $r_t^{A/RT}$ of the acquisition system to the center of the inspection system is fixed and the orientation is determined analytically, where $\gamma_t^{A/RT}$ is always set to zero. Furthermore, the agent cannot rotate the inspection object ($\Delta\varphi_{RT,t}$). In comparison to $A_{2S,0R}$, the action coding $A_{3S,0R}$ allows the agent to additionally control the distance of the acquisition system to the inspection object using the radius $r_t^{A/RT}$. The encoding variant $A_{3S,2R}$ provides the agent with the ability to freely define its orientation using Euler angles $\alpha_t^{A/RT}$ and $\beta_t^{A/RT}$. However, $\gamma_t^{A/RT}$ is still set to zero. In contrast, $A_{3S,2R,lim}$ determines the orientation ($\alpha_t^{A/RT}$, $\beta_t^{A/RT}$) analytically, but the agent can specify deviations $\Delta\alpha_t^{A/RT}$ and $\Delta\beta_t^{A/RT}$. The last action variant presented using a mapping based on spherical coordinates is $A_{3S,2R,lim,T}$. This variant enables the object to be rotated ($\Delta\varphi_{RT,t}$) but has restrictions in the angle range of the angles $\varphi_t^{A/RT}$ and $\theta_t^{A/RT}$ that can be approached. In addition, prior knowledge is integrated by limiting the distance $r_t^{A/RT}$ to the inspection object. This variant of action encoding is particularly useful and necessary when considering the agent system

Table 5.8: Spherical RL action modeling variants. Lowercase numbers in the abbreviation denote DoF. Lowercase letters denote either the coordinate system (S for spherical) or the rotation R to which the DoF are allocated. T indicates whether rotary table rotation is enabled.

Spherical coordinate mapping alternatives

Abbreviation	**DoF poses $\mathbf{p}_t^{A/RT}$ and $\mathbf{p}_t^{O/RT}$**	**Fixed parameters**	**Prior knowledge**
$A_{2S,0R}$	$a_{t,1} \hat{=} \varphi_t^{A/RT}$ $a_{t,2} \hat{=} \theta_t^{A/RT}$	$\alpha_t^{A/RT} \hat{=} analytical$ $\beta_t^{A/RT} \hat{=} analytical$ $\gamma_t^{A/RT} = 0°$ $r_t^{A/RT} = 470mm$ $\Delta\varphi_{RT,t} = 0°$	$M_{\alpha,\beta}$ $M_{\gamma,fix}$ $M_{\Delta\varphi_{RT,t},fix}$ $M_{r,fix}$
$A_{2S,0R,T}$	$a_{t,1} \hat{=} \varphi_t^{A/RT} \in [45°, 135°]$ $a_{t,2} \hat{=} \theta_t^{A/RT} \in [22.5°, 90°]$ $a_{t,3} \hat{=} \Delta\varphi_{RT,t}$	$\alpha_t^{A/RT} \hat{=} analytical$ $\beta_t^{A/RT} \hat{=} analytical$ $\gamma_t^{A/RT} = 0°$ $r_t^{A/RT} = 470mm$	$M_{\alpha,\beta}$ $M_{\gamma,fix}$ $M_{r,fix}$
$A_{3S,0R}$	$a_{t,1} \hat{=} \varphi_t^{A/RT}$ $a_{t,2} \hat{=} \theta_t^{A/RT}$ $a_{t,3} \hat{=} r_t^{A/RT}$	$\alpha_t^{A/RT} \hat{=} analytical$ $\beta_t^{A/RT} \hat{=} analytical$ $\gamma_t^{A/RT} = 0°$ $\Delta\varphi_{RT,t} = 0°$	$M_{\alpha,\beta}$ $M_{\gamma,fix}$ $M_{\Delta\varphi_{RT,t},fix}$
$A_{3S,2R}$	$a_{t,1} \hat{=} \varphi_t^{A/RT}$ $a_{t,2} \hat{=} \theta_t^{A/RT}$ $a_{t,4} \hat{=} \alpha_t^{A/RT}$ $a_{t,3} \hat{=} r_t^{A/RT}$ $a_{t,5} \hat{=} \beta_t^{A/RT}$	$\gamma_t^{A/RT} = 0°$ $\Delta\varphi_{RT,t} = 0°$	$M_{\gamma,fix}$ $M_{\Delta\varphi_{RT,t},fix}$
$A_{3S,2R,lim}$	$a_{t,1} \hat{=} \varphi_t^{A/RT}$ $a_{t,2} \hat{=} \theta_t^{A/RT}$ $a_{t,3} \hat{=} r_t^{A/RT}$ $a_{t,4} \hat{=} \Delta\alpha_t^{A/RT}$ $a_{t,5} \hat{=} \Delta\beta_t^{A/RT}$	$\gamma_t^{A/RT} = 0°$ $\Delta\varphi_{RT,t} = 0°$	$M_{\Delta\alpha,\Delta\beta}$ $M_{\gamma,fix}$ $M_{\Delta\varphi_{RT,t},fix}$
$A_{3S,2R,lim,T}$	$a_{t,1} \hat{=} \varphi_t^{A/RT} \in [45°, 135°]$ $a_{t,2} \hat{=} \theta_t^{A/RT} \in [22.5°, 90°]$ $a_{t,3} \hat{=} r_t^{A/RT} \in [400mm, 600mm]$ $a_{t,4} \hat{=} \Delta\alpha_t^{A/RT}$ $a_{t,5} \hat{=} \Delta\beta_t^{A/RT}$ $a_{t,6} \hat{=} \Delta\varphi_{RT,t}$	$\gamma_t^{A/RT} = 0°$	$M_{r,range}$ $M_{\gamma,fix}$ $M_{\varphi,\theta,range}$

Table 5.9: Cartesian RL action modeling variants. Lowercase numbers in the abbreviation denote DoF. Lowercase letters denote either the coordinate system (C for Cartesian) or the rotation R to which the DoF are allocated.

Cartesian coordinate mapping alternatives			
Abbreviation	**DoF poses p$_t^{A/RT}$ and p$_t^{O/RT}$**	**Fixed parameters**	**Prior knowledge**
$A_{3C,0R}$	$a_{t,1}\hat{=}X_t^{A/RT}$ $a_{t,2}\hat{=}Y_t^{A/RT}$ $a_{t,3}\hat{=}Z_t^{A/RT}$	$\alpha_t^{A/RT}\hat{=}analytical$ $\beta_t^{A/RT}\hat{=}analytical$ $\gamma_t = 0°$ $\Delta\varphi_{RT,t} = 0°$	$M_{\alpha,\beta}$ $M_{\gamma,fix}$ $M_{\varphi_{RT,t,t},fix}$

in conjunction with the developed robot simulation, as this variant already maps the agent outputs into accessible workspace areas of the robot system. The rotation of the inspection object then enables the agent to perform a complete object inspection. A simplified version of this action variant is $A_{2S,0R,T}$. In this case, both the radius $r_t^{A/RT}$ is fixed and the angles $\alpha_t^{A/RT}$ and $\beta_t^{A/RT}$ are determined analytically.

An analogous approach can also be used in the case of mapping to Cartesian coordinates (table 5.9). In the case of action mapping $A_{3C,0R}$, the three action outputs $a_{t,1}, a_{t,2}$ and $a_{t,3}$ of the action vector **a**$_t$ are mapped to the degrees of freedom $X_t^{A/RT}, Y_t^{A/RT}$ and $Z_t^{A/RT}$. An integration of prior knowledge to determine the remaining degrees of freedom can be done analogously to the case of mapping to spherical coordinates.

5.3.4.3 Reward module

To learn a desired behavior that solves the acquisition planning problem as well as possible and to continuously evaluate the actions performed, an RL agent receives feedback in the form of a numerical reward signal r_t. The challenge in RL lies in breaking down the overall goal, which the agent must fulfill through its strategy in interaction with the environment, to this one numerical value. The reward signal must, therefore, be expressed as a function of evaluation metrics that indicate the degree of fulfillment of an inspection target. In accordance with 2.2.1 and section 5.3.1, the evaluation metrics considered in this thesis for the planning problems NBV, RPP and IPP are chosen as follows:

- **Achieved surface coverage** Ψ_E of the target point cloud **P**$_G^{RT'}$ after one (NBV) or several acquisitions (RPP, IPP) after an episode. For NBV and RPP, **P**$_G^{RT'}$ corresponds to the ground-truth point cloud **P**$_{GT}^{RT'}$ of the inspection object, while for IPP only a subset of all surface points of the inspection object, i.e. only certain ROI, must be acquired (**P**$_G^{RT'} \subset$ **P**$_{GT}^{RT'}$).

- **Number n_E of acquisitions** required for completing the inspection goal, i.e., reaching goal state s_G, or the termination state s_T. Only relevant for RPP and IPP since for NBV $n_E = 2$.
- **Overall length d_E of traveling distances** between the successively determined poses by the agent to fulfill the inspection goal (reaching state s_G) or after termination by exceeding a predefined allowable number of acquisitions (reaching state s_T).

Further evaluation metrics not considered in this thesis may depend on the VPP solution's application area. For metrological applications, the accuracy of an acquired point cloud or its resolution may be important (Scott 2009).

Considering the evaluation metrics mentioned above, it is useful to express the reward signal R as a function of these variables as in equation 5.7.

$$R = f(\Psi_E, n_E, d_E) \tag{5.7}$$

In the literature and in this thesis, two types of reward signals, R_{dense} and R_{sparse}, are distinguished (Eschmann 2021, P. 32). The dense reward R_{dense} describes a reward signal where the agent receives a reward after each action performed. The sparse reward signal R_{sparse} describes a reward signal, where the agent only receives a reward if a terminal state s_T is reached.

Specifically for the dense reward R_{dense}, auxiliary metrics need to be considered since, particularly at the beginning of a data acquisition process, the impact of an action on the evaluation metrics Ψ_E, n_E, and d_E can be difficult to estimate. Auxiliary metrics include, for example, the percentage of surface points $\Delta\Psi_t$ acquired with an acquisition, which have not been covered in the previous acquisitions. Furthermore an auxiliary metric is the length d_t as the travel length between the previous pose ($\mathbf{p}_{t-1}^{A/RT}$) and the pose output by the agent ($\mathbf{p}_t^{A/RT}$). Dense rewards, thus, offer the advantage of making it easier for the agent to learn a strategy since it receives a reward after each step. However, the challenge is to choose auxiliary metrics to construct R_{dense} in such a way that actions are chosen so that they contribute to fulfilling the long-term goal expressed by the evaluation metrics Ψ_E, n_E, and d_E. An insufficient choice of auxiliary metrics can make it impossible for the agent to find the optimal solution to the problem.

In contrast, sparse rewards R_{sparse} are directly based on the evaluation metrics Ψ_E, n_E, and d_E after an episode ends. However, in this case, the agent receives a reward only once in an episode. This results in a difficult credit-assignment problem, which involves determining which of the actions taken by the agent were more influential on the outcome (good or bad) than the others.

The evaluation and auxiliary metrics must be quantitatively approximated from the observation $\mathbf{o}_t$ of the scan simulation environment S_{Sim} during the agent's learning process. These approximations are detailed in the following.

Calculation of surface coverage

To calculate the surface coverage Ψ_E after an episode or auxiliary metrics related to surface coverage, the point cloud $\mathbf{P}_{cov,t}^{RT'}$ acquired until interaction step t is first downsampled. This downsampling is done to match the point density (voxel grid size) of the ground truth point cloud $\mathbf{P}_{GT}^{RT'}$ using a voxel grid approach.

The first auxiliary metric is the *percentual surface coverage* Ψ_t after t acquisition steps in the acquisition process. Ψ_t can be estimated by taking into account the maximum number of possible ground truth points n_{GT} to be acquired and the $n_{cov,t}$ acquired points on the ground truth point cloud $\mathbf{P}_{GT}^{RT'}$ by the acquisition system until interaction step t. $n_{cov,t}$ is calculated by assigning each point of the acquired point cloud $\mathbf{P}_{cov,t}^{RT'}$ to a point of the ground truth point cloud $\mathbf{P}_{GT}^{RT'}$ if its nearest neighbor in $\mathbf{P}_{GT}^{RT'}$ falls below a certain threshold distance ϵ. The parameter ϵ is thereby dependent on the point cloud density of $\mathbf{P}_{cov,t}^{RT'}$ and $\mathbf{P}_{GT}^{RT'}$ and consequently the point distances between them. Therefore, the parameter ϵ is chosen heuristically, resulting in $\epsilon = 0.2$ in this thesis. Note that a higher density of $\mathbf{P}_{cov,t}^{RT'}$ and $\mathbf{P}_{GT}^{RT'}$ should result in choosing a lower ϵ. The object coverage Ψ_t can then be estimated using equation 5.8.

$$\Psi_t = \frac{n_{cov,t}}{n_{GT}} \tag{5.8}$$

Similarly, based on the same calculation method, the percentage of object surface in the last acquisition $\Delta\Psi_t$, not yet covered by previous acquisitions, can be calculated. This auxiliary metric is a function of the new surface points $\Delta n_{cov,t}$ acquired in the last acquisition at step t and the number of points in the ground truth point cloud n_{GT}. It can also be calculated via $\Delta\Psi_t = \Psi_t - \Psi_{t-1}$. As the last auxiliary metric, the percentage of *remaining object surface* to be covered $\Psi_{t,rem} = 1 - \Psi_t$ is an additional metric that can be used for the reward calculation.

It should be noted that calculating the surface coverage of individual acquisitions during the training process, which takes place during the calculation of the reward signal, is only possible based on an available three-dimensional geometry model of the object. This is permissible since geometry models must be part of the training process. The distinction between model-based and model-free only arises when using the fully trained RL agent, provided that information about the geometry model is used to calculate the state $\mathbf{s}_{t,modb}$

(model-based, the geometry model must also be available when using the RL agent) or is not used $\mathbf{s}_{t,modf}$ (model-free, no geometry model is required).

Approximation of the length of traveling distance

For industrial application cases, the length of the traveling paths is a relevant evaluation metric. It significantly influences the execution time of an acquisition plan and, therefore, directly influences the inspection time. Using *MoveIt* to determine a trajectory between points and is a simple way to afterward calculate the length of a trajectory between an initial pose $\mathbf{p}_{t-1}^{RT}$ and a final pose $\mathbf{p}_{t}^{RT}$. For the framework developed in this thesis, estimating the trajectories and calculating the associated reward should also be possible without the explicit use of *MoveIt*. Therefore, a reasonable and conservative approximation of the travel distances given the acquisition system's initial and final pose is used. The simplified approximation of the *travel distance* d_t $(t = 1, ..., T)$ between a start pose $\mathbf{p}_{t-1}^{RT}$ and an end pose $\mathbf{p}_{t}^{RT}$ of the acquisition system is carried out via the estimation of the length of a circular segment and a downstream linear travel path (see equation 5.9). This simplified approximation is visualized in Figure 5.20 and explained in the following.

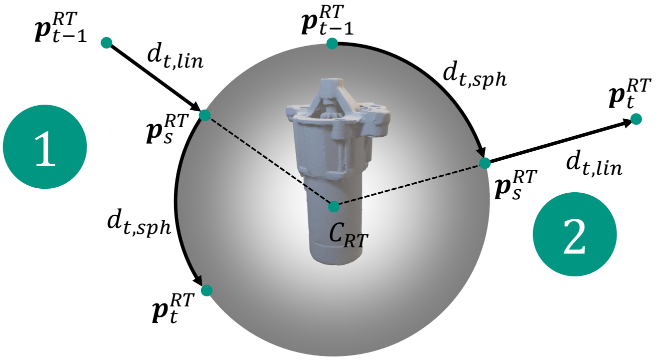


Figure 5.20: Visualization of the approximation of travel distance d_t

First, the Cartesian coordinates of the auxiliary point $\mathbf{p}_{s}^{RT}$ are calculated. To this end, it is first checked which of the two poses ($\mathbf{p}_{t}^{RT}$ or $\mathbf{p}_{t-1}^{RT}$) has the smaller distance to the origin, which is the coordinate system C_{RT} of the rotary table. Two different cases have then to be distinguished based on which point ($\mathbf{p}_{t}^{RT}$ or $\mathbf{p}_{t-1}^{RT}$) has the greater distance to C_{RT}. Both cases are visualized in Figure 5.20. An auxiliary point is calculated as the point that lies on the vector between the origin and the pose with the greater distance to the origin and has a distance to the origin that is equal to the distance of the pose with the smaller distance to the origin. $d_{t,lin}$ then equals the travel distance from $\mathbf{p}_{s}^{RT}$ to the pose with the greater distance to the origin. $d_{t,sph}$ correspondingly equals the length of the circular arc between the pose with the smaller distance to the origin and $\mathbf{p}_{s}^{RT}$.

$$d_t = d_{t,sph} + d_{t,lin} \tag{5.9}$$

In the case of an equal radius, the travel distance solely consists of the distance $d_{t,sph}$ on the spherical surface. The overall traveling distance d_E as an evaluation metric for an episode can then be calculated as the sum of the travel distances d_t of all acquisitions until s_T or s_G is reached at step $t = T$ (see equation 5.10).

$$d_E = \sum_{t=0}^{t=T} d_t \tag{5.10}$$

Overview of implemented reward alternatives

An overview of the implemented variants of dense and sparse rewards can be found in Table 5.10. The first reward variant R_1 rewards the agent after each acquisition based on the percentage of newly covered surface area scaled by the percentage of object surface area still acquirable before said acquisition. R_2 only rewards the agent by the percentage of newly acquired surface area and is thus a simpler version of R_1. The choice of designing reward R_1 this way is reasonable, since rewarding the agent only relative to the area newly covered (R_2), leads to continuously diminishing rewards in later acquisition steps. This is because the object surface to be newly covered relative to the total surface area gets smaller, in case previous acquisition steps already provided a high coverage. Therefore, the reward signal R_1 considers the object surface covered in an acquisition step relative to the object surface that can still be acquired. R_3 additionally scales R_1 with the travel distance required to reach the pose $\mathbf{p}_t^{A/RT}$ specified by the agent for acquisition. This reward signal can thus guide the agent to choose actions that provide high surface coverage with low travel distances. Variations of reward functions R_1 and R_2 are R_4 and R_5, where the agent receives a negative reward if no additional object surface is acquired.

Similar approaches are adopted based on the agent's reward after an episode in a sparse reward setting. R_6 rewards the agent only if the agent fulfills the $\Psi_E \geq 90\%$ surface coverage goal. R_7 relates that reward to the achieved surface coverage Ψ_E. R_8 additionally introduces a scaling concerning the number of acquisitions needed to achieve the respective coverage goal. Thus, the agent explicitly receives higher rewards for achieving a large object surface coverage while minimizing the number of acquisitions required. An additional scaling by the cumulative travel costs d_E in an episode is used in R_9.

Table 5.10: Implemented modeling variants of the reward signal for the reinforcement learning agent.

Abbreviation	Type	Goal	Formula
R_1	Dense	$\Psi_t \geq 90\%$	$r_t = \frac{\Delta\Psi_t}{\Psi_{rem,t-1}}$
R_2	Dense	$\Psi_t \geq 90\%$	$r_t = \Delta\Psi_t$
R_3	Dense	$\Psi_t \geq 90\%$	$r_t = \frac{\Delta\Psi_t}{\Psi_{rem,t-1}}\frac{1}{d_t}$
R_4	Dense	$\Psi_t \geq 90\%$	$r_t = \begin{cases} \Delta\Psi_t, & \Delta\Psi_t > 0 \\ -1, & otherwise \end{cases}$
R_5	Dense	$\Psi_t \geq 90\%$	$r_t = \begin{cases} \frac{\Delta\Psi_t}{\Psi_{rem,t-1}}\frac{1}{d_t}, & \Delta\Psi_t > 0 \\ -1, & otherwise \end{cases}$
R_6	Sparse	$\Psi_t \geq 90\%$	$r_t = \begin{cases} 1, & s_t = s_G \\ 0, & s_t \neq s_G \end{cases}$
R_7	Sparse	$\Psi_t \geq 90\%$	$r_t = \begin{cases} \frac{\Psi_E}{n_E}, & s_t = s_G \\ 0, & s_t \neq s_G \end{cases}$
R_8	Sparse	$\Psi_t \geq 90\%$	$r_t = \begin{cases} \frac{\Psi_E}{n_E}, & s_t = s_T \\ 0, & s_t \neq s_T \end{cases}$
R_9	Sparse	$\Psi_t \geq 90\%$	$r_t = \begin{cases} \frac{\Psi_E}{n_E d_E}, & s_t = s_T \\ 0, & s_t \neq s_T \end{cases}$

5.3.5 Modeling of the reinforcement learning agent $A_{Plan,RL}$

To configure the RL agents, the software framework *stable-baselines*[1] (Raffin & Hill et al. 2021) is used, which supports the simple use and parameterization of different RL agents. Due to the continuous state and action space considered, *proximal policy optimization* (PPO) by Schulman & Wolski et al. 2017 and *soft actor critic* (SAC) by Haarnoja & Zhou et al. 2018 are suitable state of the art methods and evaluated. Both methods belong to the group of actor-critic variants of RL. As explained in 2.3.3, these methods use a parameterized policy as an actor and parameterized value function approximation as critic. The parameters of the agent $A_{Plan,RL}$ are denoted by $\mathbf{w}$. $\mathbf{w}$ contains the parameters of the parameterized policy and the value function approximation. The parameters of the value function approximation (critic) are denoted by $\mathbf{w}_\phi$. During the training phase, weights $\mathbf{w}_\phi$ are continuously updated to approximate the value function given the current state. This, in return, enables the evaluation and optimization of the action selection strategy (actor) by updating its weights $\mathbf{w}_\psi$ by gradient-based updates.

[1] Link to online documentation of *stable-baselines*:
https://stable-baselines3.readthedocs.io/en/master/
accessed: 13.06.2024

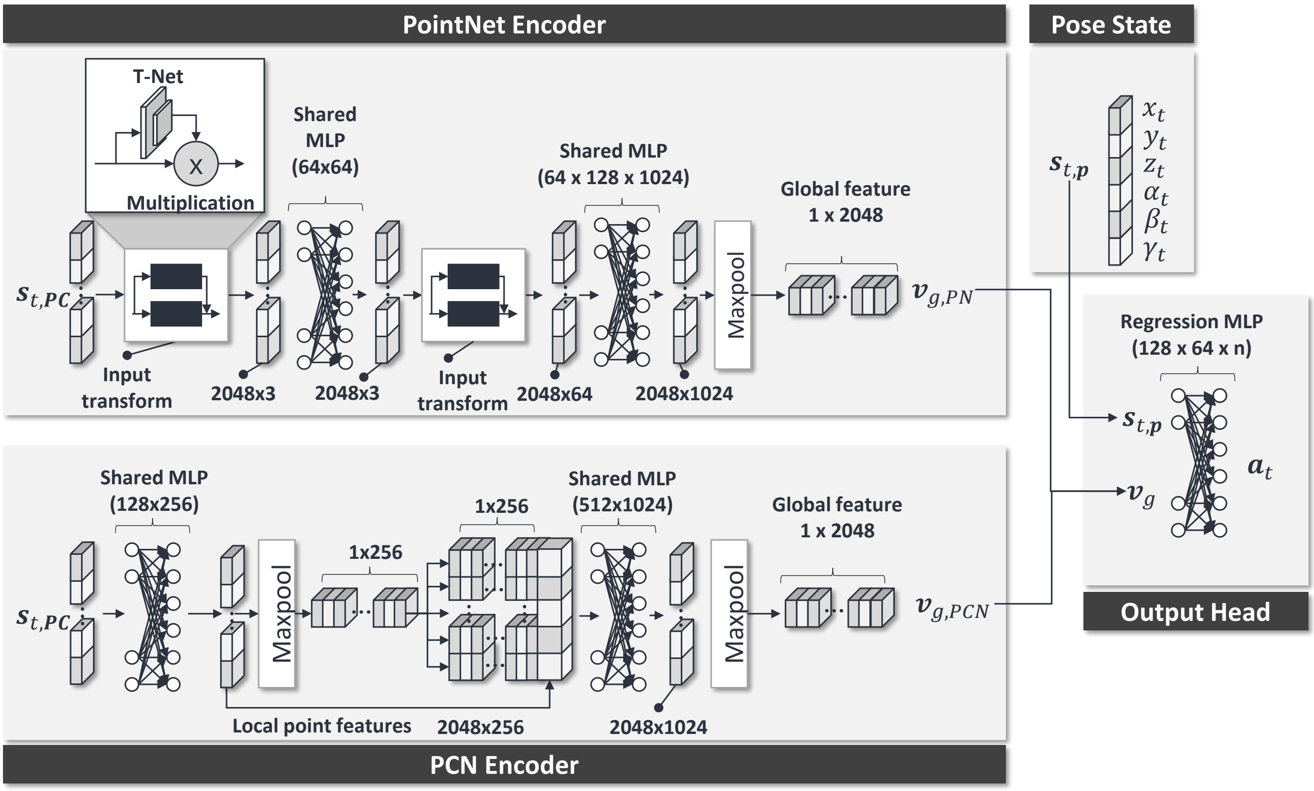


Figure 5.21: Overview of the NN architecture based on PN and PCN encoder for feature extraction and regression MLP for agent output.

The representation of the strategy or the estimators of the value function is carried out, as usual in the current state of research, using neural networks (c.f. Devrim Kaba & Gokhan Uzunbas et al. 2017; Potapova & Artemov et al. 2020; Korbach & Solbach et al. 2021; Landgraf & Meese et al. 2021; Mnih & Kavukcuoglu et al. 2015). Any network architecture that transfers a point cloud as input into a feature vector can be used. Based on preliminary tests, two suitable alternatives for processing point clouds have been identified in a joint work, resulting in a thesis supervised by the author of this dissertation (A_Gäbele 2022). The first alternative is an implementation of *PointNet* (PN) by Qi & Su et al. (2017), the first implementation of a neural network for learning on point clouds. The second option is the use of the feature vector as the output of the encoder of the *Point Completion Network* (PCN) by Yuan & Khot et al. 2018, which builds on the approach of the PN. In contrast to the PN, the PCN does not employ input transforms. Instead, a shared MLP directly calculates local point features. Intermediary global features are obtained and concatenated with the local point features using a max-pooling layer. The final global feature vector $\mathbf{v}_{g,PCN}$ that captures information about the entire point cloud is obtained using another shared MLP and max-pooling. Both alternative network structures, when used in a parameterized policy as an actor to determine the next action $\mathbf{a}_t$ based on state $\mathbf{s}_t = (\mathbf{s}_{t,\mathbf{PC}}, \mathbf{s}_{t,\mathbf{p}})$, can be seen in Figure 5.21. Thereby, $\mathbf{s}_{t,\mathbf{PC}}$ and $\mathbf{s}_{t,\mathbf{p}}$ are either the model-based versions $((\mathbf{s}_{t,\mathbf{PC}}, \mathbf{s}_{t,\mathbf{p}}) = (\mathbf{s}_{t,modb,\mathbf{PC}}, \mathbf{s}_{t,modb,\mathbf{p}}))$ or the model-free version $((\mathbf{s}_{t,\mathbf{PC}}, \mathbf{s}_{t,\mathbf{p}}) = (\mathbf{s}_{t,modf,\mathbf{PC}}, \mathbf{s}_{t,modf,\mathbf{p}}))$ of the state modeling alternatives. It should be noted that when the network is defined, either the PN or the PCN is used as the encoder, and the display of both networks in Figure 5.21 is purely for visualization purposes.

Only the state $\mathbf{s}_{t,\mathbf{PC}}$ is passed through the respective encoder to extract features of the point cloud-based state. The pose-related state $\mathbf{s}_{t,\mathbf{p}}$ is concatenated to the global feature vector after the global feature vector has been extracted by the PCN or PN and passed through a regression MLP to obtain the output of the NN. The output dimension n of the regression MLP depends on the number of DoF of the action modeling used. Compared to the NN of the parameterized policy for action selection, the NN of the critic for estimating the state value function (critic) differs only in that only one output (the value of a state) has to be estimated. The basic structure, consisting of an encoder (either PN or PCN) and a downstream regression MLP, is identical. For details on the hyperparameters used and the network architectures, please refer to section 6.2.2.2 as well as the original works of Qi & Su et al. 2017 and Yuan & Khot et al. 2018.

6 Evaluation and computational results

The approaches presented in the methodological part of this thesis (chapter 5) are evaluated below. The section 6.1 presents the results of the validation of the approaches for the 3D reconstruction of the inspection object and the additional integration of semantic information into the 3D reconstruction process. The following section 6.2 then details the training results of the agents trained with the RL simulation framework developed in this thesis to solve the sub-problems of VPP in remanufacturing.

6.1 Results for the approaches of 3D reconstruction and semantic 3D reconstruction

This section first discusses the validation of the 3D reconstruction approach (section 6.1.1). The segmentation results using the semantic and instance segmentation models are then presented (section 6.1.2). Finally, the semantic 3D reconstruction is evaluated as a whole (section 6.1.3).

6.1.1 Validation of the 3D Reconstruction approach

The following results confirm the effectiveness of the developed reconstruction procedure for 3D modeling of starter motors and potentially any inspection object. This includes validation of the reconstruction process and calibration algorithms (calculation of $\mathbf{T}^{A/E}$ and $\mathbf{T}^{RT/R}$), with results detailed in Table 6.1. An excerpt of the collaborative results presented in the thesis of A_Scheiger 2022 under the author's supervision is presented. The methodology and relevance of these results in supporting the reconstruction approach are then discussed.

The analysis compares point clouds from the reconstruction approach with those from the highly accurate handheld laser scanning system *Zeiss T-Scan*[1]. A small point distance between the models indicates that the accuracy of the reconstruction method is close to that of the reference system. This is verified using four different starter motors, with detailed product information of the *Zeiss T-Scan* available in the Appendix A6.

Figure 6.1 shows a point cloud model of a clamped starter motor: subplot a) from the reconstruction approach of the thesis and subplot b) from the model obtained by the *Zeiss T-Scan* reference system. A heat map visualizing point distances between the reconstruction approach and the reference cloud is presented in subplot c). The clamping system was manually removed from all four starter motors for comparison.

[1] Link to product website:
https://www.handsonmetrology.com/de/loesungen/t-scan/
accessed: 14.06.2024

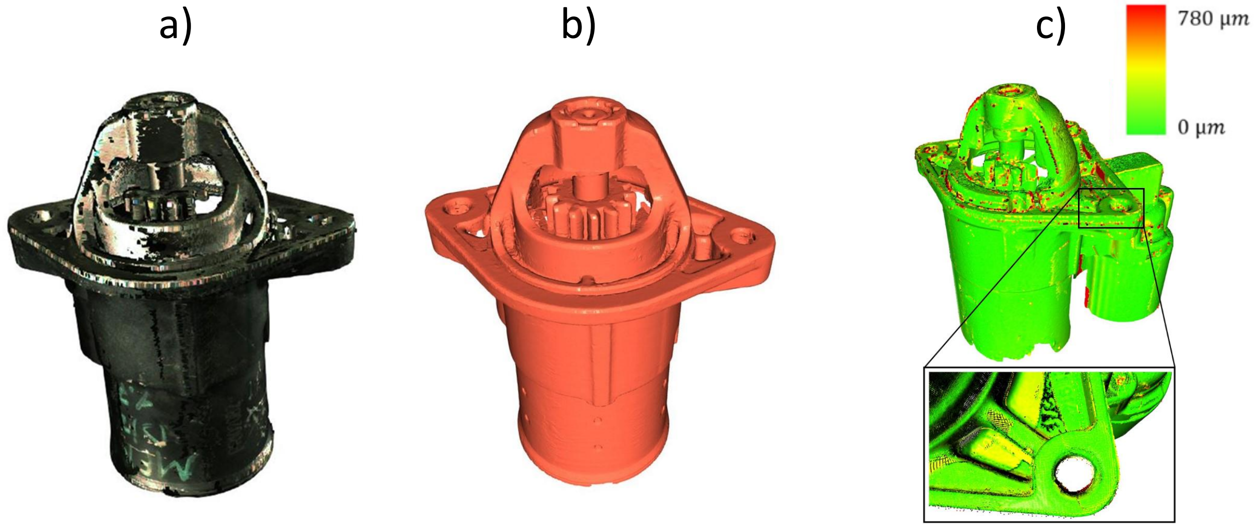


Figure 6.1: Visualization of the reconstructed 3D model (subplot a), the reference model (subplot b), and a heat map of the point-to-point distance for the reconstructed model and reference model (subplot c).

The agreement between point cloud models obtained by the reconstruction approach and those obtained by the *Zeiss T-Scan* reference system was assessed using Hausdorff Distance (see Reinke & Tizabi et al. (2021, P. 24)) and Mean Error (see Hodson (2022, P. 5482)) metrics. Hausdorff Distance measures the maximum discrepancy between the clouds by identifying the furthest point-to-point distance, highlighting the maximum misalignment. Mean Error calculates the average deviation among corresponding points, indicating overall alignment accuracy. The analysis primarily focuses on the 95th percentile of the Hausdorff Distance to mitigate outlier effects. Further details on the metrics used in this thesis can be found in Appendix A7.

Due to the analytical nature of the reconstruction approach proposed in this thesis, comparing four starter motor models generated with the reconstruction approach with their counterparts obtained by the *Zeiss T-Scan* reference system has been deemed sufficient. The only small source of variability in the results is the non-deterministic ICP, which relies on random initial transformation values. To account for this, the three-dimensional modeling process was repeated ten times for each of the four starter motors. Table 6.1 shows these iterations' averaged Hausdorff distance and mean errors values.

All four models show similar 95th percentile Hausdorff distances ($HD_{95}^{C \to L}$) and mean errors ($ME^{C \to L}$). Thereby, C and L denote point clouds and the metric space is the Cartesian Space. Model 4 shows the best reconstruction results due to the lowest error values. In contrast, model 2 shows the least accurate reconstruction. However, the average point distance across

Table 6.1: Hausdorff Distance and Mean Error für the reconstructed 3D models evaluated.

	$HD_{95}^{C \to L}$			$ME^{C \to L}[\mu m]$		
Model Nr.	**Mean**	**Min**	**Max**	**Mean**	**Min**	**Max**
1	627.9	623.2	629.9	251.4	252.4	256.1
2	723.1	701.6	747.2	311.1	303.1	319.8
3	647.0	645.7	648.0	285.3	282.9	287.6
4	420.2	414.1	424.6	157.5	157.4	158.0

models is below $350\mu m$, which is acceptable for fine planning using reinforcement learning (RL) agent solving the IPP.

6.1.2 Results of the semantic segmentation and instance segmentation approaches

After validating the developed reconstruction method for three-dimensional modeling, the segmentation approaches proposed in this thesis are evaluated using the IoU metric (see section 5.2.2) of the validation dataset. It's important to note that only extracts of the quantitative results are presented in the following main text, while the complete data are available in Appendix A8. Three NN were trained for each model configuration.

6.1.2.1 Results of the segmentation models using only RGB data

Table 6.2: Hyperparameter configuration for training the U-Net and M-Net configurations.

Parameter
Model
Backbone
Optimizer & Learning rate[a]
Loss
Pretrain
Freeze & Epochs
Batch size

(a)

Value	
M-Net	
VGG16	ResNet34
SGD	Adam[b]
0.01/0.005/0.001	
Default	
True	False
True	False
50/150	200
8	

(b)

Value	
U-Net	
VGG16	ResNet34
SGD	Adam[b]
0.01/0.005/0.001	
Focal	Dice
True	False
True	False
50/150	200
8	

(c)

[a] : Initial learning rate, reduced by 20% every 10 epochs.
[b] : Learning rate for Adam is 0.01 times the learning rate of SGD.

The training results for segmentation networks using only RGB data are discussed. Table 6.2 shows the selected hyperparameter levels for M-Net and U-Net networks, including variations (subplots b and c) based on the hyperparameters (subplot a). These networks are trained over all hyperparameter permutations, with detailed results in Appendix A8. The rationale behind certain hyperparameter choices (backbone, loss, pre-train, and freeze) is explained in the section on the segmentation approach (section 5.2.2), while others are discussed below. Adam and SGD are used as optimizers, with the learning rate adjustments informed by previous collaborative research documented in a supervised thesis (A_Hollinger 2022). Adam's learning rate is 100 times lower than SGD's due to its faster convergence speed. The training covers 200 epochs, with the initial freezing of the encoder removed after 50 epochs to allow full training over the remaining 150 epochs to balance performance improvement with the risk of overfitting. The batch size is fixed at eight to balance computational efficiency and network performance.

Table 6.4: Overview of IoU values of the best U-Net configurations trained.

Model Abbr.	Model	Backbone	Optimizer	Loss	Learning rate	Pretrain	Freeze	$\mathrm{IoU}_{\mathrm{max}}$	$\mathrm{IoU}_{\mathrm{mean}}$	$\mathrm{IoU}_{\mathrm{std}}$
U1	U-Net	VGG16	SGD	Dice	$1 * 10^{-2}$	True	True	0.8081	0.8050	0.0036
U2	U-Net	VGG16	SGD	Focal	$1 * 10^{-2}$	True	True	0.8124	0.7638	0.0344
U3	U-Net	VGG16	Adam	Dice	$1 * 10^{-4}$	True	False	**0.9217**	**0.9187**	0.0027
U4	U-Net	ResNet34	SGD	Dice	$1 * 10^{-2}$	True	True	0.6596	0.6561	0.0025
U5	U-Net	ResNet34	Adam	Dice	$1 * 10^{-4}$	True	False	0.9033	0.9005	0.0028

Table 6.4 shows the best-performing U-Net configurations, highlighting U3 as the best-performing model. Using a VGG16 backbone, Adam optimizer, dice loss function, learning rate of $1 * 10^{-4}$, pre-trained weights, and no frozen layers, U3 achieves the highest IoU values (max: 0.9217, mean: 0.9187) with minimal standard deviation (std: 0.0027). The results demonstrate the effectiveness of the Adam optimizer in improving U-Net performance. In comparison, configurations using the SGD optimizer show less satisfactory convergence and performance. In addition, VGG16 outperforms ResNet34 as a backbone in direct comparisons, indicating its slight advantage. The choice between Dice and Focal loss does not show a consistent trend across different configurations, although the best-performing U3 uses Dice loss.

Table 6.5: Overview of IoU values of the best M-Net configurations trained.

Model Abbr.	Model	Backbone	Optimizer	Loss	Learning rate	Pretrain	Freeze	$\mathrm{IoU}_{\mathrm{max}}$	$\mathrm{IoU}_{\mathrm{mean}}$	$\mathrm{IoU}_{\mathrm{std}}$
M1	M-RCNN	VGG16	Adam	Default	$1 * 10^{-4}$	True	True	0.9517	0.9462	0.0040
M2	M-RCNN	ResNet34	Adam	Default	$1 * 10^{-4}$	True	True	**0.9566**	**0.9537**	0.0026

Table 6.5 lists training results of the best M-Net configurations trained. In contrast to the U-Net, the SGD optimization algorithm has also proven to be a valid variant when using the M-Net. However, the Adam optimizer still shows slightly better performance. In contrast to U-Net, the ResNet34 encoder consistently provides slightly better performance on M-Net.

Freezing encoder weights for the first 50 epochs consistently yields slightly better IoU values with the M-Net. However, this finding can not be confirmed for the U-Net configurations, as the best-trained model (U3) is trained without freezing the encoder weights. Using encoders with pre-trained weights leads to better performance for both U-Net and M-Net. A direct comparison of the best configurations (U3 and M2) reveals that both exhibit high IoU values. The M-Net outperforms the U-Net with a higher average IoU value (mean IoU of 0.9537 compared to 0.9217), with similar standard deviations.

6.1.2.2 Results of the segmentation approaches integrating depth information

In the experiments incorporating depth data into segmentation network training, the hyperparameter configurations are detailed in Tables 6.6 and 6.9. Due to suboptimal performance with the ResNet34 encoder and the SGD optimizer in U-Net training using RGB data, these parameters are excluded from the evaluation of U-Nets incorporating depth information (UD-Nets). The exclusion also applies to the VGG16 encoder for M-Nets integrating depth information (MD-Nets). Similarily, due to the tendency towards worse performance using lower learning rates, the lowest learning rate ($1 * 10^{-5}$ for Adam and $1 * 10^{-3}$ for SGD) was excluded. Apart from these exceptions, the hyperparameter configurations for training the different variants of UD-Nets and MD-Nets remain unchanged.

Table 6.6: Hyperparameter configuration of the UD-Net configurations

Parameter	**Value**		
Model	UD-Add	UD-Net	UD-4D
Fusion type	Intermediate	Late|	Early
Loss	Focal	Dice	
Pretrain	True	False	
Freeze & Epochs	True	False	
	50/150	200	
Batch size	8		
Backbone	VGG16		
Optimizer & Learning rate[a]	Adam		
	$1 * 10^{-4}$	$5 * 10^{-5}$	

(a) *(b)*

[a] : Initial learning rate, reduced by 20% every 10 epochs.

The results from the evaluated UD-Nets in Table 6.8 yield similar conclusions to those from training without depth data integration. The dice and focal losses perform comparably well, exemplified by UD1 and UD2. Both UD-4D model variants show strong performance. Specifically, UD1, with the dice loss, yields the best average networks (mean: 0.9086), while the best-performing network, UD2, is trained with the focal loss (max: 0.9165).

Table 6.8: Overview of IoU values of the best UD-Net configurations trained.

Model Abbr.	Model	Fusion type	Backbone	Optimizer	Loss	Learning rate	Pretrain	Freeze	$\mathrm{IoU}_{\mathrm{max}}$	$\mathrm{IoU}_{\mathrm{mean}}$	$\mathrm{IoU}_{\mathrm{std}}$
UD1	UD-4D	Early	VGG16	Adam	Dice	$5*10^{-5}$	True	True	0.9135	0.9086	0.0039
UD2	UD-4D	Early	VGG16	Adam	Focal	$1*10^{-4}$	True	True	0.9165	0.9062	0.0045
UD3	UD-Net	Intermediate	VGG16	Adam	Focal	$1*10^{-4}$	True	True	0.9022	0.9017	0.0005
UD4	UD-Net	Late	VGG16	Adam	Focal	$5*10^{-5}$	True	True	0.9143	0.9129	0.0016
UD5	UD-Add	Intermediate	VGG16	Adam	Focal	$1*10^{-4}$	True	True	**0.9296**	**0.9291**	0.0006
UD6	UD-Add	Late	VGG16	Adam	Dice	$1*10^{-4}$	True	False	0.9272	0.9251	0.0015

The late fusion variant outperforms when comparing the top variants of UD-Net with net-based feature fusion at either the intermediate or late stage (UD3 and UD4). This is true for the maximum (max: 0.9143) and the mean (mean: 0.9129). Conversely, the opposite is observed for feature fusion by addition. The intermediate fusion is more effective than the best late fusion, with higher scores (max: 0.9296 and mean: 0.9291). However, the gap between the two variants (UD5 and UD6) is not as pronounced as when comparing the net-based fusion variants.

In summary, the UD5 model with an intermediate, addition-based feature fusion is the best-performing model. In this model, the encoder is frozen at the beginning of training, unlike training without depth data integration. Comparing the best hyperparameter configuration without depth data integration (U3) with the best configuration with depth data integration (UD5), UD5 shows superior performance. On average, UD5 with depth integration outperforms U3, with $IoU_{mean,UD5}$=0.9291 compared to $IoU_{mean,U3}$=0.9187, and similarly for the best model, with $IoU_{max,UD5}$=0.9296 compared to $IoU_{max,U3}$=0.9217.

Table 6.9: Hyperparameter configuration of the MD-Net configurations

Parameter	**Value**			
Model	MD-ADD	MD-Net	MD-4D	MD-Late
Fusion type	Intermediate		Early	Late
Pretrain	True		False	
Freeze & Epochs	True		False	
	50/150		200	
Loss	Default			
Batch size	8			
Backbone	ResNet34			
Optimizer & Learning rate[a]	Adam[b]		SGD	
	$1*10^{-2}$		$5*10^{-3}$	

(a) *(b)*

[a] : Initial learning rate, reduced by 20% every 10 epochs.

[b] : Learning rate for Adam is 0.01 times the learning rate of SGD.

Early image and depth data fusion yield the best results when comparing M-Net with integrated depth data. The MD1 configuration produced the best model overall ($IoU_{max,MD1}$=0.9547), while the MD2 configuration produced the best trained models on average ($IoU_{mean,MD2}$=0.9527). However, the differences between these configurations are minimal, the only difference being the different learning rates.

Table 6.11: Overview of IoU values of the best MD-Net configurations trained.

Model Abbr.	Model	Fusion type	Backbone	Optimizer	Loss	Learning rate	Pretrain	Freeze	$\mathrm{IoU_{max}}$	$\mathrm{IoU_{mean}}$	$\mathrm{IoU_{std}}$
MD1	MD-4D	Early	ResNet34	Adam	Default	$1 * 10^{-4}$	True	True	**0.9547**	0.9504	0.0038
MD2	MD-4D	Early	ResNet34	Adam	Default	$5 * 10^{-5}$	True	True	0.9542	**0.9527**	0.0013
MD3	MD-Add	Intermediate	ResNet34	Adam	Default	$1 * 10^{-4}$	True	True	0.9436	0.9410	0.0021
MD4	MD-Net	Intermediate	ResNet34	Adam	Default	$1 * 10^{-4}$	True	True	0.9400	0.9337	0.0047
MD5	MD-Late	Late	ResNet34	Adam	Default	$1 * 10^{-4}$	True	True	0.9515	0.9482	0.0023

In a direct comparison between the best UD-Net and MD-Net configurations, the results are consistent with the comparison between U-Net and M-Net variants without depth information integration. All of the best models from individual MD-Net fusion variants, as listed in Table 6.11, outperform the best UD-Net configuration. Additionally, it's noteworthy that all configurations with depth integration, except UD5, were initially trained with frozen encoder weights. This suggests that using frozen weights in the initial training should be preferred when incorporating depth information in the application.

6.1.2.3 Ensemble evaluation and uncertainty estimation

The subsequent analysis includes ensemble evaluation and uncertainty estimation results based on the identified best model configurations for U-Net, UD-Net, M-Net, and MD-Net.

Ensemble evaluation

Ten models (M_1 to M_{10}) are trained for each of the model configurations U3, UD5, M2, and MD2, following the methodology described in section 5.2.3. These models are then evaluated on the test dataset. It's important to note that no data augmentation is applied during the segmentation model predictions on the test data, thus evaluating the models' performance on real image and depth data. The IoU values for all models within each configuration and the IoU for the ensemble prediction are shown in Table 6.12.

Table 6.12: Results of the ensemble evaluation for the best model configurations.

Model Abbr.	M 1	M 2	M 3	M 4	M 5	M 6	M 7	M 8	M 9	M 10	IoU E
U3	0.9628	0.9627	0.9631	0.9630	0.9618	0.9633	0.9629	0.9640	0.9636	0.9633	0.9656
UD5	0.9662	0.9668	0.9675	0.9636	0.9662	0.9675	0.9682	0.9671	0.9671	0.9663	**0.9707**
M2	0.9555	0.9559	0.9536	0.9571	0.9530	0.9556	0.9574	0.9562	0.9556	0.9562	0.9650
MD2	0.9430	0.9454	0.9449	0.9394	0.9423	0.9493	0.9456	0.9435	0.9464	0.9492	0.9574

The main finding from the ensemble approach is that without data augmentation, both U-Net versions (U3 and UD5) perform significantly better regarding ensemble IoU (IoU E) than their counterparts, the M-Net versions (M2 and MD2). This contrasts the validation data evaluation, where the M-Net configurations outperformed the U-Net configurations. In this scenario, the U-Net configurations show superiority. Ultimately, the UD5 model configuration is the best configuration variant.

This result emphasizes the impact of augmentation on U-Net configurations and their performance. A plausible explanation is that augmentation introduces notable noise into the neural network (NN), leading to less robust semantic segmentation than instance segmentation. Semantic segmentation relies on pixel-based classification, while instance segmentation internally conducts object detection first. This two-part process will likely enhance segmentation robustness by initially detecting objects and then performing binary segmentation of the detected objects.

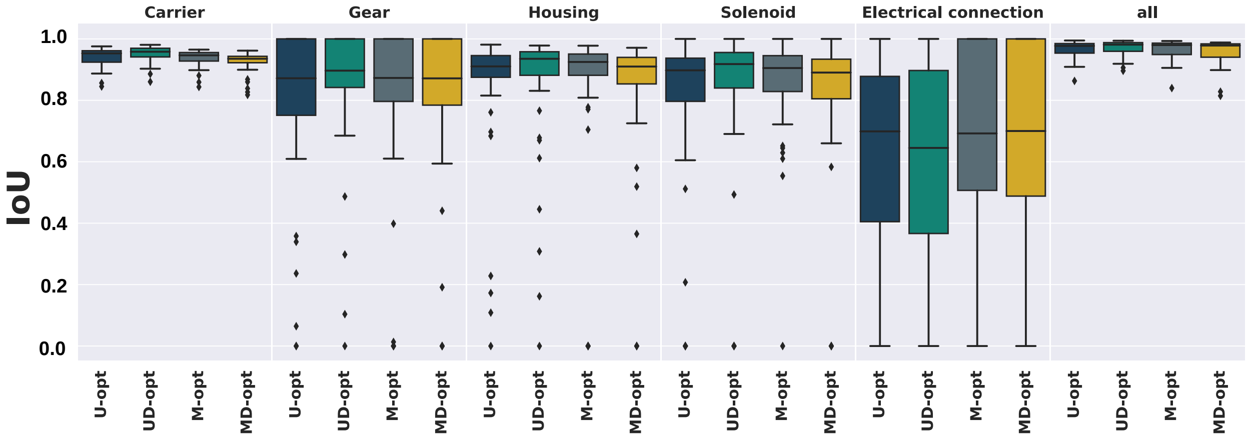


Figure 6.2: IoU of the best model configurations for all starter motor components. The following applies: U-opt $\hat{=}$ U3, UD-opt $\hat{=}$ UD5, M-opt $\hat{=}$ M2, MD-opt $\hat{=}$ MD2.

Figure 6.2 presents a visual summary of component-specific IoU values. It displays box plots illustrating ensemble performance for the NN configurations in segmenting starter motor components and the IoU for segmenting the complete motor ("all"). The overall IoU value of the whole segmentation result ("all") is high with a low boxplot spread, indicating an overall good segmentation performance. Consistently good segmentation is observed for the carrier and housing across all ensembles. However, all NN configurations encounter challenges in segmenting the electrical connection (evident from the wide boxplot spread), indicating a significant variation in segmentation quality. This difficulty likely arises from the small size and complexity of the electrical connection, where even minor mis-segmentations have a notable impact on the IoU value. Similar challenges may affect the gear and solenoid components due to their smaller size than the carrier or housing.

Uncertainty estimation

The uncertainty estimation results offer valuable insights. Figure 6.3 displays RGB images alongside segmentation results from various model configurations. Additionally, uncertainty maps generated by uncertainty estimation are depicted for all model configurations. Significantly sharper uncertainty maps are observed for configurations that utilize depth data for segmentation (UD5 and MD2) compared to those relying solely on RGB data (U3 and M2). Configurations U3 and M2 exhibit extensive uncertainties, particularly in areas bordering the starter motor, indicating challenges in distinguishing background from motor components. This suggests that configurations integrating depth data effectively utilize it to differentiate between background and engine components. Furthermore, areas of high uncertainty are observed at transition pixels between components and between components and the background. This is reasonable considering the potential overlap between different segmentations in these regions across individual NNs in the ensemble, leading to heightened uncertainty.

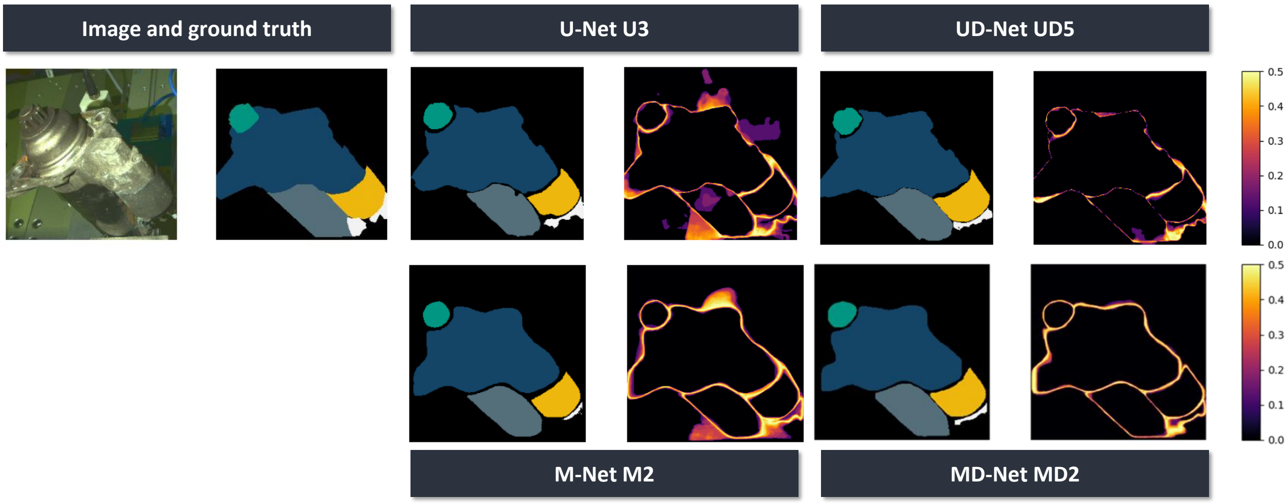


Figure 6.3: Visualization of the uncertainty maps of the best model configurations.

Table 6.13 presents the $F1$ score, false discovery rate (FDR), and false positive rate (FPR) for the best model configurations, contrasting with the earlier qualitative analyses. This comparison quantitatively evaluates unfiltered and filtered cases, where segmentations of individual pixels are discarded if their segmentation uncertainty, determined by the NN ensemble, exceeds a threshold ($\hat{\sigma}(D_{T,i})_{uv} \geq 0.5$). The $F1$ score represents the harmonic mean of precision and recall. At the same time, FDR is the proportion of false positive predictions, and FPR is the ratio of false positive predictions to the total number of actual negative results. Further details on these metrics can be found in the Appendix A9.

The quantitative results in Table 6.13 suggest that filtering out pixels with high prediction uncertainty decreases prediction quality ($F1$ score is consistently lower in the filtered case $F_{1,filt}$ than in the unfiltered case F_1). However, it can increase segmentation robustness (FDR

Table 6.13: Results of the best model configurations' unfiltered and filtered segmentation results. The IoU, $F1$ score, false discovery rate (FDR), and false positive rate (FPR) are compared.

Model Abbr.	IoU	IoU_{filt}	F_1	$F_{1,filt}$	FDR	FDR_{filt}	FPR	FPR_{filt}
U3	0.9656	0.9284	0.7220	0.6308	0.1249	0.0507	0.0051	0.0014
UD5	0.9707	**0.9529**	0.7287	**0.6642**	0.0770	0.0315	0.0064	0.0024
M2	0.9650	0.9136	0.7158	0.5607	0.0650	**0.0209**	0.0082	**0.0013**
MD2	0.95574	0.9359	0.7043	0.6413	0.0759	0.0248	0.0091	0.0021

and FPR are also lower in the filtered cases FDR_{filt} and FPR_{filt} in contrast to the unfiltered cases FDR and FPR). Filtering pixels with high uncertainty results in lower IoU and $F1$ scores across all model ensembles, indicating that more pixels predicting correct component membership are filtered out than those predicting incorrect components. However, FPR and FDR also decrease, suggesting that the remaining pixels are more likely to be correctly assigned to starter components (or background) by the segmentation. This represents a trade-off between model quality and completeness regarding surface coverage percentage during the semantic reconstruction process. Filtering by the NN ensemble excludes pixels with high uncertainty, potentially improving model quality by increasing the likelihood of correct class assignment for all surface points in the reconstructed 3D semantic model. However, this may reduce the completeness of the model because surface areas with high prediction uncertainty may be excluded from the reconstruction.

6.1.3 Evaluation of the semantic 3D reconstruction approach

The preceding sections demonstrate the successful reconstruction of three-dimensional models and the semantic and instance segmentation capability to segment starter motor components. Furthermore, it is shown that an ensemble slightly improves segmentation compared to a single model and that such an ensemble can refine models further through uncertainty assessment.

This section evaluates the model completeness and quality of the semantic 3D models generated by the developed approach. Ten additional starter motors are acquired using RGB, depth data, and point clouds according to the method described in section 5.2.1. These data are not used to train segmentation networks. Semantic 3D models of the starter motors are reconstructed using the respective optimal NN model configurations, as described in section 5.2.4. An example of a reconstructed semantic 3D model for the four NN model configurations is shown in Figure 6.4.

The acquired data includes reconstruction and manual labeling of 3D models for each of the ten starter motors, resulting in ten ground truth semantic 3D models. The quality and com-

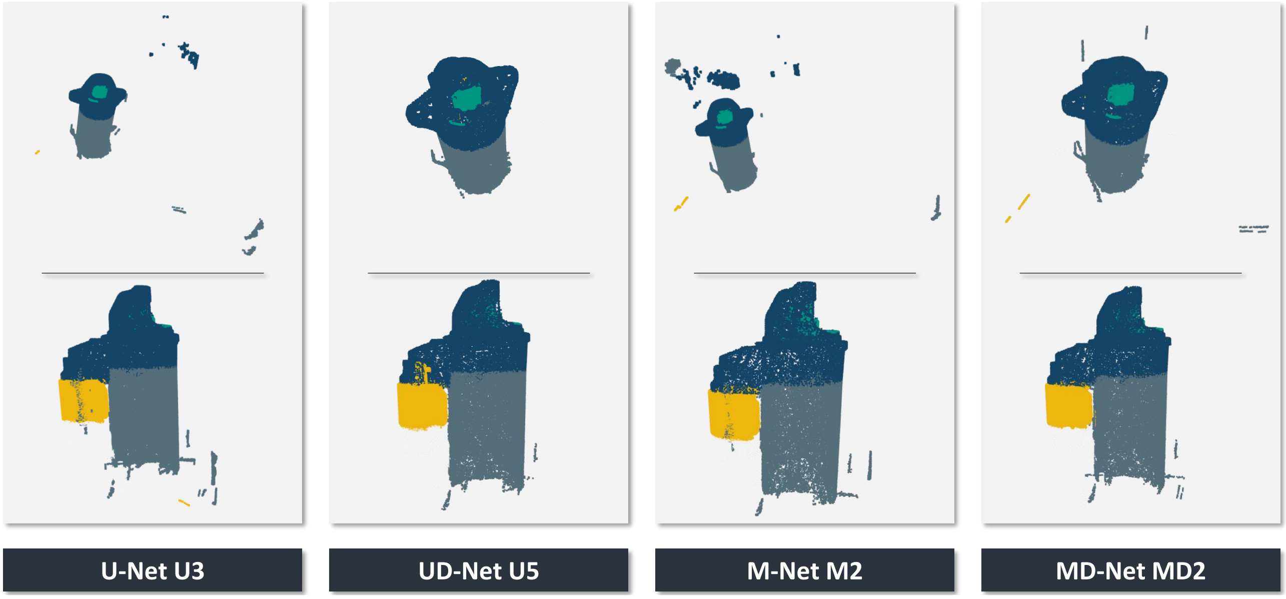


Figure 6.4: Visualization of the reconstructed semantic 3D models based on the segmentation of the image data (RGB and/or depth) using the best model configurations and the acquired point clouds.

Table 6.14: Tabular representation of the average accuracy of the semantic 3D reconstruction of the best model configurations for all components of the starter motors of the test data set.

	Accuracy					
Model Abbr.	**Carrier**	**Housing**	**Solenoid**	**Electrical connection**	**Gear**	Mean
U3	89.2%	89.2%	89.0%	82.0%	80.7%	86.0%
UD5	**96.1%**	**94.4%**	93.1%	**90.7%**	**95.3%**	**93.9%**
M2	89.1%	91.8%	**93.6%**	90.5%	91.7%	91.3%
MD2	94.0%	91.6%	93.5%	84.4%	94.9%	91.7%

pleteness of these ground truth models are evaluated by comparing them to the reconstructed semantic 3D models generated by the segmentation networks. The percentage of correctly segmented surface points on the starter motors is examined to evaluate model quality. Meanwhile, for model completeness, the focus is on determining the percentage of the starter motor surface covered by the semantic 3D reconstruction approach. The average model accuracy results are summarized in Table 6.14, while the model completeness summary is presented in Table 6.15.

The tabular results clearly show that the U-Net model configuration U3 has the highest average model completeness across all components. In addition, examination of the variation in component completeness, as shown in the box plots in Figure 6.5 for all model configurations

Table 6.15: Tabular representation of the average model completeness of the semantic 3D reconstruction of the best model configurations for all components of the starter motors of the test data set.

	Percentage seen					
Model Abbr.	**Carrier**	**Housing**	**Solenoid**	**Electrical connection**	**Gear**	Mean
U3	**95.9%**	**98.3%**	**99.1%**	**94.5%**	**98.1%**	**97.2%**
UD5	93.3%	97.9%	99.0%	94.2%	96.9%	96.2%
M2	95.3%	97.9%	99.0%	89.7%	95.7%	95.5%
MD2	93.1%	97.2%	98.0%	91.6%	92.4%	94.4%

considered, reveals that this variation is minimal for the U3 model configuration. This trend is consistent for all other model configurations, with few outliers observed.

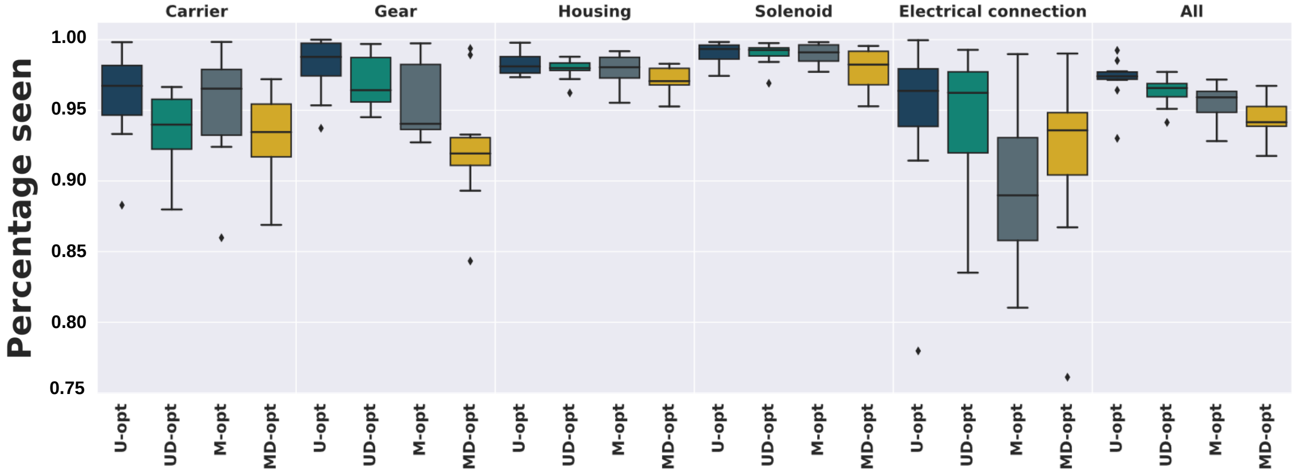


Figure 6.5: Semantic 3D reconstruction completeness of the best model configurations for all starter motor components. The following applies: U-opt ≙ U3, UD-opt ≙ UD5, M-opt ≙ M2, MD-opt ≙ MD2.

However, a more detailed quantitative examination reveals a critical issue, particularly of the individual semantic 3D reconstruction results shown in Figure 6.4. Model configurations U3 and M2, which do not integrate depth data, often segment the environment around the starter motor, leading to significant distortions in the semantic 3D reconstruction result. This effect is less pronounced in the two model configurations integrating depth information.

The model accuracy further confirms these findings (refer to Table 6.14). Both model configurations integrating depth data exhibit the highest accuracy, with the U-Net model configuration UD5 achieving the highest accuracy for all components except the solenoid. Regarding model completeness, Figure 6.6 provides a visual overview of the accuracy for the best segmentation model configurations.

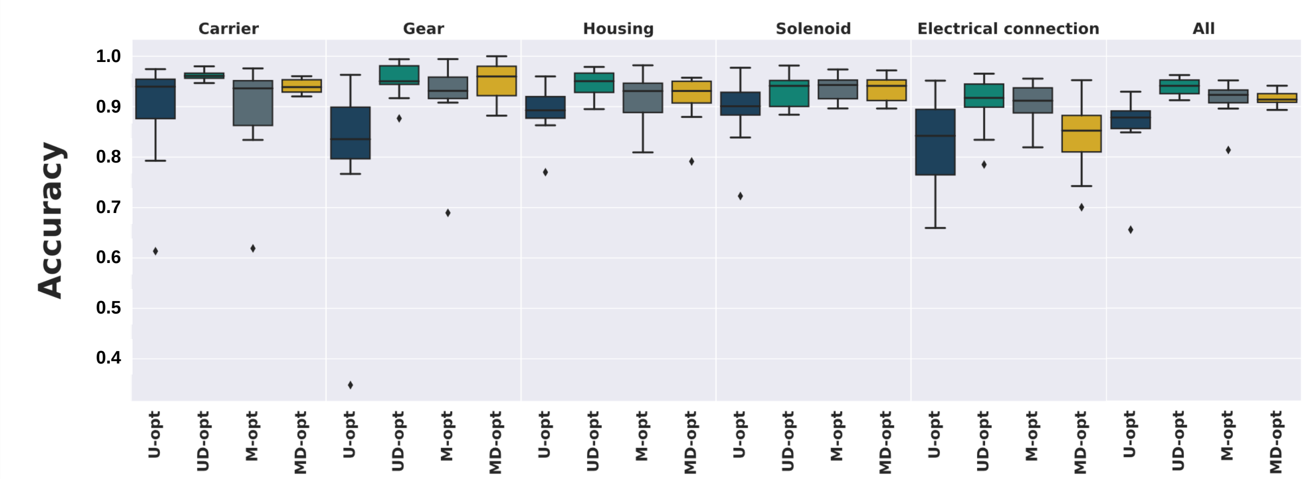


Figure 6.6: Semantic 3D reconstruction accuracy of the best model configurations for all starter motor components. The following applies: U-opt $\hat{=}$ U3, UD-opt $\hat{=}$ UD5, M-opt $\hat{=}$ M2, MD-opt $\hat{=}$ MD2.

These results demonstrate that the semantic 3D reconstruction approach can be robustly applied to unknown starter variants. First, the best model configurations from all investigated segmentation approaches (semantic and instance segmentation) achieve sufficiently high model coverage with an acceptable level of variation. Furthermore, all investigated model configurations of these segmentation approaches show equally good model accuracy. Furthermore, the variation of model completeness and model accuracy is low, with only a few outliers. In addition, integrating depth information is shown to prevent co-segmentation of the environment.

6.2 RL for the solution of VPPs in remanufacturing

This section presents the results of the modeling alternatives outlined in section 5.3. It begins with a description and overview of the data sets used and the evaluation approach in sections 6.2.1 and 6.2.2. The results for the VPP problem cases, including NBV selection, RPP and IPP, are then detailed in the following sections 6.2.3 (NBV), 6.2.4 (RPP) and 6.2.5 (IPP).

6.2.1 Dataset description

This thesis evaluates problem cases using four different datasets. These datasets vary in realism and complexity, making them suitable for evaluating the developed agent models and their applicability in remanufacturing. Each dataset in this thesis consists of STL models of the objects to be inspected and the ground truth point cloud $\mathbf{P}_{GT}$. Additional information such as the membership of individual points of $\mathbf{P}_{GT}$ to component classes may also be available. This allows the derivation of inspection targets for the case of IPP, if specific components need to be inspected, highlighting the distinction from supervised learning. None of the datasets contain ground truth labels in the form of poses $\mathbf{p}_t^{A/RT}$ of the acquisition system. Instead, only the object model (STL) and the ground truth point cloud ($\mathbf{P}_{GT}$) are provided, requiring the RL agent to learn full (RPP) or partial inspection of individual regions of interest (IPP).

Synthetic motor dataset D1: The first dataset, D1, consists of STL models and ground truth point clouds $\mathbf{P}_{GT}$ of starter motors synthetically generated using the approach of Wu & Zhou et al. (2022). These generated starter motors vary in their geometric properties to represent the range of existing starter motor variants. It's assumed that each starter motor consists of a fixed set of geometric primitives representing specific components (e.g., solenoid, main housing, gear, etc.). These geometric primitives can be parameterized (e.g., the main housing is a cylinder with parameters such as height and radius), and parameter values are sampled to create unique starter models. Realistic parameter bounds and defined relationships between geometric primitives ensure that parameters stay within realistic ranges. For example, the gear diameter cannot exceed the size of the gear housing. This approach results in a diverse and realistic data set of 200 motors with randomly generated parameters.

Real motor dataset D2: The second data set, D2, consists of 40 real starter motor models. These STL models were created from real starter motors using a handheld laser scanner system (*Zeiss T-Scan*). More technical details about the laser scanner system can be found in Appendix A6. During the measurement, the laser scanner system generated the ground truth point clouds $\mathbf{P}_{GT}$. The measurement system's software generated the corresponding STL models.

Station motor dataset D3: The third dataset, D3, represents actual clamping scenarios of starter motors within the clamping system of the inspection station. To construct the D3 dataset, point clouds of the dataset that were collected to train the segmentation networks for 3D semantic reconstruction were used. For each engine clamping scenario, multiple partial point clouds were generated from different perspectives and angles of the rotary table were generated during this collection process. These partial point clouds of the 40 existing starter motors clamped in the system were converted into a STL model. This conversion was achieved using the 3D reconstruction approach described in section 5.1.4, which first transforms the partial point clouds into a globally registered and post-processed point cloud representing the ground truth point cloud $\mathbf{P}_{GT}$. The *screened Poisson*[1] reconstruction approach of Kazhdan & Hoppe (2013) was used to generate the STL models based on $\mathbf{P}_{GT}$. The *screened Poisson* algorithm extends the *Poisson* reconstruction approach of Kazhdan

[1] Link to algorithm documentation in PyMeshlab:
https://pymeshlab.readthedocs.io/en/0.1.8/filter_list.html
accessed: 14.06.2024

& Bolitho et al. (2006). The *Poisson* surface reconstruction approach treats points and their normals in a point cloud as a vector field to solve the standard Poisson problem (Kazhdan & Bolitho et al. 2006, P. 1). The solution of the Poisson problem is an implicit function whose values are zero at points of the vector field and whose gradient equals the normal vector of the vector field. Thus, a surface can be extracted from the solution of the Poisson problem. *Screened Poisson* reconstruction additionally incorporates interpolation constraints to improve stability and surface quality (Kazhdan & Hoppe 2013, P. 1).

Dataset D3 can represent a real clamping scenario and ensure that all points of $\mathbf{P}_{GT}$ can be classified. This is possible because all RGB and depth images used to train the segmentation networks were labeled. Due to the clear relationship between the pixels of the RGB and depth images and the corresponding points of the partial point clouds, each point of the globally registered and post-processed point cloud $\mathbf{P}_{GT}$ (the result of the 3D reconstruction process) can also be assigned to a component class.

ShapeNet dataset D4: The previously discussed datasets, D1-3, focus solely on the product category of starter motors. Dataset D4, on the other hand, consists of STL models and point clouds of various model types from the ShapeNet dataset. ShapeNet, developed by Stanford University (Chang & Funkhouser et al. 2015), provides a collection of 51,300 unique 3D models in 55 different object categories. This dataset serves as a widely used benchmark for comparing various computer graphics algorithms (Chang & Funkhouser et al. 2015, P. 1).

To keep complexity manageable, dataset D4 includes 150 randomly selected STL models along with their corresponding ground truth point clouds $\mathbf{P}_{GT}$ from seven object categories: bus, ship, car, skateboard, train, motorcycle, and airplane from the ShapeNet dataset. In addition, dataset D4 contains the 40 engine models and their ground truth point clouds from dataset D2.

An overview of the appearance of the STL models for the individual data sets D1-4 can be found in Figure 6.7. The experiments conducted in this thesis use the presented datasets to control complexity and realism and provide fundamental insights into different modeling variants of the RL agents developed in this thesis.

6.2.2 Evaluation approach

The following section details the overall evaluation approach.

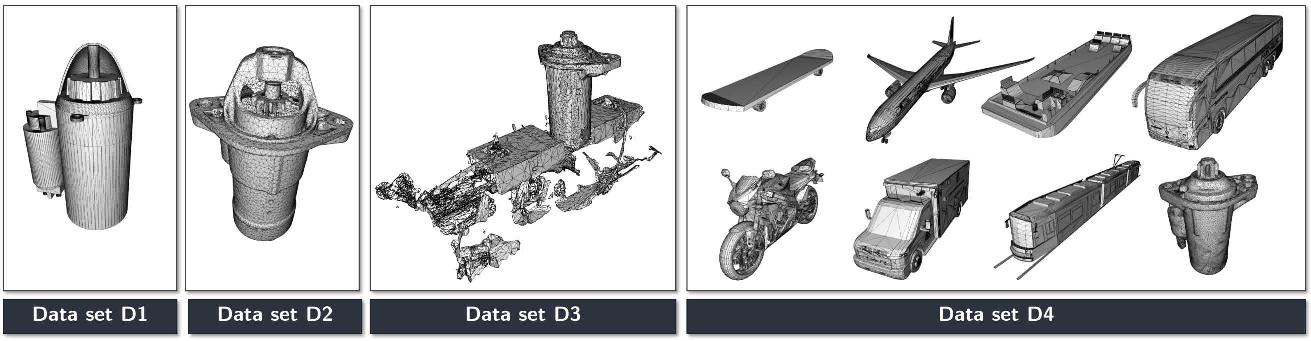


Figure 6.7: Overview of the appearance of the STL models for the individual data sets D1-4.

6.2.2.1 Evaluation setup

The RL agent is trained in episodes, each representing the acquisition process of a single STL model from the dataset. To emulate the manual clamping process in the inspection station's clamping system and to introduce variability, each randomly selected STL model is first placed upright in the simulation. It is then rotated around its longitudinal axis (z-axis) in the range $[-180°, 180°]$ and tilted along the x- and y-axes in the range $[-25°, 25°]$. The acquisition process continues until the terminal state s_T or the goal state s_G is reached. This endpoint is determined by either reaching 90% surface area coverage or reaching the threshold of ten acquisitions, depending on the episode design. The agent's strategy π for action selection is optimized based on state, action, and reward transitions. It's worth noting that the initial rotation is not applied when working with dataset D3 since the manual clamping process is already included in the STL model, which represents a starter motor clamped in the inspection system's clamping system.

Except for data set D1, data sets D2 to D4 are divided into training and test STL models and point clouds. An 80% portion is allocated for training, while the remaining 20% is allocated for testing. Dataset D1 is specifically used to evaluate basic modeling variants and the behavior of the RL agents in the studied problem cases (NBV, RPP, and IPP). For datasets D2 to D4, RL agents trained on the training models and point clouds are then tested on the test models and point clouds to assess the adaptability of the agents to previously unseen models.

In addition to testing for unseen data, the best RL modeling variants are compared with benchmark algorithms. These are the following:

1. **Random:** These benchmark algorithm randomly chooses a position in space and automatically compute the orientation to the center of the coordinate system C_{RT}. Therefore, these algorithms use the action mapping $A_{2S,0R}$.

2. **Heuristic:** The simple solution method involves placing a rectangular bounding box around the object mesh and generating an acquisition system pose for each face of the bounding box. The pose is determined at a fixed distance along the bounding box surface's normal, and the acquisition system's orientation is aligned with the surface normal. This method results in a total of six acquisition poses.

3. **SCP:** The analytical solution of the VPP involves sampling multiple poses of the acquisition system as VPs (VP). These VPs are then simulated for acquisition and evaluated to analytically compute the shortest view plan to achieve the optimization goal. Two implemented variants, $\mathbf{SCP}_{\mathbf{\Psi}}$ and $\mathbf{SCP}_{\mathbf{\Psi},\mathbf{d}}$, are used. $\mathbf{SCP}_{\mathbf{\Psi}}$ focuses solely on maximizing the total surface area coverage Ψ_E after each episode, while $\mathbf{SCP}_{\mathbf{\Psi},\mathbf{d}}$ aims to maximize the surface area coverage Ψ_E while minimizing the required travel distance d_E. Both variants of the **SCP** benchmark algorithm are explained in detail in Appendix 10.

6.2.2.2 Evaluation parameters

The parameters and selected standard configurations for the following evaluations of agent modeling for solving the NBV problem, RPP, and IPP subproblems of VPP can be found in Table 6.16. The chosen parameters and configurations are mostly based on preliminary studies carried out in works (A_Schmid 2021, A_Gäbele 2022 and A_Koch 2022) under the supervision of the author of this thesis, which have achieved good learning results (learning success and convergence speed).

The required surface coverage to complete an episode and reach the target state s_G is set to 90% for the RPP and Inspection Planning Problem (IPP). For the NBV, an episode ends after a fixed number of agent acquisitions, and no fixed value of required surface coverage is required. In contrast, episodes end after 10 acquisitions for the RPP or after 5 acquisitions for the IPP (reaching the terminal state s_T) without reaching the goal state s_G. The maximum number of interactions (steps) of the agent is defined as $1 * 10^6$ (NBV), $2 * 10^6$ (RPP), and $5 * 10^6$ (IPP), where the learning process is terminated in case of premature convergence.

The discount factor is the same for all subproblems ($\gamma_{RL} = 0.9$), whereas the learning rate for the NBV problem is initially higher ($\eta = 10^{-4}$) than for the RPP and IPP ($\eta = 4 * 10^{-5}$).

The standard RL algorithm used for all subproblems is the SAC with the PCN as encoder for the NN and a downstream MLP structure of 256-128-64 neurons. Training is performed with a batch size of 256, where the agents for the NBV problem are trained after each step. In contrast, the agents for the RPP and IPP problems are trained after eight consecutive episodes. For the agents solving the NBV problem, learning starts at the beginning of the

Table 6.16: Parameters and configuration, as well as the respective default values and default modeling choices of the RL agent.

Training cycle design	**Values NBV**	**Values RPP**	**Values IPP**
Coverage of $\mathbf{P}_G$ to reach s_G	-	90%	90%
Number of acquisitions to reach s_T	1	10	5
Max. number of steps ^{a}	$1 * 10^6$	$2 * 10^6$	$5 * 10^6$
Learning parameters	**Values NBV**	**Values RPP**	**Values IPP**
Learning rate η ^{b}	10^{-4}	$4 * 10^{-5}$	$4 * 10^{-5}$
Discount factor γ_{RL}	0.9	0.9	0.9
Modeling configuration	**Setup NBV**	**Setup RPP**	**Setup IPP**
Dataset	D1	D1	D1
Algorithm	SAC	SAC	SAC
Encoder	PCN	PCN	PCN
MLP structure^c	256-128-64	256-128-64	256-128-64
Batch size	256	256	256
Training frequency^d	$1S$	$8E$	$8E$
Learning start	0	10^4	10^4
Entropy coefficient^e	0.005	0.01	0.5

^{a}: Maximum number, training is ended when convergence happens earlier.
^{b}: Linear decay with a final value of 0.
^{c}: Number of last layer neurons corresponds to DoF of action.
^{d}: Learning happens after a specified number of steps (S) or episodes (E).
^{e}: Higher values indicate an initially increased exploration of the agent.

interaction, while for the agents solving the RPP and IPP problems, learning starts after 10^4 steps. Random actions are performed at the beginning of the training process. This allows the RL agents to collect this random experience at the beginning of training after 10^4 steps. This is favored by the higher values of the entropy coefficient for the RPP and IPP agents, which allows for more exploration. The choice of this configuration is favored by the fact that both RPP and IPP represent more complex problems compared to the NBV problem. Therefore, the agents must initially explore more to learn a successful solution strategy.

The acquisition system parameters for the evaluated problem cases are detailed in the Table 6.17. These parameters remain nearly identical for all problem cases, with variations observed primarily in the OWD and working area for the IPP scenario. This distinction arises because detailed evaluation of ROI often requires acquisition systems capable of closer proximity to the object for higher resolution. Consequently, the varied parameters accommodate this requirement. In addition, the altered parameters introduce additional complexity to the IPP problem, as less area is typically acquired per acquisition. Therefore, more emphasis is placed on the RL agent to accurately determine the pose of the acquisition system.

Table 6.17: Parameters and chosen default values for the sensor model.

Sensor model parameters	Values RPP & NBV	Values IPP
Near and far bounds b_{near}, b_{far}	$[300, 500]mm$	$[150, 300]mm$
Resolution R_{uv}	$(430 \times 300)_{xy}$	$(430 \times 300)_{xy}$
Aperture angles $\boldsymbol{\vartheta}_{A,uv}$	$(27°, 25°)_{xy}$	$(27°, 25°)_{xy}$
Sensor OWD	$470mm$	$270mm$
Cut-off angle $\vartheta_{A,C}$	60%	60%

6.2.2.3 Evaluation procedure

The main evaluation metrics for comparing different modeling alternatives of RL agents have already been introduced in 5.3.4.3. These include the surface coverage Ψ_E after an episode, the number n_E of acquisitions required to reach Ψ_E, and the cumulative travel distances d_E. Additional evaluation metrics used in this thesis will be introduced and explained where appropriate. While the trajectory of the reward signal is crucial for evaluating the learning behavior of the RL agents, it is not used as a central evaluation metric. Ideally, the reward function is functionally related to the problem's evaluation metrics, such that maximizing the reward is equivalent to maximizing/minimizing (depending on the optimization variable) the evaluation metrics. Therefore, the effect of the reward signal on the problem at hand is directly examined by monitoring the evaluation metrics defined above.

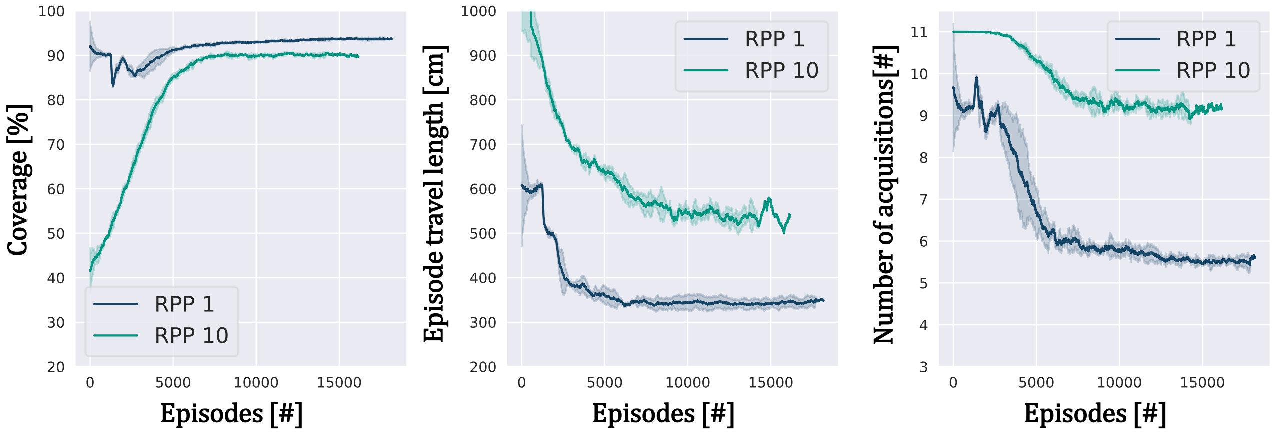


Figure 6.8: Graph of the evaluation metrics Coverage Ψ_E, Episode travel distances d_E and Required number of acquisitions n_E for the exemplary agents RPP 1 and RPP10 introduced in later sections of this work.

Multiple simulation runs with the same configuration are performed and then averaged. Previous supervised work by A_Schmid 2021, A_Gäbele 2022, and A_Koch 2022 has shown that using three runs per configuration allows for quantitative comparisons. The evaluation metrics shown in Figure 6.8 are plotted over the number of episodes. An episode includes the acquisition process of an object model by the RL agent. Since multiple agents are trained for each agent configuration, the plots in Figure 6.8 include the averaged course of the evaluation metric in a bold line. In addition, the minimum and maximum bounds of all trained agents across the training run are shown as an area around the averaged evaluation metrics. As shown, the agents reach a converged state after a certain number of episodes. In the converged state, there is no significant improvement in the evaluation metrics until the final value of episodes is reached. The Figure shows that the agents train for different numbers of episodes until they converge. For this reason, in the following sections, only a part of the training process of the compared agents is shown in the Figures to visualize the agents' training behavior, draw conclusions, and visually support the facts explained. For this reason, the quantitative values of the evaluation metrics of the RL agents in the converged state may differ slightly from the representations in the Figures. In addition to the illustrations of the evaluation metrics throughout the training, a table with the final average values of the evaluation metrics in the converged state is always provided. The numerical value in the converged state is calculated by averaging the respective evaluation metric over the last 100 episodes of a simulation run and the simulation runs performed with that particular modeling alternative.

6.2.3 Results of the RL agent training using the proposed simulation framework for the NBV problem

First, the developed agent model's basic learning capability is demonstrated by tackling the simplest planning problem: determining the NBV. To do this, the four agents, NBV 1-4 are compared against two baseline methods, Random 1 and Random 2. The dataset used is the synthetic motor dataset D_1.

The baselines, Random 1 and Random 2, randomly select actions using the action mapping $A_{2S,0R}$ to convert a two-entry random vector in the range $[-1, 1]$ into a pose in spherical coordinates. The key difference lies in the initial pose $p_0^{A/RT}$: for Random 1, it remains constant, while for Random 2, it varies randomly based on the method described above. The subsequent pose $\mathbf{p}_1^{A/RT}$ is chosen for both baselines.

The learning agents, NBV 1 and NBV 2, differ in their initial pose p_0^A: it is fixed for NBV 1 but randomly chosen for NBV 2. Both agents define VPs as poses on a sphere with a fixed radius of 470 mm, using $M_{r,fix}$. NBV 1 demonstrates the ability of an agent to learn the NBV from a fixed initial pose, while NBV 2 extends this capability to varying initial conditions. These

Table 6.18: Overview of the evaluation metrics of the agents and baselines evaluated for the NBV problem.

Agent information							
Model Abbr.	**S**	**A**	**R**	**Initial acquisition**	$\mathbf{\Delta\Psi_0}$	$\mathbf{\Delta\Psi_1}$	$\mathbf{\Psi_E}$
Random 1	-	$A_{2S,0R}$	-	fix	25.95%	13.56%	39.51%
Random 2	-	$A_{2S,0R}$	-	random	26.27%	13.61%	39.88%
NBV 1	S_1	$A_{2S,0R}$	R_1	fixed	25.98%	30.54%	56.52%
NBV 2	S_1	$A_{2S,0R}$	R_1	random	26.50%	29.26%	55.76%
NBV 3	S_1	$A_{3S,0R}$	R_1	random	26.55%	29.71%	56.26%
NBV 4	S_1	$A_{3C,0R}$	R_1	random	25.60%	22.87%	48.47%

agents operate within the state S_1 and adhere to the reward function R_1. The results of their learning processes are documented in Table 6.18.

A comparison of the surface coverage of the initial acquisition shows no differences between learning agents (NBV 1 and NBV 2) and baselines (Random 1 and Random 2). This lack of difference is expected since the initial poses $\mathbf{p}_0^{A/RT}$ are predetermined and not influenced by learning. It should also be noted that the additional surface coverage gained by a VP is about 26% of the object's total surface coverage if no prior acquisition has occurred.

The additional surface covered by the pose $\mathbf{p}_1^{A/RT}$ is more interesting. Random baselines cover only about 13-14%, resulting in a total surface coverage Ψ_E of about 39%. In contrast, learning agents identify VPs that cover, on average, about 29-31% of the object's surface. This additional coverage, $\Delta\Psi_1$, exceeds the average initial coverage of 26%, resulting in a higher Ψ_E of approximately 56% for agents NBV 1 and NBV 2 compared to Random 1 and Random 2. This suggests that both fixed and random initial VPs allow NBV 1 and NBV 2 to select subsequent VPs to maximize the acquired surface coverage $\Delta\Psi_1$.

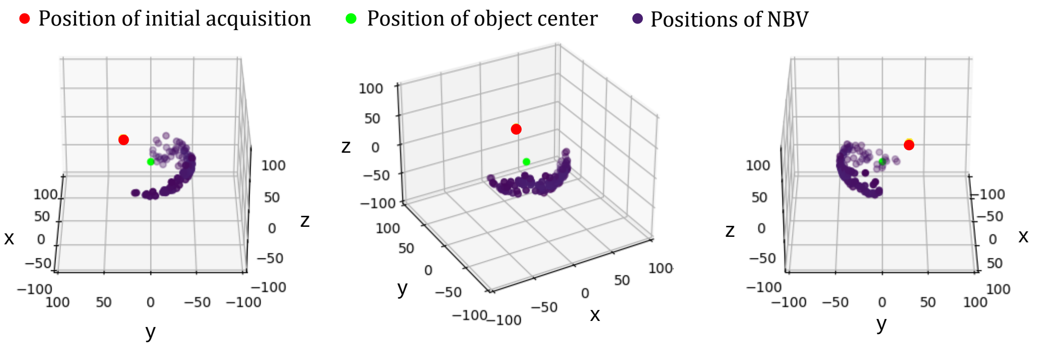


Figure 6.9: Distribution of acquisition system positions of the poses $\mathbf{p}_1^{A/RT}$ issued by the agent NBV 1 in the NBV problem with fixed initial acquisition system pose $\mathbf{p}_0^{A/RT}$.

Figure 6.9 illustrates the distribution of acquisition system poses in space of the NBV for agent NBV 1. Shown are the acquisition system positions from three different perspectives to make the distribution of the acquisition system poses more tangible. This distribution shows that NBV 1 strategically positions its VP $\mathbf{p}_1^{A/RT}$ relative to the initial fixed pose $\mathbf{p}_0^{A/RT}$ to maximize surface coverage $\Delta\Psi_1$. The crescent shape of the distribution is probably due to the initial random rotation of the inspected starter motors before each acquisition process. Figure 6.10 shows the poses generated by agent NBV 2. Despite the randomness of the initial VP $\mathbf{p}_0^{A/RT}$, the agent seems to converge on preferred poses $\mathbf{p}_1^{A/RT}$, presumably offering the highest average surface coverage $\Delta\Psi_1$ depending on the initial pose $\mathbf{p}_0^{A/RT}$.

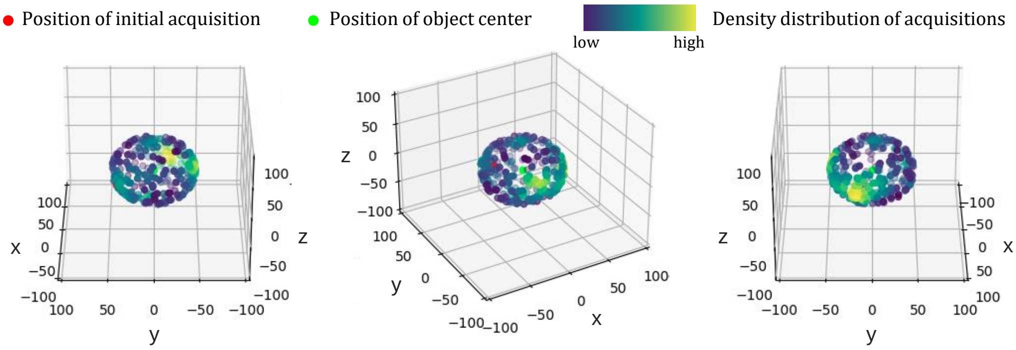


Figure 6.10: Distribution of acquisition system positions of the poses $\mathbf{p}_1^{A/RT}$ issued by the agent NBV 2 in the NBV problem with random acquisition system pose $\mathbf{p}_0^{A/RT}$.

Further insight can be gained by allowing the agent to vary the radius (NBV 3, action mapping $A_{3S,0R}$). NBV 3 selects VPs $\mathbf{p}_1^{A/RT}$ as VPs, resulting in approximately the same newly acquired surface $\Delta\Psi_1$ of the inspected object as NBV 1 and NBV 2. This also leads to approximately the same object surface coverage $\Delta\Psi_E$ after two acquisitions. Figure 6.11 illustrates the visualization of evaluation metrics, including reward and surface coverage of NBV for pose $\mathbf{p}_1^{A/RT}$ for agents NBV 1-3. It can be seen that the additional integration of a degree of freedom to be learned (radius $r_t^{A/RT}$) does not seem to have a negative impact on the speed of convergence. However, the agents (NBV 2 and NBV 3) that start with a random initial pose are slower to converge (need more episodes), as seen in Figure 6.11

A visual comparison of Agent NBV 3 and NBV 4 can be seen in Figure 6.12. NBV 4 uses Cartesian coordinates for action mapping (action mapping $A_{3C,0R}$). In contrast to NBV 3, the surface coverage achieved by NBV 4's selected VPs is inferior. On average, it covers only about 22.87% of the surface $\Delta\Psi_1$ of the inspected object with its chosen pose $\mathbf{p}_1^{A/RT}$, about 7% less than agents NBV 1-3. In addition, there were notable differences in the learning speed of the individual agents during training, as shown by the different limits of the curves in Figure 6.12 for agent NBV 4. This indicates that, despite the DoF of both agent variants (NBV 3 and NBV 4) being the same, the action mapping $A_{3C,0R}$ using Cartesian Coordinates is

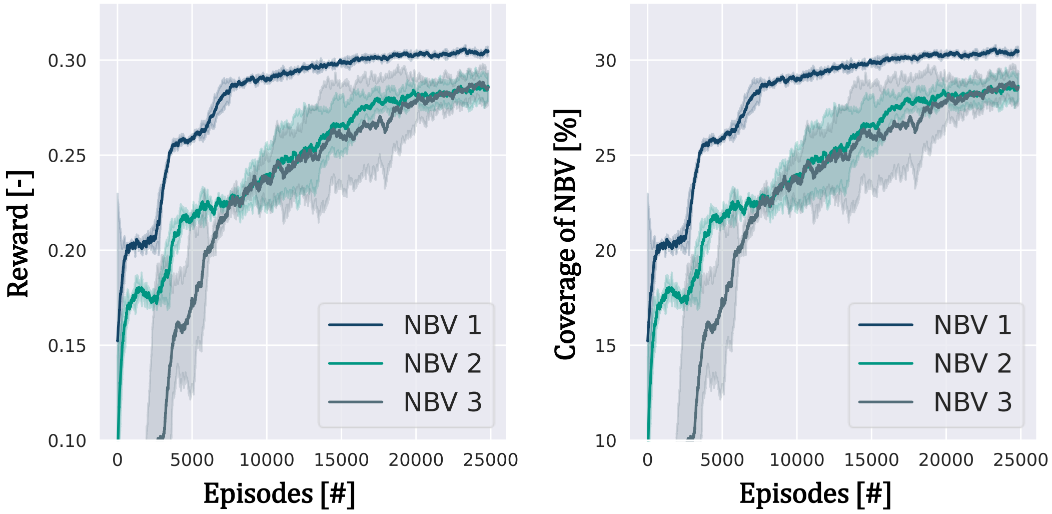


Figure 6.11: Graph of the evaluation metrics Reward and Coverage of NBV for the agents NBV1, NBV 2 and NBV 3.

preventing the agent NBV 4 from learning a favorable action selection strategy. The following simulations are conducted with action mapping variants using Spherical coordinates.

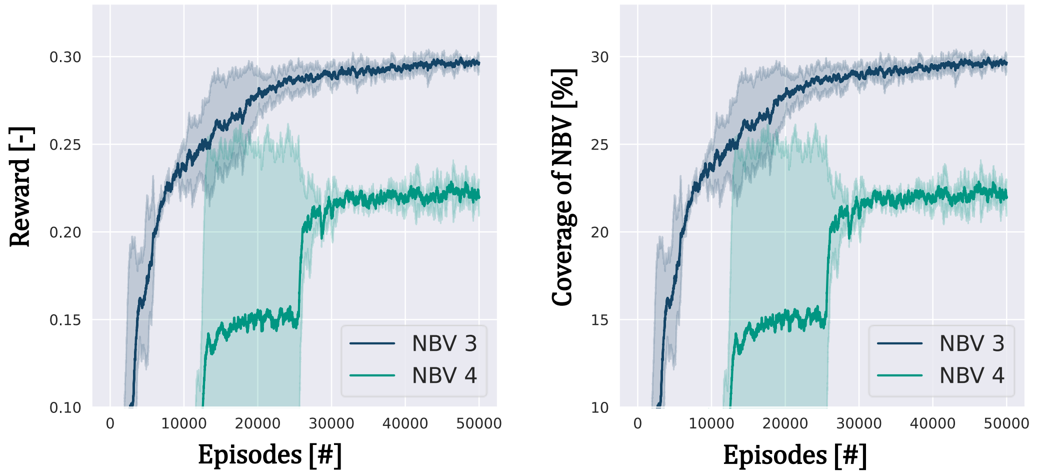


Figure 6.12: Graph of the evaluation metrics Reward and Coverage of NBV for the agents NBV 3 and NBV 4.

6.2.4 Results of the RL agent training using the proposed simulation framework for the RPP problem

Having demonstrated the effectiveness of the RL agent modeling approach in solving NBV, the performance on the RPP problem will be evaluated in the following.

6.2.4.1 Basic agent modeling results

Four basic agents (RPP 1-4) and their corresponding evaluation metrics in solving RPP are presented in Table 6.19. These agents differ in their choice of action mapping ($A_{2S,0R}$ for RPP 1, $A_{3C,0R}$ for RPP 2, and $A_{3S,0R}$ for RPP 3 and RPP 4) and the encoder used (PCN for RPP 1-3 and PN for RPP 4). The evaluation metrics graphs, including coverage, travel distances, required number of acquisitions, and distance to the origin, are shown in Figure 6.13.

Table 6.19: Overview of the agents and evaluation metrics evaluated in the RPP.

Agent information					Metrics		
Model Abbr.	**S**	**A**	**R**	**Encoder**	$\mathbf{\Psi_E}$	$\mathbf{n_E}$	$\mathbf{d_E}$
RPP 1	S_1	$A_{2S,0R}$	R_1	PCN	93.95%	5.43	340.14
RPP 2	S_1	$A_{3C,0R}$	R_1	PCN	89.70%	7.81	427.80
RPP 3	S_1	$A_{3S,0R}$	R_1	PCN	92.94%	5.72	341.29
RPP 4	S_1	$A_{3S,0R}$	R_1	PN	92.08%	6.46	358.29

All agents exhibit learning behavior aimed at optimizing the goal enforced by the reward signal R_1, maximizing surface coverage by sequentially adjusting the acquisition system. This behavior is evident from the upper left graph in Figure 6.13, which illustrates the surface coverage Ψ_E at the end of episodes throughout the agents' learning cycle.

Remarkably, to achieve the required surface coverage $\Psi_T > 90\%$, the agents adjust the distance of their poses from the origin during the learning process (lower right graph in Figure 6.13). This is also visualized for one of the RPP 3 agents in the left graph of Figure 6.14 in the left graph where also the standard deviation of the distance to the inspection object is visualized. The adjustment is necessary because the sensor model operates at a fixed working distance and is ideally positioned as far away from the inspected object as possible to maximize the surface area covered. However, the distance cannot be too large, as this would cause the acquisition system to be too far away from the object, resulting in the object being out of the system's field of view, resulting in empty acquisitions (right graph of Figure 6.14). Thus, by adjusting the distance of poses from the origin, the amount of empty acquisitions (visualized in the right graph of Figure 6.14) is minimized.

Using state S_1 (see Table 6.19 and section 5.3.4 for details) for agent training yields the primary insight: solving RPP in the model-free scenario is feasible using RL. This is explained in section 5.3.4.1, where it's explained that state S_1 relies only on information from the current acquisition step. Unlike state S_3, which explicitly requires an object model, state S_1 doesn't require any knowledge of the geometry of the inspection object.

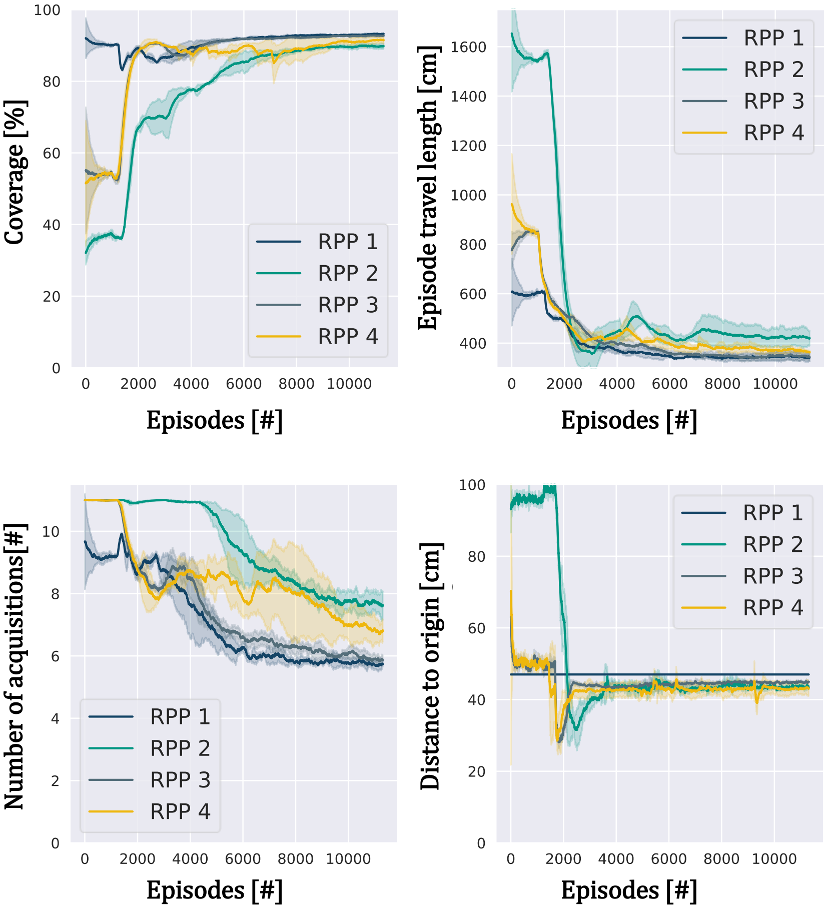


Figure 6.13: Graph of the evaluation metrics Coverage Ψ_E, Episode travel distances d_E, Required number of acquisitions n_E and Distance to origin C_{RT} for the agents RPP 1, RPP 2, RPP 3 and RPP 4.

It's interesting to note that, although not explicitly enforced by the reward signal R_1, the number of acquisitions needed to reach Ψ_T decreases throughout the learning cycle (lower left graph of Figure 6.13). This decrease can be attributed to the discount factor $\gamma_{RT} = 0.9$, which incentivizes the agent to prioritize acquisitions with higher rewards earlier in the process. As a result, the desired 90% surface area coverage is achieved in fewer steps, reducing the total number of acquisitions required. Figure 6.15 illustrates this trend. Although the surface coverage of the initial acquisition remains fixed throughout the learning process (approximately 26%, similar to NBV), the agents excel at maximizing the additional surface area gained in the initial acquisitions. In particular, agent RPP 3 demonstrates particularly efficient maximization of the additional surface coverage acquired in the first four acquisitions, namely $\Delta\Psi_1$, $\Delta\Psi_2$, $\Delta\Psi_3$, and $\Delta\Psi_4$.

Comparing agents RPP 1-4 in terms of achieved surface area coverage, it's evident that only agent RPP 2, which uses mapping to Cartesian coordinates, fails to achieve the required

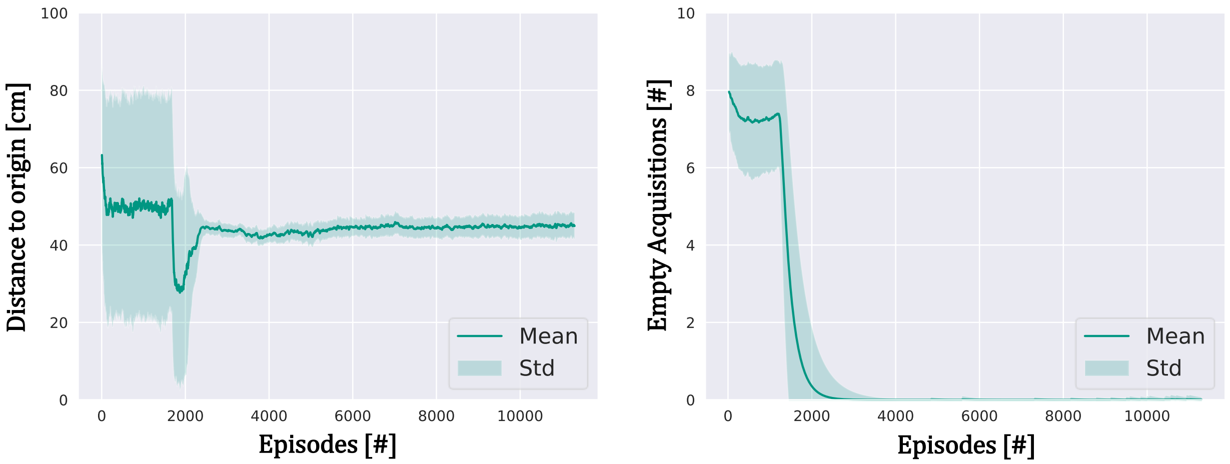


Figure 6.14: Graph of the distance to origin C_{RT} and empty acquisitions over the learning process of one of the RPP 3 agents.

average surface area coverage of $\Psi_T > 90\%$. Conversely, all three agents (RPP 1, RPP 3, and RPP 4) that use mapping to spherical coordinates achieve $\Psi_T > 90\%$ on average. These results are consistent with those observed in the NBV problem experiments, where agent NBV 4 with Cartesian mapping also performed worse than agents with spherical mapping. Additionally, it's noteworthy that despite having a lower average total surface coverage Ψ_E, agent RPP 2 requires more acquisitions n_E on average, resulting in longer travel distances d_E.

Furthermore, insights from the NBV problem suggest that action mappings A_{2S0R} and A_{3S0R} yield similar evaluation metrics since the agent can learn the optimal working distance independently. In addition, a comparison between agents RPP 3 and RPP 4, which differ only in the encoder network used (PCN for RPP 3 and PN for RPP 4), sheds further light on agent design. Although both agents achieve similar average surface coverage Ψ_E after an episode, it's clear that using PN as the encoder results in higher average acquisitions. RPP 4 converges as fast as its counterpart RPP 3 in terms of surface coverage optimization (upper left graph of Figure 6.13). Yet, it continues to minimize the necessary acquisitions throughout the learning cycle, as shown in the lower left graph in Figure 6.13. Notably, while RPP 4 achieves convergence in surface coverage after approximately 4000 episodes, it also exhibits significant variance in convergence speed regarding the number of needed acquisitions (lower left graph in Figure 6.13) of the RL agents trained with this configuration, indicated through the wide spread of lower und upper bound. In contrast, the variance between learning runs for agents RPP 1 and RPP 3 is significantly lower while additionally converging faster.

Figures 6.16 and 6.17 show the distribution of VP positions for the ten observations in space for agent RPP 3. The VP positions show spatial clustering, which is particularly noticeable

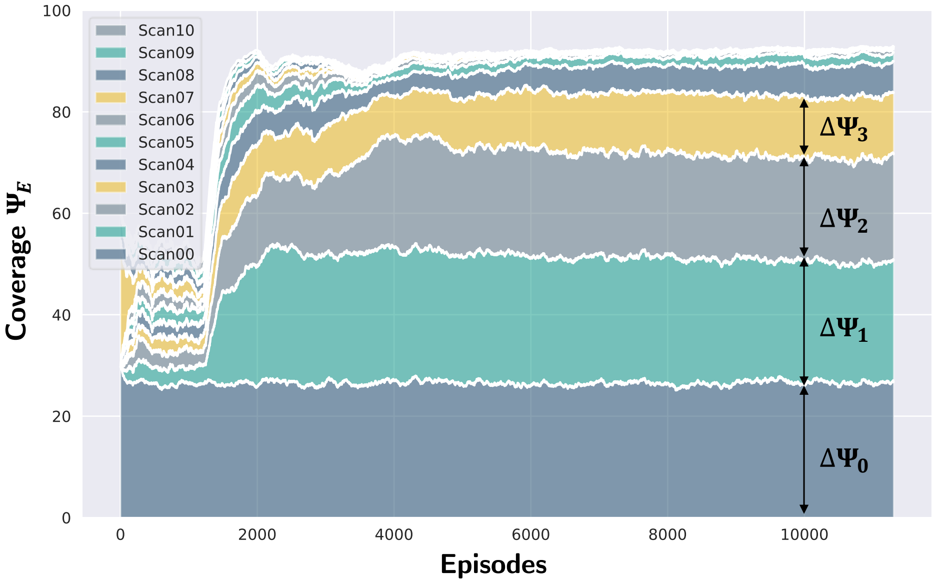


Figure 6.15: Plot of the distribution of the difference values $\Delta\Psi_t$ of the surface coverage Ψ_E over the learning process of the agent RPP 3.

in the initial observations. Similar to NBV, the first acquisition after the first one forms a crescent-shaped cluster compared to the first acquisition. With successive acquisitions, these clusters become larger and less structured.

In the initial acquisitions, certain points, highlighted by intense yellow coloring, are approached disproportionately often. However, this pattern diminishes in subsequent acquisitions. By acquisition 5 ($t = 5$), points on a hemisphere are uniformly approached. This suggests that individual poses are favored in the early stages of acquisition due to their potential for high surface coverage. As acquisition progresses and surface coverage accumulates, the agent adjusts its approach based on acquisition progress and available surface coverage.

6.2.4.2 Comparison of the basic agent modeling alternatives with benchmark algorithms

Table 6.20 compares the best-performing basic agent model against random and heuristic benchmark algorithms. The results show that the random benchmark, which has no learning behavior, achieves an average surface coverage Ψ_E of 74.86% after six acquisitions in a single episode.

Conversely, the heuristic achieves an average surface coverage of 96.33% with the same number of acquisitions as the random benchmark. However, the heuristic approach entails a high average travel distance d_E of 619.37. This is attributed to the heuristic algorithm determining acquisition system poses around the object without actively considering minimizing

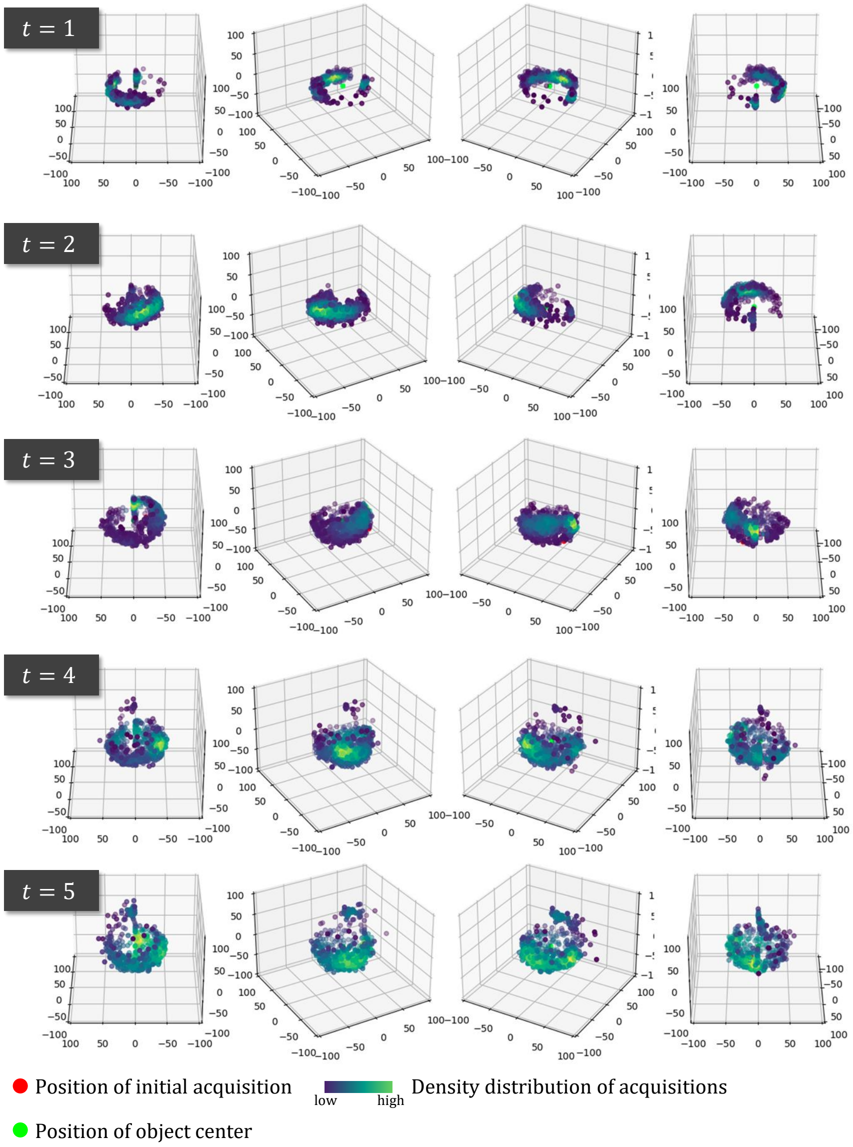


Figure 6.16: Plots of the first 5 poses $\mathbf{p}_t^{A/RT} (t \in [1, 5])$ issued by the agent RPP 3 after convergence.

travel distances or reordering the sequence of poses to minimize travel distances between successive acquisitions.

RPP1 has an average surface coverage Ψ_E of 93.95% after six acquisitions, close to the heuristic benchmark. In addition, it has the lowest average travel distance to reach Ψ_E, with an average of 5.43 acquisitions and the shortest total travel distance of 340.14 among the three approaches compared.

In summary, RPP1 outperforms the random benchmark in surface coverage Ψ_E and travel distance d_E. While it falls short of the heuristic method's surface coverage, RPP1 requires

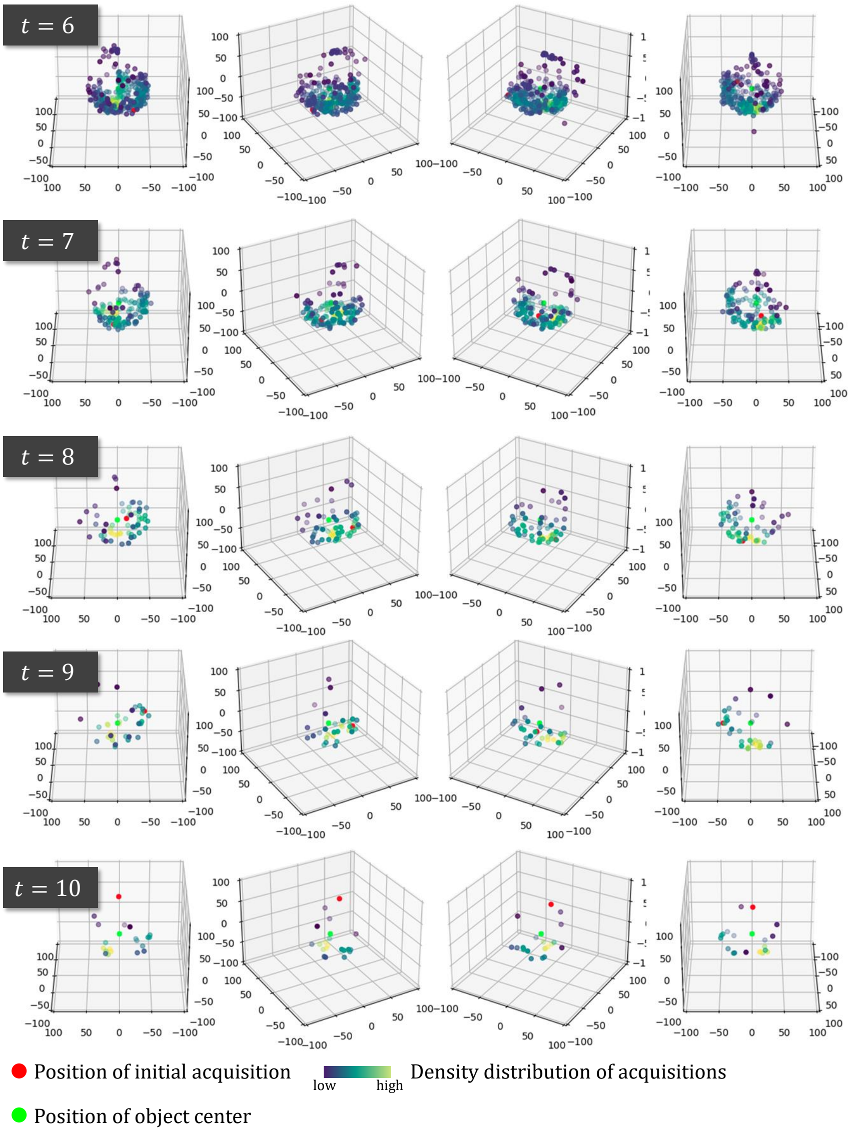


Figure 6.17: Plots of the last 5 poses $\mathbf{p}_t^{A/RT}(t \in [6,10])$ *issued by the agent RPP 3 after convergence.*

fewer acquisitions on average. It covers less travel distance, highlighting the superiority of the RL approach over the simple heuristic benchmark algorithm.

6.2.4.3 State modeling alternative results

While the previous section focused on basic agents, the following section delves deeper into the variations resulting from different state, action, and agent modeling approaches. The quantitative results of these variations are detailed in Table 6.21.

Comparing agents RPP 3 and RPP 5, there are no significant differences in the evaluation metrics Ψ_E, n_E, and d_E. RPP 3 uses the model-free state S_1, while RPP 5 uses the binary-coded model-based state S_4. This suggests that a geometry model may not be essential

Table 6.20: Overview of the evaluation metrics of the random and heuristic benchmark and the RPP1 agent.

Benchmark	$\mathbf{\Psi_E}$	$\mathbf{n_E}$	$\mathbf{d_E}$	$\mathbf{\Psi_1}$	$\mathbf{\Psi_2}$	$\mathbf{\Psi_3}$	$\mathbf{\Psi_4}$	$\mathbf{\Psi_5}$	$\mathbf{\Psi_6}$
Random	74.86%	6	386.14	23.63%	41.01%	53.36%	63.22%	70.30%	74.86%
Heuristic	96.33%	6	619.37	-	-	-	-	-	96.33%
RPP 1	93.95%	5.43	340.14	26.98%	50.67%	70.24%	82.91%	90.96%	93.39%

Table 6.21: Overview of the agents' evaluation metrics for varying agent modeling of state modeling and action mapping used.

Agent information						**Metrics**		
Model Abbr.	**State type**	**S**	**A**	**R**	**Agent**	$\mathbf{\Psi_E}$	$\mathbf{n_E}$	$\mathbf{d_E}$
RPP 3	Model-free	S_1	$A_{3S,0R}$	R_1	SAC	92.94%	5.72	341.29
RPP 5	Model-based	S_4	$A_{3S,0R}$	R_1	SAC	93.31%	5.40	348.32
RPP 6	Model-free	S_2	$A_{3S,0R}$	R_1	SAC	93.09%	5.67	348.90
RPP 7	Model-based	S_5	$A_{3S,0R}$	R_1	SAC	92.82%	5.67	336.57
RPP 8	Model-based	S_5	$A_{3S,2R}$	R_1	SAC	31.94%	11	580.89
RPP 9	Model-free	S_2	$A_{3S,2R,lim}$	R_1	SAC	92.91%	6.01	324.32
RPP 10	Model-free	S_2	$A_{3S,2R,lim}$	R_1	PPO	90.22%	9.09	535.80
RPP 11	Model-based	S_5	$A_{3S,2R,lim}$	R_1	SAC	93.23%	5.90	339.70
RPP 12	Model-free	S_3	$A_{3S,2R,lim}$	R_1	SAC	93.04%	5.79	394.28
RPP 13	Model-based	S_6	$A_{3S,2R,lim}$	R_1	SAC	93.24%	5.84	404.60

for determining the optimal view plan, particularly in applications involving different variants of starter motors. This could be due to the similarity of sequential poses for geometrically similar products. With minimal variation in boundary conditions (such as starter type and initial orientation), an agent such as RPP 3 can adapt without requiring knowledge of the geometry model or its spatial orientation.

The agents RPP 6 and RPP 7 are extensions of RPP 3 and RPP 5. These agents only differ from RPP 3 and RPP 5 in their state, which contains normal vectors in addition to the point cloud coding (S_2 in the model-free case for RPP 6 and S_5 in the model-based case for RPP 6). However, when examining the evaluation metrics, RPP 6 and RPP 7 do not differ significantly from their counterparts, RPP 3 and RPP 5, in the converged state. This is evident both in the quantitative tabular overview (Table 6.21) and in Figure 6.18. Interestingly, the inclusion of normal vectors can be detrimental to convergence speed. This can be seen by comparing, for example, RPP 5 and RPP 7. While RPP 7 integrates normal vector information, it converges slower than its counterpart, RPP 5, which does not. RPP 5, which uses the model-based state S_4 based on the ground truth point cloud $\mathbf{P}_{GT}$ and binary coding of the covered points, shows the fastest convergence in terms of required acquisitions compared to RPP 3, RPP 6, and RPP 7. Thus, a model-based state supports faster convergence than a model-free state

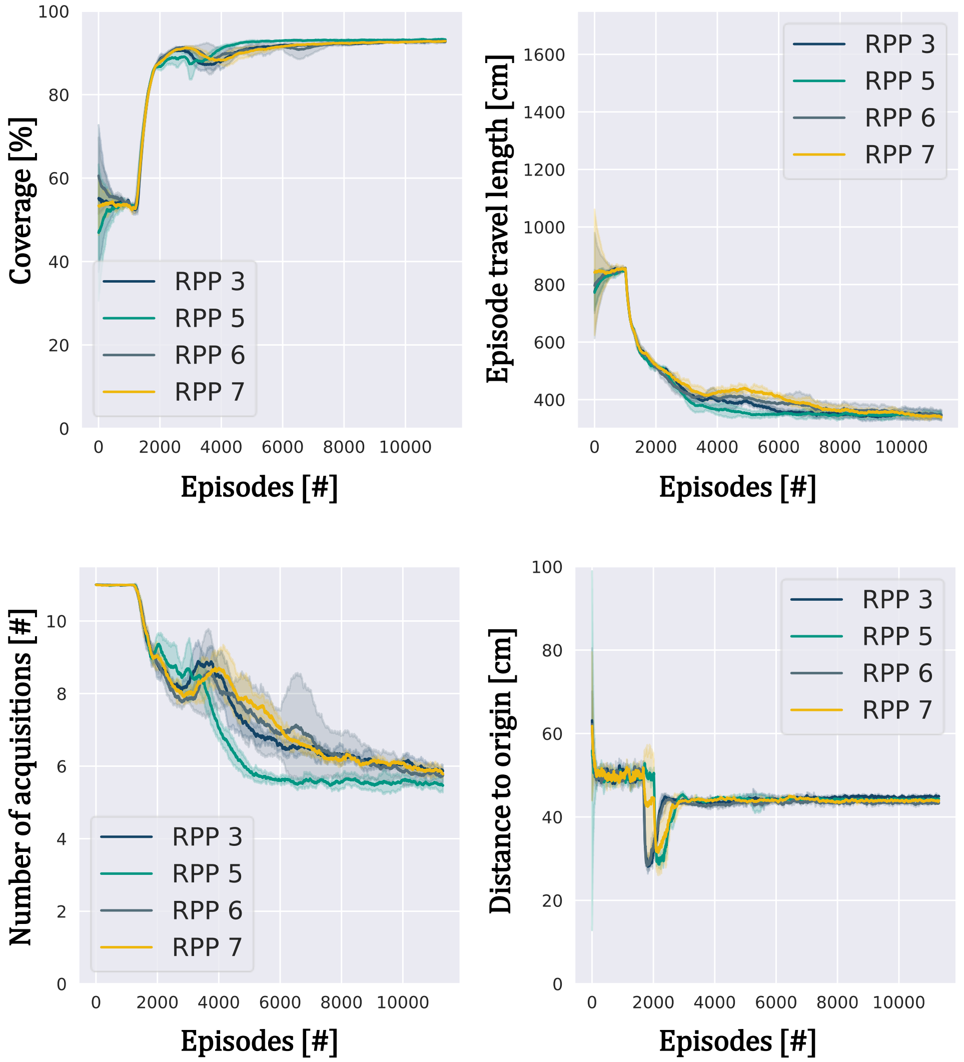


Figure 6.18: Graph of the evaluation metrics Coverage Ψ_E, Episode travel distances d_E, Required number of acquisitions n_E and Distance to origin C_{RT} for the agents RPP 3, RPP 5, RPP 6 and RPP 7.

when only point information is passed. However, this advantage is negated when normal vectors are integrated into the state, since RPP 6 (model-free with normal vector information) and RPP 7 (model-based with normal vector information) show similar convergence speed.

Up to this point, the discussed agents have only been able to determine the position of the pose $\mathbf{p}_t^{A/RT}$ in space. The orientation of these agents is determined solely by their orientation towards the origin of the coordinate system C_{RT}. However, agents RPP 8-13 also allow for the orientation of the acquisition system to be defined. Agent RPP 8 employs the $A_{3S,2R}$ action mapping. This agent modeling variant is able to set the pose of the acquisition system as well as the orientation of the inspected starter motor by controlling the rotary degree of freedom of the rotary table without integrating any prior knowledge. However, quantitative results of evaluation metrics in Table 6.21 and associated graphs in Figure 6.19 indicate that the trained agents for agent modeling RPP 8 are not able to learn a satisfactory strategy, based on the poor performance regarding surface coverage and the needed amount of acquisitions, since the agent modeling is not able to achieve $\Psi_T > 90\%$.

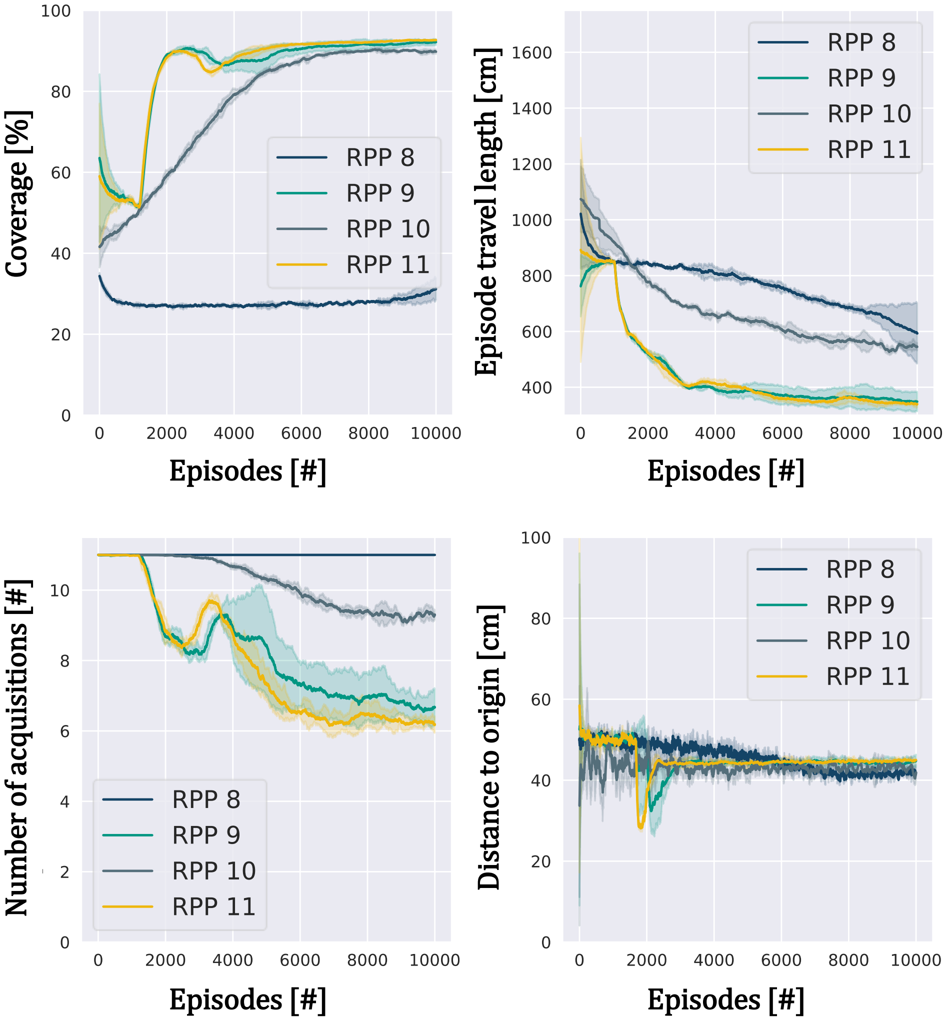


Figure 6.19: Graph of the evaluation metrics Coverage Ψ_E, Episode travel distances d_E, Required number of acquisitions n_E and Distance to origin C_{RT} for the agents RPP 8-11.

Building on this result, agents RPP 9-13 adopt the action mapping $A_{3S,2R,lim}$, allowing only angle-based deviations from the predefined orientation of the acquisition system to the origin of C_{RT}. Despite initial challenges in convergence, state models S_2 and S_5 are used because they have minimal impact on evaluation metrics in the converged state, primarily influencing convergence speed while providing more comprehensive information. In addition, agent modeling for RPP 10 uses PPO as the RL algorithm.

An analysis of the behavior of agent RPP 10 compared to agents RPP 9 and RPP 11 shows that while RPP 10 achieves the required surface coverage of $\Psi_E \geq 90\%$ on average, it requires a significantly higher number of acquisitions n_E, resulting in longer travel distances d_E. Figure 6.19 shows that RPP 10 converges much slower than RPP 9 and RPP 11. Overall, RPP 10, which uses PPO as the RL method, shows inferior performance in the VPP problem. Additionally, comparing agents RPP 9 and RPP 11 with their counterparts RPP 6 and RPP 7, which cannot output an angle-based deviation from C_{RT}, shows that RPP 9 and RPP 11 tend to perform even worse. Although they achieve comparable surface coverage Ψ_E on average, RPP 9 and RPP 11 require a higher average number n_E of acquisitions despite slightly smaller average travel distances d_E.

Finally, agents RPP 12 and RPP 13 are analyzed. Their state models S_3 and S_6 contain information about the acquired surface points from the last acquisition and the pose approached by the acquisition system. The results in Table 6.21 show that these agents also achieve $\Psi_E \geq 90\%$ and require fewer acquisitions on average compared to RPP 9 and RPP 11. However, RPP 12 and RPP 13 show significantly higher travel distances than their counterparts, RPP 9 and RPP 11, which lack additional information about the last pose and the acquired surface points in their state.

In summary, it's clear that state modeling and action mapping significantly impact agent behavior. Incorporating additional information doesn't always lead to improved performance on all evaluation metrics. A reward signal was used in the experiments that only enforced surface coverage maximization. Other evaluation metrics, such as number of acquisitions n_E until the end of an episode and distance traveled d_E in an episode, are not explicitly addressed in the reward signal, resulting in different levels of optimality across agent modeling variants. Therefore, in the following section, selected variants of different reward functions are evaluated to explore the possibility of optimizing multiple objective variables simultaneously.

6.2.4.4 Reward modeling alternative results

Reward signals used in this work are introduced in section 5.3.5. The reward signal R_1 used to reward based on the area $\Delta\Psi_t$ not yet covered in the last acquisition relative to the theoretically still possible area $\Psi_{rem,t-1}$ to be covered. The reward signal R_2, on the other hand, only rewards based on the area $\Delta\Psi_t$ not yet covered after each acquisition step. The reward signal R_3 further scales R_1 based on the length of the travel distance d_t between the position of the last acquisition and the current acquisition pose. R_3 is also a dense reward signal and rewards after each acquisition step. Agent RPP 14 uses R_2, while RPP 15 uses R_3. For an overview of the evaluation metrics of the agent modeling variants considered in this section, see Table 6.22.

Table 6.22: Overview of the agents' evaluation metrics for varying reward modeling alternatives used.

Agent information						Metrics		
Model Abbr.	**Reward type**	**S**	**A**	**R**	**Agent**	$\mathbf{\Psi_E}$	$\mathbf{n_E}$	$\mathbf{d_E}$
RPP 1	Dense	S_1	$A_{2S,0R}$	R_1	SAC	93.95%	5.43	340.14
RPP 12	Dense	S_3	$A_{3S,2R,lim}$	R_1	SAC	93.04%	5.79	394.28
RPP 14	Dense	S_3	$A_{3S,2R,lim}$	R_2	SAC	92.64%	6.37	419.20
RPP 15	Dense	S_3	$A_{3S,2R,lim}$	R_3	SAC	93.54%	6.69	251.08
RPP 16	Sparse	S_3	$A_{2S,0R}$	R_8	SAC	93.19%	4.73	340.19
RPP 17	Sparse	S_3	$A_{2S,0R}$	R_9	SAC	93.49%	4.80	340.74
RPP 18	Sparse	S_3	$A_{3S,2R,lim}$	R_6	SAC	53.08%	11	588.97
RPP 19	Sparse	S_3	$A_{3S,2R,lim}$	R_7	SAC	43.00%	11	400.46

A direct comparison between RPP 12 and RPP 14 shows that the simpler modeling of the reward signal R_2 has a negative impact on the evaluation metrics of travel distance d_E and required number of acquisitions n_E. While both agents achieve the required surface coverage $\Psi_E \geq 90\%$, RPP 14 converges more slowly with this reward signal, as shown in Figure 6.20. Comparing RPP 12 with RPP 15, which uses reward signal R_3, demonstrates the potential of multi-criteria evaluation metric optimization. By integrating the scaling term of the travel distance d_t between acquisitions, RPP 15 minimizes the cumulative travel distance d_E for the acquisition process of an inspection object. However, the slightly more complex optimization problem results in a slower convergence process for the trained agents.

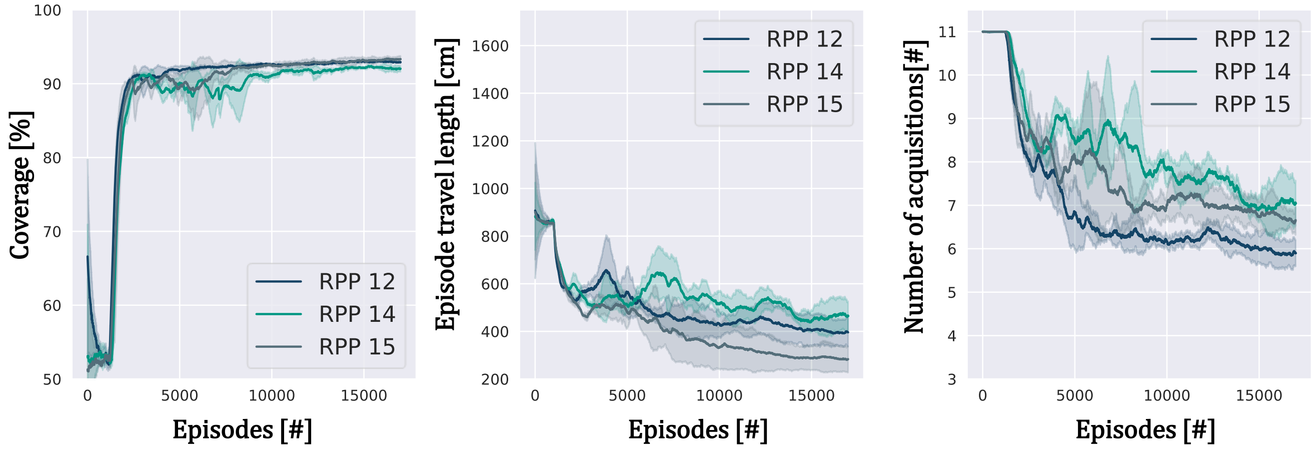


Figure 6.20: Graph of the evaluation metrics Coverage Ψ_E, Episode travel distances d_E and Required number of acquisitions n_E for the agents RPP 12, RPP 14 and RPP 15.

When using sparse reward signals, the comparison starts with agents RPP 18 and RPP 19. RPP 18 uses the reward signal R_6, which rewards the RL agent with a constant value (1) when the target state s_G is reached at $\Psi_T \geq 90\%$. Similarly, RPP 19's reward is based on the achieved surface area coverage Ψ_E divided by the required number of acquisitions n_E. However, the results indicate that neither agent, RPP 18 or RPP 19, achieves comparable surface coverage to agents with dense reward signals. This could be attributed to the rarity of reaching the target state s_G at $\Psi_T \geq 90\%$ within the 11 allowed acquisitions for RPP 18 and RPP 19. Consequently, these agents receive sparse rewards, which hinders their ability to discriminate between beneficial and less beneficial actions in states.

In the case of sparse rewards, two simplifications are used, resulting in agent modeling variants RPP 16 and RPP 17. First, these agent models use action modeling $A_{2S,0R}$, and second, both agent models use sparse reward signals that do not reward the agent upon reaching the goal state s_G, but rather upon reaching a terminal state s_T. This ensures that these agent models are consistently rewarded at the end of an episode, i.e., after an acquisition process of the inspection object. By choosing the action modeling $A_{2S,0R}$ instead of $A_{3S,2R,lim}$, the degrees of freedom of the agent models RPP 16 and RPP 17 are further restricted, which simplifies the learning process for the agents.

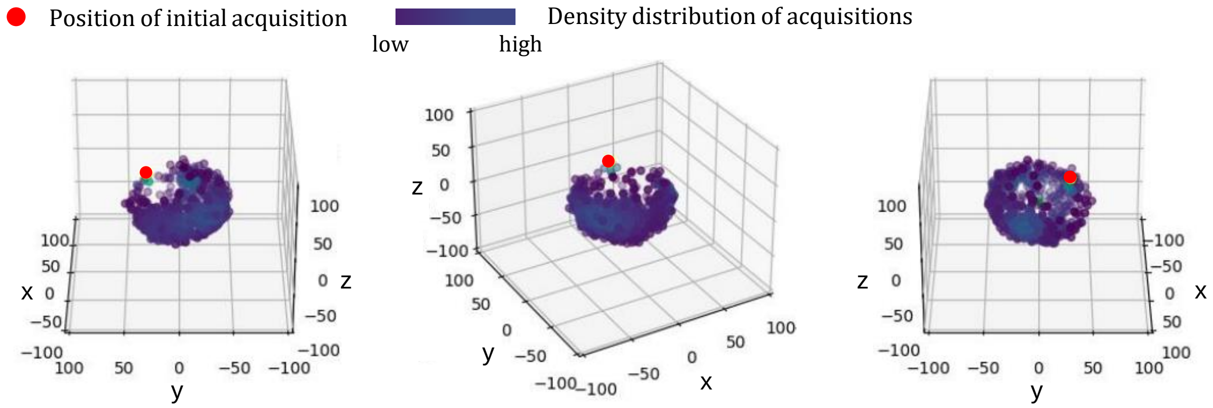


Figure 6.21: Positional components of the poses $\mathbf{p}_t^{A/RT}$ issued by the agent with agent modeling RPP 12 in the last 200 training episodes.

Comparing RPP 16 and RPP 19 shows the benefits of restricting the agents' degrees of freedom (DoF) and choosing a reward signal that consistently rewards the agents at the end of an episode. Unlike the agent modeling in RPP 19, RPP 16 demonstrates learning and convergence, while agent 19 does not maximize surface coverage or minimize the amount of acquisitions. It always performs the maximum amount of 11 acquisitions in an episode until said episode is terminated. Surprisingly, RPP 16 outperforms agent models RPP 1 and RPP 12, which use dense reward signals. While these agents achieve comparable surface coverage Ψ_E at the end of an episode, RPP 16 exhibits lower travel distance d_E than RPP 12 and comparable travel distance d_E compared to RPP 1. The notable difference lies in the number of acquisitions n_E required to achieve surface coverage Ψ_E, which is lower for RPP 16 than for RPP 1 and RPP 12 in both cases.

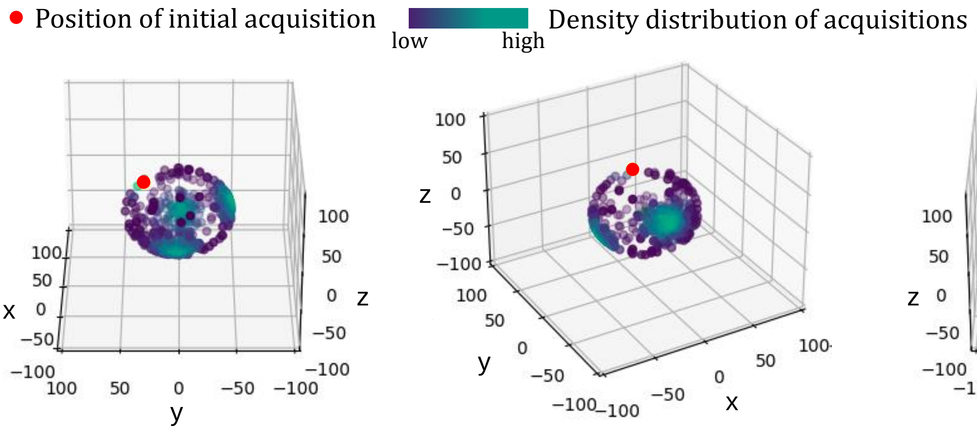


Figure 6.22: Positional components of the poses $\mathbf{p}_t^{A/RT}$ issued by the agent with agent modeling RPP 16 in the last 200 training episodes.

A possible explanation can be found in Figures 6.21 and 6.22. Figure 6.22 shows the density distribution of the positional components in space for the last 200 episodes of poses $\mathbf{p}_t^{A/RT}$ approached by one of the trained agents using agent modeling RPP 16. Conversely, Figure 6.21 shows the density distribution of the last 200 episodes of poses $\mathbf{p}_t^{A/RT}$ approached by an agent using agent modeling RPP 12. The concentration of poses in Figure 6.22 is significantly higher than in Figure 6.21. This suggests that agents using agent modeling RPP 16 learn a globally optimal distribution of pose positions to achieve the final surface coverage $\Psi_E \geq 90\%$ with the fewest acquisitions n_E, as dictated by the reward signal R_8. In contrast, agents using agent modeling RPP 12 tend to have a more widely distributed density distribution. Here, the reward signal R_1 implicitly minimizes the number of acquisitions, and agent modeling RPP 12 also implicitly emphasizes a globally optimal solution for VPP. Consequently, the agent modeling prioritizes solving the NBV problem over the RPP problem. This results in a higher number of necessary acquisitions to reach the goal $\Psi_E \geq 90\%$, considering that the choice of NBV is not always optimal for solving RPP (see section 2.2.1).

6.2.4.5 Training results on real starter motor data and comparison with benchmark algorithm SCP for the RPP

The previous results discussed the results of agent modeling using the D1 dataset. Now the focus shifts to examining the transferability of these results to the D2 dataset and comparing them to the analytical solution approaches using the SCP benchmark algorithms.

Table 6.23 presents the results of RPP 12 and RPP 16, as discussed earlier. Additionally, the outcomes of agents RPP 20 and RPP 21 are included. These agents differ only in their

Table 6.23: Overview of the evaluation metrics of selected agent modeling with dense and sparse reward configuration trained with the data sets D1 and D2.

Agent information						**Metrics train**			**Metrics test**		
Abbr.	**Reward**	**S**	**A**	**R**	**Data**	$\mathbf{\Psi_E}$	$\mathbf{n_E}$	$\mathbf{d_E}$	$\mathbf{\Psi_E}$	$\mathbf{n_E}$	$\mathbf{d_E}$
RPP 12	Dense	S_3	$A_{3S,2R,lim}$	R_1	D1	93.04%	5.79	394.28	-	-	-
RPP 16	Sparse	S_3	$A_{2S,0R}$	R_8	D1	93.19%	4.73	340.19	-	-	-
RPP 20	Dense	S_3	$A_{3S,2R,lim}$	R_1	D2	93.11%	5.59	351.77	93.56%	5.57	347.23
RPP 21	Sparse	S_3	$A_{2S,0R}$	R_8	D2	93.05%	4.02	275.19	92.06%	5.47	417.41

training dataset, being trained on dataset D2 instead of D1. Without delving into specific numerical values, it was demonstrated that successfully training the modeled RL agents is achievable using dataset D2, with comparable evaluation metrics to those trained with dataset D1. RPP 20 and RPP 21 were additionally evaluated on previously unseen starter motor models (5 out of the 40 starter motor models from dataset D2 were excluded from training and used for testing). It was observed that these agents generalize well to unseen models, indicating their robustness. However, in the case of RPP 21, which uses a sparse reward, there is a noticeable reduction in performance on the test data, particularly regarding the required number of acquisitions (n_E) and travel distance (d_E).

Table 6.24: Overview of the evaluation metrics of the analytical SCP VPP solution algorithms.

			Acquisition step $\mathbf{t}$					
	Dataset	**Metric**	$t = 1$	$t = 2$	$t = 3$	$t = 4$	$t = 5$	$t = 6$
$\mathbf{SCP^{D1,Cov}_{\Psi}}$	D1	Average Ψ_t	35.54%	68.75%	86.43%	94.03%	97.20%	98.59%
		Average d_t	0	140.57	224.13	347.65	433.47	544.26
$\mathbf{SCP^{D1,Cov}_{\Psi,d}}$	D1	Average Ψ_t	35.15%	63.96%	85.40%	93.59%	96.56%	98.04%
		Average d_t	0	108.51	192.60	255.08	323.20	398.22
$\mathbf{SCP^{D2,Cov}_{\Psi}}$	D2	Average Ψ_t	44.03%	81.50%	92.51%	97.02%	98.88%	99.45%
		Average d_t	0	138.02	223.95	318.48	422.21	517.43
$\mathbf{SCP^{D2,Cov}_{\Psi,d}}$	D2	Average Ψ_t	43.95%	76.44%	92.33%	96.30%	98.34%	99.22%
		Average d_t	0	111.16	176.24	228.70	305.33	378.76

The quantitative comparison between the benchmark algorithms $SCP_{\Psi}^{D1,cov}$ and $SCP_{\Psi}^{D2,cov}$, both aiming to maximize surface coverage and evaluated on datasets D1 and D2, respectively, yields comparable results. Table 6.24 shows the tabular results of evaluating the analytical SCP benchmark algorithms. Evaluated over the number of acquisitions, they each achieve similar values for the evaluation metrics Ψ_t and d_E. Notably, $SCP_{\Psi}^{D2,cov}$, evaluated on the dataset D2 containing real starter motor models, tends to achieve a larger average surface coverage Ψ_t with a smaller necessary average travel distance d_t. Similar results are observed when comparing $SCP_{\Psi,d}^{D1,cov}$ and $SCP_{\Psi,d}^{D2,cov}$. Both SCP variants seek a balance between maximizing surface coverage and minimizing the required distance. This can be seen in a direct comparison, such as between $SCP_{\Psi}^{D1,cov}$ and $SCP_{\Psi,d}^{D1,cov}$. While $SCP_{\Psi}^{D1,cov}$ achieves an average surface coverage of $\Psi_4 = 94.03\%$ with an average path length of $d_4 = 347.65$ after four acquisitions, $SCP_{\Psi,d}^{D1,cov}$ achieves a slightly lower surface coverage ($\Psi_4 = 93.59\%$) but significantly reduces the average path length required ($d_4 = 255.08$).

A direct comparison between the agents and the SCP benchmark algorithms shows that, despite the learning behavior of the agents, the benchmark algorithms outperform them. On average, the SCP benchmark algorithms reach a surface coverage Ψ_E comparable to that of the agents after only three acquisitions. Thus, on average, the agents require two to three more acquisitions than the analytical SCP benchmark algorithm to reach a surface coverage of $\Psi_E \sim 93\%$. However, a significant drawback of these SCP benchmark algorithms is that they must virtually evaluate 900 acquisitions to solve the VPP. This takes about 0.5 seconds per acquisition for a total of 450 seconds. In contrast, the agent can output the next pose in a few milliseconds due to the short computation time of the NN used as policy. Thus, the RL agent-based approach provides a real-time solution, albeit with more required acquisitions during the inspection than is ideal.

6.2.4.6 Application including the robot simulation

The previous results were obtained by allowing the RL agent to freely position itself around the object to be inspected, which only partially reflects the real application case of an acquisition system positioned on a robot. The following results discuss how the previously evaluated RL agents can solve the RPP problem when they also have to learn the reachability of selected poses. When a robotic system with reachability constraints is considered, Table 6.25 presents quantitative results for selected modeling variants of the RL agents trained on datasets D1-D3. While the state and reward modeling remains the same for these agents, new action modeling variants, $A_{3S,2R,lim,T}$ and $A_{2S,0R,T}$, were used. These variants allow rotation of the inspection object and use a restricted mapping of the agent's output to spherical coordinates to better represent the robot's workspace, as introduced in section 5.3.4.2. As a result, the agent must learn to control the rotation of the inspection object to compensate for the restricted motion due to the workspace constraint. It has to be noted that agents RPP 28 and RPP 29 trained on dataset D3 were also tested using five starter motors not used during training.

The insights gained from previous sections are further confirmed when considering the restriction of the agent's movement space, the possibility for the agent to define unreachable poses, and the additional degree of freedom of rotation of the rotary table. For instance, integrating the travel length between poses into the reward signal can impact the overall travel length of the trained RL agent. This is evident from comparing evaluation metric d_E for agents RPP 22 and RPP 23, where RPP 22 uses R_1 and RPP 23 uses R_3. Additionally,

Table 6.25: Overview of the evaluation metrics of the agents that have been trained using the robot simulation.

Agent information						**Metrics train**			**Metrics test**		
Abbr.	**Reward**	**S**	**A**	**R**	**Data**	$\mathbf{\Psi_E}$	$\mathbf{n_E}$	$\mathbf{d_E}$	$\mathbf{\Psi_E}$	$\mathbf{n_E}$	$\mathbf{d_E}$
RPP 22	Dense	S_3	$A_{3S,2R,lim,T}$	R_1	D1	90.72%	6.42	207.14	-	-	-
RPP 23	Dense	S_3	$A_{3S,2R,lim,T}$	R_3	D1	90.65%	6.76	166.30	-	-	-
RPP 24	Sparse	S_3	$A_{3S,2R,lim,T}$	R_8	D1	51.53%	10.99	148.60	-	-	-
RPP 25	Sparse	S_3	$A_{2S,0R,T}$	R_8	D1	91.33%	4.95	45.73	-	-	-
RPP 26	Dense	S_3	$A_{3S,2R,lim,T}$	R_1	D2	90.35%	5.65	151.30	87.73%	5.95	150.77
RPP 27	Sparse	S_3	$A_{2S,0R,T}$	R_8	D2	91.35%	4.43	49.07	91.21%	4.75	36.38
RPP 28	Dense	S_3	$A_{3S,2R,lim,T}$	R_1	D3	90.84%	5.60	103.70	91.46%	5.70	113.49
RPP 29	Sparse	S_3	$A_{2S,0R,T}$	R_8	D3	90.82%	5.07	64.85	91.41%	4.63	70.36

agent RPP 24, rewarded by a sparse reward signal R_8 with action modeling $A_{3S,2R,lim,T}$, fails to converge. Similar to evaluations without integrating a robot model, the sparse reward setting is effectively addressed by the RL agent only when the action space is restricted to fewer degrees of freedom. This is observed in the comparison of RPP 24 and RPP 25, both configured similarly, with the only difference being action modeling $A_{2S,0R,T}$ for agent RPP 25, which has fewer degrees of freedom compared to action modeling $A_{3S,2R,lim,T}$ of RPP 24.

Similar to the experiments without robot model integration, agents RPP 25, RPP 27, and RPP 29 with sparse reward signals show comparable results to each other and outperform their counterparts with dense reward signals (RPP 22 as the dense counterpart of RPP 25, RPP 26 as the dense counterpart of RPP 27, and RPP 28 as the dense counterpart of RPP 29).

The insights from training agents using dataset D1 can be directly applied to datasets D2 and D3. Additionally, similar to the scenario without integrating the robot model into the simulation, the trained agents generalize their learned strategies to data not present during training, as evidenced by the test evaluation metrics for agents RPP 26-29.

The quantitative results indicate that all trained RL agents effectively minimize the number of unreachable poses and empty acquisitions (resulting from the positioning of the acquisition system too far away or too near to the object or selecting an unreachable pose) defined by the agent. This is exemplified by one of the trained RL agents from the RPP 28 configuration and one from the RPP 29 configuration, as depicted in Figures 6.23 and 6.24.

A notable observation is that the RL agent in agent configuration RPP 28 initially has a significantly higher number of empty acquisitions and unreachable poses than its counterpart in agent configuration RPP 29. However, both configurations quickly adapt during training by consistently selecting reachable poses and performing acquisitions that improve surface coverage. There is a brief period where the RL agent in the RPP 29 configuration exhibits

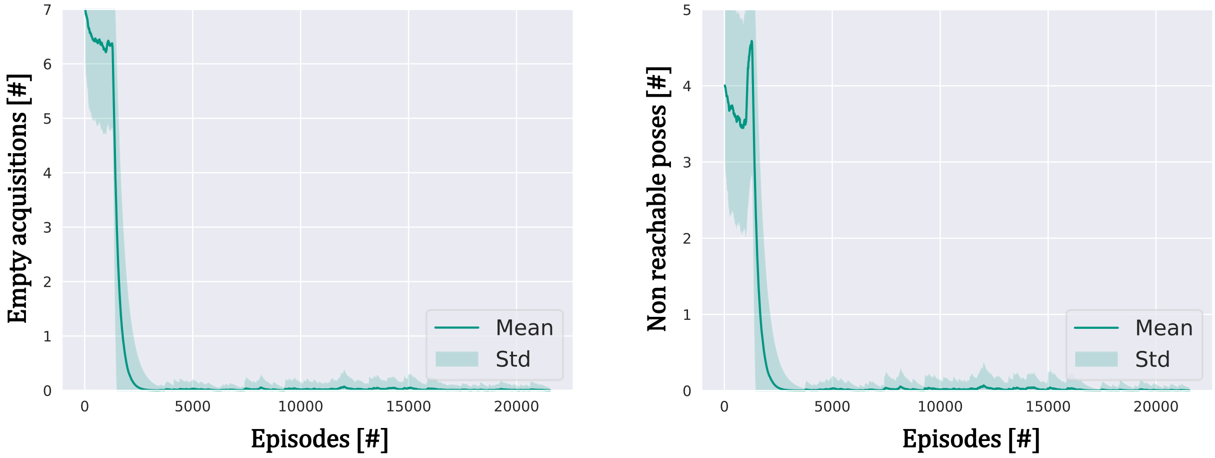


Figure 6.23: Graph of the empty acquisitions and non-reachable poses over the learning cycle for one agent of the agent configuration RPP 28.

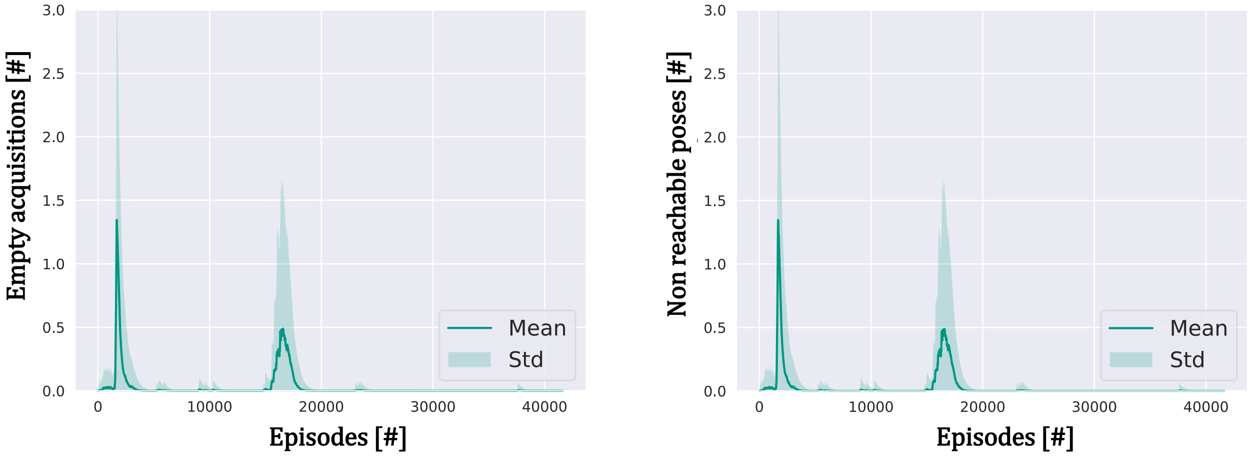


Figure 6.24: Graph of the empty acquisitions and non-reachable poses over the learning cycle for one agent of the agent configuration RPP 29.

a temporary decline in its learned policy, as evidenced by a temporary increase in empty acquisitions and unreachable poses. This may be due to the fact, that in the sparse setting, more variance in the observed rewards can be experienced, which may, in turn, favor bad gradient estimation by the RL algorithm and thus bad weight updates of the NN representing the action selection strategy (actor and critic when using SAC).

Comparing the positional components of poses issued by the RL agents in agent configurations RPP 28 and RPP 29 (displayed in Figures 6.25 and 6.26), along with the rotation angles (illustrated in Figures 6.27 and 6.28), provides interesting insights about their behavior across different acquisition steps.

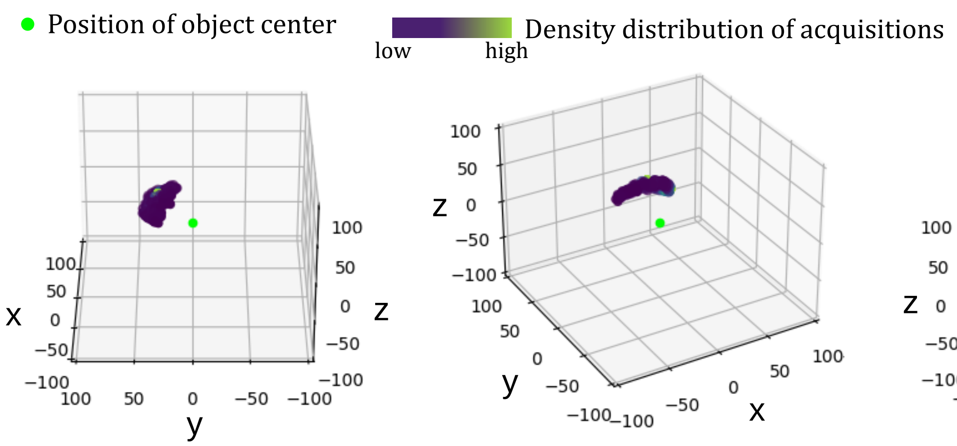


Figure 6.25: Positional components of the poses $\mathbf{p}_t^{A/RT}$ issued by the agent with agent modeling RPP 28 in the last 200 training episodes.

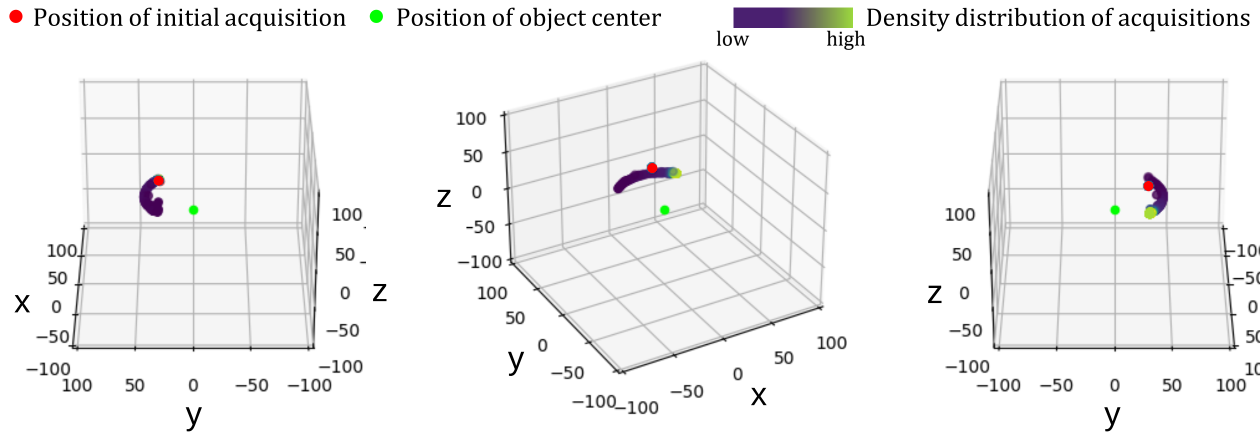


Figure 6.26: Positional components of the poses $\mathbf{p}_t^{A/RT}$ issued by the agent with agent modeling RPP 29 in the last 200 training episodes.

A direct comparison shows that the position components of the poses generated by the RL agent in agent configuration RPP 28 are relatively evenly distributed on a spherical surface within the workspace of the robot-mounted acquisition system. In contrast, the position components of the poses in agent configuration RPP 29 tend to cluster along a circular arc.

Comparing the output rotation angles of the rotary table, particularly in the early acquisitions (e.g., acquisitions 1 and 2), it's evident that the variance of these output rotation angles for the RL agent in agent configuration RPP 28 is significantly lower than that of the RL agent in agent configuration RPP 29. The RL agent in RPP 28 appears to achieve surface coverage maximization through a fixed rotation pattern and adjusting the acquisition system's position on the spherical surface. In contrast, the RL agent in RPP 29 maintains a constant positioning

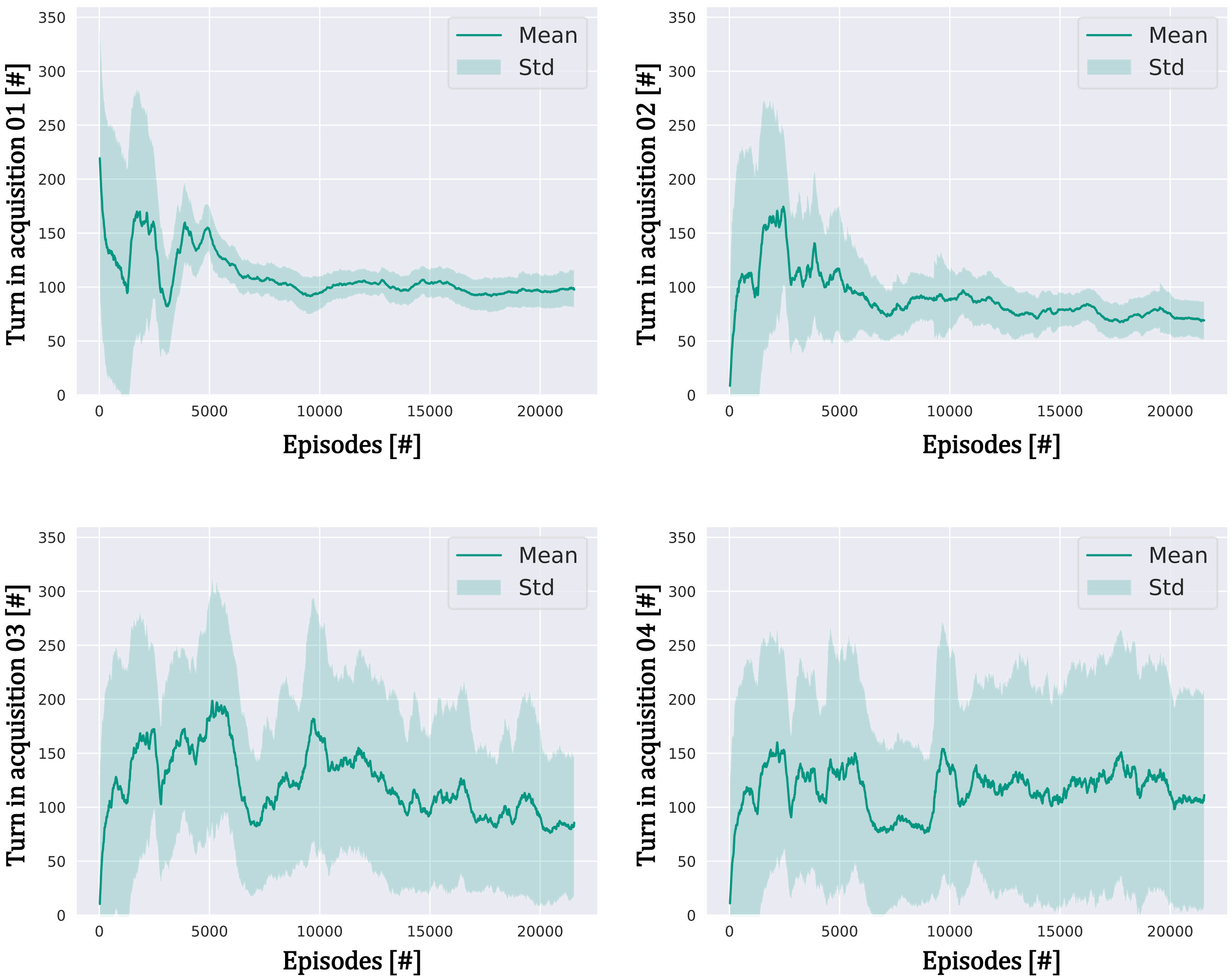


Figure 6.27: Mean curve and standard deviation of the rotation angles of the rotary motor in the individual acquisition steps throughout training for an RL agent of agent configuration RPP 28.

of the acquisition system along the circular arc while varying the rotation angle of the rotary table to maximize surface coverage.

6.2.4.7 Results for the RPP with multiple product categories on the Dataset D4

The results up to this point have been generated with products from one product category (starter motor) with variations of its variants. It has been shown that RL agents can also be used model-free (state modeling variants S_1-S_3), i.e. if no knowledge about the 3D model of the product to be inspected is available to derive the state of the RL agent.

For the following experiments, data set D4 is used, which contains not only the starter motor product category but also other object categories (bus, motorcycle, ...). The tabular

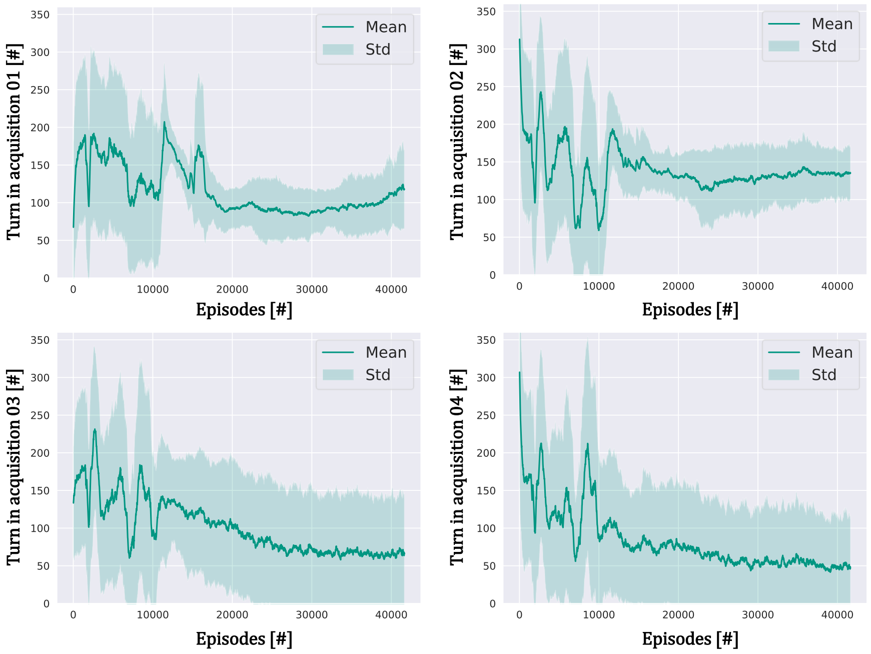


Figure 6.28: Mean curve and standard deviation of the rotation angles of the rotary motor in the individual acquisition steps throughout training for an RL agent of agent configuration RPP 29.

presentation of the quantitative results of the following explanations can be found in table 6.26.

Table 6.26: Overview of the evaluation metrics of the agents trained on dataset D4.

Agent information						Metrics		
Model Abbr.	**Reward type**	**S**	**A**	**R**	**Dataset**	$\Psi_\mathbf{E}$	$\mathbf{n_E}$	$\mathbf{d_E}$
Val 1	Dense	S_3	$A_{3S,2R,lim}$	R_1	D4	88.37%	8.56	577.62
Val 2	Dense	S_6	$A_{3S,2R,lim}$	R_1	D4	91.11%	6.41	492.18
Val 3	Sparse	S_3	$A_{2S,0R}$	R_8	D4	88.03%	7.51	416.91
Val 4	Sparse	S_6	$A_{2S,0R}$	R_8	D4	91.69%	5.7	436.88

First, it has been shown that the learning behavior of the RL agents also occurred in the case of multiple product categories. However, it can be noted that the variants of the RL agents that were trained with a model-free condition (Val 1 with dense reward signal R_1 and Val 3 with sparse reward signal R_8) performed significantly worse than their model-based counterparts (Val 2 and Val 4). The agents Val 2 and Val 4, with both dense and sparse reward signals, managed to maximize the surface coverage and reduce the number of acquisitions required for this.

Although the agents (especially Val 2 and Val 4) show a clear learning behavior, deficits are still identifiable. For example, the poses approached by the trained RL agents can be

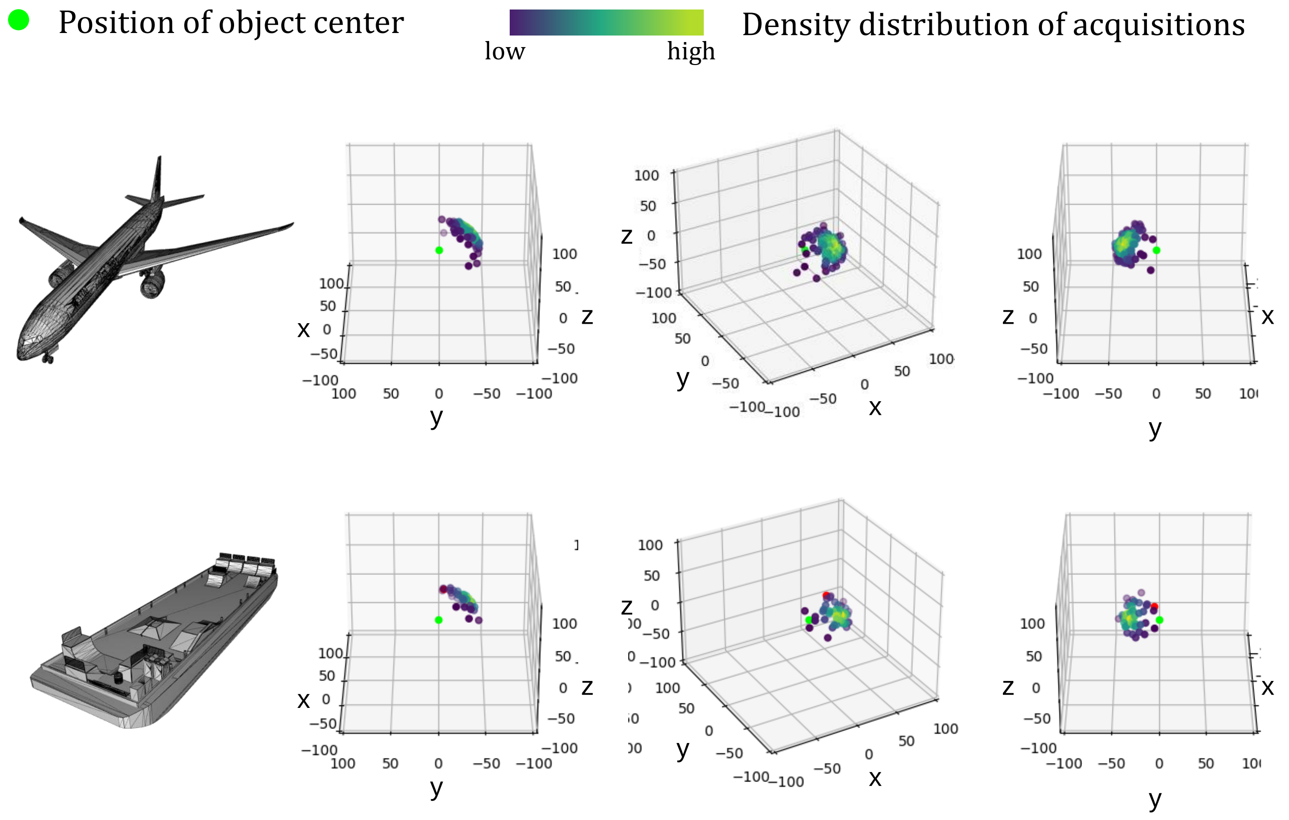


Figure 6.29: Distribution of the poses approached for one RL agent of the agent modeling Val 2 in the converged state for the object categories ship and airplane.

examined in dependence on the inspection object at hand. It shows the distribution of the first pose after the initial acquisition for one of the RL agents of agent modeling Val 2 in the converged state for the object categories *ship* and *airplane*. Results have shown that the trained RL agents maximize the surface area and minimize the number of acquisitions required for this but do so independently of the inspection object at hand. This means that independent of the inspection object, the agent always chooses the same poses as VP for the episode's second, third, and further acquisitions, resulting in a local optimum found by the agent.

It can thus be concluded that the RL agents solve the credit assignment problem in RL this problem class (RPP multiple object categories) but do so insufficiently for the realization of adaptive/ autonomous agent behavior.

6.2.5 Results of the RL agent training using the proposed simulation framework for the IPP problem

After the applicability of RL for solving the problems of NBV planning and RPP has been shown, it remains to be examined whether it is suitable for the problem of IPP.

The following results demonstrate the main findings that have emerged in this work when considering the IPP. Detailed investigations can be found in the work of A_Koch 2022, which was carried out under the supervision of the author of this thesis.

Quantitative results for investigating an exemplary problem of the IPP can be found in Table 6.27. For this purpose, two regions with 150 surface points each were randomly defined on each starter motor of the respective investigated data sets (D1, D2 and D3) in each episode. The task in training the RL agents was to select acquisition poses with as few acquisitions as possible to acquire these surface points fully. Variation of this problem is given by varying the number of regions and surface points per region (which ultimately corresponds to the size of the region to be acquired). These have been investigated in detail in the work of A_Koch 2022.

Table 6.27: Overview of the evaluation metrics of the agents trained using the robot simulation.

Agent information						**Metrics train**			**Metrics test**		
Abbr.	**Reward**	**S**	**A**	**R**	**Data**	$\mathbf{\Psi_E}$	$\mathbf{n_E}$	$\mathbf{d_E}$	$\mathbf{\Psi_E}$	$\mathbf{n_E}$	$\mathbf{d_E}$
IPP 1	Dense	S_4	$A_{3S,0R}$	R_1	D1	91.67%	3.16	123.79	-	-	-
IPP 2	Dense	S_4	$A_{3S,0R}$	R_1	D2	92.62%	3.17	128.38	92.60%	3.18	123.76
IPP 3	Dense	S_4	$A_{3S,0R}$	R_1	D3	85.12%	3.44	127.03	83.50%	4.30	113.96
IPP 4	Dense	S_4	$A_{3S,2R,lim}$	R_1	D3	86.15%	3.37	115.2	86.00%	4.55	100.17
IPP 5	Dense	S_6	$A_{3S,2R,lim}$	R_1	D3	88.09%	3.02	136.81	88.61%	3.85	113.00
IPP 6	Dense	S_6	$A_{3S,2R,lim}$	R_5	D3	91.13%	2.73	107.19	90.47%	3.21	103.20
IPP 7	Dense	S_6	$A_{2S,0R,T}$	R_1	D3	63.55%	5.00	51.93	-	-	-

As in the case of RPP, the results are presented below with results of the benchmark algorithms $\mathbf{SCP_\Psi^{D1,Ins}}$, $\mathbf{SCP_\Psi^{D2,Ins}}$ and $\mathbf{SCP_\Psi^{D3,Ins}}$, which give the analytical solution to the problem for the data sets D1, D2 and D3, respectively, when only the surface coverage is optimized and the travel distances required for this are neglected. It can be seen that $\mathbf{SCP_\Psi^{D1,Ins}}$, $\mathbf{SCP_\Psi^{D2,Ins}}$ and $\mathbf{SCP_\Psi^{D3,Ins}}$ achieve over 90% surface coverage after only two acquisitions ($t = 2$). This is expected, as two surface regions should be covered on average with two acquisitions. The comparatively high coverage after just one acquisition can be explained by the fact that there are often regions sampled next to each other that can theoretically be covered with one acquisition.

By comparing agent IPP 1, which was trained on data set D1, with its counterpart $\mathbf{SCP_\Psi^{D1,Ins}}$, it can be seen that although it also achieves over 90 % surface coverage of the regions, it requires three instead of two acquisitions on average. Analogous insights can also be gained for the agents IPP 2 and IPP 3 if they are compared with their counterparts $\mathbf{SCP_\Psi^{D2,Ins}}$ and $\mathbf{SCP_\Psi^{D3,Ins}}$. It is also noticeable that the performance of agent IPP 3 is worse for data set D3, in particular compared to IPP 1 and IPP 2, which have the same hyperparameters. On the one hand, this is already evident in the training, as a lower surface coverage is achieved with a greater number of necessary acquisitions. On the other hand, this is even more evident when testing the IPP 3 agent on starter motors that have not been trained on.

Here, it can be seen that, on average, a larger number of acquisitions is necessary for a similar surface coverage as during training. However, the integration of additional information into the agent modeling proved to be advantageous here. IPP 4 has more degrees of freedom (rotation of the acquisition system) due to the action modeling $A_{3S,2R,lim}$. Compared to IPP 3, this enables increased surface coverage during training with slightly fewer necessary acquisitions. IPP 5 additionally integrates the complete state vector S_6, which in turn enables an increase in surface coverage and a reduction in the number of necessary acquisitions compared to IPP 4 during training. This is presumably because more information is given to the agent through the state S_6, enabling IPP 5 to learn a better strategy than instead of using state S_4.

However, as with IPP 3, the fact remains that when testing the agents IPP 4 and IPP 5, a significantly higher number of acquisitions is carried out than in training, and that would theoretically be necessary (as can be seen from the results of the $\mathbf{SCP}$ benchmark algorithms from Table 6.28).

In the case of IPP, the variation of the reward signal also has an influence on the agent's behavior. Comparing IPP 5 and IPP 6, they differ only in the reward signal (R_1 for IPP 5 and R_5 for IPP 6). More specifically, only a constant punishment (negative reward) is integrated in R_5, which punishes acquisitions that do not include any surface points of a ROI to be acquired. Compared to IPP 5, it can be observed that with this definition of the reward signal, an increase in the average surface coverage can be achieved while at the same time reducing the required acquisitions and travel distances of the acquisition system. This can presumably be attributed to the fact that the IPP 6 agent learns faster and more efficiently through negative rewards to associate the passed state with actions that achieve a gain in the object surface to be covered.

Table 6.28: Tabular overview of the results of the $\mathbf{SCP}$ benchmark algorithms for the IPP problem evaluated on datasets D1, D2 and D3.

	Acquisition step t					
	Metric	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$
$\mathbf{SCP_{\Psi}^{D1,Ins}}$	Average Ψ_t	74.34%	94.38%	98.69%	99.22%	99.04%
	Average d_t	0	49.13	102.13	144.37	159.66
$\mathbf{SCP_{\Psi}^{D2,Ins}}$	Average Ψ_t	72.63%	92.96%	97.86%	98.95%	99.18%
	Average d_t	0	47.19	94.49	136.72	169.84
$\mathbf{SCP_{\Psi}^{D3,Ins}}$	Average Ψ_t	71.27%	88.27%	93.25%	95.26%	96.10%
	Average d_t	0	49.45	99.81	144.32	184.36

While the agents IPP 1-6, which were trained without integrating the robot model, achieve comparatively good results for the IPP, IPP 7 is trained with integration of the robot model. In addition to controlling the position of the acquisition system, in this case the IPP 7 agent must also control the relative rotation angle of the rotary table to acquire the ROI systematically. In this case, it can be seen that the agent only inadequately solves the IPP. The IPP 7 agent does not acquire the required surface coverage of $\Psi_T > 90\%$ of the ROI with 5 acquisitions and only achieves an average surface coverage of approx. 63%. In comparison to Agent IPP 3, only the state (from S_4 to S_6) and the additional degree of freedom due to the rotary table have changed in addition to the integration of the robot model. Since the integration of the state S_6 had a positive effect on the agent behavior in the remaining investigations in IPP and the agents in RPP have learned to output only reachable poses of the acquisition system as actions, it can be assumed that in the case of IPP, the degree of freedom of the rotary table is detrimental. It is reasonable to assume that the rotation of the rotary table also constantly changes the position and orientation of the ROI during the acquisition process and thus induces additional complexity (comparable to the acquisition of different objects, see investigations in 6.2.4.7), which could not be learned by the agent IPP 7.

Overall, it should be noted that, as in the case of RPP, when training an RL agent on several object variants, the problem complexity induced by the IPP results in comparatively suboptimal strategies being learned by the RL agents. This is not only the case for IPP 7 but also for the remaining agents IPP 1-6. In principle, the parameterization effort for these RL agents was significantly greater than for the agents of the NBV problem and the RPP. The RL agents for solving the IPP problem have always required a higher exploration (higher entropy coefficient in Table 6.16) rate and a lower learning rate to converge. It cannot be ruled out that this work has merely laid the foundation for further attempts at more optimal parameterization and solution of the IPP. In principle, however, there is a trade-off between the number of agent trainings to be carried out and the choice of hyperparameterization. For higher exploration rates and lower learning rates, RL agents need longer to converge with the same computing capacity, which is why the in-depth investigation of these hyperparameterizations was dispensed within this thesis.

6.2.5.1 Sim-to-Real: RPP results on the inspection station

All previous results of the trained RL agents have been obtained in the simulation. In the following, the quantitative results of RL agents trained in the simulation including the robot model are presented, which are evaluated (tested) on the real inspection station using real starter motors (Sim-to-Real setting). The following results show the results of the experiments of agents RPP 28 (dense reward signal R_1) and RPP 29 (sparse reward signal R_8) in comparison to two benchmarks (random and heuristic). Since it is assumed in the present experiments that the object geometry of the starter motors to be inspected is not available, the heuristic from section 6.2.1, which explicitly extracts bounding boxes from known geometry models and calculates the necessary VP from them, cannot be used. For the following experiments, a fixed VP was defined above the rotary table, and the acquisition system was aligned horizontally to the starter motors in the following six acquisitions, whereby the starter motor with the rotary table was rotated by 60° between the individual acquisitions. The random benchmark corresponds to the benchmark from section 6.2.1, whereby a random permissible position in space has been sampled for each acquisition, and the orientation of the acquisition system has been automatically calculated to face the starter motor.

Figure 6.30 shows the percentage of surface area acquired over the number of acquisitions performed for the agents and benchmarks. The respective agents and benchmarks were tested for the five different test starter motors in dataset D2. The underlying ground truth model (see data set D2 in section 6.2.1) was used to calculate the percentage surface coverage for each motor. The respective percentage of surface coverage in Figure 6.30 is the averaged surface coverage after the respective acquisitions for all five starter motors.

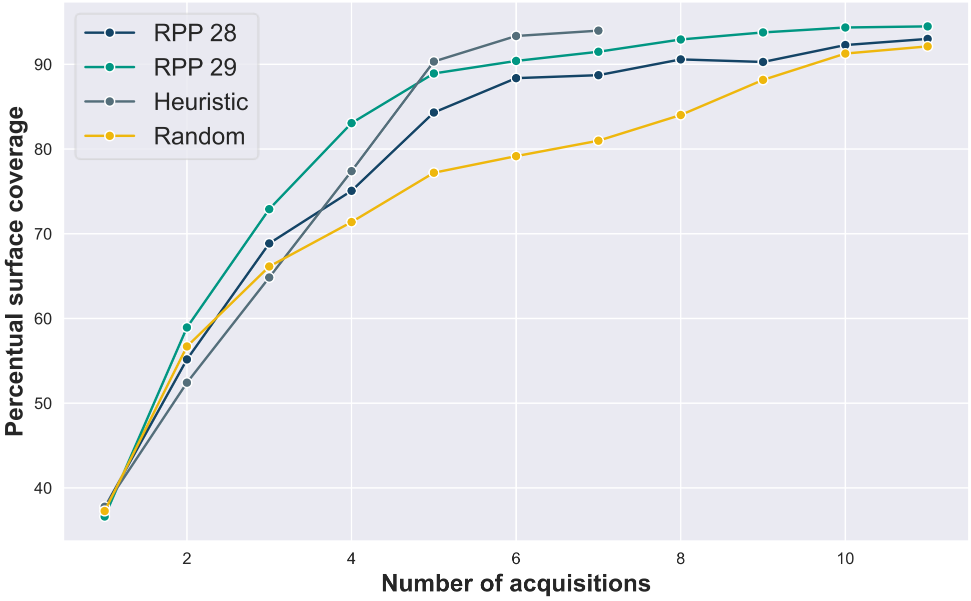


Figure 6.30: Percentage of surface coverage over the number of acquisitions for agents RPP 28 and RPP 29 as well as benchmarks Random and Heuristic.

The results show that both RPP 28 and RPP 29, trained in the virtual inspection station environment, can be used on the real inspection station. Both agents perform better than the random benchmark. After just five acquisitions, both agents have a percentage surface coverage between 80% and 90%. RPP 29, which was trained using a sparse reward signal R_8, shows, as in the simulations, a comparable surface coverage to its dense counterpart

RPP 28 with fewer acquisitions. Therefore, the simulation's findings can be transferred directly to the real system. It also becomes apparent that the agents RPP 28 and RPP 29 have a significantly greater percentage surface coverage than the benchmark heuristic in early acquisitions (acquisitions 1 to 4). This is reasonable because the heuristic rotates the starter motors with a certain number of angle increments (60°) and thus requires several rotations to enable the acquisition system to cover the entire surface. In contrast, the agents RPP 28 and RPP 29 have been trained to maximize the percentage surface coverage, whereby the learned strategies have probably forced a maximization of the surface coverage in each acquisition. This has probably resulted in problems in the later course of the acquisition process, which have already been discussed in section 2.2.1, since the agents have tended to solve the NBV problem.

Overall, these results show that the results from the simulation can also be confirmed in reality. The trained RL agents perform comparably to a heuristic, whereby the heuristic performs slightly better in the real application. However, these deficits can possibly be easily remedied in further work by fine-tuning the RL agents on the real use case and thus outperforming the heuristic.

7 Discussion and outlook

This thesis presents a methodological approach to using RL agents for autonomous view planning in remanufacturing. Based on the real use case of initial visual inspection, its challenges have been addressed as part of the solution approach of this thesis, and the solution approach has been implemented and evaluated. In the following, the insights gained are summarized and critically assessed in relation to the research deficits (see section 3.5) derived in this thesis (section 7.1). These findings provide the basis for deriving possible future research activities (section 7.2).

7.1 Critical appraisal

The approach of this thesis consists of a RL simulation framework. Differently modeled RL agents are trained to solve VPP relevant for remanufacturing. To this end, the inspection station used within the scope of this thesis has been modeled and virtualized to allow, e.g., for virtual analyses of the accessibility of the poses issued by the RL agent. Further, a method for semantic 3D reconstruction was developed using semantic mapping employing instance and semantic segmentation methods. This semantically reconstructed 3D model can then be used as input for pose planning by the RL agent in case components or defects, generally referred to as ROI, must be inspected in more detail during the acquisition process. Based on the digital representation of the inspection station, the RL simulation framework was developed. The semantic 3D reconstruction approach was used to generate initial training and test data for the RL agents to be trained. The entire RL simulation framework consists of three components. Firstly, the actual RL agent A_{Plan}, secondly S_{Sim} consisting of the virtual station representation including a developed acquisition simulation and thirdly, I_{Agent}, an interface for communication between A_{Plan} and S_{Sim} with the possibility to define differently modeled state, action and reward functions. This thesis presents a solution approach (see chapter 4) that addresses the research deficits (see section 3.5) identified in the current state of research. In the following section, the approach will be critically assessed.

Research deficit 1: *Missing evaluation of differently modeled RL Agents*
The development of the RL simulation framework has made it possible to freely design the configuration of the RL agents. Firstly, the open-source library stable-baselines 3 (see section 5.3.5) enables seamless integration of different RL algorithms. Furthermore, developing the interface I_{Agent} allowed maximum freedom to define the state, action, and reward definitions. Particular attention has been paid to a general implementation when defining the various configuration options. For example, the RL agent operates on a state function that is derived from acquired point clouds. This state definition is therefore highly universal, as many optical measurement systems available today can acquire point clouds. The same applies to the

modeling of action mapping, whereby, in this case, the RL agent directly manipulates degrees of freedom (DoF) of the system (pose of the acquisition system and rotation angle of the rotary table). The results have shown that a variation in the modeling of the RL agents has a decisive influence on their performance. For instance, it was shown that the appropriate integration of prior knowledge and the associated restriction of the RL agents' action space positively affects their performance. Furthermore, it has been shown that the additional integration of a large amount of state information does not provide any significant benefit. These insights can be traced back to the so-called curse of dimensions, according to which finding an optimal strategy for the RL agents becomes more difficult with increasing state and action space, even though additional potential information has been encoded in the state and the RL agent has more degrees of freedom. Overall, the modeling results of the present work provide an ideal starting point for further research. Different modeling variants have been presented and interpreted, whereby successful and less successful modeling alternatives have been compared quantitatively concerning defined performance indicators.

Research deficit 2: *Dependence on available object geometries at system runtime*
Existing RL approaches for solving the VPP are model-dependent. This implies that at execution time, they require knowledge of the geometry model or knowledge of the product at hand or its variant to plan on this geometry model (derivation of the acquisition state based on the geometry model, e.g., Potapova & Artemov et al. (2020)) or the selection of a RL agent specifically trained for the product at hand to solve the problem (e.g., Landgraf & Meese et al. (2021)). However, this does not imply that geometry models are not required to train the RL agents. Therefore, two alternative state representations, model-based and model-free, have been defined and evaluated in this thesis. The model-free version is primarily required and validated for the solution of RPP as part of the general inspection of remanufacturing. The model-based version must be chosen and evaluated for the individual inspection (IPP), as the reconstructed semantic 3D model of the inspection object is available when using the methodology proposed and its procedure for inspection. Both state definitions have shown comparable results, indicating the ability of the RL agents to interfere knowledge about not yet acquired surface points of the inspection from inputting information about the surface points already acquired in the previous acquisitions as state when using the model-free variant. This has successfully been demonstrated in the sim-to-real, where agents trained in the simulation have successfully been transferred to the real inspection station and solved the RPP for starter motors with an unknown object geometry model.

Research deficit 3: *Missing investigation of varying planning tasks*
Another significant deficit in the state of research is the lack of investigation of RL agent behavior for varying planning tasks. On the one hand, this concerns the training of RL agents

to solve RPP for a large number of different objects, but also, for example, the solving of IPP, in which there is a variable positioning of the regions to be acquired on the inspection object. Both problems have been investigated in this thesis. Regarding the generalization to different inspection objects for the RPP, the investigated RL agents showed learning behavior but only found local optima, shown by the inferior performance in comparison to the analytically found (optimal) solutions. The RL agents learned a strategy that successively positions the acquisition system so that an acceptable surface coverage is achieved on average for all inspection objects, regardless of the inspection object. The more desirable behavior of the RL agents, an inspection object-specific adaptation of the positioning strategy of the detection system by the RL agents, could not be learned. Similarly, the use case of IPP with varying ROI to be inspected with constant variants of the inspection objects (starter motors) has been investigated. Except for the RL agents integrating the robot model, the RL agents trained here were able to solve this problem and, depending on the positioning of the ROI on the inspection object, select acquisition poses that also acquire the ROI. However, the comparison with the analytical SCP benchmark algorithms has shown that for both the case of the RPP and the case of the IPP, the trained RL agents require on average 1-2 more acquisitions to fulfill the respective inspection goals than necessary according to the optimal solution provided by the SCP algorithms.

7.2 Outlook

This thesis extends the state of research in the field of view planning using RL by applying it to the use case of initial visual inspection in remanufacturing. Various problems of VPP have been considered, solved, and analyzed with different modeled RL agents. The findings of this work serve as a foundation for further research.

Fine-tuning of pre-learned strategies: In the present work, all RL agents have been trained from scratch, i.e., with a randomly initialized strategy. The research of this work can be built upon if already pre-trained strategies are merely fine-tuned by RL. One example is imitation learning, in which an agent has to learn the behavior of an expert by demonstrating its problem solution (Hussein & Gaber et al. 2017). The expert can be a human, but also an analytical solution approach (e.g., the solution of a VPP using SCP algorithm), whose solution behavior is to be imitated. The formalism of imitation learning is directly based on that of RL. Thus, the knowledge gained in this work makes the RL simulation framework developed in this work suitable for further work in this direction.

Model-based RL: In comparison to the model-free RL approach of this work, in which an RL agent learns an ideally optimal strategy by intelligently exploring the environment, model-based RL integrates a planning component. Model-based RL combines a model of the environment (explicitly not the geometry model of the inspection object) with a learning process to approximate the optimal strategy. (Moerland & Broekens et al. 2023). Usually, in addition to the optimal strategy, this environment model is also learned during the learning process, whereby various possibilities exist here. One possibility is, for example, the prediction of the successor state given a current state and a theoretically executed action. If such a model is known, several actions can be evaluated in one state and the subsequent state they lead to. An evaluation of the possible subsequent states with regard to their quality allows the best possible evaluated action to be selected and carried out.

In addition to these fundamentally methodologically motivated research directions, further steps can be taken. The results of this work are mainly based on point cloud-based state definitions. However, in the past, RL agents have also been evaluated very successfully in the domain with image-based states (cf. Mnih & Kavukcuoglu et al. 2013). The findings from applying RL agents can be transferred to the problem of VPP. Extending the acquisition simulation to generate virtual image data is easily possible. This makes it possible to train RL agents with multimodal data (image data, depth data, point clouds and their processed forms, e.g. voxel grids).

Increasing the degree of realism also plays an important role in future research efforts. The point clouds acquired in this work were determined solely by visibility and distance to the simulated acquisition system without considering, for example, the reflection effects of surfaces. However, reflections may make the acquisition of individual surface areas of an inspection object invisible, even though they lie within the working range of the acquisition system. Considering possible boundary conditions when acquiring individual ROI when viewing the IPP also falls into this context. A feature of an inspection object may have to be acquired at a predefined distance or at a predefined angle of the acquisition system to the feature. The integration of such boundary conditions poses a significant difficulty, especially for model-free RL, since these boundary conditions must be modeled via a quantitative value (the reward signal), which in turn induces an enormous difficulty for the learning behavior of the RL agents.

This work has also shown that integrating multiple optimization variables into a reward function is possible. Optimization by the RL agents in the use case of RPP allowed two variables to be optimized simultaneously, whereby the evaluated agents no longer showed any significant improvement with regard to this variable when a third optimization variable was integrated.

8 Conclusion

Due to uncertainty regarding the variant and condition of returned used products in remanufacturing, the industry currently performs manual visual inspections of these products. This inspection process is complex, and returned used products must be inspected as a whole, but individual components may also require dedicated inspection. A human inspector implicitly solves the so-called View Planning Problem (VPP), which involves choosing acquisition poses to fully inspect a product or specific features of the product.

The research conducted in this thesis underscores that in the absence of a three-dimensional model of the object to be inspected, intelligent methods like reinforcement learning (RL) can be harnessed to devise a strategy for solving the VPP in remanufacturing. However, the current RL solutions have seen limited application or evaluation, particularly when a 3D model of the object is not available at runtime. Moreover, the targeted inspection of specific features of the object with RL remains unexplored.

In this thesis, a method has been developed to enable RL agents to solve the View Planning Problem (VPP) in remanufacturing. To approach this problem, an inspection station was first modeled and virtually represented, serving as the basis for subsequent evaluations. This virtually modeled and represented inspection station was integrated into a simulation framework developed in this thesis for training variously modeled RL agents. The framework facilitates the training of RL agents for the *Next Best view Planning Problem* (NBV), the *Reconstruction Planning Problem* (RPP) and the *Inspection Planning Problem* (IPP).

Given that it is often assumed in remanufacturing that there is no 3D model of the inspection object available, an approach was developed with the aid of segmentation algorithms. This approach enables constructing a semantic product model, which includes 3D information of the inspection object and the locations of different components on the product model. This semantic product model can be generated during system runtime with the results from the overall inspection of the object's surface (RPP) and subsequently serve as a basis of information for RL agents inspecting individual features of the inspection object (IPP).

The consistent description of different VPP and the standardized modular framework introduced in this work enable the easy integration of different solution approaches for VPP (heuristics, analytical solutions, and learning approaches). By analyzing the derived planning problems with data sets of increasing complexity under consideration of the different modeling variants of RL agents developed in the thesis, influences of existing RL approaches on the agent performance have been systematically investigated and discussed.

The results have shown that the modeled RL agents can solve all derived planning problems. It is worth highlighting that this is also possible in the model-free case, where an agent has not been specifically trained on the geometry model at hand, but only with a large number of different geometry models of the same product group (in the case of this work, starter motors). This offers significant advantages in that generating a view plan is possible, even without an existing geometry model, by sequential output of VP by an RL agent. It is also possible to optimize various target variables of VPP (maximizing the surface coverage of the inspection object and minimizing the required travel path lengths). Nevertheless, although the RL agents have shown to perform on par with a benchmark heuristic, it has also been shown that the trained RL agents perform worse in terms of evaluation metrics (especially the number of acquisitions required to achieve a given surface coverage) compared to analytical solutions that calculate a near-optimal view plan. This was specifically the case when the RL agent was assigned an increasing number of degrees of freedom (e.g., in addition to influencing the pose of the acquisition system, also influencing the rotation of the rotation table). However, the modular simulation framework derived in this work and the uniform, consistent, and systematic approach to modeling the RL agents provide an essential basis for using more novel algorithms of the RL to obtain agent behavior closer to the global optimum for a real-time capable solution of VPP. This modularity, in particular, also made it possible to successfully validate and verify the results of this thesis directly on the real inspection system. As in the simulation, RL agents trained in the simulation thereby also show comparable performance to a benchmark heuristic when tested in the real application case.

Overall, the present work thus provides a significant contribution to the selection, modeling, and interpretation of learning RL methods for solving VPP. Remanufacturing is selected as a use case, where prior knowledge regarding the planning task to be solved is not or only partially available, and thus autonomous, adaptive system behavior is required. The individual components of the solution approach developed in this thesis for semantic 3D product model generation with the help of agents for solving the RPP for complete surface coverage of the inspection object and downstream solution of the IPP for individual inspection of ROI have been successfully evaluated. In particular, the uniform and standardized modeling of the agent framework allows the promising results to be systematically improved in further work.

Bibliography

References according to the scheme (A_<name><year>) refer to student work at the wbk Institute of Production Science of the Karlsruhe Institute of Technology (KIT), which was guided by the author of the dissertation.

A_Gäbele 2022
Gäbele, J. (2022), „Development and Implementation of a Reinforcement-Learning Approach for Automated Product Acquisition“. Bachelors Thesis. Karlsruhe: Karlsruhe Institute of Technology (KIT), wbk Institute of Production Science.

A_Hollinger 2022
Hollinger, V. (2022), „Development and implementation of a deep-learning method for semantic segmentation of electric engines based on RGB-D data“. Masters Thesis. Karlsruhe: Karlsruhe Institute of Technology (KIT), wbk Institute of Production Science.

A_Koch 2022
Koch, D. (2022), „Implementation and Evaluation of Supervised and Reinforcement Learning Approaches for the Selective Optical Inspection of Product Features“. Masters Thesis. Karlsruhe: Karlsruhe Institute of Technology (KIT), wbk Institute of Production Science.

A_Scheiger 2022
Scheiger, V. (2022), „Automated Product Inspection and Computer Vision-based Camera Position Replanning“. Masters Thesis. Karlsruhe: Karlsruhe Institute of Technology (KIT), wbk Institute of Production Science.

A_Schmid 2021
Schmid, J. (2021), „Development and implementation of an online capable, model free, machine learning based approach for automated product capturing“. Masters Thesis. Karlsruhe: Karlsruhe Institute of Technology (KIT), wbk Institute of Production Science.

A_Schnaberich 2022
Schnaberich, M. (2022), „Development and commissioning of an autonomously operating diagnostic station for optical inline measurement in a remanufacturing use-case“. Masters Thesis. Karlsruhe: Karlsruhe Institute of Technology (KIT), wbk Institute of Production Science.

A_Yingqi 2022
Yingqi, Q. (2022), „Implementation of an approach for three-dimensional product modeling and projection-based corrosion detection“. Masters Thesis. Karlsruhe: Karlsruhe Institute of Technology (KIT), wbk Institute of Production Science.

Achlioptas & Diamanti et al. 2018

Achlioptas, P.; Diamanti, O.; Mitliagkas, I. & Guibas, L. (2018), „Learning representations and generative models for 3d point clouds“. *Proceedings of Machine Learning Research (PMLR)*. 35th International Conference on Machine Learning (ICML) (Stockholm, Sweden, July 10–15, 2018). Ed. by J. Dy & A. Krause. Vol. 80, pp. 40–49.

Ahmed & Saint et al. 2018

Ahmed, E.; Saint, A.; Shabayek, A. E. R.; Cherenkova, K.; Das, R.; Gusev, G.; Aouada, D. & Ottersten, B. E. (2018), „Deep Learning Advances on Different 3D Data Representations: A Survey“, arXiv preprint: `1808.01462`. URL: `http://arxiv.org/abs/1808.01462`.

Andrew-Munot & Ibrahim et al. 2015

Andrew-Munot, M.; Ibrahim, R. N. & Junaidi, E. (2015), „An overview of used-products remanufacturing“, *Mechanical Engineering Research* 5.1, pp. 12–23. DOI: `10.5539/MER.V5N1P12`.

ANSI RIC001.2-2021 2021

ANSI RIC001.2-2021 (2021), *Specifications For The Process Of Remanufacturing*. American National Standards Institute (ANSI), Washington.

Aras & Boyaci et al. 2004

Aras, N.; Boyaci, T. & Verter, V. (2004), „The effect of categorizing returned products in remanufacturing“, *IIE transactions* 36.4, pp. 319–331. DOI: `10.1080/07408170490279561`.

Ashutosh & Kumar et al. 2022

Ashutosh, K.; Kumar, S. & Chaudhuri, S. (2022), „3D-NVS: A 3d supervision approach for next view selection“. Proceedings of the 26th International Conference on Pattern Recognition (ICPR) (Montreal, Canada, Aug. 21–25, 2022). Ed. by M. Jenkin; H. I. Christensen & C.-L. Liu. IEEE, pp. 3929–3936. DOI: `10.1109/ICPR56361.2022.9956377`.

Banta & Zhien et al. 1995

Banta, J. E.; Zhien, Y.; Wang, X. Z.; Zhang, G.; Smith, M. T. & Abidi, M. A. (1995), „Best-next-view algorithm for three-dimensional scene reconstruction using range images“, *Intelligent Robots and Computer Vision XIV: Algorithms, Techniques, Active Vision, and Materials Handling* 2588, pp. 418–429. DOI: `10.1117/12.222691`.

Beer & Fisk et al. 2014

Beer, J. M.; Fisk, A. D. & Rogers, W. A. (2014), „Toward a framework for levels of robot autonomy in human-robot interaction“, *Journal of human-robot interaction* 3.2, pp. 74–99. DOI: `10.5898/JHRI.3.2.Beer`.

Berner & Brockman et al. 2019

Berner, C.; Brockman, G.; Chan, B.; Cheung, V.; Dębiak, P.; Dennison, C.; Farhi, D.; Fischer, Q.; Hashme, S.; Hesse, C. et al. (2019), „Dota 2 with large scale deep reinforcement learning“, arXiv preprint: `1912.06680`. URL: `https://arxiv.org/abs/1912.06680`.

Bishop & Nasrabadi 2006

Bishop, C. M. & Nasrabadi, N. M. (2006), *Pattern recognition and machine learning*. Vol. 4. 4. Springer, New York. DOI: 10.1007/978-0-387-45528-0.

Bras & Hammond et al. 1996

Bras, B.; Hammond, R. & Woodruff, G. W. (1996), „Towards Design for Remanufacturing - Metrics for Assessing Remanufacturability“. Proceedings of the 1st International Workshop on Reuse (Eindhoven, The Netherlands, Nov. 11–13, 1996). Ed. by S. D. Flapper & A. J. de Ron, pp. 5–22.

Bronstein & Bruna et al. 2017

Bronstein, M. M.; Bruna, J.; LeCun, Y.; Szlam, A. & Vandergheynst, P. (2017), „Geometric Deep Learning: Going beyond Euclidean data“, *IEEE Signal Processing Magazine* 34.4, pp. 18–42. DOI: 10.1109/MSP.2017.2693418.

BS8887-2:2009 2009

BS8887-2:2009 (2009), *Design for manufacture, assembly, disassembly and end-of-life processing (MADE) - Terms and definitions*. British Standards Institution (BSI), Britain.

Chang & Funkhouser et al. 2015

Chang, A. X.; Funkhouser, T.; Guibas, L.; Hanrahan, P.; Huang, Q.; Li, Z.; Savarese, S.; Savva, M.; Song, S.; Su, H.; Xiao, J.; Yi, L. & Yu, F. (2015), *ShapeNet: An Information-Rich 3D Model Repository - Stanford University - Princeton University - Toyota Technological Institute at Chicago*.

Chen & Li et al. 2011

Chen, S.; Li, Y. & Kwok, N. M. (2011), „Active vision in robotic systems: A survey of recent developments“, *The International Journal of Robotics Research* 30.11, pp. 1343–1377. DOI: 10.1177/0278364911410755.

Chu & Madhavan et al. 2016

Chu, B.; Madhavan, V.; Beijbom, O.; Hoffman, J. & Darrell, T. (2016), „Best practices for fine-tuning visual classifiers to new domains“. *Proceedings of Computer Vision–ECCV 2016 Workshops Part III*. ECCV 2016 Workshops (Amsterdam, The Netherlands, Oct. 8–16, 2016). Ed. by G. Hua & H. Jégou. Springer, Cham, pp. 435–442. DOI: 10.1007/978-3-319-49409-8_34.

Coleman & Sucan et al. 2014

Coleman, D.; Sucan, I.; Chitta, S. & Correll, N. (2014), „Reducing the barrier to entry of complex robotic software: a moveit! case study“, arXiv preprint: 1404.3785. URL: https://arxiv.org/abs/1404.3785.

Colledani & Battaïa 2016

Colledani, M. & Battaïa, O. (2016), „A decision support system to manage the quality of End-of-Life products in disassembly systems“, *CIRP Annals* 65.1, pp. 41–44. DOI: https://doi.org/10.1016/j.cirp.2016.04.121.

Connolly 1985

Connolly, C. (1985), „The determination of next best views“. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) (St. Louis, Missouri, USA, Mar. 25–25, 1985). Ed. by K. S. FU & M. A. Wesley, pp. 432–435. DOI: 10.1109/ROBOT.1985.1087372.

CoremanNet 2022

CoremanNet (2022), *Bosch Core acceptance criteria for starter motors*. Accessed: 24.06.2024. URL: https://www.coremannet.com/assets/docs/return-criteria/new-2019/Starter.pdf.

Corvellec & Böhm et al. 2020

Corvellec, H.; Böhm, S.; Stowell, A. & Valenzuela, F. (2020), „Introduction to the special issue on the contested realities of the circular economy“, *Culture and Organization* 26.2, pp. 97–102. DOI: 10.1080/14759551.2020.1717733.

Dawson-Haggerty et al. 2019

Dawson-Haggerty et al. (2019), *trimesh*. Version 3.2.0. Accessed: 24.06.2024. URL: https://trimsh.org/.

Deinzer & Denzler et al. 2003

Deinzer, F.; Denzler, J. & Niemann, H. (2003), „Viewpoint selection–planning optimal sequences of views for object recognition“. 10th International Conference on Computer Analysis of Images and Patterns (CAIP) (Groningen, The Netherlands, Aug. 25–27, 2003). Ed. by N. Petkov & M. A. Westenberg. Springer, Berlin Heidelberg, pp. 65–73.

Devrim Kaba & Gokhan Uzunbas et al. 2017

Devrim Kaba, M.; Gokhan Uzunbas, M. & Nam Lim, S. (2017), „A reinforcement learning approach to the view planning problem“. *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Honolulu, HI, USA, July 21–26, 2017). Ed. by Y. Liu; J. M. Rehg; C. J. Taylor & Y. Wu, pp. 6933–6941. DOI: 10.1109/CVPR.2017.541.

DIN 31051:2019-06 2019

DIN 31051:2019-06 (2019), *Fundamentals of maintenance*. Deutsches Institut für Normung e.V. (DIN). Beuth Verlag GmbH, Berlin. DOI: https://dx.doi.org/10.31030/3048531.

DIN EN 13306:2018-02 2018

DIN EN 13306:2018-02 (2018), *Maintenance - Maintenance terminology; Trilingual version*. Deutsches Institut für Normung e.V. (DIN). Beuth Verlag GmbH, Berlin. DOI: https://dx.doi.org/10.31030/2641990.

DIN SPEC 91472:2023-06 2023

DIN SPEC 91472:2023-06 (2023), *Remanufacturing (Reman) - Quality classification for circular processes*. Deutsches Institut für Normung e.V. (DIN). Beuth Verlag GmbH, Berlin. DOI: https://dx.doi.org/10.31030/3434252.

Domingos 2012

Domingos, P. (2012), „A few useful things to know about machine learning“, *Communications of the ACM* 55.10, pp. 78–87. DOI: 10.1145/2347736.2347755.

Elharrouss & Akbari et al. 2022

Elharrouss, O.; Akbari, Y.; Almaadeed, N. & Al-Maadeed, S. (2022), „Backbones-review: Feature extraction networks for deep learning and deep reinforcement learning approaches“, arXiv preprint: 2206.08016. URL: https://arxiv.org/abs/2206.08016.

Emek Soylu & Guzel et al. 2023

Emek Soylu, B.; Guzel, M. S.; Bostanci, G. E.; Ekinci, F.; Asuroglu, T. & Acici, K. (2023), „Deep-Learning-Based Approaches for Semantic Segmentation of Natural Scene Images: A Review“, *Electronics* 12.12. DOI: 10.3390/electronics12122730.

Endsley 2017

Endsley, M. R. (2017), „From here to autonomy: lessons learned from human–automation research“, *Human factors* 59.1, pp. 5–27. DOI: 10.1177/0018720816681350.

Errington & Childe 2013

Errington, M. & Childe, S. J. (2013), „A business process model of inspection in remanufacturing“, *Journal of Remanufacturing* 3, pp. 1–22. DOI: 10.1186/2210-4690-3-7.

Ertel 2018

Ertel, W. (2018), *Introduction to artificial intelligence*. Springer, London. DOI: 10.1007/978-3-319-58487-4.

Eschmann 2021

Eschmann, J. (2021), „Reward function design in reinforcement learning“. *Reinforcement learning algorithms: Analysis and Applications*. Ed. by B. Belousov; H. Abdulsamad; P. Klink; S. Parisi & J. Peters. Springer, Cham, pp. 25–33. DOI: 10.1007/978-3-030-41188-6_3.

Fahim & Amin et al. 2021

Fahim, G.; Amin, K. & Zarif, S. (2021), „Single-View 3D reconstruction: A Survey of deep learning methods“, *Computers and Graphics* 94, pp. 164–190. DOI: https://doi.org/10.1016/j.cag.2020.12.004.

Ferrer 2001

Ferrer, G. (2001), „On the widget remanufacturing operation“, *European Journal of Operational Research* 135.2, pp. 373–393. DOI: https://doi.org/10.1016/S0377-2217(00)00318-0.

Füvesi & Kovács et al. 2010

Füvesi, V.; Kovács, E. & Blága, C. (2010), „Measurement and identification of a starter motor system". Proceedings of the 2nd Conference on Recent Achievements in Mechatronics, Automation, Computer Sciences and Robotics (MACRo2010) (Tirgu Mureş , Rumänien, May 14–15, 2010). Ed. by D. László; S. Gyula; M. László Ferenc & K. András, pp. 129–134.

García & Luengo et al. 2015

García, S.; Luengo, J. & Herrera, F. (2015), *Data preprocessing in data mining*. Springer, Cham. DOI: `10.1007/978-3-319-10247-4`.

Garcia-Garcia & Orts-Escolano et al. 2018

Garcia-Garcia, A.; Orts-Escolano, S.; Oprea, S.; Villena-Martinez, V.; Martinez-Gonzalez, P. & Garcia-Rodriguez, J. (2018), „A survey on deep learning techniques for image and video semantic segmentation", *Applied Soft Computing* 70, pp. 41–65. DOI: `https://doi.org/10.1016/j.asoc.2018.05.018`.

Gawlikowski & Tassi et al. 2023

Gawlikowski, J.; Tassi, C. R. N.; Ali, M.; Lee, J.; Humt, M.; Feng, J.; Kruspe, A.; Triebel, R.; Jung, P.; Roscher, R. et al. (2023), „A survey of uncertainty in deep neural networks", *Artificial Intelligence Review* 56, pp. 1–77. DOI: `10.1007/s10462-023-10562-9`.

Germani & Mandorli et al. 2010

Germani, M.; Mandorli, F.; Mengoni, M. & Raffaeli, R. (2010), „CAD-based environment to bridge the gap between product design and tolerance control", *Precision engineering* 34.1, pp. 7–15. DOI: `10.1016/j.precisioneng.2008.10.002`.

Germani & Mengoni et al. 2009

Germani, M.; Mengoni, M. & Raffaeli, R. (2009), „Automation of 3D view acquisition for geometric tolerances verification". 12th IEEE International Conference on Computer Vision (ICCV) Workshops (Kyoto, Japan, Sept. 27–Oct. 4, 2009). Ed. by R. Cipolla; M. Hebert; X. Tang & N. Yokoya. IEEE, pp. 1710–1717. DOI: `10.1109/ICCVW.2009.5457489`.

Gezawa & Zhang et al. 2020

Gezawa, A. S.; Zhang, Y.; Wang, Q. & Yunqi, L. (2020), „A review on deep learning approaches for 3d data representations in retrieval and classifications", *IEEE access* 8, pp. 57566–57593. DOI: `10.1109/ACCESS.2020.2982196`.

Ghahramani 2003

Ghahramani, Z. (2003), „Unsupervised learning". *Advanced Lectures on Machine Learning*. Ed. by O. Bousquet; U. von Luxburg & G. Rätsch. Springer, Berlin Heidelberg, pp. 72–112. DOI: `10.1007/978-3-540-28650-9_5`.

Gibbons & Pierce et al. 2018

Gibbons, T.; Pierce, G.; Worden, K. & Antoniadou, I. (2018), „A Gaussian mixture model for automated corrosion detection in remanufacturing“, pp. 63–68. DOI: 10.3233/978-1-61499-902-7-63.

Globisch & Thäter et al. 2019

Globisch, S.; Thäter, S. & Döpper, F. (2019), „Optical inspection for the characterization and classification of component surfaces in the field of remanufacturing“. Proceedings of the 8th Congress of the German Academic Association for Production Technology (WGP) (Aachen, Deutschland, Nov. 19–19, 2018). Ed. by R. Schmitt & G. Schuh. Springer, Cham, pp. 44–51. DOI: doi.org/10.1007/978-3-030-03451-1_5.

Goodfellow & Bengio et al. 2016

Goodfellow, I.; Bengio, Y. & Courville, A. (2016), *Deep learning*. MIT press, Cambridge, MA.

Gu & Wang et al. 2018

Gu, J.; Wang, Z.; Kuen, J.; Ma, L.; Shahroudy, A.; Shuai, B.; Liu, T.; Wang, X.; Wang, G.; Cai, J. & Chen, T. (2018), „Recent advances in convolutional neural networks“, *Pattern Recognition* 77, pp. 354–377. DOI: https://doi.org/10.1016/j.patcog.2017.10.013.

Gungor & Gupta 1998

Gungor, A. & Gupta, S. M. (1998), „Disassembly sequence planning for products with defective parts in product recovery“, *Computers and Industrial Engineering* 35.1, pp. 161–164. DOI: https://doi.org/10.1016/S0360-8352(98)00047-3.

Haarnoja & Zhou et al. 2018

Haarnoja, T.; Zhou, A.; Abbeel, P. & Levine, S. (2018), „Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor“. *Proceedings of Machine Learning Research (PMLR)*. 35th International Conference on Machine Learning (ICML) (Stockholm, Sweden, July 10–15, 2018). Ed. by J. Dy & A. Krause. Vol. 80.

Hafiz & Bhat 2020

Hafiz, A. M. & Bhat, G. M. (2020), „A survey on instance segmentation: state of the art“, *International journal of multimedia information retrieval* 9.3, pp. 171–189. DOI: 10.1007/s13735-020-00195-x.

Hammond & Amezquita et al. 1998

Hammond, R.; Amezquita, T. & Bras, B. (1998), „Issues in the automotive parts remanufacturing industry: a discussion of results from surveys performed among remanufacturers“, *Engineering Design and Automation* 4, pp. 27–46.

Hastie & Tibshirani et al. 2009

Hastie, T.; Tibshirani, R.; Friedman, J. H. & Friedman, J. H. (2009), *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. Springer, New York. DOI: 10.1111/J.1541-0420.2010.01516.X.

Haynsworth & Lyons 1987

Haynsworth, H. C. & Lyons, R. T. (1987), „Remanufacturing by design, the missing link“, *Production and inventory management* 28.2, pp. 24–29.

Hazirbas & Ma et al. 2017

Hazirbas, C.; Ma, L.; Domokos, C. & Cremers, D. (2017), „Fusenet: Incorporating depth into semantic segmentation via fusion-based cnn architecture“. Proceedings of the 13th Asian Conference on Computer Vision (ACCV) Part I (Taipei, Taiwan, Nov. 20–24, 2016). Ed. by S.-H. Lai; V. Lepetit; K. Nishino & Y. Sato. Springer, Cham, pp. 213–228. DOI: 10.1007/978-3-319-54181-5_14.

He & Gkioxari et al. 2017

He, K.; Gkioxari, G.; Dollár, P. & Girshick, R. (2017), „Mask R-CNN“. *Proceedings of the IEEE international conference on computer vision*. Proceedings of the 16th IEEE International Conference on Computer Vision (ICCV) (Venedig, Italien, Oct. 22–29, 2017). Ed. by R. Cucchiara; Y. Matsushita; N. Sebe & S. Soatto, pp. 2961–2969. DOI: 10.1109/ICCV.2017.322.

Hjorth & Chrysostomou 2022

Hjorth, S. & Chrysostomou, D. (2022), „Human–robot collaboration in industrial environments: A literature review on non-destructive disassembly“, *Robotics and Computer-Integrated Manufacturing* 73, p. 102208. DOI: https://doi.org/10.1016/j.rcim.2021.102208.

Hodson 2022

Hodson, T. O. (2022), „Root mean square error (RMSE) or mean absolute error (MAE): When to use them or not“, *Geoscientific Model Development Discussions* 2022, pp. 1–10. DOI: 10.5194/gmd-15-5481-2022.

Hussein & Gaber et al. 2017

Hussein, A.; Gaber, M. M.; Elyan, E. & Jayne, C. (2017), „Imitation learning: A survey of learning methods“, *ACM Computing Surveys (CSUR)* 50.2, pp. 1–35. DOI: 10.1145/3054912.

Ijomah & Bennett et al. 1999

Ijomah, W. L.; Bennett, J. P. & Pearce, J. (1999), „Remanufacturing: evidence of environmentally conscious business practice in the UK“. Proceedings 1st International Conference on Environmentally conscious Design and Inverse Manufacturing (Tokyo, Japan, Feb. 1–3, 1999). Ed. by H. Yoshikawa & R. Yamamoto. IEEE, pp. 192–196. DOI: 10.1109/ECODIM.1999.747607.

Ijomah & Childe et al. 2004

Ijomah, W. L.; Childe, S. & McMahon, C. (2004), „Remanufacturing: A Key Strategy for Sustainable Development“. Proceedings of the 3rd International Conference on Design and

Manufacture for Sustainable Development (Loughborough, UK, Sept. 1–2, 2004). Ed. by T. Bhamra & B. Hon, pp. 51–64.

Ioannidou & Chatzilari et al. 2017

Ioannidou, A.; Chatzilari, E.; Nikolopoulos, S. & Kompatsiaris, I. (2017), „Deep learning advances in computer vision with 3d data: A survey“, *ACM computing surveys (CSUR)* 50.2, pp. 1–38. DOI: 10.1145/3042064.

ISO/IEC Guide 98-3 2008

ISO/IEC Guide 98-3 (2008), *Evaluation of measurement data — Guide to the expression of uncertainty in measurement*. Joint Committee for Guides in Metrology (JCGM), France.

ISO/IEC Guide 98-3-1 2008

ISO/IEC Guide 98-3-1 (2008), *Evaluation of measurement data — Supplement 1 to the "Guide to the expression of uncertainty in measurement" — Propagation of distributions using a Monte Carlo method*. Joint Committee for Guides in Metrology (JCGM), France.

Jadon 2020

Jadon, S. (2020), „A survey of loss functions for semantic segmentation“. Proceedings of the 2020 IEEE conference on computational intelligence in bioinformatics and computational biology (CIBCB) (Via del Mar, Chile, Oct. 27–29, 2020). Ed. by G. A. Ruz. IEEE, pp. 1–7. DOI: 10.1109/CIBCB48159.2020.9277638.

Jing & Goh et al. 2018

Jing, W.; Goh, C. F.; Rajaraman, M.; Gao, F.; Park, S.; Liu, Y. & Shimada, K. (2018), „A computational framework for automatic online path generation of robotic inspection tasks via coverage planning and reinforcement learning“, *IEEE Access* 6, pp. 54854–54864. DOI: 10.1109/ACCESS.2018.2872693.

Jing & Polden et al. 2017

Jing, W.; Polden, J.; Tao, P. Y.; Goh, C. F.; Lin, W. & Shimada, K. (2017), „Model-based coverage motion planning for industrial 3D shape inspection applications“. Proceedings of the 13th IEEE Conference on Automation Science and Engineering (CASE) (Xi'an, China, Aug. 20–23, 2017). Ed. by X. Guan; Q. Zhao; Q.-S. Jia & M. Dotoli. IEEE, pp. 1293–1300. DOI: 10.1109/COASE.2017.8256278.

Kaiser & Gäbele et al. 2024

Kaiser, J.-P.; Gäbele, J.; Koch, D.; Schmid, J.; Stamer, F. & Lanza, G. (2024), „Adaptive acquisition planning for visual inspection in remanufacturing using reinforcement learning“, *Journal of Intelligent Manufacturing*, pp. 1–27. DOI: https://doi.org/10.1007/s10845-024-02478-0.

Kaiser & Koch et al. 2024

Kaiser, J.-P.; Koch, D.; Gäbele, J.; May, M. C. & Lanza, G. (2024), „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. DOI: https://doi.org/10.1016/j.cirpj.2024.07.006.

Kara & Hauschild et al. 2022

Kara, S.; Hauschild, M.; Sutherland, J. & McAloone, T. (2022), „Closed-loop systems to circular economy: A pathway to environmental sustainability?“, *CIRP Annals* 71.2, pp. 505–528. DOI: 10.1016/j.cirp.2022.05.008.

Kazhdan & Bolitho et al. 2006

Kazhdan, M.; Bolitho, M. & Hoppe, H. (2006), „Poisson surface reconstruction“. Proceedings of the 4th Eurographics symposium on Geometry processing (Cagliari, Italien, June 26–28, 2006). Ed. by A. Sheffer & K. Polthier. Vol. 7. 4. DOI: 10.2312/SGP/SGP06/061-070.

Kazhdan & Hoppe 2013

Kazhdan, M. & Hoppe, H. (2013), „Screened poisson surface reconstruction“, *ACM Transactions on Graphics (ToG)* 32.3, pp. 1–13. DOI: 10.1145/2487228.2487237.

Kerin & Pham 2020

Kerin, M. & Pham, D. T. (2020), „Smart remanufacturing: a review and research framework“, *Journal of Manufacturing Technology Management* 31.6, pp. 1205–1235. DOI: 10.1108/jmtm-06-2019-0205.

Khan & Mineo et al. 2021

Khan, A.; Mineo, C.; Dobie, G.; Macleod, C. & Pierce, G. (2021), „Vision guided robotic inspection for parts in manufacturing and remanufacturing industry“, *Journal of Remanufacturing* 11.1, pp. 49–70. DOI: 10.1007/s13243-020-00091-x.

Khor & Udin 2012

Khor, K. & Udin, Z. (2012), „Impact of Reverse Logistics Product Disposition towards Business Performance in Malaysian E&E Companies“, *Journal of Supply Chain and Customer Relationship Management*, pp. 1–19. DOI: 10.5171/2012.699469.

Kin & Ong et al. 2014

Kin, S. T. M.; Ong, S. & Nee, A. (2014), „Remanufacturing process planning“, *Procedia CIRP* 15, pp. 189–194. DOI: 10.1016/j.procir.2014.06.087.

Kirchherr & Reike et al. 2017

Kirchherr, J.; Reike, D. & Hekkert, M. (2017), „Conceptualizing the circular economy: An analysis of 114 definitions“, *Resources, conservation and recycling* 127, pp. 221–232. DOI: 10.2139/ssrn.3037579.

Kober & Bagnell et al. 2013

Kober, J.; Bagnell, J. A. & Peters, J. (2013), „Reinforcement learning in robotics: A survey“, *The International Journal of Robotics Research* 32.11, pp. 1238–1274. DOI: 10.1007/978-3-642-27645-3_18.

Koch & Kaiser et al. 2024

Koch, D.; Kaiser, J.-P.; Stamer, F.; Stark, R. & Lanza, G. (2024), „Enhancing Visual Inspection in Remanufacturing: A Reinforcement Learning Approach with Integrated Robot Simulation“, *Tech Report Preprint of Karlsruhe Institute of Technology*. DOI: doi.org/10.5445/IR/1000176932.

Korbach & Solbach et al. 2021

Korbach, C.; Solbach, M. D.; Memmesheimer, R.; Paulus, D. & Tsotsos, J. K. (2021), „Next-Best-View Estimation based on Deep Reinforcement Learning for Active Object Classification“, arXiv preprint: 2110.06766. URL: https://arxiv.org/abs/2110.06766.

Kostavelis & Gasteratos 2015

Kostavelis, I. & Gasteratos, A. (2015), „Semantic mapping for mobile robotics tasks: A survey“, *Robotics and Autonomous Systems* 66, pp. 86–103. DOI: https://doi.org/10.1016/j.robot.2014.12.006.

Koutecký & Paloušek et al. 2016

Koutecký, T.; Paloušek, D. & Brandejs, J. (2016), „Sensor planning system for fringe projection scanning of sheet metal parts“, *Measurement* 94, pp. 60–70. DOI: 10.1016/J.MEASUREMENT.2016.07.067.

Kurilova-Palisaitiene & Sundin et al. 2018

Kurilova-Palisaitiene, J.; Sundin, E. & Poksinska, B. (2018), „Remanufacturing challenges and possible lean improvements“, *Journal of Cleaner Production* 172, pp. 3225–3236. DOI: 10.1016/J.JCLEPRO.2017.11.023.

Lacher & Vasconcelos et al. 2019

Lacher, R. M.; Vasconcelos, F.; Williams, N. R.; Rindermann, G.; Hipwell, J.; Hawkes, D. & Stoyanov, D. (2019), „Nonrigid reconstruction of 3D breast surfaces with a low-cost RGBD camera for surgical planning and aesthetic evaluation“, *Medical Image Analysis* 53, pp. 11–25. DOI: 10.1016/j.media.2019.01.003.

Landgraf & Meese et al. 2021

Landgraf, C.; Meese, B.; Pabst, M.; Martius, G. & Huber, M. F. (2021), „A Reinforcement Learning Approach to View Planning for Automated Inspection Tasks“, *Sensors (Basel, Switzerland)* 21.6, p. 2030. DOI: 10.3390/s21062030.

Lange 2017

Lange, U. (2017), *Ressourceneffizienz durch Remanufacturing - Industrielle Aufarbeitung von Altteilen: Kurzanalyse Nr. 18*. Ed. by VDI ZRE. Accessed 26.6.2023. URL: https://www.ressource-deutschland.de/service/publikationen/detailseite/ka-18-remanufacturing/.

LeCun & Bengio et al. 2015

LeCun, Y.; Bengio, Y. & Hinton, G. (2015), „Deep learning“, *nature* 521.7553, pp. 436–444. DOI: 10.1038/nature14539.

Liu & Zhu et al. 2019

Liu, C.; Zhu, Q.; Wei, F.; Rao, W.; Liu, J.; Hu, J. & Cai, W. (2019), „A review on remanufacturing assembly management and technology“, *The International Journal of Advanced Manufacturing Technology* 105, pp. 4797–4808. DOI: 10.1007/s00170-019-04617-x.

Lund 1984

Lund, R. T. (1984), „Remanufacturing: the experience of the United States and implications for developing countries“, *Integrated resource recovery series* 2.

Lund & Hauser 2010

Lund, R. T. & Hauser, W. M. (2010), „Remanufacturing - An American Perspective“, ed. by N. Gindy; R. Farr & Y. Wang. DOI: 10.1049/cp.2010.0404.

Lundmark & Sundin et al. 2009

Lundmark, P.; Sundin, E. & Björkman, M. (2009), „Industrial challenges within the remanufacturing system“. Proceedings of the 3rd Swedish Production Symposium (SPS) (Göteborg, Schweden, Dec. 2–3, 2009), pp. 132–138.

Mareczek 2020

Mareczek, J. (2020), *Grundlagen der Roboter-Manipulatoren–Band 1: Modellbildung von Kinematik und Dynamik*. Springer, Berlin Heidelberg. DOI: 10.1007/978-3-662-52759-7.

Martín & González et al. 2021

Martín, F.; González, F.; Guerrero, J. M.; Fernández, M. & Ginés, J. (2021), „Semantic 3D Mapping from Deep Image Segmentation“, *Applied Sciences* 11.4. DOI: 10.3390/app11041953.

Matsumoto & Ijomah 2013

Matsumoto, M. & Ijomah, W. L. (2013), „Remanufacturing“. *Handbook of Sustainable Engineering*. Ed. by J. Kauffmann & K.-M. Lee. Springer, Dodrecht, pp. 389–408. DOI: 10.1007/978-1-4020-8939-8_93.

Matsumoto & Umeda 2011

Matsumoto, M. & Umeda, Y. (2011), „An analysis of remanufacturing practices in Japan“, *Journal of Remanufacturing* 1, pp. 1–11. DOI: doi.org/10.1186/2210-4690-1-2.

McCormac & Handa et al. 2017

McCormac, J.; Handa, A.; Davison, A. & Leutenegger, S. (2017), „Semanticfusion: Dense 3d semantic mapping with convolutional neural networks“. Proceedings of the 2017 IEEE International Conference on Robotics and automation (ICRA) (Marina Bay Sands, Singapur, May 29–June 3, 2017). Ed. by A. Okamura. IEEE, pp. 4628–4635. DOI: 10.1109/ICRA.2017.7989538.

Mendoza & Vasquez-Gomez et al. 2020

Mendoza, M.; Vasquez-Gomez, J. I.; Taud, H.; Sucar, L. E. & Reta, C. (2020), „Supervised learning of the next-best-view for 3d object reconstruction“, *Pattern Recognition Letters* 133, pp. 224–231. DOI: 10.1016/j.patrec.2020.02.024.

Minaee & Boykov et al. 2021

Minaee, S.; Boykov, Y.; Porikli, F.; Plaza, A.; Kehtarnavaz, N. & Terzopoulos, D. (2021), „Image segmentation using deep learning: A survey“, *IEEE transactions on pattern analysis and machine intelligence* 44.7, pp. 3523–3542. DOI: 10.1109/TPAMI.2021.3059968.

Mnih & Kavukcuoglu et al. 2013

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D. & Riedmiller, M. (2013), „Playing atari with deep reinforcement learning“, arXiv preprint: 1312.5602. URL: https://arxiv.org/abs/1312.5602.

Mnih & Kavukcuoglu et al. 2015

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G. et al. (2015), „Human-level control through deep reinforcement learning“, *nature* 518.7540, pp. 529–533. DOI: 10.1038/nature14236.

Moerland & Broekens et al. 2023

Moerland, T. M.; Broekens, J.; Plaat, A.; Jonker, C. M. et al. (2023), „Model-based reinforcement learning: A survey“, *Foundations and Trends® in Machine Learning* 16.1, pp. 1–118.

Monica & Aleotti 2021

Monica, R. & Aleotti, J. (2021), „A probabilistic next best view planner for depth cameras based on deep learning“, *IEEE Robotics and Automation Letters* 6.2, pp. 3529–3536. DOI: 10.1109/LRA.2021.3064298.

Newman & Jain 1995

Newman, T. S. & Jain, A. K. (1995), „A survey of automated visual inspection“, *Computer vision and image understanding* 61.2, pp. 231–262. DOI: 10.1006/cviu.1995.1017.

Nüchter & Hertzberg 2008

Nüchter, A. & Hertzberg, J. (2008), „Towards semantic maps for mobile robots“, *Robotics and Autonomous Systems* 56.11, pp. 915–926. DOI: https://doi.org/10.1016/j.robot.2008.08.001.

Nwankpa & Ijomah et al. 2021

Nwankpa, C. E.; Ijomah, W. & Gachagan, A. (2021), „Design for automated inspection in remanufacturing: A discrete event simulation for process improvement“, *Cleaner Engineering and Technology* 4, pp. 1–10. DOI: https://doi.org/10.1016/j.clet.2021.100199.

Pan & Hu et al. 2022

Pan, S.; Hu, H. & Wei, H. (2022), „SCVP: Learning one-shot view planning via set covering for unknown object reconstruction“, *IEEE Robotics and Automation Letters* 7.2, pp. 1463–1470. DOI: 10.1109/LRA.2022.3140449.

Panzer & Bender 2021

Panzer, M. & Bender, B. (2021), „Deep reinforcement learning in production systems: a systematic literature review“, *International Journal of Production Research*, pp. 1–26. DOI: 10.1080/00207543.2021.1973138.

Parker & Riley et al. 2015

Parker, D.; Riley, K.; Robinson, S.; Symington, H.; Tewson, J.; Jansson, K.; Ramkumar, S. & Peck, D. (2015), *Remanufacturing Market Study*. Ed. by D. Parker & European Remanufacturing Network. Accessed: 26.06.2023. URL: https://ec.europa.eu/research/participants/documents/downloadPublic?documentIds=080166e5a4bc7898&appId=PPGMS.

Peuzin-Jubert & Polette et al. 2021

Peuzin-Jubert, M.; Polette, A.; Nozais, D.; Mari, J.-L. & Pernot, J.-P. (2021), „Survey on the View Planning Problem for Reverse Engineering and Automated Control Applications“, *Computer-Aided Design* 141, p. 103094. DOI: 10.1016/j.cad.2021.103094.

Pfeifer & Schmitt 2010

Pfeifer, T. & Schmitt, R. (2010), *Fertigungsmesstechnik*. Oldenbourg Wissenschaftsverlag GmbH, München. DOI: 10.1524/9783486711356.

Pito 1999

Pito, R. (1999), „A solution to the next best view problem for automated surface acquisition“, *IEEE Transactions on pattern analysis and machine intelligence* 21.10, pp. 1016–1030. DOI: 10.1109/34.799908.

Potapova & Artemov et al. 2020

Potapova, S.; Artemov, A.; Sviridov, S.; Musatkina, D.; Zorin, D. & Burnaev, E. (2020), „Next best view planning via reinforcement learning for scanning of arbitrary 3d shapes“, *Journal of Communications Technology and Electronics* 65, pp. 1484–1490. DOI: 10.1134/S1064226920120141.

Powers 2020

Powers, D. M. (2020), „Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation“, arXiv preprint: 2010.16061. URL: https://arxiv.org/abs/2010.16061.

Qi & Su et al. 2017

Qi, C. R.; Su, H.; Mo, K. & Guibas, L. J. (2017), „PointNet: Deep learning on point sets for 3d classification and segmentation“. Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (Honolulu, HI, USA, July 21–26, 2017). Ed. by Y. Liu; J. M. Rehg; C. J. Taylor & Y. Wu, pp. 652–660. DOI: 10.1109/CVPR.2017.16.

Raffaeli & Mengoni et al. 2013a

Raffaeli, R.; Mengoni, M. & Germani, M. (2013a), „Context dependent automatic view planning: the inspection of mechanical components“, *Computer-Aided Design and Applications* 10.1, pp. 111–127. DOI: 10.3722/CADAPS.2013.111-127.

Raffaeli & Mengoni et al. 2013b

Raffaeli, R.; Mengoni, M.; Germani, M. & Mandorli, F. (2013b), „Off-line view planning for the inspection of mechanical parts“, *International Journal on Interactive Design and Manufacturing (IJIDeM)* 7, pp. 1–12. DOI: 10.1007/s12008-012-0160-1.

Raffin & Hill et al. 2021

Raffin, A.; Hill, A.; Gleave, A.; Kanervisto, A.; Ernestus, M. & Dormann, N. (2021), „Stable-Baselines3: Reliable Reinforcement Learning Implementations“, *Journal of Machine Learning Research* 22.268, pp. 1–8.

Redmon & Divvala et al. 2016

Redmon, J.; Divvala, S.; Girshick, R. & Farhadi, A. (2016), „You only look once: Unified, real-time object detection“. Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (Las Vegas, NV, USA, June 27–30, 2016). Ed. by T. Tuytelaars; F.-F. Li & R. Bajcsy, pp. 779–788. DOI: 10.1109/CVPR.2016.9.

Reinhard & Heidrich et al. 2010

Reinhard, E.; Heidrich, W.; Debevec, P.; Pattanaik, S.; Ward, G. & Myszkowski, K. (2010), *High dynamic range imaging: acquisition, display, and image-based lighting*. Morgan Kaufmann, Burlington, Massachusetts. DOI: 10.1016/B978-0-12-585263-0.X5000-3.

Reinke & Tizabi et al. 2021

Reinke, A.; Tizabi, M. D.; Sudre, C. H.; Eisenmann, M.; Rädsch, T.; Baumgartner, M.; Acion, L.; Antonelli, M.; Arbel, T.; Bakas, S. et al. (2021), „Common limitations of image processing metrics: A picture story“, arXiv preprint: 2104.05642. URL: https://arxiv.org/abs/2104.05642.

Ren & Fang et al. 2022

Ren, Z.; Fang, F.; Yan, N. & Wu, Y. (2022), „State of the art in defect detection based on machine vision“, *International Journal of Precision Engineering and Manufacturing-Green Technology* 9.2, pp. 661–691. DOI: 10.1007/s40684-021-00343-6.

Rickli & Dasgupta et al. 2014

Rickli, J. L.; Dasgupta, A. K. & Dinda, G. P. (2014), „A descriptive framework for additive remanufacturing systems“, *International Journal of Rapid Manufacturing* 4.2-4, pp. 199–218. DOI: 10.1504/IJRAPIDM.2014.066043.

Robotis & Boyaci et al. 2012

Robotis, A.; Boyaci, T. & Verter, V. (2012), „Investing in reusability of products of uncertain remanufacturing cost: The role of inspection capabilities“, *International Journal of Production Economics* 140.1, pp. 385–395. DOI: https://doi.org/10.1016/j.ijpe.2012.04.017.

Ronneberger & Fischer et al. 2015

Ronneberger, O.; Fischer, P. & Brox, T. (2015), „U-net: Convolutional networks for biomedical image segmentation“. Proceedings of the 18th Medical Image Computing and Computer-Assisted Intervention (MICCAI) Part III 18 (München, Deutschland, Oct. 5–9, 2015). Ed. by N. Navab. Springer, pp. 234–241. DOI: 10.1007/978-3-319-24574-4_28.

Russell & Norvig 2010

Russell, S. J. & Norvig, P. (2010), *Artificial intelligence - A modern approach*. Pearson Education, Inc., London.

Saavedra & Barquet et al. 2013

Saavedra, Y. M.; Barquet, A. P.; Rozenfeld, H.; Forcellini, F. A. & Ometto, A. R. (2013), „Remanufacturing in Brazil: case studies on the automotive sector“, *Journal of Cleaner Production* 53, pp. 267–276. DOI: 10.1016/j.jclepro.2013.03.038.

Saiz & Alfaro et al. 2021

Saiz, F. A.; Alfaro, G. & Barandiaran, I. (2021), „An inspection and classification system for automotive component remanufacturing industry based on ensemble learning“, *Information* 12.12, p. 489. DOI: 10.3390/info12120489.

Salkin & Oner et al. 2018

Salkin, C.; Oner, M.; Ustundag, A. & Cevikcan, E. (2018), „A conceptual framework for Industry 4.0“. *Industry 4.0: managing the digital transformation*. Springer, pp. 3–23. DOI: 10.1007/978-3-319-57870-5_1.

Schlüter & Lickert et al. 2021

Schlüter, M.; Lickert, H.; Schweitzer, K.; Bilge, P.; Briese, C.; Dietrich, F. & Krüger, J. (2021), „AI-enhanced identification, inspection and sorting for reverse logistics in remanufacturing“, *Procedia CIRP* 98, pp. 300–305. DOI: 10.1016/j.procir.2021.01.107.

Schlüter & Niebuhr et al. 2018

Schlüter, M.; Niebuhr, C. A.; Lehr, J. & Krüger, J. (2018), „Vision-based Identification Service for Remanufacturing Sorting“, *Procedia Manufacturing* 21, pp. 384–391. DOI: 10.1016/j.promfg.2018.02.135.

Schulman & Wolski et al. 2017

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A. & Klimov, O. (2017), „Proximal Policy Optimization Algorithms.“, arXiv preprint: 1707.06347. URL: https://arxiv.org/abs/1707.06347.

Scott & Roth et al. 2003

Scott, W.; Roth, G. & Rivest, J.-F. (Mar. 2003), „View planning for automated three-dimensional object reconstruction and inspection“, *ACM Comput. Surv.* 35, pp. 64–96. DOI: 10.1145/641865.641868.

Scott 2009

Scott, W. R. (2009), „Model-based view planning“, *Machine Vision and Applications* 20.1, pp. 47–69. DOI: 10.1007/s00138-007-0110-2.

Seaver 1994

Seaver, W. B. (1994), „Design considerations for remanufacturability, recyclability and reusability of user interface modules“. IEEE International Symposium on Electronics and The Environment (San Francisco, CA, USA, May 2–4, 1994), pp. 241–245. DOI: 10.1109/ISEE.1994.337251.

See & Drury et al. 2017

See, J. E.; Drury, C. G.; Speed, A.; Williams, A. & Khalandi, N. (2017), „The role of visual inspection in the 21st century“, 61.1, pp. 262–266. DOI: 10.1177/1541931213601548.

Seitz & Peattie 2004

Seitz, M. A. & Peattie, K. (2004), „Meeting the closed-loop challenge: the case of remanufacturing“, *California management review* 46.2, pp. 74–89. DOI: 10.2307/41166211.

Shorten & Khoshgoftaar 2019

Shorten, C. & Khoshgoftaar, T. M. (2019), „A survey on image data augmentation for deep learning“, *Journal of big data* 6.1, pp. 1–48. DOI: 10.1186/s40537-019-0197-0.

Silver & Schrittwieser et al. 2017

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A. et al. (2017), „Mastering the game of go without human knowledge“, *nature* 550.7676, pp. 354–359. DOI: 10.1038/nature24270.

Solar & Parada et al. 2002

Solar, M.; Parada, V. & Urrutia, R. (2002), „A parallel genetic algorithm to solve the set-covering problem“, *Computers and Operations Research* 29.9, pp. 1221–1235. DOI: 10.1016/S0305-0548(01)00026-0.

Stanford Artificial Intelligence Laboratory et al. 2018

Stanford Artificial Intelligence Laboratory et al. (May 23, 2018), *Robotic Operating System*. Version ROS Melodic Morenia. Acessed: 04.07.2024. URL: https://www.ros.org.

Steinhilper 1998

Steinhilper, R. (1998), *Remanufacturing*. Fraunhofer IRB Verlag, Stuttgart.

Sünderhauf & Pham et al. 2017

Sünderhauf, N.; Pham, T. T.; Latif, Y.; Milford, M. & Reid, I. (2017), „Meaningful maps with object-oriented semantic mapping“. Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (Vancouver, Canada, Sept. 24–28, 2017). Ed. by H. Zhang. IEEE, pp. 5079–5085.

Sundin 2004

Sundin, E. (2004), „Product and process design for successful remanufacturing“. Dissertation. Linköping University.

Sundin & Bras 2005

Sundin, E. & Bras, B. (2005), „Making functional sales environmentally and economically beneficial through product remanufacturing“, *Journal of cleaner production* 13.9, pp. 913–925. DOI: 10.1016/j.jclepro.2004.04.006.

Sutton & Barto 2018

Sutton, R. S. & Barto, A. G. (2018), *Reinforcement learning: An introduction*. MIT press, Cambridge, MA.

Tan & Sun et al. 2018

Tan, C.; Sun, F.; Kong, T.; Zhang, W.; Yang, C. & Liu, C. (2018), „A survey on deep transfer learning“. Proceedings of the 27th International Conference on Artificial Neural Networks (ICANN) Part III 27 (Rhodos, Griechenland, Oct. 4–7, 2018). Ed. by V. Kůrková; Y. Manolopoulos; B. Hammer; L. Iliadis & I. Maglogiannis. Springer, Cham, pp. 270–279. DOI: 10.1007/978-3-030-01424-7_27.

Tant & Mulholland et al. 2019

Tant, K. M.; Mulholland, A. J.; Curtis, A. & Ijomah, W. L. (2019), „Design-for-testing for improved remanufacturability“, *Journal of Remanufacturing* 9, pp. 61–72. DOI: 10.1007/s13243-018-0057-7.

Tarabanis & Tsai et al. 1995

Tarabanis, K. A.; Tsai, R. Y. & Allen, P. K. (1995), „The MVP sensor planning system for robotic vision tasks“, *IEEE Transactions on Robotics and Automation* 11.1, pp. 72–85. DOI: 10.1109/70.345939.

Tarbox & Gottschlich 1995

Tarbox, G. H. & Gottschlich, S. N. (1995), „Planning for complete sensor coverage in inspection“, *Computer vision and image understanding* 61.1, pp. 84–111. DOI: 10.1006/cviu.1995.1007.

Thierry & Salomon et al. 1995

Thierry, M.; Salomon, M.; van Nunen, J. & van Wassenhove, L. N. (1995), „Strategic Issues in Product Recovery Management“, *California Management Review* 37.2, pp. 114–136. DOI: 10.2307/41165792.

Tolio & Bernard et al. 2017

Tolio, T.; Bernard, A.; Colledani, M.; Kara, S.; Seliger, G.; Duflou, J.; Battaia, O. & Takata, S. (2017), „Design, management and control of demanufacturing and remanufacturing systems“, *CIRP Annals* 66.2, pp. 585–609. DOI: 10.1016/j.cirp.2017.05.001.

Vaquero & Turk et al. 2010

Vaquero, D.; Turk, M.; Pulli, K.; Tico, M. & Gelfand, N. (2010), „A survey of image retargeting techniques", 7798, pp. 328–342. DOI: 10.1117/12.862419.

Vasquez-Gomez & Sucar et al. 2014

Vasquez-Gomez, J. I.; Sucar, L. E. & Murrieta-Cid, R. (2014), „View planning for 3d object reconstruction with a mobile manipulator robot" (Chicago, USA, Sept. 14–18, 2014). Ed. by K. Lynch & L. Parker. IEEE, pp. 4227–4233. DOI: 10.1109/IROS.2014.6943158.

Vasquez-Gomez & Troncoso et al. 2021

Vasquez-Gomez, J. I.; Troncoso, D.; Becerra, I.; Sucar, E. & Murrieta-Cid, R. (2021), „Next-best-view regression using a 3D convolutional neural network", *Machine Vision and Applications* 32, pp. 1–14. DOI: 10.1007/s00138-020-01166-2.

Vongbunyong & Chen et al. 2015

Vongbunyong, S.; Chen, W. H.; Vongbunyong, S. & Chen, W. H. (2015), *Disassembly automation*. Springer, Cham. DOI: 10.1007/978-3-319-15183-0.

Wang & Wang et al. 2021

Wang, C.; Wang, C.; Li, W. & Wang, H. (2021), „A brief survey on RGB-D semantic segmentation using deep learning", *Displays* 70, p. 102080. DOI: 10.1016/j.displa.2021.102080.

Wang & Ma et al. 2020

Wang, Q.; Ma, Y.; Zhao, K. & Tian, Y. (2020), „A comprehensive survey of loss functions in machine learning", *Annals of Data Science*, pp. 1–26. DOI: 10.1007/s40745-020-00253-5.

Wang & Peng et al. 2023

Wang, Y.; Peng, T.; Wang, W. & Luo, M. (2023), „High-efficient view planning for surface inspection based on parallel deep reinforcement learning", *Advanced Engineering Informatics* 55, p. 101849. DOI: 10.1016/j.aei.2022.101849.

Wen & Li et al. 2020

Wen, X.; Li, T.; Han, Z. & Liu, Y.-S. (2020), „Point cloud completion by skip-attention network with hierarchical folding". Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (Seattle, USA, June 13–19, 2020). Ed. by T. Boult; G. Medioni & R. Zabih, pp. 1939–1948. DOI: 10.1109/CVPR42600.2020.00201.

Wu & Zhou et al. 2022

Wu, C.; Zhou, K.; Kaiser, J.-P.; Mitschke, N.; Klein, J.-F.; Pfrommer, J.; Beyerer, J.; Lanza, G.; Heizmann, M. & Furmans, K. (2022), „MotorFactory: A Blender Add-on for Large Dataset Generation of Small Electric Motors", *Procedia CIRP* 106, pp. 138–143. DOI: 10.1016/j.procir.2022.02.168.

Wu & Lu et al. 2015

Wu, Q.; Lu, J.; Zou, W. & Xu, D. (2015), „Path planning for surface inspection on a robot-based scanning system“. Proceedings of the 2015 IEEE International Conference on Mechatronics and Automation (ICMA) (Beijing, China, Aug. 2–5, 2015). Ed. by K. Kosuge & J. Chen. IEEE, pp. 2284–2289. DOI: 10.1109/ICMA.2015.7237842.

Wu & Lin et al. 2020

Wu, Z.-G.; Lin, C.-Y.; Chang, H.-W. & Lin, P. T. (2020), „Inline inspection with an industrial robot (IIIR) for mass-customization production line“, *Sensors* 20.11, p. 3008. DOI: 10.3390/s20113008.

Yuan & Khot et al. 2018

Yuan, W.; Khot, T.; Held, D.; Mertz, C. & Hebert, M. (2018), „PCN: Point Completion Network“. Proceedings of the 2018 International Conference on 3D Vision, Processing, Visualization and Transmission (3DIMPVT) (Verona, Italien, Sept. 5–8, 2018). Ed. by A. Fusiello, pp. 728–737. DOI: 10.1109/3DV.2018.00088.

Zeng & Wen et al. 2020

Zeng, R.; Wen, Y.; Zhao, W. & Liu, Y.-J. (2020), „View planning in robot active vision: A survey of systems, algorithms, and applications“, *Computational Visual Media* 6, pp. 225–245. DOI: 10.1007/s41095-020-0179-3.

Zeng & Zhao et al. 2020

Zeng, R.; Zhao, W. & Liu, Y.-J. (2020), „Pc-nbv: A point cloud based deep network for efficient next best view planning“. Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (Las Vegas, USA, Oct. 25–29, 2020). Ed. by P. Oh. IEEE, pp. 7050–7057. DOI: 10.1109/IROS45743.2020.9340916.

Zeng & Zaenker et al. 2022

Zeng, X.; Zaenker, T. & Bennewitz, M. (2022), „Deep reinforcement learning for next-best-view planning in agricultural applications“. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) (Philadelhpia, USA, May 23–27, 2022). Ed. by V. Kumar & G. J. Pappas. IEEE, pp. 2323–2329. DOI: 10.1109/ICRA46639.2022.9811800.

Zhang & Yu 2020

Zhang, H. & Yu, T. (2020), „Taxonomy of reinforcement learning algorithms“, *Deep Reinforcement Learning: Fundamentals, Research and Applications*, pp. 125–133. DOI: 10.1007/978-981-15-4095-0_3.

Zhang & Sidibé et al. 2021

Zhang, Y.; Sidibé, D.; Morel, O. & Mériaudeau, F. (2021), „Deep multimodal fusion for semantic image segmentation: A survey“, *Image and Vision Computing* 105, p. 104042. DOI: 10.1016/j.imavis.2020.104042.

Zheng 2021

Zheng, Y. (2021), *Intelligent and Automatic Inspection, Reconstruction and Process Planning Methods for Remanufacturing and Repair*. DOI: 10.7939/R3-5QQK-8408.

List of Figures

List of Tables

Own Publications

Gerlitz, E.; Greifenstein, M.; Kaiser, J.-P.; Mayer, D.; Lanza, G. & Fleischer, J. (2022), „Systematic Identification of Hazardous States and Approach for Condition Monitoring in the Context of Li-ion Battery Disassembly“, *Procedia CIRP* 107, pp. 308–313. DOI: 10.1016/j.procir.2022.02.062.

Kaiser, J.-P.; Becker, S. N.; Wurster, M.; Stricker, N. & Lanza, G. (2021), „Framework for simulation-based Trajectory Planning and Execution of Robots equipped with a Laser Scanner for Measurement and Inspection“, *Procedia CIRP* 103, pp. 292–297. DOI: 10.1016/j.procir.2021.10.047.

Kaiser, J.-P.; Bolender, M.; Eschner, N. & Lanza, G. (2021), „View-Planning im Remanufacturing : Optisches Erfassen von Produkten im Remanufacturing durch View-Planning“, German. *WT Werkstattstechnik* 111.11-12, pp. 781–785. DOI: 10.37544/1436-4980-2021-11-12-11.

Kaiser, J.-P.; Gäbele, J.; Koch, D.; Schmid, J.; Stamer, F. & Lanza, G. (2024), „Adaptive acquisition planning for visual inspection in remanufacturing using reinforcement learning“, *Journal of Intelligent Manufacturing*, pp. 1–27.

Kaiser, J.-P.; Koch, D.; Gäbele, J.; May, M. C. & Lanza, G. (2024), „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.

Kaiser, J.-P.; Lang, S.; Wurster, M. & Lanza, G. (2022), „A concept for autonomous quality control for core inspection in remanufacturing“, *Procedia CIRP* 105, pp. 374–379.

Kaiser, J.-P.; Mitschke, N.; Stricker, N.; Heizmann, M. & Lanza, G. (2021), „Konzept einer automatisierten und modularen Befundungsstation in der wandlungsfähigen Produktion: Am Anwendungsfall des Remanufacturings“, *Zeitschrift für wirtschaftlichen Fabrikbetrieb* 116.5, pp. 313–317. DOI: 10.1515/zwf-2021-0070.

Koch, D.; Kaiser, J.-P.; Stamer, F.; Stark, R. & Lanza, G. (2024), „Enhancing Visual Inspection in Remanufacturing: A Reinforcement Learning Approach with Integrated Robot Simulation“, *Tech Report Preprint of Karlsruhe Institute of Technology*. DOI: 10.5445/IR/1000176932.

Kuhnle, A.; Kaiser, J.-P.; Theiß, F.; Stricker, N. & Lanza, G. (2021), „Designing an adaptive production control system using reinforcement learning“, *Journal of Intelligent Manufacturing* 32, pp. 855–876. DOI: 10.1007/s10845-020-01612-y.

Lanza, G.; Asfour, T.; Beyerer, J.; Deml, B.; Fleischer, J.; Heizmann, M.; Furmans, K.; Hofmann, C.; Cebulla, A.; Dreher, C. et al. (2022), „Agiles Produktionssystem mittels lernender Roboter bei ungewissen Produktzuständen am Beispiel der Anlasser-Demontage“, *at-Automatisierungstechnik* 70.6, pp. 504–516. DOI: 10.1515/auto-2021-0158.

Merz, D.; Gerlitz, E.; Kaiser, J.-P. & Fleischer, J. (2023), „Anomalieerkennung bei der Li-Ionen-Zellkontaktierung: Innovative Anomalieerkennung im Laserschweißen: Eine Pipeline basierend auf der Analyse von Strahlungsemissionen und maschinellem Lernen", *Zeitschrift für wirtschaftlichen Fabrikbetrieb* 118.11, pp. 790–794. DOI: 10.1515/zwf-2023-1146.

Schild, L.; Sasse, F.; Kaiser, J.-P. & Lanza, G. (2022), „Assessing the optical configuration of a structured light scanner in metrological use", *Measurement Science and Technology* 33.8, pp. 1–16. DOI: 10.1088/1361-6501/ac6e2f.

Wu, C.; Zhou, K.; Kaiser, J.-P.; Mitschke, N.; Klein, J.-F.; Pfrommer, J.; Beyerer, J.; Lanza, G.; Heizmann, M. & Furmans, K. (2022), „MotorFactory: A Blender Add-on for Large Dataset Generation of Small Electric Motors", *Procedia CIRP* 106, pp. 138–143. DOI: 10.5445/IR/1000147768.

Wurster, M.; Exner, Y.; Kaiser, J.-P.; Stricker, N. & Lanza, G. (2021), „Towards planning and control in cognitive factories-A generic model including learning effects and knowledge transfer across system entities", *Procedia CIRP* 103, pp. 158–163. DOI: 10.1016/j.procir.2021.10.025.

Wurster, M.; Klein, J.-F.; Kaiser, J.-P.; Mangold, S.; Furmans, K.; Heizmann, M.; Fleischer, J. & Lanza, G. (2022), „Integrierte Steuerungsarchitektur für ein agiles Demontagesystem mit autonomer Produktbefundung", *at-Automatisierungstechnik* 70.6, pp. 542–556. DOI: 10.1515/auto-2021-0157.

Appendix

A1 General solution approach for SCP, CM and NBV approaches

Solution of the VPP via solving the SCP

The SCP is defined in Peuzin-Jubert & Polette et al. 2021 as follows. Given a parent set $U = \{u_1, ..., u_n\}$ with elements u_i ($i = 1, ..., n$). Another set $G = \{g_1, ..., g_m\}$ contains elements g_j ($j = 1, ..., m$) which are subsets of U such that $g_j \subset U$. Furthermore, the union of all subsets in G corresponds to the set U ($\cup g_j = U$). Solving SCP is then equivalent to finding the smallest list of subsets in G such that the union of this list is equal to U (Peuzin-Jubert & Polette et al. 2021, P. 2). An extension to the traditional SCP is adding costs to each element g_j, where then the goal is to find the smallest list of subsets in G with the least costs that fully cover U.

Based on this definition, a model-based approach can solve the problem of the SCP. The elements u_i of the set U correspond to the object's total surface to be acquired, divided into sections. The indice j ($j \in [1, ..., m]$) of g_j denotes the jth of the m VP. Each VP can acquire a certain section of the total surface. g_j then represents surface sections visible from VP j. Thus, G is the set of all combinations of surface sections seen from all m VPs. Accordingly, a list of VPs must be found that covers all sections of the total surface, hence the set U. To solve SCP, Peuzin-Jubert & Polette et al. (2021) describes the common procedure in the literature in five steps.

The first step divides the object surface into a fixed number of sections. Each section corresponds to an element u_i of the set U. For example, the faces of a mesh or points of a point cloud can be used as sections. Another possibility is representing sections as voxels after the voxelization of the object. In the second step, a number of m VP are sampled. For the sampled VPs, all vectors g_j ($j = 1, ..., m$) containing the acquirable surface sections of the VP of the acquisition system are empty at the beginning of the reconstruction process. Therefore, the set of surface sections u_i that can be acquired by each VP j are calculated and stored in g_j. A visibility matrix is suitable to evaluate which sections can be acquired by a VP. This binary matrix introduced by Tarbox & Gottschlich (1995) contains as rows all VP j, as well as the surface sections u_i as columns (Tarbox & Gottschlich 1995, P. 91). One row for VP j corresponds to g_j. The fourth step calculates the needed VPs to fully cover U. To determine which of the VPs to choose, a selection algorithm is used, whose selection of the VPs solves the optimization problem. Two kinds of selection algorithms can be distinguished. First, selection algorithms that iteratively choose a VP j ($j \in [1, ..., m]$) and add g_j to the set

G until a termination criterion is met. Second, selection algorithms that determine the whole set G in one computational step and optimize it iteratively. To solve the first kind of selection problem, a simple greedy algorithm might be suitable, which selects in each step the VP that covers the maximum number of surface sections u_i not yet covered by the VPs currently in G (Peuzin-Jubert & Polette et al. 2021, P. 8). A selection algorithm for the second problem might be realized via evolutionary algorithms (Solar & Parada et al. 2002). In this case, several candidate sets of VP would be directly generated and optimized until a solution based on a given termination criterion is found. The fifth step is then concerned with postprocessing (e.g., finding the optimal sequence of the VP) of the found solution.

Solution of the VPP via coverage maximization
The following methods are primarily aimed at maximizing the surface coverage of an object. The number of acquisitions required to achieve that goal is only considered a secondary objective, in contrast to the solution of SCP.

In the first step, the object's surface is divided into sections. The quality of this step is of crucial importance in the case of applying coverage optimization methods since their success mainly depends on the subdivision of the surface. Different possibilities are proposed in the literature to subdivide the object into sections (segmentation of the object surface) according to Peuzin-Jubert & Polette et al. (2021). For instance, if the object model is represented as a point cloud, clusters can be generated based on point distances and their orientation. These clusters are then grouped into sections.

In the second step, the VPs are generated for the complete coverage of all segmented surface sections. In contrast to the methods for solving the SCP, the VPs are not generated and evaluated in a subsequent step. However, they are determined directly from the surface segmentation. The approaches in Peuzin-Jubert & Polette et al. (2021) define a VP for each surface segment. According to Germani & Mengoni et al. (2009), a visibility map can be used (Germani & Mengoni et al. 2009, P. 1714). Starting from a VP oriented using the normal of a surface section, the directions of occluded contours are determined using the visibility map. This map is computed for each point of a section by projecting the surfaces to be determined onto a unit sphere. New poses are then determined to include the largest possible number of points. Analogous to the solution of the SCP, a visibility matrix is then constructed to link the VPs to the corresponding sections they can acquire. Last, the number of acquisitions needed to maximize the surface coverage is determined.

Since these methods aim not to minimize the number of acquisitions needed, a solution is to choose one acquisition per surface section. Current research strives to reduce the required acquisitions after the initial view plan generation. Methods are described in more detail in Peuzin-Jubert & Polette et al. (2021), to which the interested reader is referred.

Solution of the VPP via NBV planning

Peuzin-Jubert & Polette et al. (2021) define the procedure as follows. At the beginning of the procedure, an initial acquisition is performed with a VP j. After the initial acquisition, the acquired surface sections $g^0 = g_j$ and the estimated remaining unknown surface sections $g^{R,0}$ are deduced. Based on $g^{R,0}$, m new VPs are generated. Each of these m VP gets assigned a vector g_o^1 ($o = 1, ..., m$). Each of the vectors g_o^1 contains a subset of the estimated remaining unknown surface sections $g^{R,0}$ of the previous acquisition with VP j that can be acquired with VP o ($o \in [1, ..., m]$).

The next VP is then selected based on a selection algorithm. This can be achieved, for example, using a greedy algorithm that chooses the VP o ($o \in [1, ..., m]$) associated with the vector g_o^1 containing the most surface segments estimated to be not yet covered. Consequently, the sets of known surface segments are calculated as the union of surface sections covered by both previous acquisitions $g^1 = g^0 \cup g_o^1$. Furthermore, the estimated unknown surface segments $g^{R,0}$ are updated to $g^{R,1}$, and the procedure is repeated until a termination criterion is met.

Different approaches represent the acquired surface sections and estimate the unknown surface sections to be acquired by the next views. Pito (1999) uses a simplified mesh to represent the acquired surface (Pito 1999, P. 1026). At the edges of the mesh, the author defines small rectangular sections. This results in three areas based on the acquisition position and orientation. A distinction is made between the seen surfaces, the unseen areas close to an edge of the known area, and the unknown space. New VPs are determined based on an overlap criterion with previous acquisitions to maximize coverage of unseen areas close to seen areas. Furthermore, a voxel representation is commonly used. This has been applied by Vasquez-Gomez & Sucar et al. (2014) and others (Peuzin-Jubert & Polette et al. 2021, P. 16). In the works of Vasquez-Gomez & Sucar et al. (2014), each voxel assumes either the state occupied, not occupied, or unknown. Through an iterative procedure, VPs are searched, from which the unknown voxels can be acquired. This allows for the inclusion of multiple objects located within the workspace. However, voxel representation is very inefficient in memory usage. Since empty voxels must also be considered and stored, the memory and computational overhead increases greatly with the resolution of the voxel grid. New VPs are randomly generated on a sphere whose center is at the object's centroid.

A2 In-depth basics of coordinate transformations

The following is based on Mareczek (2020).

Exemplary calculation of a rotation matrix based on Euler angles of the X-Y'-Z" convention

In the context of kinematics and robotics, Euler angles are a set of three angles used to represent the orientation of a rigid body in three-dimensional space. Rotations with the specified Euler angles are applied to the principal axes in a specific sequence. Euler angles with the rotation convention of X-Y'-Z" are called Cardan angles. The rotations are applied first about the X-axis, then the new Y-axis (Y'), and finally the new Z-axis (Z").

Given the Euler angles α for the rotation around the X-axis, β for the rotation around the Y'-axis, and γ for the rotation around the Z"-axis, the rotation matrices for each of these axes are:

- Rotation about the X-axis (α):

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}$$

- Rotation about the Y'-axis (β):

$$R_{y'}(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}$$

- Rotation about the Z"-axis (γ):

$$R_{z''}(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The combined rotation matrix R for the X-Y'-Z" convention is obtained by multiplying these matrices in the order of the rotations:

$$R = R_x(\alpha) \cdot R_y(\beta) \cdot R_z(\gamma) \qquad \text{A2.1}$$

When performing this matrix multiplication, the result is the final rotation matrix representing the orientation described by the given Euler angles in the X-Y'-Z" rotation order. This matrix is used to transform coordinates from one coordinate system to another or to describe the orientation of an object in space.

Conversion from spherical coordinates to Cartesian coordinates
Given that a point is not represented in Cartesian coordinates, the transformation of said point from spherical coordinates to Cartesian coordinates needs to be calculated. In spherical coordinates, a point is represented by three values: the radial distance r, the polar angle θ (measured from the positive Z-axis), and the azimuthal angle φ (measured from the positive X-axis in the X-Y plane). The Cartesian coordinates (x, y, z) of the point can be calculated from the spherical coordinates (r, θ, φ) using the following equations:

$$x = r\sin(\theta)\cos(\varphi) \tag{A2.2}$$

$$y = r\sin(\theta)\sin(\varphi) \tag{A2.3}$$

$$z = r\cos(\theta) \tag{A2.4}$$

Here, r is the distance from the origin to the point, θ is the angle between the positive Z-axis and the vector from the origin to the point, and φ is the angle between the positive X-axis and the projection of the vector from the origin to the point onto the X-Y plane.

A3 Basics of the perceptron, multi-layer perceptron and backpropagation algorithm

The perceptron

The Perceptron is the basic building block of a simple NN and the mathematical approximation of a human neuron. It can be used as a simple classification algorithm that, according to Ertel (2018), maps an input feature vector $\mathbf{x_f}$ to an output value y_f (Ertel 2018, P. 184, 185). This is done by first calculating the weighted sum $Z(\mathbf{x_f}, \mathbf{w})$ of the input feature vector $\mathbf{x_f}$ and a weight vector $\mathbf{w}$. After computing the weighted sum $Z(\mathbf{x_f}, \mathbf{w})$, an activation function $a(Z(\mathbf{x_f}, \mathbf{w}))$ is applied to it to obtain the actual output y_f of the perceptron according to A3.5:

$$y_f = g(Z(\mathbf{x_f}, \mathbf{w})) = g(\sum_{i=0}^{|X|} w_i \cdot x_{f,i}) = g(w_0 + \sum_{i=1}^{|X|} w_i \cdot x_{f,i}) \qquad \text{A3.5}$$

The transformation step in equation A3.5 can be carried out as the zeroth component $x_{f,0} = 1$ of the input feature vector is the so-called bias, whose value is always one and which is always added to the actual variable input feature vector (Ertel 2018, P. 193, Russell & Norvig 2010, P. 728). In the simplest case, the activation function $a(Z(\mathbf{x_f}, \mathbf{w}))$ is a step function according to formula A3.6 (Ertel 2018, P. 184):

$$y_f = a(Z(\mathbf{x_f}, \mathbf{w})) = \begin{cases} 1, & \text{if } w_0 + \sum_{i=1}^{|X|} w_i \cdot x_{f,i} > 0 \\ 0, & \text{otherwise} \end{cases} \qquad \text{A3.6}$$

The perceptron is thus a linear classifier (Ertel 2018, P. 183). The function of the bias is directly evident from equation A3.6, since it shifts the hyperplane given by the weighted sum $Z(\mathbf{x_f}, \mathbf{w})$ with constant value w_0 (Ertel 2018, P. 187). Thus, data that are not linearly separable with a hyperplane given by $Z(\mathbf{x_f}, \mathbf{w})$, going through the origin, can also be separated. The learning task is then to find the weight vector $\mathbf{w}$ to separate the different classes in the training data. For more information, please refer to Ertel (2018).

Multilayer feed-forward neural networks

Perceptrons can only solve linearly separable problems. A generalization of a perceptron is called a neuron, which may use an arbitrary activation function $a(Z(\mathbf{x_f}, \mathbf{w}))$ (Russell & Norvig 2010, P. 729). Common activation functions are the ***Rectified Linear Unit*** (ReLu, $a(Z(\mathbf{x_f}, \mathbf{w})) = max(0, Z(\mathbf{x_f}, \mathbf{w}))$) or ***Tangens Hyperbolicus*** (tanh, $a(Z(\mathbf{x_f}, \mathbf{w})) = tanh(Z(\mathbf{x_f}, \mathbf{w}))$). To solve nonlinear problems, several neurons may be arranged in successive layers (cf. Ertel 2018, P. 271, Russell & Norvig 2010, P. 729), where each layer is composed of an arbitrary number of neurons. This network topology is also called ***Multilayer Perceptron*** (MLP) (Bishop & Nasrabadi 2006, P. 229) [1]. In so-called feed-forward networks, connections between a neuron only exist to the neurons of neighboring layers in the direction of the network output but not to neurons in its own layer (Russell & Norvig 2010, P. 729). MLPs consist of an input layer propagating the input data into the network. Input layers of neural networks often employ linear activation functions (Arena & Fortuna et al. 1998, P. 7). The result of the classification or regression is the neuron's output in the neural network's last layer. The number of neurons, as well as their activation function thereby, is dependent on the task at hand. For a regression task of one continuous variable, one output neuron with linear activation $a(Z(\mathbf{x_f}, \mathbf{w})) = Z(\mathbf{x_f}, \mathbf{w})$ is useful (Bishop & Nasrabadi 2006, P. 228). For a classification task of more than two classes, for example, when classifying learning to distinguish between images of handwritten digits, it is common to use one output neuron for each digit (Russell & Norvig 2010, P. 729). As an activation function, the softmax function (see equation A3.7) is then used (Ertel 2018, P. 236). It outputs for each neuron, e.g., the digit class, the probability of the input belonging to said class based on the normalization of the exponential output of each neuron k to the exponential outputs of all K neurons in the output layer.

$$y_k = \frac{e^{a_k(Z(\mathbf{x_{f,k}}, \mathbf{w_k}))}}{\sum_{j=1}^{|K|} e^{a_j(Z(\mathbf{x_{f,j}}, \mathbf{w_j}))}} \tag{A3.7}$$

All further layers between the input and output layers are called hidden layers (Ertel 2018, P. 267) and are used for the actual processing of the network input to the output by sequentially passing the input through the layers of the neural network to obtain the output. The number of hidden layers characterizes the depth of an MLP. In contrast to the perceptron, weights $w_{i,j}^{l}$ in MLP are typically denoted with three indices. These represent the layer of the neuron the weight is associated with (l), the number of the neuron in the previous layer (i), which forms the connection with the neuron number (j) in the layer the weight is associated with.

The backpropagation algorithm

Based on Bishop & Nasrabadi (2006), learning in neural networks is performed by continuing adjustment of the weights of the neural network until a satisfying accuracy of the prediction $\hat{y}_{f,i}$ for the training examples and the actual label $y_{f,i}$ is achieved across all training examples (Bishop & Nasrabadi 2006, P. 269 f.). The prediction error is determined by a loss function $\mathcal{L}$ (see equation A3.8). Since $y_{f,i}$ is calculated based on a function of the input $x_{f,i}$ and the weight vector $\mathbf{w}$, the loss function can be expressed as a function of the weights.

[1] Please note that while a MLP is a NN that always contains neurons, the term NN also refers to network types that contain layers that are not made of neurons, for example ***Convolutional neural networks*** (CNNs).

$$\mathcal{L} = f_L(\hat{y}_{f,i}, y_{f,i}) = f_L(\hat{y}_{f,i}, \mathbf{w}, x_{f,i}) \qquad \text{A3.8}$$

The backpropagation algorithm is applied to adjust the weights based on the calculated error. Each weight w_n is corrected according to its contribution to the error, with a learning rate η.

$$\Delta w_{n,t} = -\eta \nabla_{w_n} \mathcal{L} = -\eta \frac{\partial \mathcal{L}}{w_n} \qquad \text{A3.9}$$

The choice of loss function depends on the problem to solve. The mean square error can be used for regression as a loss function, while the cross entropy loss can be used for classification. For more information on the backpropagation algorithm and an overview of loss functions, see Russell & Norvig (2010, P. 733, ff.), Bishop & Nasrabadi (2006, P. 269, ff.) and Wang & Ma et al. (2020).

A4 Details on point cloud transformation using $\mathbf{T}^{A/RT'}$ and determination of $\mathbf{T}^{E/A}$ and $\mathbf{T}^{R/RT}$

Determination of $\mathbf{T}^{A/RT'}$ for transforming $\mathbf{P}_{t,i}^{A}$ to $\mathbf{P}_{t,i}^{RT'}$

The rules for transforming the acquired point clouds between the mentioned coordinate systems can be deduced assuming the knowledge of all transformation matrices above. Each point $\mathbf{P}_{t,i}^{A} = [X_{t,i}^{A}, Y_{t,i}^{A}, Z_{t,i}^{A}]$ of an acquired point cloud $\mathbf{PC}_{t}^{A}$ is initially represented in the acquisition system's coordinate system C_A. For transforming the point cloud to the intermediate reference coordinate system $C_{RT'}$, the transformation matrix $\mathbf{T}^{R/RT'}$ is used, which transforms a point or point cloud from coordinate system C_R to $C_{RT'}$. $\mathbf{T}^{R/RT'}$ can be obtained by a sequential multiplication of $\mathbf{T}^{RT/RT'}$ and $\mathbf{T}^{R/RT}$, since a point or point cloud in C_R has to be first expressed in C_{RT} (using $\mathbf{T}^{R/RT}$) and can then be expressed in $C_{RT'}$ (using $\mathbf{T}^{RT/RT'}$). Consequently, an acquired point cloud in C_A has to first be expressed in C_R (using $\mathbf{T}^{A/R}$) and then from C_R in $C_{RT'}$ using the relationships explained above. This leads to the equation A4.10.

$$\mathbf{T}^{A/RT'} = \mathbf{T}^{R/RT'}\ \mathbf{T}^{A/R} = \mathbf{T}^{RT/RT'}\ \mathbf{T}^{R/RT}\ \mathbf{T}^{A/R} \qquad \text{A4.10}$$

The computation of the coordinate transformation is performed for all points of the point cloud according to equation A4.11 to yield points in the reference coordinate system $\mathbf{P}_{t,i}^{RT'} = [X_{t,i}^{RT'}, Y_{t,i}^{RT'}, Z_{t,i}^{RT'}]$.

$$\mathbf{P}_{t,i}^{RT'} = \mathbf{T}^{A/RT'}\ \mathbf{P}_{t,i}^{A} \quad \forall\ \mathbf{P}_{t,i}^{A} \in \mathbf{PC}_{t}^{A} \qquad \text{A4.11}$$

This results in a transformed point cloud $\mathbf{PC}_{t}^{RT'} = [\mathbf{X}_{t}^{RT'}, \mathbf{Y}_{t}^{RT'}, \mathbf{Z}_{t}^{RT'}, \mathbf{R}_{t}^{RT'}, \mathbf{G}_{t}^{RT'}, \mathbf{B}_{t}^{RT'}]$ where now the points are represented in the rotating coordinate system of the rotary table. At the same time, the color values remain unchanged ($\mathbf{C}_{t}^{A} = \mathbf{C}_{t}^{RT'} = [\mathbf{R}_{t}^{RT'}, \mathbf{G}_{t}^{RT'}, \mathbf{B}_{t}^{RT'}]$). The transformation back to the robots coordinate system C_R can then be achieved similarly, using the transformation matrix $\mathbf{T}^{RT'/R} = \mathbf{T}^{RT/R}\ \mathbf{T}^{RT'/RT}$. It is important to note that $\mathbf{T}^{RT'/RT}$ results from the knowledge of the absolute rotation angle of the rotary table φ_{RT} and that the fundamental calculation rule of transformation matrices $\mathbf{T}^{RT'/RT} = (\mathbf{T}^{RT/RT'})^{-1}$ applies. The following also applies $\mathbf{T}^{R/RT} = (\mathbf{T}^{RT/R})^{-1}$, which is relevant to calculate $\mathbf{T}^{R/RT}$ knowing $\mathbf{T}^{RT/R}$.

Determination of $\mathbf{T}^{A/E}$ and $\mathbf{T}^{RT/R}$

$\mathbf{T}^{A/R}$ can be expressed via the homogeneous transformation matrices of the acquisition system in the coordinate system of the robot end-effector $\mathbf{T}^{A/E}$. The end-effector in the robot coordinate system $\mathbf{T}^{E/R}$ whereas $\mathbf{T}^{A/R} = \mathbf{T}^{E/R}\ \mathbf{T}^{A/E}$ holds. $\mathbf{T}^{E/R}$ can be determined via forward kinematics of the robot's known joint angles and geometric dimensions. $\mathbf{T}^{RT'/RT}$ can be expressed by a homogeneous transformation matrix only containing elements concerning a simple rotation of $C_{RT'}$ around the z-axis of C_{RT}, where the current absolute rotation angle φ_{RT} can always assume to be known. This is because, before each acquisition run, the rotary table performs a reference drive where the final state is the zero position ($C_{RT} = C_{RT'}$ with $\varphi_{RT} = 0$ and $\mathbf{T}^{RT'/RT} = \mathbf{I}_4$) and the rotational angles successively executed can always be summed up. The transformation matrices $\mathbf{T}^{A/E}$ and $\mathbf{T}^{RT/R}$ have to be determined differently.

A hand-eye calibration can be used to determine the constant transformation matrix $\mathbf{T}^{A/E}$ between the robot end-effector coordinate system C_E and the coordinate system C_A of the acquisition system. In this thesis, a checkerboard pattern is a calibration pattern, which is placed statically on the station. Several poses of the acquisition system are sequentially approached with the robot so that the spatial relationship between the coordinate system of the calibration pattern and the acquisition system can be determined. In the case of the checkerboard pattern, the corner pixels of the black and white tiles must be detectable. The constant transformation $\mathbf{T}^{CAL/R}$ between the calibration object and the robot base is unknown. However, it can be expressed by the transformation between the robot base and the end effector $\mathbf{T}^{E/R}$, the static transformation between the end effector and the acquisition system $\mathbf{T}^{A/E}$, and the variable transformation between the acquisition system and the calibration object $\mathbf{T}^{CAL/A}$, which is different for each pose. Thus, a system of equations can be set up. The task of the hand-eye calibration is now to determine the static transformation matrix $\mathbf{T}^{A/E}$ between the end-effector and acquisition system so that the system of equations is solved unambiguously. For further information, please refer to A_Yingqi 2022, a master thesis supervised by the author of this thesis. In this thesis, ten successive acquisitions from different poses of the acquisition system mounted on the robot end-effector were performed. The algorithm used is provided by Zivid itself [1] and was adapted to the needs of this thesis.

The transformation $\mathbf{T}^{RT/R}$ between the robot base coordinate system C_R and the rotary table coordinate system C_{RT} also has to be determined. This can be achieved by performing a certain number of successive acquisitions of the checkerboard pattern. In this case, the checkerboard pattern is placed on the rotary table. The acquisition system pose must be fixed, and the rotary table must be rotated between each acquisition. To determine $\mathbf{T}^{RT/R}$ the translational part of C_R to C_{RT} is calculated first. This is done by sampling a number n random absolute rotation angles $\varphi_{RT,i}$ $(i = 1, ..., n)$. Keypoints of the checkerboard are detected in

[1] Link to documentation of hand-eye calibration: https://github.com/zivid/zivid-python-samples/tree/master/source/applications/advanced/hand_eye_calibration accessed 24.08.2023

the resulting color images $I_{D,i}$. Next, n absolute rotation angles are calculated based on the rule $\varphi_{RT,j} = \varphi_{RT,i} + 180^\circ$ $(j, i = 1, ..., n)$ and again, keypoints are detected in $I_{D,j}$. The three-dimensional coordinate is then assigned to each keypoint of the acquisition results $I_{D,i}$ and $I_{D,j}$ resulting in $\mathbf{P}^{A}_{C,i}$ and $\mathbf{P}^{A}_{C,j}$. These three-dimensional points of the keypoints are then transformed in the static coordinate system C_R resulting in keypoints $\mathbf{P}^{R}_{C,i}$ and $\mathbf{P}^{R}_{C,j}$. By performing a 180° rotation between pairs of acquisitions i and j, the coordinates of the same keypoints in $\mathbf{P}^{R}_{C,i}$ and $\mathbf{P}^{R}_{C,j}$ can be averaged to obtain the coordinate of the rotation center. It is sufficient to average two acquisitions (i.e., $n = 1$). However, by averaging the estimated coordinate centers of the $n > 1$ acquisition pairs $\mathbf{P}^{R}_{C,i}$ and $\mathbf{P}^{R}_{C,j}$ $(i, j = 1, ..., n)$ it is assumed to obtain a better estimate. In this thesis, $n = 5$ is used. The rotational part of C_{RT} in relation to C_R can be determined by estimating the rotary motor's rotation axis parameters. This can be achieved by using optimization algorithms that estimate the parameters of said axis in a way that the same keypoints in $\mathbf{P}^{R}_{C,i}$ and $\mathbf{P}^{R}_{C,j}$ have minimal positional error (e.g., using the mean squared euclidian distance) after rotation with known absolute angles $\varphi_{RT,i}$ and $\varphi_{RT,j}$. However, based on the hardware setup seen in Figure 5.2, it is assumed that the z-axis of C_R and C_{RT} are approximately the same and therefore, the simplified homogeneous transformation matrix $\mathbf{T}^{RT/R}$ only contains translational components. It should be noted that errors induced by this simplification can be compensated by registration methods, as mentioned in section 5.1.4.

A5 In-depth details and foundations on the segmentation model architectures and training procedure

Chosen base model types

The basic models to be compared in this thesis are shown in Figure A5.1 and are:

- The U-Net by Ronneberger & Fischer et al. 2015 is a commonly used base architecture for semantic segmentation.
- The Mask-RCNN (in the following called M-Net) by He & Gkioxari et al. (2017) is a commonly used base architecture for object detection and instance segmentation.

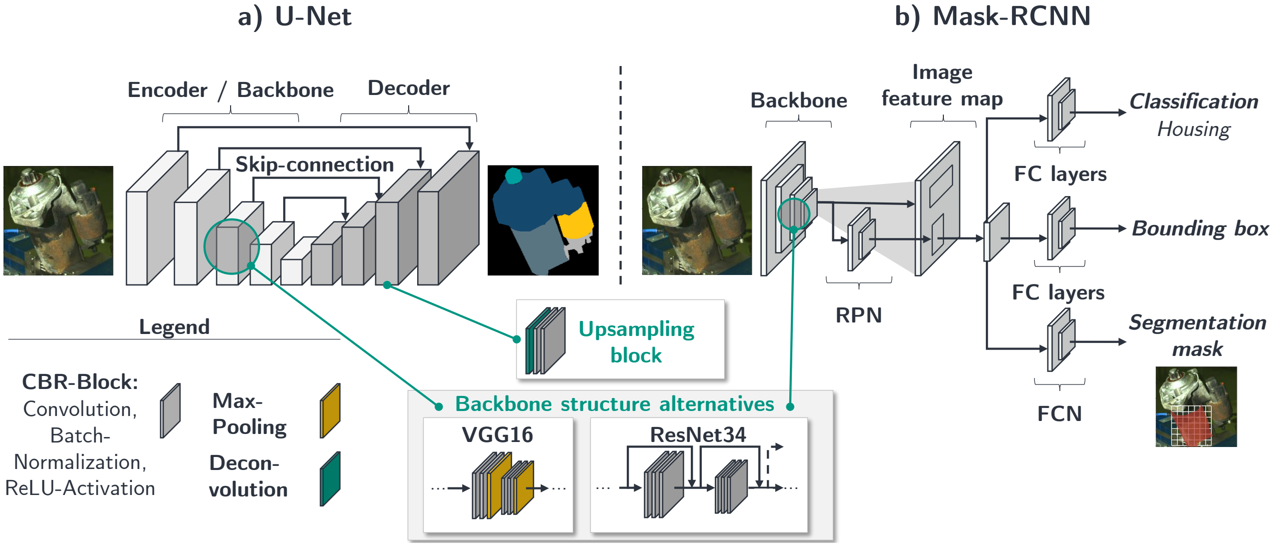


Figure A5.1: Structure of the segmentation methods used. a) Structure of the U-Net as semantic segmentation approach. b) Structure of the M-Net as instance segmentation approach.

Transfer learning and finetuning using pre-trained backbones

Although the amount of data and variability can be increased by image augmentation, the transfer learning method is often used in practice and this thesis. The basic idea is to transfer knowledge from a source task to a target task (Tan & Sun et al. 2018, P. 271). A concept often applied is transfer learning using parameter transfer. The assumption is that NNs work similarly to the processing mechanism of the human brain (Tan & Sun et al. 2018, P. 275). The network input undergoes a continual abstraction process. The first layers of the network can be treated as a feature extractor (see also section 2.3.3 about deep learning), and the extracted features are transferable to new tasks more easily than training a new network from scratch.

In transfer learning with parameter transfer, a NN is trained on a source task (e.g., classification) in a domain (e.g., classification of animals in images) with a huge base dataset. The first layers of these NN often capture general features and patterns that can be valuable for a target task with the same goal (e.g., classification) but for a different domain (e.g., classification of motor types in images). When faced with a new task, instead of training a model from scratch, the first layers of the NN pre-trained on the source task, also called the backbone, can be used and finetuned using a smaller task-specific dataset of the target task. The model then only needs to adapt its learned features from the source task to the target task, leading to faster and often more effective training.

This thesis also uses Pre-trained backbones when constructing the NN, the U-Net, and the M-Net. Available backbone NN architectures differ in the layers' structure. A recent overview of available backbone architectures is in Elharrouss & Akbari et al. (2022). In this thesis, two of the most basic and well-known representatives, the VGG-16 backbone and the ResNet-34 backbone, are used and compared for the segmentation task. VGG backbones use a series of blocks of convolutional layers with batch-normalization and ReLu activation (CBR-Block) with small filters followed by max-pooling layers. ResNet backbones use skip connections between successive CBR-blocks (as opposed to the skip connections between the encoder and decoder of a U-Net). These skip connections allow the extracted features to skip layers, which mitigates the problem of vanishing gradients in deep networks. The basic structure of the individual blocks of both backbone architectures can also be found in Figure A5.1.

The general procedure is to use a backbone as a feature extractor and build neural layers on top of the backbone used to enable the NN architecture to solve the target task. When constructing the U-Net, this results in adding the decoder consisting of deconvolution layers on top of the backbone while enabling information flow from the backbone to the decoder using skip connections. Similarly, when constructing the M-Net, the RPN and the subsequent fully connected layers (for classification and BB regression) and the fully convolutional NN for generating the segmentation mask are stacked onto the backbone used.

However, when using NN with pre-trained backbones, these consist of layers with pre-trained weights (the backbones) and layers without pre-trained weights (all other layers). A common approach is to first freeze (i.e., fix the weights) of the backbone and train the NN only adjusting the weights of the layers not contained in the backbone and then finetune all weights (Chu & Madhavan et al. 2016, P. 435) usually using a lower learning rate than before. However, Chu & Madhavan et al. (2016, P. 440) have shown that the choice of freezing pre-trained layers or not when finetuning should largely depend on the amount of training data available and the similarity between the source task and target task data.

Therefore, this thesis implements and compares three alternatives for the segmentation approach. Given a NN for semantic/ instance segmentation consisting of a backbone and additional layers,

1. no pre-trained backbone is used, i.e., the structure (e.g., a VGG-16 backbone) is used, but the weights are initialized randomly before training.
2. the pre-trained weights of the backbone used are first frozen, and the NN is trained with a comparably high learning rate just adjusting the weights of the newly added layers. After a certain amount of training, the weights of the backbone are unfrozen, and all layers of the NN are trained.
3. no freezing of the pre-trained NN weights is applied and the NN is fine-tuned directly.

Integration of depth information

By default, the U-Net and the M-Net operate exclusively on RGB data. However, approaches to data fusion for segmentation already exist in the state of research. An overview article for multimodal data fusion for segmentation is the one by Zhang & Sidibé et al. (2021). A summary of approaches specifically for the data fusion of RGB and depth data is provided by Wang & Wang et al. (2021).

Zhang & Sidibé et al. (2021) divide the existing approaches to data fusion into (1) early, (2) late, and (2) hybrid data fusion. The analysis of the state of research from Wang & Wang et al. (2021) shows that various approaches and network architectures exist to enable this data fusion. As with the choice of models for semantic segmentation and instance segmentation, simple fusion principles are, therefore, first evaluated in the use case of this thesis. For both networks (U-Net and M-Net), fusion approaches based on the categorization of Zhang & Sidibé et al. (2021) (early, late, and hybrid fusion) are therefore developed and implemented. In the following, the term intermediate fusion will be used for hybrid data fusion, in contrast to the nomenclature of Zhang & Sidibé et al. (2021). This name change is because models that include hybrid data fusion usually do not use a combination of early or late data fusion but rather fuse intermediate features of the backbone.

The alternative model architectures implemented in this thesis for the fusion of RGB and depth data can be found in figures A5.2 (extension of the U-Net by depth data - UD-Net) and A5.3 (extension of the M-Net by depth data - MD-Net). The approach commonly used in the current state of research is used in which a separate feature extractor is used for both the RGB and the depth data. This is also either a VGG-16 or a ResNet34 for the depth data.

The early and intermediate data fusion is the same for both network architectures used for semantic segmentation (UD-Net) and instance segmentation (MD-Net).

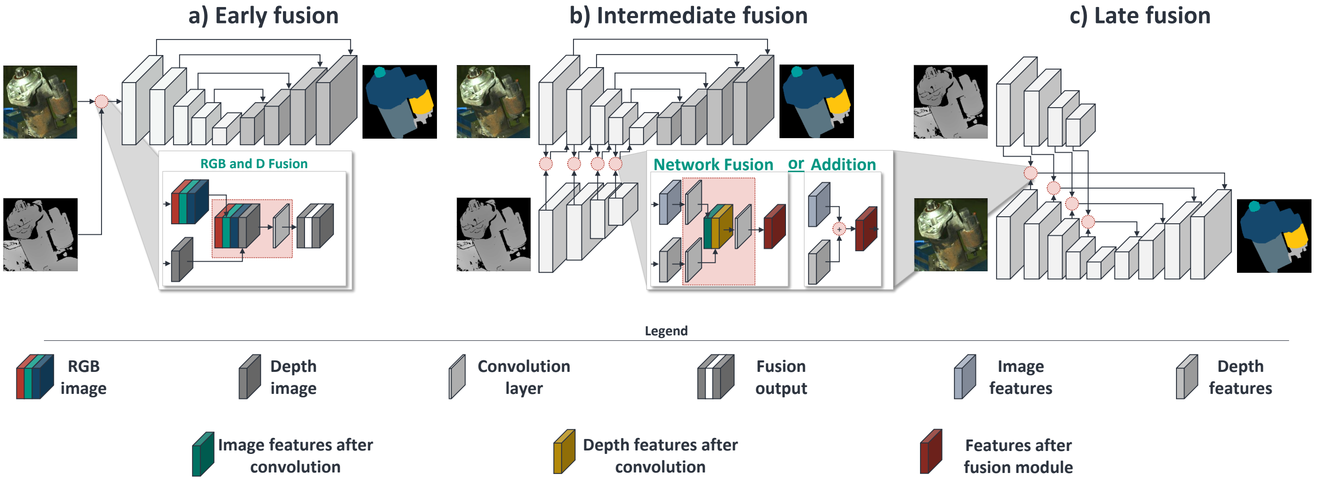


Figure A5.2: Implemented depth fusion approaches for the networks based on the U-Net architecture considered in this thesis.

- **Early fusion:** In early data fusion, RGB and depth data are combined before entering the respective network. This is done by stacking the individual channels (R, G, and B) of the color image with the depth channel of the depth image. Since the pre-trained backbones used only operate on three channels, as these are pre-trained with RGB images, a dimensional reduction takes place in addition to stacking the channels using a convolutional layer with learning weights.

- **Intermediate fusion:** In intermediate data fusion, the intermediate features of the individual layers of the backbones (VGG-16 or ResNet34) for RGB and depth data are first fused and then used as input to the subsequent layer of the backbone of RGB data is used. It should be noted that the feature space size has to remain the same when fusing, as the individual layers of the backbones used depend on a fixed size of the feature space. This results in two options for feature fusion. With **addition**, the features of a layer of the backbones of RGB and depth data are added element-wise. Using a VGG-16, this approach is similar to the one proposed by Hazirbas & Ma et al. (2017). A significant advantage of this option is that it leaves the size of the feature space unchanged. Another comparatively easy option is **network fusion** using an additional intermediate neural network. First, the channels of the initial features of the respective layer are halved using a learning convolution (as with early data fusion). The features of RGB backbone and deep backbone are then stacked to obtain features that are then processed again by a learning convolution to be used as input for the subsequent layer of the RGB backbone.

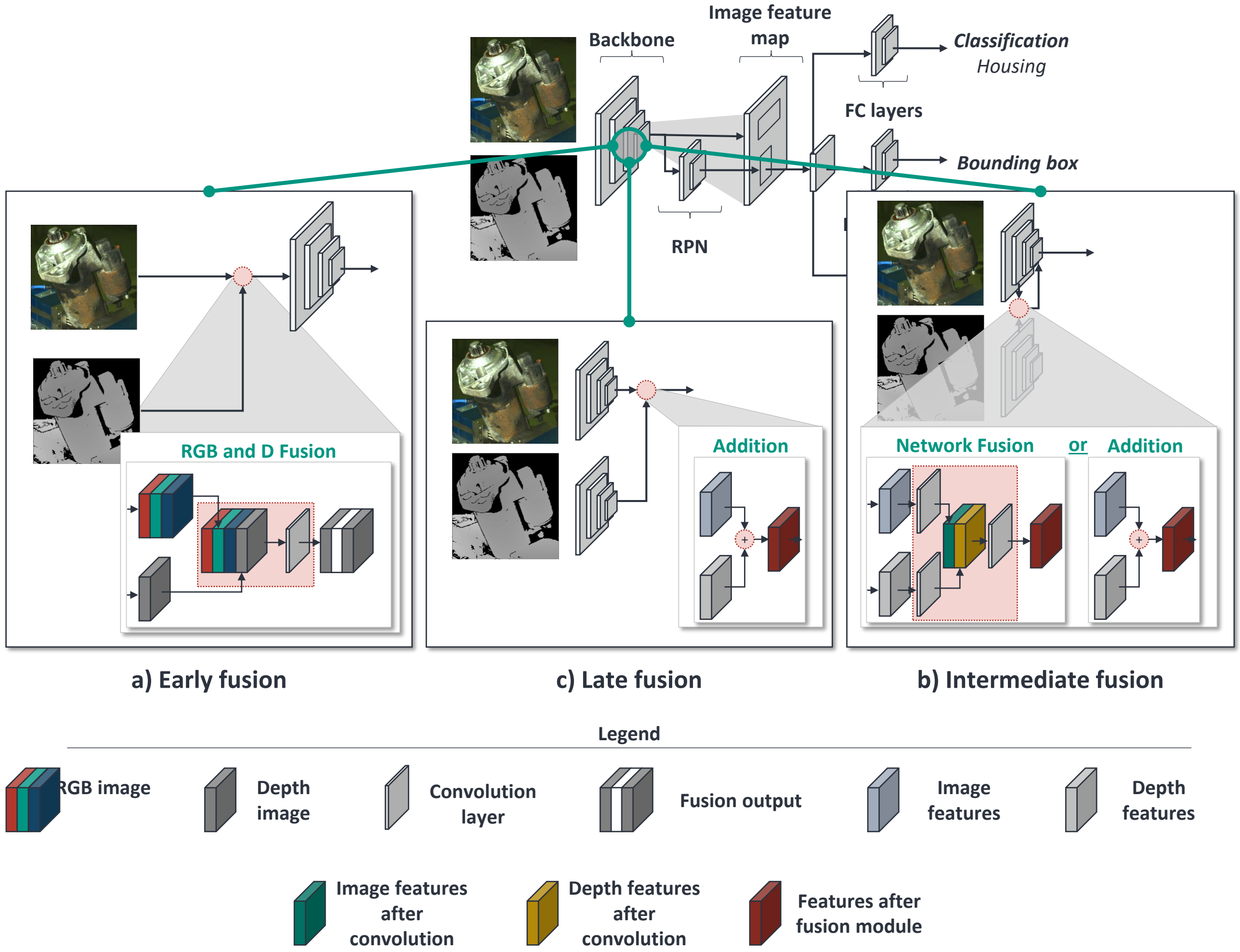


Figure A5.3: Implemented depth fusion approaches for the networks based on the M-Net architecture considered in this thesis.

In contrast to early and intermediate data fusion, the late data fusion approaches for the UD-Net and the MD-Net implemented in this thesis differ.

- **Late fusion of MD-Net:** In contrast to intermediate data fusion, in which intermediate features of individual layers of the backbones are fused, only the output of these backbones is fused in late data fusion. For the MD-Net, this involves a separate extraction of a feature extractor for RGB and depth data and the subsequent fusion. The variant implemented in this thesis corresponds to adding the extracted features and the subsequent further processing of these features for extracting region proposals and predicting the BB and their properties (class, vertices, segmentation mask).
- **Late fusion of UD-Net:** When using a UD-Net, the skip connections can be used to implement the late data fusion. For this purpose, the intermediate features of the individual layers are either added or processed by an intermediate network, as in

intermediate data fusion. These processed features can then be used directly in the decoder. When using addition as a fusion method, this procedure corresponds to a generalization of the fusion method for the MD-Net. This is because the fusion method for the MD-Net results when only the features of the last layer of the RGB backbone and the deep backbone are added.

Loss functions

The loss function for training a semantic segmentation model is calculated using the augmented ground truth label segmentation mask C'_{GT} and the model's prediction C_P. Since the segmentation is initially pixel-based, losses of ordinary classification problems can be used (e.g., pixel-based class prediction accuracy). In addition, region-based losses can be used. An overview of loss functions commonly used for semantic segmentation can be found in Jadon (2020). Two standard loss functions are dice loss and focal loss. The dice loss is based on the overlap between the predicted and true masks of the classes in an image. It aims to optimize the similarity between these class masks by increasing the overlap of all class masks (Jadon 2020, P. 2).

$$L_{Dice} = \frac{1}{C+1}\sum_{c=0}^{C}[1 - \frac{2\sum_{u,v=1}^{H,W} p_c(u,v)\cdot g_c(u,v)}{\sum_{u,v=1}^{H,W} p_c(u,v)^2 + \sum_{u,v=1}^{H,W} g_c(u,v)^2}] \qquad \text{A5.12}$$

Unlike the dice loss, which focuses on mask overlap, the focal loss is based on a pixel's prediction. It introduces a modulating factor γ_f that dynamically adjusts the loss contribution of each pixel. This factor decreases the contribution of well-classified pixels and increases the contribution of misclassified pixels. Additionally, α_c is class dependent and is a balancing factor when there is a significant class imbalance. The Focal loss enables the model to learn more effectively by putting more emphasis on challenging examples, thus performing well in tasks with class imbalance (Jadon 2020, P. 2).

$$\begin{aligned} L_{Focal} &= \frac{1}{C+1}\sum_{c=0}^{C}\sum_{u,v}^{H,W}[\alpha_c(1-p_t(u,v))^{\gamma_f}\cdot log(p_t(u,v))] \\ \text{with} \quad p_t &= \begin{cases} p_c(u,v) & \text{if} \quad g_c(u,v)=1 \\ 1-p_c(u,v) & \text{else} \end{cases} \end{aligned} \qquad \text{A5.13}$$

The formulas for Dice-Loss and Focal-Loss can be found in equations A5.12 and A5.13. Both loss functions calculate the mean value of the respective loss over all classes (number C) of the pixels (u,v) to be segmented in an image with size $H \times W$. $p_c(u,v)$ accordingly denotes the prediction of the class membership of the pixel (u, v) to class c and $g_c(u, v)$ the ground truth of the pixel (u, v) for class c.

In contrast to the comparatively simple definition of the loss function for semantic segmentation, the loss function, for instance, segmentation models, such as, e.g., M-Net, is composed of different loss functions. Since M-Net allows the localization and segmentation of an object instance using a bounding box in an image, the loss function is composed of the loss functions for the (1) localization of the object in the image using the bounding box (L_{box}), the (2) assignment (classification) of the object to a class (L_{cls}), and the (3) predicted segmented object mask (L_{mask}) (Hafiz & Bhat 2020, P. 1046). The whole loss function is then based on the sum of the individual loss functions according to equation A5.14.

$$L_{M-RCNN} = L_{box} + L_{cls} + L_{mask} \quad \text{A5.14}$$

These contributing loss functions can be individually defined. For example, L_{mask} is defined in the M-Net as the binary cross entropy loss for the per-pixel binary classification of the assignment of the pixel to the object (He & Gkioxari et al. 2017, P. 2963).

The evaluation metric IoU

Similarly, evaluation metrics based on the classification quality of individual pixels or region-based metrics can be used to evaluate the segmentation quality of the approaches used. The pixel accuracy calculates the percentage of correctly classified pixels in the entire image to be segmented. It is a simple and intuitive measure of overall segmentation accuracy but does not account for class imbalances. Region-based metrics (e.g., *intersection over union* - IoU) consider the similarity of the resulting segmentation masks. IoU calculates the ratio of the area of overlap between the predicted class mask and ground truth mask for a class to the area of their union. It is a standard metric used in segmentation tasks to evaluate the accuracy of object localization and class prediction. Another variant of a region-based metric is the dice coefficient. The dice coefficient calculates the ratio of the overlap area of the predicted and ground truth class masks in relation to the union of the area of both the predicted and ground truth class masks. It is helpful for tasks with imbalanced class distributions. Since the evaluation metrics indicate the quality of the segmentation, they can also be used to define loss functions for training the segmentation algorithms.

A6 T-Scan system product information

The table A6.1 lists the relevant parameters of the handheld T-Scan laser scanner and its tracking system (T-Track 10) based on available data sheets[1]. These systems were used in this work to generate the reference models. The Zeiss T-Scan and Zeiss T-Track 10 can be seen in Figure A6.1, which was extracted of the T-Scan systems' website[2].

Table A6.1: Technical data of the Zeiss T-Scan and Zeiss T-Track 10 system extracted from the technical datasheet.[2]

Zeiss T-Scan	
Characteristic	**Value or Description**
Measuring depth	+/- 50 mm
Line width	Up to 125 mm
Average working distance	150 mm
Line frequency	Up to 330 Hz
Data rate	210.000 Points/Second
Average point distance	0.075 mm

Zeiss T-Track 10	
Characteristic	**Value or Description**
Measuring distance object - camera	2.0 m - 6.0 m
Measuring volume	20 m^3
Field of view	Up to 3200 mm x 2500 mm
Measuring rate	Up to 2.8 kHz
Accuracy	0.04 mm + 0.04 mm/m

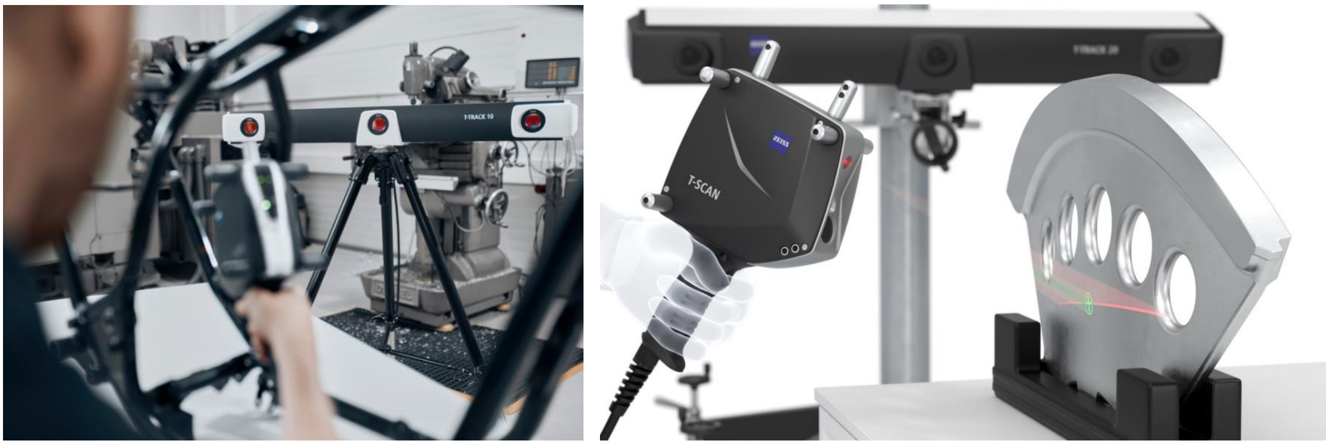


Figure A6.1: Visual display of the Zeiss T-Scan and Zeiss T-Track 10 systems based on the T-Scan system website.

[1] Link of datasheet:
https://www.handsonmetrology.com/de/loesungen/t-scan/#Features
accessed 04.04.2024

[2] Link of image:
https://www.handsonmetrology.com/de/loesungen/t-scan/#Features
accessed 04.04.2024

A7 Evaluation metrics for the 3D reconstruction approach

Hausdorff Distance

The Hausdorff Distance measures the maximum distance from any point in one set to the closest point in the other set, reflecting the extent of maximum mismatch between the two sets. Since the raw Hausdorff distance uses the maximum distance as a criterion of mismatch, it is therefore prone to outliers. Thus, the adapted metric *Hausdorff Distance 95th percentile* ($HD_{95}^{C\to L}$) is used in this work. $HD_{95}^{C\to L}$ between two sets C and L of a metric space X, is defined as (Reinke & Tizabi et al. 2021, P. 24):

$$HD_{95}^{C\to L} = \max\left\{d_{95}(C,L), d_{95}(L,C)\right\} \qquad \text{A7.15}$$

Thereby, C and L denote point clouds and the metric space is the Cartesian Space. $d_{95}(C,L)$ denotes 95th percentile of the distances between all points in C to all points in L. Similarly, $d_{95}(L,C)$ denotes the 95th percentile of the distances of all points in L to all points in C. It is therefore a more conservative approach to calculating the mismatch of two point clouds.

Mean Error

The Mean Error (ME) is used in this work as a metric to quantify the average distance between the two point clouds C and L. It can be defined as:

$$ME^{C\to L} = \frac{1}{n}\sum_{i=1}^{n}\min_{l\in L} d(c_i, l) \qquad \text{A7.16}$$

Thereby, n denotes the number of points in the point cloud C. For each point c_i, the distance to its nearest neighboring point in point cloud L is calculated. These distances are summed up to calculate $ME^{C\to L}$. $ME^{C\to L}$ is useful for assessing the spatial similarity of two point clouds, especially when direct point correspondences are not available, by focusing on the closeness of points in C relative to the geometry of L. ME is a variant of the mean absolute error (Hodson (2022, P. 5482)), whereby the point distances are used instead of the absolute value.

A8 Result tables for training the U-Net, UD-Net, M-Net and MD-Net configurations

Table A8.1: Training results for the U-Nets SGD optimizer and IoU as the evaluation metric.[a]

Model	Backbone	Optimizer	Loss	Learning rate[b]	Pretrain	Freeze	$\mathrm{IoU}_{\mathrm{max}}$	$\mathrm{IoU}_{\mathrm{mean}}$	$\mathrm{IoU}_{\mathrm{std}}$
U-Net	**VGG16**	**SGD**	**Dice**	$\mathbf{1 * 10^{-2}}$	**True**	**True**	0.8081	**0.8050**	0.0036
U-Net	VGG16	SGD	Dice	$1 * 10^{-2}$	True	False	0.6455	0.6449	0.0004
U-Net	VGG16	SGD	Dice	$1 * 10^{-2}$	False	False	0.5658	0.5564	0.0073
U-Net	VGG16	SGD	Dice	$5 * 10^{-3}$	True	True	0.6855	0.6831	0.0025
U-Net	VGG16	SGD	Dice	$5 * 10^{-3}$	True	False	0.5884	0.5396	0.0515
U-Net	VGG16	SGD	Dice	$5 * 10^{-3}$	False	False	0.3970	0.3880	0.0111
U-Net	VGG16	SGD	Dice	$1 * 10^{-3}$	True	True	0.5408	0.5032	0.0267
U-Net	VGG16	SGD	Dice	$1 * 10^{-3}$	True	False	0.4091	0.4009	0.0084
U-Net	VGG16	SGD	Dice	$1 * 10^{-3}$	False	False	0.3218	0.3155	0.0084
U-Net	**VGG16**	**SGD**	**Focal**	$\mathbf{1 * 10^{-2}}$	**True**	**True**	**0.8124**	0.7638	0.0344
U-Net	VGG16	SGD	Focal	$1 * 10^{-2}$	True	False	0.6584	0.6537	0.0056
U-Net	VGG16	SGD	Focal	$1 * 10^{-2}$	False	False	0.4910	0.4819	0.0082
U-Net	VGG16	SGD	Focal	$5 * 10^{-3}$	True	True	0.7004	0.6931	0.0056
U-Net	VGG16	SGD	Focal	$5 * 10^{-3}$	True	False	0.5958	0.5316	0.0454
U-Net	VGG16	SGD	Focal	$5 * 10^{-3}$	False	False	0.4007	0.3875	0.0094
U-Net	VGG16	SGD	Focal	$1 * 10^{-3}$	True	True	0.4860	0.4829	0.0035
U-Net	VGG16	SGD	Focal	$1 * 10^{-3}$	True	False	0.4193	0.4029	0.0168
U-Net	VGG16	SGD	Focal	$1 * 10^{-3}$	False	False	0.3118	0.2987	0.0098
U-Net	**ResNet34**	**SGD**	**Dice**	$\mathbf{1 * 10^{-2}}$	**True**	**True**	**0.6596**	**0.6561**	0.0025
U-Net	ResNet34	SGD	Dice	$1 * 10^{-2}$	True	False	0.4712	0.4690	0.0016
U-Net	ResNet34	SGD	Dice	$1 * 10^{-2}$	False	False	0.4253	0.4214	0.0028
U-Net	ResNet34	SGD	Dice	$5 * 10^{-3}$	True	True	0.4757	0.4715	0.0035
U-Net	ResNet34	SGD	Dice	$5 * 10^{-3}$	True	False	0.4259	0.4139	0.0085
U-Net	ResNet34	SGD	Dice	$5 * 10^{-3}$	False	False	0.3680	0.3655	0.0026
U-Net	ResNet34	SGD	Dice	$1 * 10^{-3}$	True	True	0.3021	0.2700	0.0248
U-Net	ResNet34	SGD	Dice	$1 * 10^{-3}$	True	False	-	-	-
U-Net	ResNet34	SGD	Dice	$1 * 10^{-3}$	False	False	-	-	-
U-Net	ResNet34	SGD	Focal	$1 * 10^{-2}$	True	True	0.5242	0.5230	0.0009
U-Net	ResNet34	SGD	Focal	$1 * 10^{-2}$	True	False	0.4699	0.4679	0.0020
U-Net	ResNet34	SGD	Focal	$1e^-4$	False	False	0.4182	0.4143	0.0031
U-Net	ResNet34	SGD	Focal	$5 * 10^{-3}$	True	True	0.5240	0.5186	0.0058
U-Net	ResNet34	SGD	Focal	$5 * 10^{-3}$	True	False	0.4142	0.3620	0.0445
U-Net	ResNet34	SGD	Focal	$5 * 10^{-3}$	False	False	0.3600	0.3477	0.0136
U-Net	ResNet34	SGD	Focal	$1 * 10^{-3}$	True	True	0.2478	0.1357	0.1008
U-Net	ResNet34	SGD	Focal	$1 * 10^{-3}$	True	False	-	-	-
U-Net	ResNet34	SGD	Focal	$1 * 10^{-3}$	False	False	0.0061	0.0037	0.0021

[a] : Rows with green text indicate best modeling options for a respective modeling variant regarding $\mathbf{IoU_{mean}}$ and/or $\mathbf{IoU_{max}}$

[b] : Initial learning rate, reduced by 20% every 10 epochs

Table A8.2: Training results for the U-Nets Adam optimizer and IoU as the evaluation metric.[a]

Model	Backbone	Optimizer	Loss	Learning rate[b]	Pretrain	Freeze	$\mathrm{IoU}_{\mathrm{max}}$	$\mathrm{IoU}_{\mathrm{mean}}$	$\mathrm{IoU}_{\mathrm{std}}$
U-Net	VGG16	Adam	Dice	$1 * 10^{-4}$	True	True	0.9100	0.9074	0.0026
U-Net	**VGG16**	**Adam**	**Dice**	$\mathbf{1 * 10^{-4}}$	**True**	**False**	**0.9217**	**0.9187**	0.0027
U-Net	VGG16	Adam	Dice	$1 * 10^{-4}$	False	False	0.8179	0.8136	0.0042
U-Net	VGG16	Adam	Dice	$5 * 10^{-5}$	True	True	0.9035	0.9015	0.0032
U-Net	VGG16	Adam	Dice	$5 * 10^{-5}$	True	False	0.8983	0.8976	0.0006
U-Net	VGG16	Adam	Dice	$5 * 10^{-5}$	False	False	0.8593	0.8564	0.0034
U-Net	VGG16	Adam	Dice	$1 * 10^{-5}$	True	True	0.8892	0.8882	0.0009
U-Net	VGG16	Adam	Dice	$1 * 10^{-5}$	True	False	0.8727	0.8702	0.0024
U-Net	VGG16	Adam	Dice	$1 * 10^{-5}$	False	False	0.7362	0.7327	0.0030
U-Net	VGG16	Adam	Focal	$1 * 10^{-4}$	True	True	0.9131	0.9129	0.0002
U-Net	VGG16	Adam	Focal	$1 * 10^{-4}$	True	False	0.9164	0.9161	0.0005
U-Net	VGG16	Adam	Focal	$1 * 10^{-4}$	False	False	0.8632	0.8603	0.0039
U-Net	VGG16	Adam	Focal	$5 * 10^{-5}$	True	True	0.9059	0.9026	0.0032
U-Net	VGG16	Adam	Focal	$5 * 10^{-5}$	True	False	0.9161	0.9153	0.0007
U-Net	VGG16	Adam	Focal	$5 * 10^{-5}$	False	False	0.8688	0.8603	0.008
U-Net	VGG16	Adam	Focal	$1 * 10^{-5}$	True	True	0.9048	0.9039	0.0013
U-Net	VGG16	Adam	Focal	$1 * 10^{-5}$	True	False	0.8864	0.8695	0.0180
U-Net	VGG16	Adam	Focal	$1 * 10^{-5}$	False	False	0.7428	0.7276	0.0247
U-Net	ResNet34	Adam	Dice	$1 * 10^{-4}$	True	True	0.8876	0.8854	0.0026
U-Net	**ResNet34**	**Adam**	**Dice**	$\mathbf{1 * 10^{-4}}$	**True**	**False**	**0.9033**	**0.9005**	0.0028
U-Net	ResNet34	Adam	Dice	$1 * 10^{-4}$	False	False	0.7708	0.7658	0.0065
U-Net	ResNet34	Adam	Dice	$5 * 10^{-5}$	True	True	0.8900	0.8890	0.0015
U-Net	ResNet34	Adam	Dice	$5 * 10^{-5}$	True	False	0.8788	0.8396	0.0572
U-Net	ResNet34	Adam	Dice	$5 * 10^{-5}$	False	False	0.7232	0.7031	0.0320
U-Net	ResNet34	Adam	Dice	$1 * 10^{-5}$	True	True	0.8614	0.8577	0.0039
U-Net	ResNet34	Adam	Dice	$1 * 10^{-5}$	True	False	0.6562	0.5764	0.0691
U-Net	ResNet34	Adam	Dice	$1 * 10^{-5}$	False	False	0.6128	0.5801	0.0039
U-Net	ResNet34	Adam	Focal	$1 * 10^{-4}$	True	True	0.8977	0.8970	0.0009
U-Net	ResNet34	Adam	Focal	$1 * 10^{-4}$	True	False	0.8893	0.8850	0.0069
U-Net	ResNet34	Adam	Focal	$1 * 10^{-4}$	False	False	0.7960	0.7935	0.0028
U-Net	ResNet34	Adam	Focal	$5 * 10^{-5}$	True	True	0.8948	0.8915	0.0029
U-Net	ResNet34	Adam	Focal	$5 * 10^{-5}$	True	False	0.7578	0.7527	0.0045
U-Net	ResNet34	Adam	Focal	$5 * 10^{-5}$	False	False	0.7711	0.7431	0.0315
U-Net	ResNet34	Adam	Focal	$1 * 10^{-5}$	True	True	0.7904	0.7629	0.0241
U-Net	ResNet34	Adam	Focal	$1 * 10^{-5}$	True	False	0.5447	0.5436	0.0010
U-Net	ResNet34	Adam	Focal	$1 * 10^{-5}$	False	False	0.5012	0.5005	0.0007

[a] : Rows with green text indicate best modeling options for a respective modeling variant regarding $\mathrm{IoU}_{\mathrm{mean}}$ and/or $\mathrm{IoU}_{\mathrm{max}}$

[b] : Initial learning rate, reduced by 20% every 10 epochs

Table A8.3: Training results for the M-Nets with IoU as the evaluation metric.[a]

Model	Backbone	Optimizer	Loss	Learning rate[b]	Pretrain	Freeze	$\mathrm{IoU}_{\mathrm{max}}$	$\mathrm{IoU}_{\mathrm{mean}}$	$\mathrm{IoU}_{\mathrm{std}}$
Mask-RCNN	**VGG16**	**Adam**	**Default**	$\mathbf{1 * 10^{-4}}$	**True**	**True**	**0.9517**	**0.9462**	0.0040
Mask-RCNN	VGG16	Adam	Default	$1 * 10^{-4}$	True	False	0.9438	0.9383	0.0040
Mask-RCNN	VGG16	Adam	Default	$1 * 10^{-4}$	False	False	0.9200	0.9166	0.0025
Mask-RCNN	VGG16	Adam	Default	$5 * 10^{-5}$	True	True	0.9422	0.9380	0.0032
Mask-RCNN	VGG16	Adam	Default	$5 * 10^{-5}$	True	False	0.9386	0.9344	0.0041
Mask-RCNN	VGG16	Adam	Default	$5 * 10^{-5}$	False	False	0.9050	0.8967	0.0110
Mask-RCNN	VGG16	Adam	Default	$1 * 10^{-5}$	True	True	0.9256	0.8972	0.0083
Mask-RCNN	VGG16	Adam	Default	$1 * 10^{-5}$	True	False	0.9038	0.8972	0.0083
Mask-RCNN	VGG16	Adam	Default	$1 * 10^{-5}$	False	False	0.8266	0.8213	0.0037
Mask-RCNN	VGG16	SGD	Default	$1 * 10^{-2}$	True	True	0.9372	0.9323	0.0038
Mask-RCNN	VGG16	SGD	Default	$1 * 10^{-2}$	True	False	0.9313	0.9254	0.0055
Mask-RCNN	VGG16	SGD	Default	$1 * 10^{-2}$	False	False	0.9073	0.8979	0.0067
Mask-RCNN	VGG16	SGD	Default	$5 * 10^{-3}$	True	True	0.9358	0.9323	0.0029
Mask-RCNN	VGG16	SGD	Default	$5 * 10^{-3}$	True	False	0.9194	0.9144	0.0040
Mask-RCNN	VGG16	SGD	Default	$5 * 10^{-3}$	False	False	0.8839	0.8708	0.0096
Mask-RCNN	VGG16	SGD	Default	$1 * 10^{-3}$	True	True	0.9090	0.9066	0.0020
Mask-RCNN	VGG16	SGD	Default	$1 * 10^{-3}$	True	False	0.8393	0.8261	0.0134
Mask-RCNN	VGG16	SGD	Default	$1 * 10^{-3}$	False	False	0.8144	0.7943	0.0194
Mask-RCNN	**ResNet34**	**Adam**	**Default**	$\mathbf{1 * 10^{-4}}$	**True**	**True**	**0.9566**	**0.9537**	0.0026
Mask-RCNN	ResNet34	Adam	Default	$1 * 10^{-4}$	True	False	0.9518	0.9487	0.0027
Mask-RCNN	ResNet34	Adam	Default	$1 * 10^{-4}$	False	False	0.9294	0.9268	0.0033
Mask-RCNN	ResNet34	Adam	Default	$5 * 10^{-5}$	True	True	0.9527	0.9482	0.0044
Mask-RCNN	ResNet34	Adam	Default	$5 * 10^{-5}$	True	False	0.9466	0.9452	0.0014
Mask-RCNN	ResNet34	Adam	Default	$5 * 10^{-5}$	False	False	0.9140	0.9041	0.0110
Mask-RCNN	ResNet34	Adam	Default	$1 * 10^{-5}$	True	True	0.9393	0.9353	0.0030
Mask-RCNN	ResNet34	Adam	Default	$1 * 10^{-5}$	True	False	0.9196	0.9150	0.0044
Mask-RCNN	ResNet34	Adam	Default	$1 * 10^{-5}$	False	False	0.8307	0.8180	0.0093
Mask-RCNN	ResNet34	SGD	Default	$1 * 10^{-2}$	True	True	0.9505	0.9488	0.0019
Mask-RCNN	ResNet34	SGD	Default	$1 * 10^{-2}$	True	False	0.9449	0.9414	0.0033
Mask-RCNN	ResNet34	SGD	Default	$1 * 10^{-2}$	False	False	0.9198	0.9106	0.0079
Mask-RCNN	ResNet34	SGD	Default	$5 * 10^{-3}$	True	True	0.9457	0.9420	0.0030
Mask-RCNN	ResNet34	SGD	Default	$5 * 10^{-3}$	True	False	0.9393	0.9358	0.0034
Mask-RCNN	ResNet34	SGD	Default	$5 * 10^{-3}$	False	False	0.8981	0.8902	0.0059
Mask-RCNN	ResNet34	SGD	Default	$1 * 10^{-3}$	True	True	0.9371	0.9292	0.0056
Mask-RCNN	ResNet34	SGD	Default	$1 * 10^{-3}$	True	False	0.9089	0.9077	0.0010
Mask-RCNN	ResNet34	SGD	Default	$1 * 10^{-3}$	False	False	0.8444	0.8252	0.0143

[a] : Rows with green text indicate best modeling options for a respective modeling variant regarding $\mathbf{IoU}_{\mathbf{mean}}$ and/or $\mathbf{IoU}_{\mathbf{max}}$

[b] : Initial learning rate, reduced by 20% every 10 epochs

Table A8.4: Training results for the UD-Nets with IoU as the evaluation metric.[a]

Model	Fusion type	Backbone	Optimizer	Loss	Learning rate[b]	Pretrain	Freeze	$\mathrm{IoU}_{\mathrm{max}}$	$\mathrm{IoU}_{\mathrm{mean}}$	$\mathrm{IoU}_{\mathrm{std}}$
UD-4D	Early	VGG16	Adam	Dice	$1 * 10^{-4}$	True	True	0.9075	0.9062	0.0018
UD-4D	Early	VGG16	Adam	Dice	$1 * 10^{-4}$	True	False	0.9124	0.9104	0.0014
UD-4D	Early	VGG16	Adam	Dice	$1 * 10^{-4}$	False	False	0.7883	0.7795	0.0101
UD-4D	**Early**	**VGG16**	**Adam**	**Dice**	$\mathbf{5 * 10^{-5}}$	**True**	**True**	0.9135	**0.9086**	0.0039
UD-4D	Early	VGG16	Adam	Dice	$5 * 10^{-5}$	True	False	0.9095	0.9078	0.0012
UD-4D	Early	VGG16	Adam	Dice	$5 * 10^{-5}$	False	False	0.7956	0.7936	0.0020
UD-4D	**Early**	**VGG16**	**Adam**	**Focal**	$\mathbf{1 * 10^{-4}}$	**True**	**True**	**0.9165**	0.9062	0.0045
UD-4D	Early	VGG16	Adam	Focal	$1 * 10^{-4}$	True	False	0.9162	0.9148	0.0011
UD-4D	Early	VGG16	Adam	Focal	$1 * 10^{-4}$	False	False	0.8066	0.7978	0.0069
UD-4D	Early	VGG16	Adam	Focal	$5 * 10^{-5}$	True	True	0.9119	0.9080	0.0029
UD-4D	Early	VGG16	Adam	Focal	$5 * 10^{-5}$	True	False	0.9096	0.9070	0.0029
UD-4D	Early	VGG16	Adam	Focal	$5 * 10^{-5}$	False	False	0.8169	0.8146	0.0028
UD-Add	Intermediate	VGG16	Adam	Dice	$1 * 10^{-4}$	True	True	0.9229	0.9220	0.0010
UD-Add	Intermediate	VGG16	Adam	Dice	$1 * 10^{-4}$	True	False	0.9259	0.9248	0.0013
UD-Add	Intermediate	VGG16	Adam	Dice	$1 * 10^{-4}$	False	False	0.8971	0.8828	0.0156
UD-Add	Intermediate	VGG16	Adam	Dice	$5 * 10^{-5}$	True	True	0.9113	0.9101	0.0010
UD-Add	Intermediate	VGG16	Adam	Dice	$5 * 10^{-5}$	True	False	0.9148	0.9140	0.0006
UD-Add	Intermediate	VGG16	Adam	Dice	$5 * 10^{-5}$	False	False	0.8913	0.8883	0.0022
UD-Add	**Intermediate**	**VGG16**	**Adam**	**Focal**	$\mathbf{1 * 10^{-4}}$	**True**	**True**	**0.9296**	**0.9291**	0.0006
UD-Add	Intermediate	VGG16	Adam	Focal	$1 * 10^{-4}$	True	False	0.9254	0.9247	0.0006
UD-Add	Intermediate	VGG16	Adam	Focal	$1 * 10^{-4}$	False	False	0.9051	0.9036	0.0012
UD-Add	Intermediate	VGG16	Adam	Focal	$5 * 10^{-5}$	True	True	0.9210	0.9195	0.0013
UD-Add	Intermediate	VGG16	Adam	Focal	$5 * 10^{-5}$	True	False	0.9212	0.9190	0.0018
UD-Add	Intermediate	VGG16	Adam	Focal	$5 * 10^{-5}$	False	False	0.8886	0.8875	0.0008
UD-Add	Late	VGG16	Adam	Dice	$1 * 10^{-4}$	True	True	0.9214	0.9186	0.0022
UD-Add	**Late**	**VGG16**	**Adam**	**Dice**	$\mathbf{1 * 10^{-4}}$	**True**	**False**	**0.9272**	**0.9251**	0.0015
UD-Add	Late	VGG16	Adam	Dice	$1 * 10^{-4}$	False	False	0.8635	0.8624	0.0010
UD-Add	Late	VGG16	Adam	Dice	$5 * 10^{-5}$	True	True	0.9251	0.9223	0.0029
UD-Add	Late	VGG16	Adam	Dice	$5 * 10^{-5}$	True	False	0.9171	0.9162	0.0011
UD-Add	Late	VGG16	Adam	Dice	$5 * 10^{-5}$	False	False	0.8873	0.8855	0.0014
UD-Add	Late	VGG16	Adam	Focal	$1 * 10^{-4}$	True	True	0.9201	0.9196	0.0004
UD-Add	Late	VGG16	Adam	Focal	$1 * 10^{-4}$	True	False	0.9223	0.9207	0.0015
UD-Add	Late	VGG16	Adam	Focal	$1 * 10^{-4}$	False	False	0.8955	0.8907	0.0035
UD-Add	Late	VGG16	Adam	Focal	$5 * 10^{-5}$	True	True	0.9174	0.9159	0.0014
UD-Add	Late	VGG16	Adam	Focal	$5 * 10^{-5}$	True	False	0.9152	0.9140	0.0009
UD-Add	Late	VGG16	Adam	Focal	$5 * 10^{-5}$	True	False	0.9002	0.8982	0.0016
UD-Net	Intermediate	VGG16	Adam	Dice	$1 * 10^{-4}$	True	True	0.8494	0.8472	0.0025
UD-Net	Intermediate	VGG16	Adam	Dice	$1 * 10^{-4}$	True	False	0.8361	0.8313	0.0035
UD-Net	Intermediate	VGG16	Adam	Dice	$1 * 10^{-4}$	True	False	0.7893	0.7746	0.0124
UD-Net	Intermediate	VGG16	Adam	Dice	$5 * 10^{-5}$	True	True	0.8844	0.8687	0.0179
UD-Net	Intermediate	VGG16	Adam	Dice	$5 * 10^{-5}$	True	False	0.8597	0.8554	0.0034
UD-Net	Intermediate	VGG16	Adam	Dice	$5 * 10^{-5}$	True	False	0.7823	0.7734	0.0079
UD-Net	Intermediate	VGG16	Adam	Focal	$1 * 10^{-4}$	True	True	0.9022	0.9017	0.0005
UD-Net	Intermediate	VGG16	Adam	Focal	$1 * 10^{-4}$	True	False	0.8877	0.8834	0.0031
UD-Net	Intermediate	VGG16	Adam	Focal	$1 * 10^{-4}$	False	False	0.8406	0.8247	0.0112
UD-Net	Intermediate	VGG16	Adam	Focal	$5 * 10^{-5}$	True	True	0.8974	0.8940	0.0025
UD-Net	Intermediate	VGG16	Adam	Focal	$5 * 10^{-5}$	True	False	0.8828	0.8732	0.0078
UD-Net	Intermediate	VGG16	Adam	Focal	$5 * 10^{-5}$	False	False	0.8272	0.8151	0.0132
UD-Net	Late	VGG16	Adam	Dice	$1 * 10^{-4}$	True	True	0.9029	0.8996	0.0040
UD-Net	Late	VGG16	Adam	Dice	$1 * 10^{-4}$	True	False	0.9028	0.9008	0.0017
UD-Net	Late	VGG16	Adam	Dice	$1 * 10^{-4}$	False	False	0.7887	0.7849	0.0027
UD-Net	Late	VGG16	Adam	Dice	$5 * 10^{-5}$	True	True	0.9087	0.9041	0.0050
UD-Net	Late	VGG16	Adam	Dice	$5 * 10^{-5}$	True	False	0.8935	0.8916	0.0013
UD-Net	Late	VGG16	Adam	Dice	$5 * 10^{-5}$	False	False	0.7930	0.7852	0.0055
UD-Net	Late	VGG16	Adam	Focal	$1 * 10^{-4}$	True	True	0.9137	0.9121	0.0020
UD-Net	Late	VGG16	Adam	Focal	$1 * 10^{-4}$	True	False	0.9141	0.9121	0.0018
UD-Net	Late	VGG16	Adam	Focal	$1 * 10^{-4}$	False	False	0.8430	0.8369	0.0044
UD-Net	**Late**	**VGG16**	**Adam**	**Focal**	$\mathbf{5 * 10^{-5}}$	**True**	**True**	**0.9143**	**0.9129**	0.0016
UD-Net	Late	VGG16	Adam	Focal	$5 * 10^{-5}$	True	False	0.9073	0.9062	0.0011
UD-Net	Late	VGG16	Adam	Focal	$5 * 10^{-5}$	False	False	0.8457	0.8422	0.0026

[a] : Rows with green text indicate best modeling options for a respective modeling variant regarding $\mathrm{IoU}_{\mathrm{mean}}$ and/or $\mathrm{IoU}_{\mathrm{max}}$

[b] : Initial learning rate, reduced by 20% every 10 epochs

Table A8.5: Training results for the MD-Nets with IoU as the evaluation metric.[a]

Model	Fusion type	Backbone	Optimizer	Loss	Learning rate[b]	Pretrain	Freeze	$\mathrm{IoU}_{\mathrm{max}}$	$\mathrm{IoU}_{\mathrm{mean}}$	$\mathrm{IoU}_{\mathrm{std}}$
MD-4D	**Early**	**ResNet34**	**Adam**	**Default**	$1 * 10^{-4}$	**True**	**True**	**0.9547**	0.9504	0.0038
MD-4D	Early	ResNet34	Adam	Default	$1 * 10^{-4}$	True	False	0.9462	0.9457	0.0006
MD-4D	Early	ResNet34	Adam	Default	$1 * 10^{-4}$	False	False	0.9315	0.9298	0.0015
MD-4D	**Early**	**ResNet34**	**Adam**	**Default**	$5 * 10^{-5}$	**True**	**True**	0.9542	**0.9527**	0.0013
MD-4D	Early	ResNet34	Adam	Default	$5 * 10^{-5}$	True	False	0.9553	0.9480	0.0054
MD-4D	Early	ResNet34	Adam	Default	$5 * 10^{-5}$	False	False	0.9314	0.9314	0.0030
MD-4D	Early	ResNet34	SGD	Default	$1 * 10^{-2}$	True	True	0.9461	0.9421	0.0030
MD-4D	Early	ResNet34	SGD	Default	$1 * 10^{-2}$	True	False	0.9390	0.9340	0.0037
MD-4D	Early	ResNet34	SGD	Default	$1 * 10^{-2}$	False	False	0.9123	0.9100	0.0019
MD-4D	Early	ResNet34	SGD	Default	$5 * 10^{-3}$	True	True	0.9445	0.9328	0.0085
MD-4D	Early	ResNet34	SGD	Default	$5 * 10^{-3}$	True	False	0.9344	0.9086	0.0313
MD-4D	Early	ResNet34	SGD	Default	$5 * 10^{-3}$	False	False	0.9230	0.8641	0.00842
MD-Add	**Intermediate**	**ResNet34**	**Adam**	**Default**	$1 * 10^{-4}$	**True**	**True**	**0.9436**	**0.9410**	0.0021
MD-Add	Intermediate	ResNet34	Adam	Default	$1 * 10^{-4}$	True	False	0.9434	0.9388	0.0033
MD-Add	Intermediate	ResNet34	Adam	Default	$1 * 10^{-4}$	False	False	0.9373	0.9358	0.0018
MD-Add	Intermediate	ResNet34	Adam	Default	$5 * 10^{-5}$	True	True	0.9389	0.9343	0.0044
MD-Add	Intermediate	ResNet34	Adam	Default	$5 * 10^{-5}$	True	False	0.9420	0.9356	0.0069
MD-Add	Intermediate	ResNet34	Adam	Default	$5 * 10^{-5}$	False	False	0.9238	0.9381	0.0044
MD-Add	Intermediate	ResNet34	SGD	Default	$1 * 10^{-2}$	True	True	0.9322	0.9303	0.0014
MD-Add	Intermediate	ResNet34	SGD	Default	$1 * 10^{-2}$	True	False	0.9164	0.8192	0.1339
MD-Add	Intermediate	ResNet34	SGD	Default	$1 * 10^{-2}$	False	False	0.9220	0.9082	0.0177
MD-Add	Intermediate	ResNet34	SGD	Default	$5 * 10^{-3}$	True	True	-	-	-
MD-Add	Intermediate	ResNet34	SGD	Default	$5 * 10^{-3}$	True	False	-	-	-
MD-Add	Intermediate	ResNet34	SGD	Default	$5 * 10^{-3}$	False	False	-	-	-
MD-Net	**Intermediate**	**ResNet34**	**Adam**	**Default**	$1 * 10^{-4}$	**True**	**True**	**0.9400**	**0.9337**	0.0047
MD-Net	Intermediate	ResNet34	Adam	Default	$1 * 10^{-4}$	True	False	0.9259	0.9223	0.0031
MD-Net	Intermediate	ResNet34	Adam	Default	$1 * 10^{-4}$	False	False	0.9278	0.9255	0.0021
MD-Net	Intermediate	ResNet34	Adam	Default	$5 * 10^{-5}$	True	True	0.9339	0.9233	0.0087
MD-Net	Intermediate	ResNet34	Adam	Default	$5 * 10^{-5}$	True	False	0.9374	0.9268	0.0083
MD-Net	Intermediate	ResNet34	Adam	Default	$5 * 10^{-5}$	False	False	0.9337	0.9319	0.0017
MD-Net	Intermediate	ResNet34	SGD	Default	$1 * 10^{-2}$	True	True	0.9206	0.9139	0.0092
MD-Net	Intermediate	ResNet34	SGD	Default	$1 * 10^{-2}$	True	False	0.8361	0.8313	0.0035
MD-Net	Intermediate	ResNet34	SGD	Default	$1 * 10^{-2}$	False	False	0.9016	0.8860	0.0113
MD-Net	Intermediate	ResNet34	SGD	Default	$5 * 10^{-3}$	True	True	0.9325	0.9294	0.0029
MD-Net	Intermediate	ResNet34	SGD	Default	$5 * 10^{-3}$	True	False	0.9045	0.8978	0.0048
MD-Net	Intermediate	ResNet34	SGD	Default	$5 * 10^{-3}$	False	False	0.9249	0.9214	0.0029
MD-Late	**Late**	**ResNet34**	**Adam**	**Default**	$1 * 10^{-4}$	**True**	**True**	**0.9515**	**0.9482**	0.0023
MD-Late	Late	ResNet34	Adam	Default	$1 * 10^{-4}$	True	False	0.9545	0.9485	0.0043
MD-Late	Late	ResNet34	Adam	Default	$1 * 10^{-4}$	False	False	0.9294	0.9279	0.0017
MD-Late	Late	ResNet34	Adam	Default	$5 * 10^{-5}$	True	True	0.9420	0.9390	0.0022
MD-Late	Late	ResNet34	Adam	Default	$5 * 10^{-5}$	True	False	0.9302	0.9293	0.0007
MD-Late	Late	ResNet34	Adam	Default	$5 * 10^{-5}$	False	False	0.9084	0.9024	0.0058
MD-Late	Late	ResNet34	SGD	Default	$1 * 10^{-2}$	True	True	0.9366	0.9232	0.0137
MD-Late	Late	ResNet34	SGD	Default	$1 * 10^{-2}$	True	False	0.9290	0.9257	0.0042
MD-Late	Late	ResNet34	SGD	Default	$1 * 10^{-2}$	False	False	0.9025	0.8260	0.0705
MD-Late	Late	ResNet34	SGD	Default	$5 * 10^{-3}$	True	True	0.9336	0.9316	0.0022
MD-Late	Late	ResNet34	SGD	Default	$5 * 10^{-3}$	True	False	0.9263	0.9232	0.0031
MD-Late	Late	ResNet34	SGD	Default	$5 * 10^{-3}$	False	False	0.8830	0.8768	0.0053

[a] : Rows with green text indicate best modeling options for a respective modeling variant regarding $\mathrm{IoU}_{\mathrm{mean}}$ and/or $\mathrm{IoU}_{\mathrm{max}}$

[b] : Initial learning rate, reduced by 20% every 10 epochs

A9 Evaluation metrics for the semantic segmentation approach using uncertainty-based filtering

The following elaborations are based on Powers (2020).

Precision (P) and Recall (R)
Precision (P) and recall (R) are the two fundamental metrics in classification tasks, focusing on the accuracy (P) and completeness (R) of the positive predictions.

Precision is the ratio of true positive (TP) predictions (correct predictions) to all positive predictions. All positive predictions consist of the sum of all true positive (TP) predictions and false positive (FP) predictions. False positives (FP) denote predictions where a model predicts an outcome to be true where it should be false.

The precision is defined as:

$$P = \frac{TP}{TP + FP} \tag{A9.17}$$

Recall, or sensitivity, measures the ratio of true positive (TP) outcomes to all actual positives. All positives consist of the sum of all true positives (TP) and false negatives (FN). False negatives (FN) denote instances, where the model predicts an outcome to be false, where it should be true. The recall reflects the classifier's ability to identify all relevant instances. It is defined as:

$$R = \frac{TP}{TP + FN} \tag{A9.18}$$

F1 Score
The F1 Score is the harmonic mean of precision (P) and recall (R). It is defined as:

$$F1 = 2\frac{PR}{P + R} \tag{A9.19}$$

A high F1 score indicates both high precision and high recall. It indicates, that the classifier correctly identifies most true positives (TP) while minimizing false positives (FP). A low F1 score suggests that the classifier is performing poorly in terms of either precision (P), recall (R), or both.

False Discovery Rate (FDR)
False Discovery Rate is the proportion of false positives (FP) among all positive predictions. It is defined as:

$$FDR = \frac{FP}{FP + TP} \tag{A9.20}$$

A high FDR indicates that a large proportion of positive predictions are false positives, implying low precision (P). A low FDR means that the classifier has a higher accuracy in predicting positives, making it more reliable for identifying true positive outcomes.

False Positive Rate (FPR) False Positive Rate is the proportion of negative instances that are incorrectly classified as positive. It is defined as:

$$FPR = \frac{FP}{FP + TN} \tag{A9.21}$$

A high FPR indicates that the classifier frequently misclassifies negative instances as positive. This can be problematic in applications where false positives are costly. A low FPR suggests that the classifier is effective at identifying negative instances accurately, minimizing the occurrence of false positives.

A10 Details on the SCP benchmark algorithms used in this work

To solve the VPP with the two benchmark algorithms $\mathbf{SCP_{\Psi}}$ and $\mathbf{SCP_{\Psi,d}}$, proceed as follows:

- First, 900 random positions are sampled in the space around the object to be inspected. The VP then results from the automated orientation to the origin of the inspection object.
- For each VP, a simulation of the acquisition process takes place first, whereby it is determined which surfaces of the inspection object are acquired with the respective VP.
- After simulating the acquisition process for all sampled VP, an iterative selection of the VP to be chosen is then carried out according to the following scheme for the two variants of the $\mathbf{SCP}$ algorithm:
  - $\mathbf{SCP_{\Psi}}$: The VP that acquires the maximum remaining surface area of the inspection object to be acquired based on the acquisition state is selected next. The objective is to choose the VP so that the percentage of newly covered object surface ($\frac{\Delta\Psi_t}{\Psi_{rem,t-1}}$) of the inspection object is maximized.
  - $\mathbf{SCP_{\Psi,d}}$: The VP is selected next, which, based on the acquisition state, maximizes the mean value of the percentage coverage of the object surface still to be acquired ($\frac{\Delta\Psi_t}{\Psi_{rem,t-1}}$) and the quotient ($\frac{d_{VP}}{d_{max}}$) of the travel distance from the last selected VP to the current VP (d_{VP}) to the maximum travel distance of the last VP to any VP in the set of all VP (d_{max}).

Forschungsberichte aus dem wbk
Institut für Produktionstechnik
Karlsruher Institut für Technologie (KIT)

Bisher erschienene Bände:

Band 0
Dr.-Ing. Wu Hong-qi

Adaptive Volumenstromregelung mit Hilfe von drehzahlgeregelten Elektroantrieben

Band 1
Dr.-Ing. Heinrich Weiß

Fräsen mit Schneidkeramik - Verhalten des System Werkzeugmaschine-Werkzeug-Werkstück und Prozessanalyse

Band 2
Dr.-Ing. Hans-Jürgen Stierle

Entwicklung und Untersuchung hydrostatischer Lager für die Axialkolbenmaschine

Band 3
Dr.-Ing. Herbert Hörner

Untersuchung des Geräuschverhaltens druckgeregelter Axialkolbenpumpen

Band 4
Dr.-Ing. Rolf-Dieter Brückbauer

Digitale Drehzahlregelung unter der besonderen Berücksichtigung von Quantisierungseffekten

Band 5
Dr.-Ing. Gerhard Staiger

Graphisch interaktive NC-Programmierung von Drehteilen im Werkstattbereich

Band 6
Dr.-Ing. Karl Peters

Ein Beitrag zur Berechnung und Kompensation von Positionierfehlern an Industrierobotern

Band 7
Dr.-Ing. Paul Stauss

Automatisierte Inbetriebnahme und Sicherung der Zuverlässigkeit und Verfügbarkeit numerisch gesteuerter Fertigungseinrichtungen

Band 8
Dr.-Ing. Günter Möckesch

Konzeption und Realisierung eines strategischen, integrierten Gesamtplanungs- und -bearbeitungssystems zur Optimierung der Drehteilorganisation für auftragsbezogene Drehereien

Band 9
Dr.-Ing. Thomas Oestreicher

Rechnergestützte Projektierung von Steuerungen

Band 10
Dr.-Ing. Thomas Selinger

Teilautomatisierte werkstattnahe NC-Programmerstellung im Umfeld einer integrierten Informationsverarbeitung

Band 11
Dr.-Ing. Thomas Buchholz

Prozessmodell Fräsen, Rechnerunterstützte Analyse, Optimierung und Überwachung

Band 12
Dr.-Ing. Bernhard Reichling

Lasergestützte Positions- und Bahnvermessung von Industrierobotern

Band 13
Dr.-Ing. Hans-Jürgen Lesser

Rechnergestützte Methoden zur Auswahl anforderungsgerechter Verbindungselemente

Band 14
Dr.-Ing. Hans-Jürgen Lauffer

Einsatz von Prozessmodellen zur rechnerunterstützten Auslegung von Räumwerkzeugen

Band 15
Dr.-Ing. Michael C. Wilhelm

Rechnergestützte Prüfplanung im Informationsverbund moderner Produktionssysteme

Band 16
Dr.-Ing. Martin Ochs

Entwurf eines Programmsystems zur wissensbasierten Planung und Konfigurierung

Band 17
Dr.-Ing. Heinz-Joachim Schneider

Erhöhung der Verfügbarkeit von hochautomatisierten Produktionseinrichtungen mit Hilfe der Fertigungsleittechnik

Band 18
Dr.-Ing. Hans-Reiner Ludwig

Beanspruchungsanalyse der Werkzeugschneiden beim Stirnplanfräsen

Band 19
Dr.-Ing. Rudolf Wieser

Methoden zur rechnergestützten Konfigurierung von Fertigungsanlagen

Band 20
Dr.-Ing. Edgar Schmitt

Werkstattsteuerung bei wechselnder Auftragsstruktur

Band 21
Dr.-Ing. Wilhelm Enderle

Verfügbarkeitssteigerung automatisierter Montagesysteme durch selbsttätige Behebung prozessbedingter Störungen

Band 22
Dr.-Ing. Dieter Buchberger

Rechnergestützte Strukturplanung von Produktionssystemen

Band 23
Prof. Dr.-Ing. Jürgen Fleischer

Rechnerunterstützte Technologieplanung für die flexibel automatisierte Fertigung von Abkantteilen

Band 24
Dr.-Ing. Lukas Loeffler

Adaptierbare und adaptive Benutzerschnittstellen

Band 25
Dr.-Ing. Thomas Friedmann

Integration von Produktentwicklung und Montageplanung durch neue rechnergestützte Verfahren

Band 26
Dr.-Ing. Robert Zurrin

Variables Formhonen durch rechnergestützte Hornprozesssteuerung

Band 27
Dr.-Ing. Karl-Heinz Bergen

Langhub-Innenrundhonen von Grauguss und Stahl mit einem elektromechanischem Vorschubsystem

Band 28
Dr.-Ing. Andreas Liebisch

Einflüsse des Festwalzens auf die Eigenspannungsverteilung und die Dauerfestigkeit einsatzgehärteter Zahnräder

Band 29
Dr.-Ing. Rolf Ziegler

Auslegung und Optimierung schneller Servopumpen

Band 30
Dr.-Ing. Rainer Bartl

Datenmodellgestützte Wissensverarbeitung zur Diagnose und Informationsunterstützung in technischen Systemen

Band 31
Dr.-Ing. Ulrich Golz

Analyse, Modellbildung und Optimierung des Betriebsverhaltens von Kugelgewindetrieben

Band 32
Dr.-Ing. Stephan Timmermann

Automatisierung der Feinbearbeitung in der Fertigung von Hohlformwerkzeugen

Band 33
Dr.-Ing. Thomas Noe

Rechnergestützter Wissenserwerb zur Erstellung von Überwachungs- und Diagnoseexpertensystemen für hydraulische Anlagen

Band 34
Dr.-Ing. Ralf Lenschow

Rechnerintegrierte Erstellung und Verifikation von Steuerungsprogrammen als Komponente einer durchgängigen Planungsmethodik

Band 35
Dr.-Ing. Matthias Kallabis

Räumen gehärteter Werkstoffe mit kristallinen Hartstoffen

Band 36
Dr.-Ing. Heiner-Michael Honeck

Rückführung von Fertigungsdaten zur Unterstützung einer fertigungsgerechten Konstruktion

Band 37
Dr.-Ing. Manfred Rohr

Automatisierte Technologieplanung am Beispiel der Komplettbearbeitung auf Dreh-/Fräszellen

Band 38
Dr.-Ing. Martin Steuer

Entwicklung von Softwarewerkzeugen zur wissensbasierten Inbetriebnahme von komplexen Serienmaschinen

Band 39
Dr.-Ing. Siegfried Beichter

Rechnergestützte technische Problemlösung bei der Angebotserstellung von flexiblen Drehzellen

Band 40
Dr.-Ing. Thomas Steitz

Methodik zur marktorientierten Entwicklung von Werkzeugmaschinen mit Integration von funktionsbasierter Strukturierung und Kostenschätzung

Band 41
Dr.-Ing. Michael Richter

Wissensbasierte Projektierung elektrohydraulischer Regelungen

Band 42
Dr.-Ing. Roman Kuhn

Technologieplanungssystem Fräsen. Wissensbasierte Auswahl von Werkzeugen, Schneidkörpern und Schnittbedingungen für das Fertigingsverfahren Fräsen

Band 43
Dr.-Ing. Hubert Klein

Rechnerunterstützte Qualitätssicherung bei der Produktion von Bauteilen mit frei geformten Oberflächen

Band 44
Dr.-Ing. Christian Hoffmann

Konzeption und Realisierung eines fertigungsintegrierten Koordinaten-messgerätes

Band 45
Dr.-Ing. Volker Frey

Planung der Leittechnik für flexible Fertigungsanlagen

Band 46
Dr.-Ing. Achim Feller

Kalkulation in der Angebotsphase mit dem selbsttätig abgeleiteten Erfahrungswissen der Arbeitsplanung

Band 47
Dr.-Ing. Markus Klaiber

Produktivitätssteigerung durch rechnerunterstütztes Einfahren von NC-Programmen

Band 48
Dr.-Ing. Roland Minges

Verbesserung der Genauigkeit beim fünfachsigen Fräsen von Freiformflächen

Band 49
Dr.-Ing. Wolfgang Bernhart

Beitrag zur Bewertung von Montagevarianten: Rechnergestützte Hilfsmittel zur kostenorientierten, parallelen Entwicklung von Produkt und Montage system

Band 50
Dr.-Ing. Peter Ganghoff

Wissensbasierte Unterstützung der Planung technischer Systeme: Konzeption eines Planungswerkzeuges und exemplarische Anwendung im Bereich der Montagesystemplanung

Band 51
Dr.-Ing. Frank Maier

Rechnergestützte Prozessregelung beim flexiblen Gesenkbiegen durch Rückführung von Qualitätsinformationen

Band 52
Dr.-Ing. Frank Debus

Ansatz eines rechnerunterstützten Planungsmanagements für die Planung in verteilten Strukturen

Band 53
Dr.-Ing. Joachim Weinbrecht

Ein Verfahren zur zielorientierten Reaktion auf Planabweichungen in der Werkstattregelung

Band 54
Dr.-Ing. Gerd Herrmann

Reduzierung des Entwicklungsaufwandes für anwendungsspezifische Zellenrechnersoftware durch Rechnerunterstützung

Band 55
Dr.-Ing. Robert Wassmer

Verschleissentwicklung im tribologischen System Fräsen: Beiträge zur Methodik der Prozessmodellierung auf der Basis tribologisher Untersuchungen beim Fräsen

Band 56
Dr.-Ing. Peter Uebelhoer

Inprocess-Geometriemessung beim Honen

Band 57
Dr.-Ing. Hans-Joachim Schelberg

Objektorientierte Projektierung von SPS-Software

Band 58
Dr.-Ing. Klaus Boes

Integration der Qualitätsentwicklung in featurebasierte CAD/CAM-Prozessketten

Band 59
Dr.-Ing. Martin Schreiber

Wirtschaftliche Investitionsbewertung komplexer Produktionssysteme unter Berücksichtigung von Unsicherheit

Band 60
Dr.-Ing. Ralf Steuernagel

Offenes adaptives Engineering-Werkzeug zur automatisierten Erstellung von entscheidungsunterstützenden Informationssystemen

Band 62
Dr.-Ing. Uwe Schauer

Qualitätsorientierte Feinbearbeitung mit Industrierobotern: Regelungsansatz für die Freiformflächenfertigung des Werkzeug- und Formenbaus

Band 63
Dr.-Ing. Simone Loeper

Kennzahlengestütztes Beratungssystem zur Verbesserung der Logistikleistung in der Werkstattfertigung

Band 64
Dr.-Ing. Achim Raab

Räumen mit hartstoffbeschichteten HSS-Werkzeugen

Band 65,
Dr.-Ing. Jan Erik Burghardt

Unterstützung der NC-Verfahrenskette durch ein bearbeitungselementorientiertes, lernfähiges Technologieplanungssystem

Band 66
Dr.-Ing. Christian Tritsch

Flexible Demontage technischer Gebrauchsgüter: Ansatz zur Planung und (teil-)automatisierten Durchführung industrieller Demontageprozesse

Band 67
Dr.-Ing. Oliver Eitrich

Prozessorientiertes Kostenmodell für die entwicklungsbegleitende Vorkalkulation

Band 68
Dr.-Ing. Oliver Wilke

Optimierte Antriebskonzepte für Räummaschinen - Potentiale zur Leistungssteigerung

Band 69
Dr.-Ing. Thilo Sieth

Rechnergestützte Modellierungsmethodik zerspantechnologischer Prozesse

Band 70
Dr.-Ing. Jan Linnenbuerger

Entwicklung neuer Verfahren zur automatisierten Erfassung der geometrischen Abweichungen an Linearachsen und Drehschwenkköpfen

Band 71
Dr.-Ing. Mathias Klimmek

Fraktionierung technischer Produkte mittels eines frei beweglichen Wasserstrahlwerkzeuges

Band 72
Dr.-Ing. Marko Hartel

Kennzahlenbasiertes Bewertungssystem zur Beurteilung der Demontage- und Recyclingeignung von Produkten

Band 73
Dr.-Ing. Jörg Schaupp

Wechselwirkung zwischen der Maschinen- und Hauptspindelantriebsdynamik und dem Zerspanprozess beim Fräsen

Band 74
Dr.-Ing. Bernhard Neisius

Konzeption und Realisierung eines experimentellen Telemanipulators für die Laparoskopie

Band 75
Dr.-Ing. Wolfgang Walter

Erfolgsversprechende Muster für betriebliche Ideenfindungsprozesse. Ein Beitrag zur Steigerung der Innovationsfähigkeit

Band 76
Dr.-Ing. Julian Weber

Ein Ansatz zur Bewertung von Entwicklungsergebnissen in virtuellen Szenarien

Band 77
Dr.-Ing. Dipl. Wirtsch.-Ing. Markus Posur

Unterstützung der Auftragsdurchsetzung in der Fertigung durch Kommunikation über mobile Rechner

Band 78
Dr.-Ing. Frank Fleissner

Prozessorientierte Prüfplanung auf Basis von Bearbeitungsobjekten für die Kleinserienfertigung am Beispiel der Bohr- und Fräsbearbeitung

Band 79
Dr.-Ing. Anton Haberkern

Leistungsfähigere Kugelgewindetriebe durch Beschichtung

Band 80
Dr.-Ing. Dominik Matt

Objektorientierte Prozess- und Strukturinnovation (OPUS)

Band 81
Dr.-Ing. Jürgen Andres

Robotersysteme für den Wohnungsbau: Beitrag zur Automatisierung des Mauerwerkabaus und der Elektroinstallation auf Baustellen

Band 82
Dr.-Ing. Dipl.Wirtschaftsing. Simone Riedmiller

Der Prozesskalender - Eine Methodik zur marktorientierten Entwicklung von Prozessen

Band 83
Dr.-Ing. Dietmar Tilch

Analyse der Geometrieparameter von Präzisionsgewinden auf der Basis einer Least-Squares-Estimation

Band 84
Dr.-Ing. Dipl.-Kfm. Oliver Stiefbold

Konzeption eines reaktionsschnellen Planungssystems für Logistikketten auf Basis von Software-Agenten

Band 85
Dr.-Ing. Ulrich Walter

Einfluss von Kühlschmierstoff auf den Zerspanprozess beim Fräsen: Beitrag zum Prozessverständniss auf Basis von zerspantechnischen Untersuchungen

Band 86
Dr.-Ing. Bernd Werner

Konzeption von teilautonomer Gruppenarbeit unter Berücksichtigung kultureller Einflüsse

Band 87
Dr.-Ing. Ulf Osmers

Projektieren Speicherprogrammierbarer Steuerungen mit Virtual Reality

Band 88
Dr.-Ing. Oliver Doerfel

Optimierung der Zerspantechnik beim Fertigungsverfahren Wälzstossen: Analyse des Potentials zur Trockenbearbeitung

Band 89
Dr.-Ing. Peter Baumgartner

Stufenmethode zur Schnittstellengestaltung in der internationalen Produktion

Band 90
Dr.-Ing. Dirk Vossmann

Wissensmanagement in der Produktentwicklung durch Qualitätsmethodenverbund und Qualitätsmethodenintegration

Band 91
Dr.-Ing. Martin Plass

Beitrag zur Optimierung des Honprozesses durch den Aufbau einer Honprozessregelung

Band 92
Dr.-Ing. Titus Konold

Optimierung der Fünfachsfräsbearbeitung durch eine kennzahlenunterstützte CAM-Umgebung

Band 93
Dr.-Ing. Jürgen Brath

Unterstützung der Produktionsplanung in der Halbleiterfertigung durch risikoberücksichtigende Betriebskennlinien

Band 94
Dr.-Ing. Dirk Geisinger

Ein Konzept zur marktorientierten Produktentwicklung

Band 95
Dr.-Ing. Marco Lanza

Entwurf der Systemunterstützung des verteilten Engineering mit Axiomatic Design

Band 96
Dr.-Ing. Volker Hüntrup

Untersuchungen zur Mikrostrukturierbarkeit von Stählen durch das Fertigungsverfahren Fräsen

Band 97
Dr.-Ing. Frank Reinboth

Interne Stützung zur Genauigkeitsverbesserung in der Inertialmesstechnik: Beitrag zur Senkung der Anforderungen an Inertialsensoren

Band 98
Dr.-Ing. Lutz Trender

Entwicklungsintegrierte Kalkulation von Produktlebenszykluskosten auf Basis der ressourcenorientierten Prozesskostenrechnung

Band 99
Dr.-Ing. Cornelia Kafka

Konzeption und Umsetzung eines Leitfadens zum industriellen Einsatz von Data-Mining

Band 100
Dr.-Ing. Gebhard Selinger

Rechnerunterstützung der informellen Kommunikation in verteilten Unternehmensstrukturen

Band 101
Dr.-Ing. Thomas Windmüller

Verbesserung bestehender Geschäftsprozesse durch eine mitarbeiterorientierte Informationsversorgung

Band 102
Dr.-Ing. Knud Lembke

Theoretische und experimentelle Untersuchung eines bistabilen elektrohydraulischen Linearantriebs

Band 103
Dr.-Ing. Ulrich Thies

Methode zur Unterstützung der variantengerechten Konstruktion von industriell eingesetzten Kleingeräten

Band 104
Dr.-Ing. Andreas Schmälzle

Bewertungssystem für die Generalüberholung von Montageanlagen –Ein Beitrag zur wirtschaftlichen Gestaltung geschlossener Facility- Managment-Systeme im Anlagenbau

Band 105
Dr.-Ing. Thorsten Frank

Vergleichende Untersuchungen schneller elektromechanischer Vorschubachsen mit Kugelgewindetrieb

Band 106
Dr.-Ing. Achim Agostini

Reihenfolgeplanung unter Berücksichtigung von Interaktionen: Beitrag zur ganzheitlichen Strukturierung und Verarbeitung von Interaktionen von Bearbeitungsobjekten

Band 107
Dr.-Ing. Thomas Barrho

Flexible, zeitfenstergesteuerte Auftragseinplanung in segmentierten Fertigungsstrukturen

Band 108
Dr.-Ing. Michael Scharer

Quality Gate-Ansatz mit integriertem Risikomanagement

Band 109
Dr.-Ing. Ulrich Suchy

Entwicklung und Untersuchung eines neuartigen Mischkopfes für das Wasser Abrasivstrahlschneiden

Band 110
Dr.-Ing. Sellal Mussa

Aktive Korrektur von Verlagerungsfehlern in Werkzeugmaschinen

Band 111
Dr.-Ing. Andreas Hühsam

Modellbildung und experimentelle Untersuchung des Wälzschälprozesses

Band 112
Dr.-Ing. Axel Plutowsky

Charakterisierung eines optischen Messsystems und den Bedingungen des Arbeitsraums einer Werkzeugmaschine

Band 113
Dr.-Ing. Robert Landwehr

Konsequent dezentralisierte Steuerung mit Industrial Ethernet und offenen Applikationsprotokollen

Band 114
Dr.-Ing. Christoph Dill

Turbulenzreaktionsprozesse

Band 115
Dr.-Ing. Michael Baumeister

Fabrikplanung im turbulenten Umfeld

Band 116
Dr.-Ing. Christoph Gönnheimer

Konzept zur Verbesserung der Elektromagnetischen Verträglichkeit (EMV) in Produktionssystemen durch intelligente Sensor/Aktor-Anbindung

Band 117
Dr.-Ing. Lutz Demuß

Ein Reifemodell für die Bewertung und Entwicklung von Dienstleistungsorganisationen: Das Service Management Maturity Modell (SMMM)

Band 118
Dr.-Ing. Jörg Söhner

Beitrag zur Simulation zerspanungstechnologischer Vorgänge mit Hilfe der Finite-Element-Methode

Band 119
Dr.-Ing. Judith Elsner

Informationsmanagement für mehrstufige Mikro-Fertigungsprozesse

Band 120
Dr.-Ing. Lijing Xie

Estimation Of Two-dimension Tool Wear Based On Finite Element Method

Band 121
Dr.-Ing. Ansgar Blessing

Geometrischer Entwurf mikromechatronischer Systeme

Band 122
Dr.-Ing. Rainer Ebner

Steigerung der Effizienz mehrachsiger Fräsprozesse durch neue Planungsmethoden mit hoher Benutzerunterstützung

Band 123
Dr.-Ing. Silja Klinkel

Multikriterielle Feinplanung in teilautonomen Produktionsbereichen – Ein Beitrag zur produkt- und prozessorientierten Planung und Steuerung

Band 124
Dr.-Ing. Wolfgang Neithardt

Methodik zur Simulation und Optimierung von Werkzeugmaschinen in der Konzept- und Entwurfsphase auf Basis der Mehrkörpersimulation

Band 125
Dr.-Ing. Andreas Mehr

Hartfeinbearbeitung von Verzahnungen mit kristallinen diamantbeschichteten Werkzeugen beim Fertigungsverfahren Wälzstoßen

Band 126
Dr.-Ing. Martin Gutmann

Entwicklung einer methodischen Vorgehensweise zur Diagnose von hydraulischen Produktionsmaschinen

Band 127
Dr.-Ing. Gisela Lanza

Simulative Anlaufunterstützung auf Basis der Qualitätsfähigkeiten von Produktionsprozessen

Band 128
Dr.-Ing. Ulf Dambacher

Kugelgewindetrieb mit hohem Druckwinkel

Band 129
Dr.-Ing. Carsten Buchholz

Systematische Konzeption und Aufbau einer automatisierten Produktionszelle für pulverspritzgegossene Mikrobauteile

Band 130
Dr.-Ing. Heiner Lang

Trocken-Räumen mit hohen Schnittgeschwindigkeiten

Band 131
Dr.-Ing. Daniel Nesges

Prognose operationeller Verfügbarkeiten von Werkzeugmaschinen unter Berücksichtigung von Serviceleistungen

Im Shaker Verlag erschienene Bände:

Band 132
Dr.-Ing. Andreas Bechle

Beitrag zur prozesssicheren Bearbeitung beim Hochleistungsfertigungsverfahren Wälzschälen

Band 133
Dr.-Ing. Markus Herm

Konfiguration globaler Wertschöpfungsnetzwerke auf Basis von Business Capabilities

Band 134
Dr.-Ing. Hanno Tritschler

Werkzeug- und Zerspanprozessoptimierung beim Hartfräsen von Mikrostrukturen in Stahl

Band 135
Dr.-Ing. Christian Munzinger

**Adaptronische Strebe zur Steifigkeitssteigerung
von Werkzeugmaschinen**

Band 136
Dr.-Ing. Andreas Stepping

**Fabrikplanung im Umfeld von Wertschöpfungsnetzwerken und
ganzheitlichen Produktionssystemen**

Band 137
Dr.-Ing. Martin Dyck

**Beitrag zur Analyse thermische bedingter Werkstückdeformationen
in Trockenbearbeitungsprozessen**

Band 138
Dr.-Ing. Siegfried Schmalzried

**Dreidimensionales optisches Messsystem für eine effizientere
geometrische Maschinenbeurteilung**

Band 139
Dr.-Ing. Marc Wawerla

Risikomanagement von Garantieleistungen

Band 140
Dr.-Ing. Ivesa Buchholz

**Strategien zur Qualitätssicherung mikromechanischer Bauteile
mittels multisensorieller Koordinatenmesstechnik**

Band 141
Dr.-Ing. Jan Kotschenreuther

**Empirische Erweiterung von Modellen der Makrozerspanung
auf den Bereich der Mikrobearbeitung**

Band 142
Dr.-Ing. Andreas Knödel

Adaptronische hydrostatische Drucktascheneinheit

Band 143
Dr.-Ing. Gregor Stengel

**Fliegendes Abtrennen räumlich gekrümmter Strangpressprofile mittels
Industrierobotern**

Band 144
Dr.-Ing. Udo Weismann

Lebenszyklusorientiertes interorganisationelles Anlagencontrolling

Band 145
Dr.-Ing. Rüdiger Pabst

Mathematische Modellierung der Wärmestromdichte zur Simulation des thermischen Bauteilverhaltens bei der Trockenbearbeitung

Band 146
Dr.-Ing. Jan Wieser

Intelligente Instandhaltung zur Verfügbarkeitssteigerung von Werkzeugmaschinen

Band 147
Dr.-Ing. Sebastian Haupt

Effiziente und kostenoptimale Herstellung von Mikrostrukturen durch eine Verfahrenskombination von Bahnerosion und Laserablation

Band 148
Dr.-Ing. Matthias Schlipf

Statistische Prozessregelung von Fertigungs- und Messprozess zur Erreichung einer variabilitätsarmen Produktion mikromechanischer Bauteile

Band 149
Dr.-Ing. Jan Philipp Schmidt-Ewig

Methodische Erarbeitung und Umsetzung eines neuartigen Maschinenkonzeptes zur produktflexiblen Bearbeitung räumlich gekrümmter Strangpressprofile

Band 150
Dr.-Ing. Thomas Ender

Prognose von Personalbedarfen im Produktionsanlauf unter Berücksichtigung dynamischer Planungsgrößen

Band 151
Dr.-Ing. Kathrin Peter

Bewertung und Optimierung der Effektivität von Lean Methoden in der Kleinserienproduktion

Band 152
Dr.-Ing. Matthias Schopp

Sensorbasierte Zustandsdiagnose und -prognose von Kugelgewindetrieben

Band 153
Dr.-Ing. Martin Kipfmüller

Aufwandsoptimierte Simulation von Werkzeugmaschinen

Band 154
Dr.-Ing. Carsten Schmidt

Development of a database to consider multi wear mechanisms within chip forming simulation

Band 155
Dr.-Ing. Stephan Niggeschmidt

Ausfallgerechte Ersatzteilbereitstellung im Maschinen- und Anlagenbau mittels lastabhängiger Lebensdauerprognose

Band 156
Dr.-Ing. Jochen Conrad Peters

Bewertung des Einflusses von Formabweichungen in der Mikro-Koordinatenmesstechnik

Band 157
Dr.-Ing. Jörg Ude

Entscheidungsunterstützung für die Konfiguration globaler Wertschöpfungsnetzwerke

Band 158
Dr.-Ing. Stefan Weiler

Strategien zur wirtschaftlichen Gestaltung der globalen Beschaffung

Band 159
Dr.-Ing. Jan Rühl

Monetäre Flexibilitäts- und Risikobewertung

Band 160
Dr.-Ing. Daniel Ruch

Positions- und Konturerfassung räumlich gekrümmter Profile auf Basis bauteilimmanenter Markierungen

Band 161
Dr.-Ing. Manuel Tröndle

Flexible Zuführung von Mikrobauteilen mit piezoelektrischen Schwingförderern

Band 162
Dr.-Ing. Benjamin Viering

Mikroverzahnungsnormal

Band 163
Dr.-Ing. Chris Becke

Prozesskraftrichtungsangepasste Frässtrategien zur schädigungsarmen Bohrungsbearbeitung an faserverstärkten Kunststoffen

Band 164
Dr.-Ing. Patrick Werner

Dynamische Optimierung und Unsicherheitsbewertung der lastabhängigen präventiven Instandhaltung von Maschinenkomponenten

Band 165
Dr.-Ing. Martin Weis

Kompensation systematischer Fehler bei Werkzeugmaschinen durch self-sensing Aktoren

Band 166
Dr.-Ing. Markus Schneider

Kompensation von Konturabweichungen bei gerundeten Strangpressprofilen durch robotergestützte Führungswerkzeuge

Band 167
Dr.-Ing. Ester M. R. Ruprecht

Prozesskette zur Herstellung schichtbasierter Systeme mit integrierten Kavitäten

Band 168
Dr.-Ing. Alexander Broos

Simulationsgestützte Ermittlung der Komponentenbelastung für die Lebensdauerprognose an Werkzeugmaschinen

Band 169
Dr.-Ing. Frederik Zanger

Segmentspanbildung, Werkzeugverschleiß, Randschichtzustand und Bauteileigenschaften: Numerische Analysen zur Optimierung des Zerspanungsprozesses am Beispiel von Ti-6Al-4V

Band 170
Dr.-Ing. Benjamin Behmann

Servicefähigkeit

Band 171
Dr.-Ing. Annabel Gabriele Jondral

Simulationsgestützte Optimierung und Wirtschaftlichkeitsbewertung des Lean-Methodeneinsatzes

Band 172
Dr.-Ing. Christoph Ruhs

Automatisierte Prozessabfolge zur qualitätssicheren Herstellung von Kavitäten mittels Mikrobahnerosion

Band 173
Dr.-Ing. Steven Peters

Markoffsche Entscheidungsprozesse zur Kapazitäts- und Investitionsplanung von Produktionssystemen

Band 174
Dr.-Ing. Christoph Kühlewein

Untersuchung und Optimierung des Wälzschälverfahrens mit Hilfe von 3D-FEM-Simulation – 3D-FEM Kinematik- und Spanbildungssimulation

Band 175
Dr.-Ing. Adam-Mwanga Dieckmann

Auslegung und Fertigungsprozessgestaltung sintergefügter Verbindungen für µMIM-Bauteile

Band 176
Dr.-Ing. Heiko Hennrich

Aufbau eines kombinierten belastungs- und zustandsorientierten Diagnose- und Prognosesystems für Kugelgewindetriebe

Band 177
Dr.-Ing. Stefan Herder

Piezoelektrischer Self-Sensing-Aktor zur Vorspannungsregelung in adaptronischen Kugelgewindetrieben

Band 178
Dr.-Ing. Alexander Ochs

Ultraschall-Strömungsgreifer für die Handhabung textiler Halbzeuge bei der automatisierten Fertigung von RTM-Bauteilen

Band 179
Dr.-Ing. Jürgen Michna

Numerische und experimentelle Untersuchung zerspanungsbedingter Gefügeumwandlungen und Modellierung des thermo-mechanischen Lastkollektivs beim Bohren von 42CrMo4

Band 180
Dr.-Ing. Jörg Elser

Vorrichtungsfreie räumliche Anordnung von Fügepartnern auf Basis von Bauteilmarkierungen

Band 181
Dr.-Ing. Katharina Klimscha

Einfluss des Fügespalts auf die erreichbare Verbindungsqualität beim Sinterfügen

Band 182
Dr.-Ing. Patricia Weber

Steigerung der Prozesswiederholbarkeit mittels Analyse akustischer Emissionen bei der Mikrolaserablation mit UV-Pikosekundenlasern

Band 183
Dr.-Ing. Jochen Schädel

Automatisiertes Fügen von Tragprofilen mittels Faserwickeln

Band 184
Dr.-Ing. Martin Krauße

Aufwandsoptimierte Simulation von Produktionsanlagen durch Vergrößerung der Geltungsbereiche von Teilmodellen

Band 185
Dr.-Ing. Raphael Moser

Strategische Planung globaler Produktionsnetzwerke
Bestimmung von Wandlungsbedarf und Wandlungszeitpunkt mittels multikriterieller Optimierung

Band 186
Dr.-Ing. Martin Otter

Methode zur Kompensation fertigungsbedingter Gestaltabweichungen für die Montage von Aluminium Space-Frame-Strukturen

Band 187
Dr.-Ing. Urs Leberle

Produktive und flexible Gleitförderung kleiner Bauteile auf phasenflexiblen Schwingförderern mit piezoelektrischen 2D-Antriebselementen

Band 188
Dr.-Ing. Johannes Book

Modellierung und Bewertung von Qualitätsmanagementstrategien in globalen Wertschöpfungsnetzwerken

Band 189
Dr.-Ing. Florian Ambrosy

Optimierung von Zerspanungsprozessen zur prozesssicheren Fertigung nanokristalliner Randschichten am Beispiel von 42CrMo4

Band 190
Dr.-Ing. Adrian Kölmel

Integrierte Messtechnik für Prozessketten unreifer Technologien am Beispiel der Batterieproduktion für Elektrofahrzeuge

Band 191
Dr.-Ing. Henning Wagner

Featurebasierte Technologieplanung zum Preforming von textilen Halbzeugen

Band 192
Dr.-Ing. Johannes Gebhardt

Strukturoptimierung von in FVK eingebetteten metallischen Lasteinleitungselementen

Band 193
Dr.-Ing. Jörg Bauer

Hochintegriertes hydraulisches Vorschubsystem für die Bearbeitung kleiner Werkstücke mit hohen Fertigungsanforderungen

Band 194
Dr.-Ing. Nicole Stricker

Robustheit verketteter Produktionssysteme
Robustheitsevaluation und Selektion des Kennzahlensystems der Robustheit

Band 195
Dr.-Ing. Anna Sauer

Konfiguration von Montagelinien unreifer Produkttechnologien am Beispiel der Batteriemontage für Elektrofahrzeuge

Band 196
Dr.-Ing. Florian Sell-Le Blanc

Prozessmodell für das Linearwickeln unrunder Zahnspulen
Ein Beitrag zur orthozyklischen Spulenwickeltechnik

Band 197
Dr.-Ing. Frederic Förster

Geregeltes Handhabungssystem zum zuverlässigen und energieeffizienten Handling textiler Kohlenstofffaserzuschnitte

Band 198
Dr.-Ing. Nikolay Boev

Numerische Beschreibung von Wechselwirkungen zwischen Zerspanprozess und Maschine am Beispiel Räumen

Band 199
Dr.-Ing. Sebastian Greinacher

Simulationsgestützte Mehrzieloptimierung schlanker und ressourceneffizienter Produktionssysteme

Band 200
Dr.-Ing. Benjamin Häfner

Lebensdauerprognose in Abhängigkeit der Fertigungsabweichungen bei Mikroverzahnungen

Band 201
Dr.-Ing. Stefan Klotz

Dynamische Parameteranpassung bei der Bohrungsherstellung in faserverstärkten Kunststoffen unter zusätzlicher Berücksichtigung der Einspannsituation

Band 202
Dr.-Ing. Johannes Stoll

Bewertung konkurrierender Fertigungsfolgen mittels Kostensimulation und stochastischer Mehrzieloptimierung
Anwendung am Beispiel der Blechpaketfertigung für automobile Elektromotoren

Band 203
Dr.-Ing. Simon-Frederik Koch

Fügen von Metall-Faserverbund-Hybridwellen im Schleuderverfahren
ein Beitrag zur fertigungsgerechten intrinsischen Hybridisierung

Band 204
Dr.-Ing. Julius Ficht

Numerische Untersuchung der Eigenspannungsentwicklung für sequenzielle Zerspanungsprozesse

Band 205
Dr.-Ing. Manuel Baumeister

Automatisierte Fertigung von Einzelblattstapeln in der Lithium-Ionen-Zellproduktion

Band 206
Dr.-Ing. Daniel Bertsch

Optimierung der Werkzeug- und Prozessauslegung für das Wälzschälen von Innenverzahnungen

Band 207
Dr.-Ing. Kyle James Kippenbrock

Deconvolution of Industrial Measurement and Manufacturing Processes for Improved Process Capability Assessments

Band 208
Dr.-Ing. Farboud Bejnoud

Experimentelle Prozesskettenbetrachtung für Räumbauteile am Beispiel einer einsatzgehärteten PKW-Schiebemuffe

Band 209
Dr.-Ing. Steffen Dosch

Herstellungsübergreifende Informationsübertragung zur effizienten Produktion von Werkzeugmaschinen am Beispiel von Kugelgewindetrieben

Band 210
Dr.-Ing. Emanuel Moser

Migrationsplanung globaler Produktionsnetzwerke
Bestimmung robuster Migrationspfade und risiko-effizienter Wandlungsbefähiger

Band 211
Dr.-Ing. Jan Hochdörffer

Integrierte Produktallokationsstrategie und Konfigurationssequenz in globalen Produktionsnetzwerken

Band 212
Dr.-Ing. Tobias Arndt

Bewertung und Steigerung der Prozessqualität in globalen Produktionsnetzwerken

Band 213
Dr.-Ing. Manuel Peter

Unwuchtminimale Montage von Permanentmagnetrotoren durch modellbasierte Online-Optimierung

Band 214
Dr.-Ing. Robin Kopf

Kostenorientierte Planung von Fertigungsfolgen additiver Technologien

Band 215
Dr.-Ing. Harald Meier

Einfluss des Räumens auf den Bauteilzustand in der Prozesskette Weichbearbeitung – Wärmebehandllung – Hartbearbeitung

Band 216
Dr.-Ing. Daniel Brabandt

Qualitätssicherung von textilen Kohlenstofffaser-Preforms mittels optischer Messtechnik

Band 217
Dr.-Ing. Alexandra Schabunow

Einstellung von Aufnahmeparametern mittels projektionsbasierter Qualitätskenngrößen in der industriellen Röntgen-Computertomographie

Band 218
Dr.-Ing. Jens Bürgin

Robuste Auftragsplanung in Produktionsnetzwerken
Mittelfristige Planung der variantenreichen Serienproduktion unter Unsicherheit der Kundenauftragskonfigurationen

Band 219
Dr.-Ing. Michael Gerstenmeyer

Entwicklung und Analyse eines mechanischen Oberflächenbehandlungsverfahrens unter Verwendung des Zerspanungswerkzeuges

Band 220
Dr.-Ing. Jacques Burtscher

Erhöhung der Bearbeitungsstabilität von Werkzeugmaschinen durch semi-passive masseneinstellbare Dämpfungssysteme

Band 221
Dr.-Ing. Dietrich Berger

Qualitätssicherung von textilen Kohlenstofffaser-Preforms mittels prozessintegrierter Wirbelstromsensor-Arrays

Band 222
Dr.-Ing. Fabian Johannes Ballier

Systematic gripper arrangement for a handling device in lightweight production processes

Band 223
Dr.-Ing. Marielouise Schäferling, geb. Zaiß

Development of a Data Fusion-Based Multi-Sensor System for Hybrid Sheet Molding Compound

Band 224
Dr.-Ing. Quirin Spiller

Additive Herstellung von Metallbauteilen mit dem ARBURG Kunststoff-Freiformen

Band 225
Dr.-Ing. Andreas Spohrer

Steigerung der Ressourceneffizienz und Verfügbarkeit von Kugelgewindetrieben durch adaptive Schmierung

Band 226
Dr.-Ing. Johannes Fisel

Veränderungsfähigkeit getakteter Fließmontagesysteme
Planung der Fließbandabstimmung am Beispiel der Automobilmontage

Band 227
Dr.-Ing. Patrick Bollig

Numerische Entwicklung von Strategien zur Kompensation thermisch bedingter Verzüge beim Bohren von 42CrMo4

Band 228
Dr.-Ing. Ramona Pfeiffer, geb. Singer

Untersuchung der prozessbestimmenden Größen für die anforderungsgerechte Gestaltung von Pouchzellen-Verpackungen

Band 229
Dr.-Ing. Florian Baumann

Additive Fertigung von endlosfaserverstärkten Kunststoffen mit dem ARBURG Kunststoff-Freiform Verfahren

Band 230
Dr.-Ing. Tom Stähr

Methodik zur Planung und Konfigurationsauswahl skalierbarer Montagesysteme – Ein Beitrag zur skalierbaren Automatisierung

Band 231
Dr.-Ing. Jan Schwennen

Einbringung und Gestaltung von Lasteinleitungsstrukturen für im RTM-Verfahren hergestellte FVK-Sandwichbauteile

Band 232
Dr.-Ing. Sven Coutandin

Prozessstrategien für das automatisierte Preforming von bebinderten textilen Halbzeugen mit einem segmentierten Werkzeugsystem

Band 233
Dr.-Ing. Christoph Liebrecht

Entscheidungsunterstützung für den Industrie 4.0-Methodeneinsatz
Strukturierung, Bewertung und Ableitung von Implementierungsreihenfolgen

Band 234
Dr.-Ing. Stefan Treber

Transparenzsteigerung in Produktionsnetzwerken
Verbesserung des Störungsmanagements durch verstärkten Informationsaustausch

Band 235
Dr.-Ing. Marius Dackweiler

Modellierung des Fügewickelprozesses zur Herstellung von leichten Fachwerkstrukturen

Band 236
Dr.-Ing. Fabio Echsler Minguillon

Prädiktiv-reaktives Scheduling zur Steigerung der Robustheit in der Matrix-Produktion

Band 237
Dr.-Ing. Sebastian Haag

Entwicklung eines Verfahrensablaufes zur Herstellung von Batteriezellstapeln mit großformatigem, rechteckigem Stapelformat und kontinuierlichen Materialbahnen

Band 238
Dr.-Ing. Raphael Wagner

Strategien zur funktionsorientierten Qualitätsregelung in der Serienproduktion

Band 239
Dr.-Ing. Christopher Ehrmann

Ausfallfrüherkennung von Ritzel-Zahnstangen- Trieben mittels Acoustic Emission

Band 240
Dr.-Ing. Janna Hofmann

Prozessmodellierung des Fünf-Achs-Nadelwickelns zur Implementierung einer trajektoriebasierten Drahtzugkraftregelung

Band 241
Dr.-Ing. Andreas Kuhnle

Adaptive Order Dispatching based on Reinforcement Learning
Application in a Complex Job Shop in the Semiconductor Industry

Band 242
Dr.-Ing. Andreas Greiber

Fertigung optimierter technischer Oberflächen durch eine Verfahrenskombination aus Fliehkraft-Tauchgleitschleifen und Laserablation
Prozesseinflüsse und Prozessauslegung

Band 243
Dr.-Ing. Jan Niclas Eschner

Entwicklung einer akustischen Prozessüberwachung zur Porenbestimmung im Laserstrahlschmelzen

Band 244
Dr.-Ing. Sven Roth

Schädigungsfreie Anbindung von hybriden FVK/Metall-Bauteilen an metallische Tragstrukturen durch Widerstandspunktschweißen

Band 245
Dr.-Ing. Sina Kathrin Peukert

Robustheitssteigerung in Produktionsnetzwerken mithilfe eines integrierten Störungsmanagements

Band 246
Dr.-Ing. Alexander Jacob

Hochiterative Technologieplanung
Rekursive Optimierung produkt- und fertigungsbezogener Freiheitsgrade am Beispiel der hybrid-additiven Fertigung

Band 247
Dr.-Ing. Patrick Moll

Ressourceneffiziente Herstellung von Langfaser-Preforms im Faserblasverfahren

Band 248
Dr.-Ing. Eric Thore Segebade

Erhöhung der Verschleißbeständigkeit von Bauteilen aus Ti-6Al-4V mittels simulationsgestützer Zerspanung und mechanischer Mikrotexturierung

Band 249
Dr.-Ing. Shun Yang

Regionalized implementation strategy of smart automation within assembly systems in China

Band 250
Dr.-Ing. Constantin Carl Hofmann

Vorausschauende und reaktive Mehrzieloptimierung für die Produktionssteuerung einer Matrixproduktion

Band 251
Dr.-Ing. Paul Ruhland

Prozesskette zur Herstellung von hybriden Faser-Metall-Preforms
Modellbildung und Optimierung des Binderauftrags und der Drapierung für stabförmige Bauteile

Band 252
Dr.-Ing. Leonard Schild

Erzeugung und Verwendung von Anwendungswissen in der industriellen Computertomographie

Band 253
Dr.-Ing. Benedikt Klee

Analyse von Phaseninformationen in Videodaten zur Identifikation von Schwingungen in Werkzeugmaschinen

Band 254
Dr.-Ing. Bruno Vargas

Wälzschälen mit kleinen Achskreuzwinkeln
Prozessgrenzen und Umsetzbarkeit

Band 255
Dr.-Ing. Lucas Bretz

Function-oriented in-line quality assurance of hybrid sheet molding compound

Band 256
Dr.-Ing. Bastian Rothaupt

Dämpfung von Bauteilschwingungen durch einstellbare Werkstückdirektspannung mit Hydrodehnspanntechnik

Band 257
Dr.-Ing. Daniel Kupzik

Robotic Swing Folding of three-dimensional UD-tape-based Reinforcement Structures

Band 258
Dr.-Ing. Bastian Verhaelen

(De-)Zentralisierung von Entscheidungen in globalen Produktionsnetzwerken
Strategie- und komplexitätsorientierte Gestaltung der Entscheidungsautonomie

Band 259
Dr.-Ing. Hannes Wilhelm Weinmann

Integration des Vereinzelungs- und Stapelbildungsprozesses in ein flexibel und kontinuierlich arbeitendes Anlagenmodul für die Li-Ionen Batteriezellfertigung

Band 260
Dr.-Ing. Florian Stamer

Dynamische Lieferzeit-Preisgestaltung in variantenreicher Produktion
Ein adaptiver Ansatz mithilfe von Reinforcement Learning

Band 261
Dr.-Ing. Patrick Neuenfeldt

Modellbildung des Tauchgleitschleifens zur Abtrag- und Topografievorhersage an komplexen Geometrien

Band 262
Dr.-Ing. Boris Matuschka

Energieeffizienz in Prozessketten: Analyse und Optimierung von Energieflüssen bei der Herstellung eines PKW-Getriebebauteils aus 16MnCr5

Band 263
Dr.-Ing. Tobias Schlagenhauf

Bildbasierte Quantifizierung und Prognose des Verschleißes an Kugelgewindetriebspindeln
Ein Beitrag zur Zustandsüberwachung von Kugelgewindetrieben mittels Methoden des maschinellen Lernens

Band 264
Dr.-Ing. Benedict Stampfer

Entwicklung eines multimodalen Prozessmodells zur Oberflächenkonditionierung beim Außenlängsdrehen von 42CrMo4

Band 265
Dr.-Ing. Carmen Maria Krahe

KI-gestützte produktionsgerechte Produktentwicklung
Automatisierte Wissensextraktion aus vorhandenen Produktgenerationen

Band 266
Dr.-Ing. Markus Netzer

Intelligente Anomalieerkennung für hochflexible Produktionsmaschinen
Prozessüberwachung in der Brownfield Produktion

Band 267
Dr.-Ing. Simon Raphael Merz

Analyse der Kinematik und Kinetik von Planetenwälzgewindetrieben

Band 268
Dr.-Ing. Rainer Maria Silbernagel

Funktionsorientierte Qualitätsregelung in Produktionsnetzwerken
Qualitätsmanagement in der Produktion hochpräziser Produkte durch netzwerkweite Datenintegration

Band 269
Dr.-Ing. Jonas Nieschlag

Gestaltung und Prozessanalyse für im Schleuderverfahren hergestellte FKV-Metall-Hohlstrukturen

Band 270
Dr.-Ing. Lukas Matthias Weiser

In-Process Porositätserkennung für den PBF-LB/M-Prozess

Band 271
Dr.-Ing. Leonard Vincent Overbeck

Digital Twins of production systems
Automated validation and update of material flow simulation models with real data

Band 272
Dr.-Ing. Felix Klenk

Transparenzsteigerung in der Rückführungslogistik zur Verbesserung der Materialbedarfsplanung für das Remanufacturing

Band 273
Dr.-Ing. Benjamin Bold

Kompensation der Wrinkle-Bildung beim Kalandrieren von Lithium-Ionen-Kathoden
Vom Prozessverständnis des Kalandrierens bis zur Prozessoptimierung mittels Anti-Wrinkle-Modul

Band 274
Dr.-Ing. Daniel Gauder

Adaptive in-line Qualitätsregelung in der Mikro-Verzahnungsfertigung

Band 275
Dr.-Ing. Fabian Sasse

Ontologie-basierte Entscheidungsunterstützung für die Auswahl von Messsystemen in unreifen Produktionsprozessen

Band 276
Dr.-Ing. Jonas Hillenbrand

Unsupervised Condition-Monitoring für Kugelgewindetriebe mittels Acoustic Emission

Band 277
Dr.-Ing. Manuela Neuenfeldt

Untersuchung des Einflusses der PBF-LB-Stellgrößen auf die zerspanende Bearbeitung additiv gefertigter Stahlbauteile

Band 278
Dr.-Ing. Marvin Carl May

Intelligent production control for time-constrained complex job shops

Band 279
Dr.-Ing. Philipp Gönnheimer

Automatisierte Bereitstellung von Maschinensteuerungsdaten in Brownfield-Produktionssystemen
Ein Beitrag zur Digitalisierung von Bestandsanlagen am Beispiel von Werkzeugmaschinen

Band 280
Dr.-Ing. Markus Schäfer

Kollisionsvermeidung für Endeffektoren mit integriertem LiDAR-System in der MRK
Ein Beitrag zur Mensch-Roboter-Kollaboration

Band 281
Dr.-Ing. Oliver Brützel

Decision Support System for the Optimisation of Global Production Networks
Development of a Digital Twin for Product Allocation and Robust Line Configuration

Band 282
Dr.-Ing. Gregor Graf

Qualifizierung der Legierung FeNiCoMoVTiAl im LPBF-Prozess unter Verwendung einer Doppellaser-Belichtungsstrategie

Band 283
Dr.-Ing. Maximilian Torsten Halwas

Kompaktwickelprozess zur Erhöhung der Performance von Statoren elektrischer Traktionsantriebe

Band 284
Dr.-Ing. Magnus Kandler

Menschzentriertes Implementierungsvorgehen für das Digitale Shopfloor Management - Förderung der Selbstorganisation unter Berücksichtigung der Mitarbeiterakzeptanz

Band 285
Dr.-Ing. Michael Baranowski

Additive Herstellung endlosfaserverstärkter Kunststoffbauteile mit dem Laser-Sinterprozess
Maschinentechnik, Prozessentwicklung und -modellierung

Band 286
Dr.-Ing. Tobias Storz

Flexibel automatisierte Assemblierung von Li-Ionen-Pouchzellen
Agile Anlagentechnik für die Prozesskette Stapelbildung, Kontaktierung und Heißsiegeln

Band 287
Dr.-Ing. Nikolas Sven Matkovic

Additive Individualization of Continuous-Discontinuous Reinforced Thermoplastics

Band 288
Dr.-Ing. Marco Wurster

Planung und Steuerung agiler hybrider Demontagesysteme im Remanufacturing

Band 289
Dr.-Ing. Felix Johannes Wirth

Prozessgeregelte Formgebung von Hairpin-Steckspulen für elektrische Traktionsmotoren

Band 290
Dr.-Ing. Patrizia Konstanze Gartner

Konzept eines Selbstheilungsmechanismus für Polymerelektrolytmembranen
Optimierung der Lebensdauer und der Effizienz von Brennstoffzellen

Band 291
Dr.-Ing. Jens Schäfer

Funktionsintegriertes Handhabungssystem zur geometrieflexiblen, positionsgenauen Einzellagenstapelung in der Brennstoffzellenstackfertigung

Band 292
Dr.-Ing. Gwen Louis Steier

Strategischer Fit in globalen Produktionsnetzwerken
Entscheidungsunterstützung für die strategische Netzwerkkonfiguration

Band 293
Dr.-Ing. Louis Schäfer

Assistierte, modellbasierte Grobplanung von Produktionssystemen mittels Mehrzieloptimierung:
Anwendung am Beispiel hochautomatisierter Schweißlinien für die Automobilzuliefererindustrie

Band 294
Dr.-Ing. Jan-Philipp Kaiser

Autonomous View Planning using Reinforcement Learning
Modeling and Application for Visual Inspection in Remanufacturing