Universidad de Oviedo

Departamento de Ingeniería Eléctrica, Electrónica, de Computadores y Sistemas

Programa de Doctorado en Energía y Control de Procesos

ROBOTIC SYSTEM AND PROFILOMETRIC SENSORS INTEGRATION FOR SURFACE DEFECTS DETECTION IN PRODUCTION LINES

Doctorando:

Sara Roos Hoefgeest Toribio

Directores:

Dr. Ignacio Álvarez García

Dr. Rafael C. González de los Reyes

Oviedo, 29 de Agosto 2024

Universidad de Oviedo

Departamento de Ingeniería Eléctrica, Electrónica, de Computadores y Sistemas

Programa de Doctorado en Energía y Control de Procesos

INTEGRACIÓN DE SISTEMAS ROBÓTICOS Y SENSORES PERFILOMÉTRICOS PARA DETECCIÓN DE DEFECTOS SUPERFICIALES EN LÍNEAS DE PRODUCCIÓN

Doctorando:

Sara Roos Hoefgeest Toribio

Oviedo, 29 de Agosto 2024

Universidad de Oviedo

Departamento de Ingeniería Eléctrica, Electrónica, de Computadores y Sistemas

Programa de Doctorado en Energía y Control de Procesos

INTEGRACIÓN DE SISTEMAS ROBÓTICOS Y SENSORES PERFILOMÉTRICOS PARA DETECCIÓN DE DEFECTOS SUPERFICIALES EN LÍNEAS DE PRODUCCIÓN

Doctorando:

Sara Roos Hoefgeest Toribio

Directores:

Dr. Ignacio Álvarez García

Dr. Rafael C. González de los Reyes

Oviedo, 29 de Agosto 2024

Universidad de Oviedo

RESUMEN DEL CONTENIDO DE TESIS DOCTORAL

1.- Título de la Tesis	
Español/Otro Idioma: Integración de sistemas robóticos y sensores perfilométricos para detección de defectos superficiales en líneas de producción	Inglés: Robotic system and profilometric sensors integration for surface defects detection in production lines

2.- Autor	
Nombre: Sara Roos Hoefgeest Toribio	
Programa de Doctorado: Energía y Control de Procesos	
Órgano responsable: Universidad de Oviedo	

RESUMEN (en español)

La investigación se enmarca en el contexto de la Industria 4.0, donde las industrias están digitalizando y optimizando sus procesos productivos para mejorar la calidad, la eficiencia y la seguridad de los productos fabricados. Esta revolución industrial impulsa a las empresas a adoptar tecnologías avanzadas que integran la automatización, la interconexión de dispositivos y la digitalización de datos en toda la cadena de producción.

En este escenario, la inspección superficial de productos industriales juega un papel crucial. Tradicionalmente, las técnicas de medición con contacto eran comunes para obtener medidas precisas, pero estas presentaban limitaciones en cuanto a velocidad y adaptabilidad a diferentes formas y materiales. La evolución hacia sistemas de inspección sin contacto ha permitido superar estas limitaciones al eliminar la necesidad de contacto físico directo entre la pieza evaluada y el sistema de medición. Estos sistemas utilizan tecnologías como la visión artificial y las técnicas interferométricas para obtener mediciones precisas en 3D y detectar defectos superficiales.

Los avances en métodos de inspección 3D, como la triangulación láser, han mejorado la detección de defectos complejos en superficies, capturando variaciones sutiles que son difíciles de identificar en 2D. Los sensores de alta resolución, capaces de detectar defectos tan pequeños como unas decenas de micras, se utilizan cada vez más para mantener los estándares de calidad identificando y corrigiendo defectos microscópicos.

Sin embargo, la detección precisa de defectos mediante sensores de triangulación láser presenta desafíos significativos. El escaneo de una pieza para la inspección sin contacto suele depender del movimiento relativo entre el sensor y la pieza. Las pequeñas variaciones pueden introducir errores importantes, especialmente cuando se trata de defectos micrométricos. Por lo tanto, la integración de la robótica con sensores perfilométricos es crucial, aunque configurar estos sistemas de manera eficiente suele ser costoso y lento. Es esencial encontrar soluciones que mejoren la precisión de las mediciones y optimicen los procesos de inspección.

Para enfrentar estos desafíos, esta investigación propone un *framework* que aborda múltiples aspectos críticos de la inspección industrial. Este *framework* se centra en la simulación precisa de escaneos utilizando sensores perfilométricos, la evaluación de la detección de defectos y la generación de trayectorias óptimas de inspección. Hasta la fecha, no se ha identificado ninguna herramienta en la literatura científica que cumpla con estos requisitos específicos en el ámbito de los sensores perfilométricos, lo que resalta una clara oportunidad para avanzar en este campo.

El objetivo principal de esta investigación es mejorar la calidad y precisión en la detección de defectos superficiales. Para lograrlo, se desarrolló un simulador de inspección que replica

fielmente las condiciones del mundo real. Se utilizan modelos CAD para simular escaneos realistas, considerando los parámetros operativos del sensor y simulando ruidos como el *speckle*.

En este contexto, se exploró la simulación de defectos superficiales en modelos CAD para evaluar las capacidades de detección de los sistemas de inspección. Los defectos pueden incorporarse con precisión deformando el modelo según un mapa de elevación, y se propusieron modelos matemáticos específicos para simular abolladuras, picos y grietas. Este método permite la creación de bases de datos de escaneos 3D realistas y variados, esenciales para entrenar y validar algoritmos de detección avanzados, optimizar trayectorias de escaneo y fomentar la innovación en la inspección industrial.

La investigación también exploró el uso del Aprendizaje por Refuerzo para generar trayectorias de escaneo optimizadas. Este método ajusta dinámicamente la posición y orientación del sensor durante el escaneo para maximizar la cobertura y minimizar los errores, adaptándose a la geometría de cada producto. Este proceso se realiza utilizando el simulador propuesto, lo que permite validar y optimizar las trayectorias de escaneo, asegurando una inspección efectiva y eficiente en entornos industriales.

En conjunto, esta investigación tiene como objetivo reducir costos y tiempos de desarrollo, mejorando significativamente la precisión y eficiencia de los sistemas de inspección industrial, contribuyendo a la competitividad de la Industria 4.0 mediante el aprovechamiento de tecnologías digitales y automatizadas.

RESUMEN (en Inglés)

The research is set within the context of Industry 4.0, where industries are digitizing and optimizing their production processes to enhance the quality, efficiency, and safety of manufactured products. This industrial revolution drives companies to adopt advanced technologies integrating automation, device interconnection, and data digitization across the production chain.

In this scenario, the surface inspection of industrial products plays a crucial role. Traditionally, contact measurement techniques were common for obtaining precise measurements. However, these methods have limitations regarding speed and adaptability to different shapes and materials. Non-contact inspection systems have emerged to overcome these limitations, using computer vision and interferometric techniques to provide accurate 3D measurements and detect surface defects without physical contact.

Advancements in 3D inspection methods, such as laser triangulation, have enhanced the detection of complex surface defects, capturing subtle variations that are difficult to identify in 2D. High-resolution sensors, capable of detecting defects as small as tens of microns, are increasingly used to maintain quality standards by identifying and correcting microscopic defects.

However, accurately detecting defects using laser triangulation sensors is challenging. Typically, scanning a piece for non-contact inspection relies on the relative movement between the sensor and the piece. Small variations can introduce significant errors, especially when dealing with micrometric defects. Therefore, integrating robotics with profilometric sensors is crucial, but configuring these systems efficiently is often costly and time-consuming. Finding solutions that enhance measurement accuracy and optimize inspection processes is vital for modern industrial applications.

In response to these challenges, this research proposes a comprehensive framework that addresses multiple critical aspects of industrial inspection. This framework focuses on the precise simulation of scans using profilometric sensors, evaluating defect detection, and generating optimal inspection trajectories. To date, no tool has been found in the scientific literature that meets these specific requirements, especially in the realm of profilometric sensors, highlighting a clear opportunity to advance in this field.

Universidad de Oviedo

The main goal of this research is to improve the quality and accuracy of surface defect detection. To achieve this, an inspection simulator was developed that replicates real-world conditions. CAD models are used to simulate realistic scans, considering the sensor's operational parameters and simulating noise such as speckle.

In this context, the simulation of surface defects on CAD models was examined to assess the detection capabilities of inspection systems. Defects can be accurately incorporated by deforming the model based on an elevation map, with specific mathematical models proposed for simulating dents, peaks, and cracks. This method enables the creation of realistic and varied 3D scan databases, which are essential for training and validating advanced detection algorithms, optimizing scanning paths, and driving innovation in industrial inspection.

The research also explored the use of Reinforcement Learning to generate optimized scanning trajectories. This method dynamically adjusts the sensor’s position and orientation during scanning to maximize coverage and minimize errors, adapting to the geometry of each product. This process is conducted using the proposed simulator, allowing for validation and optimization of scan trajectories, ensuring effective and efficient inspection in industrial settings.

Overall, this research aims to reduce costs and development time while significantly improving the precision and efficiency of industrial inspection systems, contributing to the competitiveness of Industry 4.0 by leveraging digital and automated technologies.

SR. PRESIDENTE DE LA COMISIÓN ACADÉMICA DEL PROGRAMA DE DOCTORADO EN _____

Agradecimientos

Me gustaría dar las gracias a todas las personas e instituciones que hicieron posible esta tesis doctoral. Sin su colaboración y apoyo, este trabajo no habría sido posible. Especialmente, gracias:

- A mis directores de tesis, Rafael Corsino González de los Reyes e Ignacio Álvarez García, por su apoyo y motivación durante estos años y por darme la oportunidad de sumergirme en el mundo de la investigación.
- Al Departamento de Ingeniería Eléctrica, Electrónica, de Computadores y Sistemas de la Universidad de Oviedo, del cual he formado parte estos años y que me ha brindado la oportunidad de iniciar mi carrera docente.
- A los miembros del grupo PRISMA de la Universidad Federico II de Napoli, por acogerme durante un período de 3 meses y darme la oportunidad de colaborar en sus investigaciones.
- A CIN Advanced Systems, por su colaboración y por proporcionar los datos utilizados en los experimentos.
- Al Gobierno del Principado de Asturias, que a través de la ayuda predoctoral Severo Ochoa (PA-20-PF-BP19-067) me ha proporcionado el apoyo económico necesario para la realización de la presente tesis.

Acknowledgments

I would like to express my gratitude to all the individuals and institutions who made this doctoral thesis possible. Without their collaboration and support, this work would not have been achievable. Special thanks to:

- My thesis advisors, Rafael Corsino González de los Reyes and Ignacio Álvarez García, for their support and motivation throughout these years and for giving me the opportunity to delve into the world of research.
- The Department of Electrical, Electronic, Computer, and Systems Engineering at the University of Oviedo, where I have been a part of during these years and which has given me the opportunity to begin my teaching career.
- The members of the PRISMA group at the University of Naples Federico II, for welcoming me during a 3-month period and allowing me to collaborate on their research projects.
- A CIN Advanced Systems, for their collaboration and for providing the data used in the experiments.
- The Government of the Principality of Asturias, which, through the Severo Ochoa predoctoral grant (PA-20-PF-BP19-067), provided the financial support necessary for the completion of this thesis.

Resumen

La investigación se enmarca en el contexto de la Industria 4.0, donde las industrias están digitalizando y optimizando sus procesos productivos para mejorar la calidad, la eficiencia y la seguridad de los productos fabricados. Esta revolución industrial impulsa a las empresas a adoptar tecnologías avanzadas que integran la automatización, la interconexión de dispositivos y la digitalización de datos en toda la cadena de producción.

En este escenario, la inspección superficial de productos industriales juega un papel crucial. Tradicionalmente, las técnicas de medición con contacto eran comunes para obtener medidas precisas, pero estas presentaban limitaciones en cuanto a velocidad y adaptabilidad a diferentes formas y materiales. La evolución hacia sistemas de inspección sin contacto ha permitido superar estas limitaciones al eliminar la necesidad de contacto físico directo entre la pieza evaluada y el sistema de medición. Estos sistemas utilizan tecnologías como la visión artificial y las técnicas interferométricas para obtener mediciones precisas en 3D y detectar defectos superficiales.

Los avances en métodos de inspección 3D, como la triangulación láser, han mejorado la detección de defectos complejos en superficies, capturando variaciones sutiles que son difíciles de identificar en 2D. Los sensores de alta resolución, capaces de detectar defectos tan pequeños como unas decenas de micras, se utilizan cada vez más para mantener los estándares de calidad identificando y corrigiendo defectos microscópicos.

Sin embargo, la detección precisa de defectos mediante sensores de triangulación láser presenta desafíos significativos. El escaneo de una pieza para la inspección sin contacto suele depender del movimiento relativo entre el sensor y la pieza. Las pequeñas variaciones pueden introducir errores importantes, especialmente cuando se trata de defectos micrométricos. Por lo tanto, la integración de la robótica con sensores perfilométricos es crucial, aunque configurar estos sistemas de manera efi-

ciente suele ser costoso y lento. Es esencial encontrar soluciones que mejoren la precisión de las mediciones y optimicen los procesos de inspección.

Para enfrentar estos desafíos, esta investigación propone un *framework* que aborda múltiples aspectos críticos de la inspección industrial. Este *framework* se centra en la simulación precisa de escaneos utilizando sensores perfilométricos, la evaluación de la detección de defectos y la generación de trayectorias óptimas de inspección. Hasta la fecha, no se ha identificado ninguna herramienta en la literatura científica que cumpla con estos requisitos específicos en el ámbito de los sensores perfilométricos, lo que resalta una clara oportunidad para avanzar en este campo.

El objetivo principal de esta investigación es mejorar la calidad y precisión en la detección de defectos superficiales. Para lograrlo, se desarrolló un simulador de inspección que replica fielmente las condiciones del mundo real. Se utilizan modelos CAD para simular escaneos realistas, considerando los parámetros operativos del sensor y simulando ruidos como el *speckle*.

En este contexto, se exploró la simulación de defectos superficiales en modelos CAD para evaluar las capacidades de detección de los sistemas de inspección. Los defectos pueden incorporarse con precisión deformando el modelo según un mapa de elevación, y se propusieron modelos matemáticos específicos para simular abolladuras, picos y grietas. Este método permite la creación de bases de datos de escaneos 3D realistas y variados, esenciales para entrenar y validar algoritmos de detección avanzados, optimizar trayectorias de escaneo y fomentar la innovación en la inspección industrial.

La investigación también exploró el uso del Aprendizaje por Refuerzo para generar trayectorias de escaneo optimizadas. Este método ajusta dinámicamente la posición y orientación del sensor durante el escaneo para maximizar la cobertura y minimizar los errores, adaptándose a la geometría de cada producto. Este proceso se realiza utilizando el simulador propuesto, lo que permite validar y optimizar las trayectorias de escaneo, asegurando una inspección efectiva y eficiente en entornos industriales.

En conjunto, esta investigación tiene como objetivo reducir costos y tiempos de desarrollo, mejorando significativamente la precisión y eficiencia de los sistemas de inspección industrial, contribuyendo a la competitividad de la Industria 4.0 mediante el aprovechamiento de tecnologías digitales y automatizadas.

Abstract

The research is set within the context of Industry 4.0, where industries are digitizing and optimizing their production processes to enhance the quality, efficiency, and safety of manufactured products. This industrial revolution drives companies to adopt advanced technologies integrating automation, device interconnection, and data digitization across the production chain.

In this scenario, the surface inspection of industrial products plays a crucial role. Traditionally, contact measurement techniques were common for obtaining precise measurements. However, these methods have limitations regarding speed and adaptability to different shapes and materials. Non-contact inspection systems have emerged to overcome these limitations, using computer vision and interferometric techniques to provide accurate 3D measurements and detect surface defects without physical contact.

Advancements in 3D inspection methods, such as laser triangulation, have enhanced the detection of complex surface defects, capturing subtle variations that are difficult to identify in 2D. High-resolution sensors, capable of detecting defects as small as tens of microns, are increasingly used to maintain quality standards by identifying and correcting microscopic defects.

However, accurately detecting defects using laser triangulation sensors is challenging. Typically, scanning a piece for non-contact inspection relies on the relative movement between the sensor and the piece. Small variations can introduce significant errors, especially when dealing with micrometric defects. Therefore, integrating robotics with profilometric sensors is crucial, but configuring these systems efficiently is often costly and time-consuming. Finding solutions that enhance measurement accuracy and optimize inspection processes is vital for modern industrial applications.

In response to these challenges, this research proposes a comprehensive framework that addresses multiple critical aspects of industrial inspection. This framework

focuses on the precise simulation of scans using profilometric sensors, evaluating defect detection, and generating optimal inspection trajectories. To date, no tool has been found in the scientific literature that meets these specific requirements, especially in the realm of profilometric sensors, highlighting a clear opportunity to advance in this field.

The main goal of this research is to improve the quality and accuracy of surface defect detection. To achieve this, an inspection simulator was developed that replicates real-world conditions. CAD models are used to simulate realistic scans, considering the sensor's operational parameters and simulating noise such as speckle.

In this context, the simulation of surface defects on CAD models was examined to assess the detection capabilities of inspection systems. Defects can be accurately incorporated by deforming the model based on an elevation map, with specific mathematical models proposed for simulating dents, peaks, and cracks. This method enables the creation of realistic and varied 3D scan databases, which are essential for training and validating advanced detection algorithms, optimizing scanning paths, and driving innovation in industrial inspection.

The research also explored the use of Reinforcement Learning to generate optimized scanning trajectories. This method dynamically adjusts the sensor's position and orientation during scanning to maximize coverage and minimize errors, adapting to the geometry of each product. This process is conducted using the proposed simulator, allowing for validation and optimization of scan trajectories, ensuring effective and efficient inspection in industrial settings.

Overall, this research aims to reduce costs and development time while significantly improving the precision and efficiency of industrial inspection systems, contributing to the competitiveness of Industry 4.0 by leveraging digital and automated technologies.

Contents

List of figures **12**

List of tables **21**

1 Introduction **24**

1.1 Motivation of the thesis 24

1.2 Problem Statement 29

1.3 Objectives 30

1.4 Structure of the document 32

2 Measurement Techniques for Surface Inspection **34**

2.1 Contact Measurement Systems 34

2.2 Non-Contact Measurement Systems 36

2.2.1 Time of flight 37

2.2.2 Photogrammetry 38

2.2.3 Structured light 40

2.2.4 Confocal Microscopy 41

2.2.5 Interferometry 42
2.2.6 Conoscopic Holography 43
2.3 Laser Triangulation 44
2.3.1 Sensor Parameters 46
2.3.2 Challenges in Light-Surface Interactions 48
2.3.3 Surface Inspection Strategies 50

3 Reinforcement Learning: Theoretical Foundations 53

3.1 Basic principles 53
3.1.1 States 55
3.1.2 Actions 55
3.1.3 Reward and Return 55
3.1.4 Policy 56
3.1.5 The RL Problem 57
3.1.6 Value Functions 58
3.1.7 Bellman Equations 58
3.1.8 Advantage Functions 59
3.2 RL Algorithms 59
3.2.1 Q-Learning 61
3.2.2 SARSA 62
3.2.3 Deep Q-Network 63
3.2.4 Policy Gradient Methods 64
3.2.5 Trust Region Policy Optimization (TRPO) 65

3.2.6 Proximal Policy Optimization (PPO) . . . 65
3.2.7 Deep Deterministic Policy Gradient (DDPG) . . . 67
3.2.8 TD3 . . . 69
3.2.9 SAC . . . 71
3.2.10 A2C . . . 72

4 Simulation of a Laser Triangulation Profilometric Sensor 74

4.1 Related work . . . 75
4.2 Proposed Method . . . 80
4.2.1 Geometrical Model . . . 81
4.2.2 Speckle Noise Model: Perlin Noise . . . 84
4.2.3 Sensor Uncertainty Model: Gaussian Noise . . . 91
4.3 Results . . . 93
4.3.1 Geometrical Model . . . 93
4.3.2 Noise Model . . . 96
4.4 Discussion . . . 106

5 Simulation of surface defects in 3D models 109

5.1 Related work . . . 110
5.2 3D model deformation using FFD for surface defect generation . . . 115
5.2.1 Mesh remeshing . . . 116
5.2.2 Generalities of Free-form deformation (FFD) using NURBS parametrization . . . 118

5.2.3 Lattice and NURBS parametrization for surface defect simulation . . . 120
5.3 Mathematical model of defects . . . 126
5.3.1 Crack model . . . 127
5.3.2 Bump model . . . 134
5.3.3 Peak model . . . 135
5.4 Resultados . . . 136
5.4.1 Comparison of Simulated and Real Defects . . . 138
5.4.2 Machine Learning-based defect detection . . . 142
5.5 Discussion . . . 149

6 Reinforcement Learning-Based Inspection Path Planning 152

6.1 Related work . . . 153
6.2 Characteristics of surface inspection scanning with profilometric sensors 158
6.3 Proposed Method . . . 162
6.3.1 Simulated Environment . . . 164
6.3.2 State Space . . . 166
6.3.3 Action Space . . . 166
6.3.4 Reward Function . . . 168
6.3.5 RL Algorithm Configuration . . . 172
6.4 Experiments and results . . . 174
6.4.1 Inspection system configuration . . . 175
6.4.2 Training process . . . 176

6.4.3 Car door . . . 180
6.4.4 Pen Holder . . . 191
6.4.5 Parrot Drone . . . 200
6.5 Discussion . . . 209

7 Conclusions and Future Work **212**
7.1 Discussion and Final Conclusions . . . 212
7.2 Thesis Contributions . . . 216
7.3 Future work . . . 218

8 Conclusiones y trabajo futuro **221**
8.1 Discusion y Conclusiones Finales . . . 221
8.2 Contribuciones de la Tesis . . . 225
8.3 Trabajo Futuro . . . 228

A Publications **230**
A.1 Publications in journals . . . 231
A.2 Publications in Congress . . . 233

References **238**

List of Figures

1.1 Examples of surface defects on cast-iron (a) and stamped parts (b). It can be seen that the inspected parts may present complex geometries and that defects may also appear on high curvatura areas of the surface. 25

2.1 Coordinate Measuring Machine. Image extracted from [33] 35

2.2 Principle of operation of Time of Flight (ToF) systems 38

2.3 Principle of operation of Photogrammetry 39

2.4 Principle of operation of Structured Light systems 40

2.5 Principle of operation of Confocal Microscopy systems 41

2.6 Principle of operation of interferometry systems 43

2.7 Principle of operation of Conoscopic Holography 44

2.8 Operating principle of laser triangulation. 46

2.9 Common parameters of a laser triangulation profilometry sensor. . . . 48

2.10 Image of a laser spot over a rough surface. The localization of the center cannot be done without certain uncertainty, due to the speckle effect. Image obtained from [35] 50

2.11 Comparison of different scanning techniques: (a) Linear scan, (b) Area scan. 51

2.12 The triangulation angle and occlusion. 52

3.1 Agent-Environment Interaction Cycle 54

4.1 Laser triangulation sensor: Key parameters scheme. 82

4.2 Perlin noise image. 85

4.3 2D Perlin noise generation process. Left image shows the 2D grid with the pseudo-random gradient vectors for each control point. Right image represents the process in one input coordinate $P(x, y)$. $C(x_i, y_j)$ are the corners of the grid, blue arrows represent the displacement vectors $\vec{d}(x_i, y_j)$ and orange arrows the gradient vectors. 86

4.4 Common roughness parameters in a profile. P_i are all the peaks in the profile. 88

4.5 Interface of the developed simulator. The CAD model of a loaded car door and a scan path between the marked points P_0 and P_1 is shown. 95

4.6 Result obtained during the simulation of a scan of the section of the CAD model shown in figure 4.5. (a) Result in point cloud form (b) Result as 2D image. Each row corresponds to a scan profile, so the size of the image is determined by the total number of profiles scanned during the acquisition and the number of points per profile. (Size: 2000x4096) 95

4.7 Noise simulation results. Images on the left show the complete profile and those on the right show a zoom of the section outlined in orange. (a) y (b) Simulation profile before adding any noise. (c) y (d) Simulation profile after Gaussian noise to simulate sensor uncertainty. (e) y (f) Simulation profile after Perlin and Gaussian noise. 97

4.8 (a) Bearing cap. Image obtained from [71] (b) Car door (c) Heavy steel plate 98

4.9 Bearing cap experiment: Comparison between real and simulated scan. (a) Real 2D scan image. (b) Simulated 2D scan image. (c) Real scan profile. (d) Simulated scan profile. 100

4.10 Comparison of distributions. (a) and (b) Normalized histogram of each of the captures (real and simulated) in the bearing cap experiment. (c) Comparison of the CFD's of the captures. 102

4.11 Car door experiment: Comparison between real and simulated scan. (a) Real 2D scan image. (b) Simulated 2D scan image. (c) Real scan profile. (d) Simulated scan profile. 102

4.12 Comparison of distributions. (a) and (b) Normalized histogram of each of the captures (real and simulated) in the carDoor experiment. (c) Comparison of the CFD's of the captures. 104

4.13 Plate experiment: Comparison between real and simulated scan. (a) Real 2D scan image. (b) Simulated 2D scan image. (c) Real scan profile. (d) Simulated scan profile. 104

4.14 Comparison of distributions. (a) and (b) Normalized histogram of each of the captures (real and simulated) in the heavy steel plate experiment. (c) Comparison of the CFD's of the captures. 106

5.1 Scheme of the process for adding a modeled defect to a CAD model of any product 116

5.2 Remeshing of the selected area to be deformed according to the required defect. (a) Before remeshing: The number of triangles is 370. (b) After remeshing: The number of triangles is 123108 117

5.3 Illustration of the lattice and control points displacement process to deform a surface. The lattice structure is shown in the u, v, w space, with initial control points depicted in red and new control points highlighted in blue 120

5.4 Representation of the normal vectors of the faces only in the selected area, indicated by blue triangles. (a) Simple surface. Normals have practically the same direction in the selected area. (b) Complex surface. Normal direction changes over the selected surface 121

5.5 (a) Construction of the lattice around the 3D model area. (b) Result of the lattice after displacement of the control nodes according to the defect. (c) Result of applying the deformation of the lattice to the 3D model. Final defect 123

5.6 (a) Construction of the lattice with the complex surface fitting into it. (b) Cross-section of the complex surface. It shows a 2D representation of displacements of the lattice control points to get the defect shape in complex surfaces. In red: control points before displacement. In blue: control points after displacement. (c) Result of applying the deformation of the lattice to the 3D model. Final defect 125

5.7 Images of some common defects in steel products. (a) Bump. (b) Peak. (c)(d) Cracks 127

5.8 Model of the crack medial axis. First, a polygonal approximation is defined using a set of vertices (blue line with square markers). Then, this curve is sampled to obtain the initial position of each section center (red crosses). To finish, random noise in the vertical direction is added to each center (dotted yellow line) 128

5.9 Crack depth (a) and width (b) present a trapezoidal profile with an initial linearly increasing ramp extending over m_1 samples, and a final linearly decreasing ramp that is m_2 samples wide 129

5.10 Effect of adding a random variation to each section. (a) Medial axis and external contour of the crack without noise. (b) The same information after adding noise to each section. A large amount of noise has been added to show the effect more clearly 130

5.11 Volumetric representation of a crack. (a) The medial line that defines the crack and the set of sections. (b) Resulting surface representing the crack 133

5.12 Result of the bump parametrization. (a) 3D surface representation. (b) 2D section of the surface 134

5.13 Result of the peak parametrization. (a) 3D surface representation. (b) 2D section of the surface. 135

5.14 CAD model of the car door employed in these experiments. 137

5.15 (a) Surface reconstruction from scans without any processing. The defect is not visible due to its small size in relation to the shape of the part. Red lines represent individual profiles from profilometric sensor. (b) Surface reconstruction after preprocessing to emphasize small variations. The defect is now clearly visible. 139

5.16 An isolated profile of the scan of figure 5.15 in the defect area. Orange circle enclose the defect. (a) Raw measurement without any processing, corresponding to figure 5.15a. (b) Measurement after processing to emphasize small variations, corresponding to figure 5.15b 139

5.17 Inspection profiles of each type of defect. Orange circles mark the shape of the defect. (a) Real inspection. (b) Simulation. 140

5.18 Reconstruction of defects from simulated scans after preprocessing to emphasize small variations. (a) Real inspection. (b) Simulation . . . 141

5.19 (a) Diagram of the layout used to simulate a crack based on a real scan image. (b) Simulated scan of a crack constructed from the real crack shown in (a). White crosses mark the points that define the shape of the crack. 142

5.20 Depth image of real versus simulated defect samples used for training the different deep learning models. (a) Real samples. (b) Simulate samples 144

5.21 Precision-Recall of both models. In blue, the model trained with synthetic data, and in orange, the model trained with real data. . . . 148

6.1 Representation of the profilometric sensor and its main parameters. (a) 3D representation of the sensor with its coordinate system. (b) Front view of the sensor. The optimal working distance W_d and depth of field Z_r are depicted. 159

6.2 Incidence angle ($alpha$): angle between the orientation of the sensor $\overrightarrow{l}$ and the normal vector of the surface of the workpiece $\overrightarrow{n}$ 160

6.3 Distance between consecutive profiles Δs. 161

6.4 Top view of the scanning trajectory, where multiple parallel passes are made, each separated by a distance d. An overlapping area is defined between each pass. The gray square represents the sensor. The red lines show the trajectories where the sensor captures data. The black lines indicate the intermediate movements to position the sensor for the next pass. 162

6.5 Scheme of the process. 163

6.6 View of the simulated environment with the profilometric sensor and a part to be inspected. 164

6.7 Profile obtained during the simulation of a scan of the car door handle section of the CAD model shown in Figure 6.6. 165

6.8 Point cloud result obtained during the simulation of a scan of the CAD model section shown in Figure 6.6. The start and end points of the trajectory can be seen. 165

6.9 The figure shows the simulation environment and represents the action space as unit vectors (in orange). Δy refers to the increment in position in the scanning direction, Δz refers to the increment in the vertical direction (Z), and $\Delta\theta$ denotes the change in the sensor's pitch orientation. 167

6.10 Graph of the Distance Reward Function R_D. The graph shows the relationship between the observed distance D and the reward R_D based on the difference from the optimal working distance W_d. 170

6.11 Graph of the Orientation Reward Function R_α. The graph shows the relationship between the incidence angleα and the reward R_α according to a maximum incidence angle α_{max}. 171

6.12 Graph of the Movement Reward Function $R_{\Delta s}$. The graph shows the relationship between the scanning spacing between the current and previous profile Δs and the reward $R_{\Delta s}$ according to an optimal spacing Δs_{opt}. 172

6.13 Parts used for the experiments. (a) Car door. (b) Parrot drone. (c) Pen holder 174

6.14 Experimental setup: The UR3e robotic arm from Universal Robots, equipped with an AT sensor. 176

6.15 Environment used for the reinforcement learning model training. The CAD model of the piece used and the start and end poses of the trajectory to be optimized are shown. 177

6.16 Training metrics PPO algorithm: (a) mean episodic reward, (b) mean episode length, and (c) normalized reward over the course of training. 179

6.17 Comparison of overall reward (R) and partial rewards ($R_D, R_\alpha, R_{\Delta s}$) during training 180

6.18 Initial Boustrophedon Trajectory for Car Door Scanning. Red lines represent the inspection passes to be optimized, while the white lines indicate movements between passes. 181

6.19 (a) Resulting trajectories after applying the RL model. (b) Area covered during the scanning process for one of the trajectories. (c) Detail of the trajectory shown in (b), highlighting specific orientations. . . . 182

6.20 Distance error map obtained during the scanning with profilometric sensor. The colors indicate the difference between the measured distance and the optimal sensor distance, normalized based on defined distance values. (a) Using trajectories that adapt to the surface of the piece, calculated by the RL algorithm. (b) Using straight trajectories defined by a start point and an end point. 183

6.21 Comparation of the distance error distribution for all car door sections. 185

6.22 Orientation error map showing angular deviations from the optimal sensor orientation at each point, normalized based on defined orientation values. (a) Using trajectories that adapt to the surface of the piece, calculated by the RL algorithm. (b) Using straight trajectories defined by a start point and an end point. 186

6.23 Comparation of the orientation error distribution for all car door sections. 188

6.24 Zoom of a specific area of the scan of the car door. First row: Segment of the trajectory followed in the area where the defect was inserted. Second row: Results of defect detection in the scans. Third row: Density maps obtained from the scan trajectories. (The color scale ranges from [0,1]). 190

6.25 Zenital view of the CAD model of the pen holder and the straight trajectory planned for scanning. 191

6.26 Scanning trajectory generated by the RL algorithm for the pen holder. 191

6.27 Distance error map obtained during the scanning with profilometric sensor. The colors indicate the difference between the measured distance and the optimal sensor distance, normalized based on defined distance values. (a) Using trajectory that adapt to the surface of the piece, calculated by the RL algorithm. (b) Using straight trajectory. . 192

6.28 Comparison of the distance error distribution for the pen holder experiment. 194

6.29 Orientation error map showing angular deviations from the optimal sensor orientation at each point. (a) Using trajectory that adapt to the surface of the piece, calculated by the RL algorithm. (b) Using straight trajectory. 195

6.30 Comparison of the orientation error distribution for the pen holder experiment. 197

6.31 Experimental setup: The UR3e robotic arm from Universal Robots, equipped with an AT sensor. 198

6.32 Pen Holder Scan: 2D images of scanning results comparing RL and straight trajectories in both real and simulated scenarios. 199

6.33 Image of the Parrot AR. 2.0 drone body and its CAD model. 201

6.34 Zenital view of the straight path followed during scanning. 201

6.35 Scanning trajectory generated by the RL algorithm. 202

6.36 Distance error map obtained during the scanning with profilometric sensor. The colors indicate the difference between the measured distance and the optimal sensor distance, normalized based on defined distance values. (a) Using trajectory that adapt to the surface of the piece, calculated by the RL algorithm. (b) Using straight trajectory defined by a start point and an end point. 203

6.37 Comparison of the distance error distribution for the drone experiment.204

6.38 Orientation error map showing angular deviations from the optimal sensor orientation at each point. (a) Using trajectory that adapt to the surface of the piece, calculated by the RL algorithm. (b) Using straight trajectory defined by a start point and an end point. Two problematic areas marked by squares blue and orange. 205

6.39 Comparison of the orientation error distribution for the drone experiment. 206

6.40 Parrot Drone Scan: 2D images of scanning results comparing RL and straight trajectories in both real and simulated scenarios. 208

A.1 "Simulation of Laser Profilometer Measurements in the Presence of Speckle Using Perlin Noise". Paper published in MDPI's Sensors magazine [113] with the results of the chapter 4. 231

A.2 "Realistic Defect Simulation in 3D Models for Defect Detection Using Machine Learning", authored by S. Roos-Hoefgeest, M. Roos-Hoefgeest, D. García Peña, I. Álvarez, and R. C. González. Paper **currently in its third revision** with the results of the chapter 5. . 232

A.3 "A Vision-based Approach for Unmanned Aerial Vehicles to Track Industrial Pipes for Inspection Tasks". Article published and presented in the international congress ICUAS 2023 [117] 233

A.4 "Mobile robot localization in industrial environments using a ring of cameras and ArUco markers". Article published and presented in the international congress IECON 2021 [115] 234

A.5 "Block Matching Mosaicing for Surface Inspection Using an Autonomous Mobile Robot". Article published and presented in the international congress IECON 2021 [116] 235

A.6 "Localización de robots móviles en entornos industriales usando un anillo de cámaras". Article published and presented in the national Spanish congress JJAA 2019 [118] 236

A.7 "Algoritmo de generación de trayectorias en el interior de chapas para la subsanación de defectos". Article published and presented in the national Spanish congress JJAA 2019 [119] 237

List of tables

4.1 Parameters of the profilometric sensor from its datasheet and travel speed used in these experiments. 94

4.2 Bearing Cap: Comparison of roughness parameters between real and simulated scans. Estimated as the average of the parameters calculated for each profile. 101

4.3 Car door: Comparison of roughness parameters between real and simulated scans. Estimated as the average of the parameters calculated for each profile. 103

4.4 Heavy Steel Plate: Comparison of roughness parameters between real and simulated scans. Estimated as the average of the parameters calculated for each profile. 105

5.1 Parameters and variables of a Crack type defect. 131

5.2 Parameters of the profilometric sensor used in the experiments. 138

5.3 Model Performance for the model trained with simulated data. 147

5.4 Model Performance for the model trained with real data. 147

6.1 Parameters of the profilometric sensor extracted from its datasheet. . 175

6.2 Hyperparameters for the PPO algorithm used. 178

6.3 Distance Error Metrics for Optimized Scanning Paths Using RL (mm). 184

6.4 Distance Error Metrics for Scanning Paths Using straight trajectory (mm) 184

6.5 Orientation Error Metrics for Optimized Scanning Paths Using RL (º) 187

6.6 Orientation Error Metrics for Straight Scanning Paths (º) 187

6.7 Comparison of Distance Errors (mm) 193

6.8 Comparison of Orientation Errors (º): RL and Straight trajectory . . 196

6.9 Comparison of Distance Errors for RL Trajectory (mm) 199

6.10 Comparison of Distance Errors for Straight Trajectory (mm) 200

6.11 Comparison of Distance Errors (mm) 204

6.12 Comparison of Orientation Errors (Degrees) 206

6.13 Distance Error Metrics for Real Drone Experiments (RL Trajectory) (mm) 207

6.14 Distance Error Metrics for Real Drone Experiments (Straight Trajectory) (mm) 207

Chapter 1

Introduction

In this first chapter, the motivation and the main objectives established for this thesis will be presented. Finally, the chapter will conclude with a detailed outline of the structure of the rest of the document.

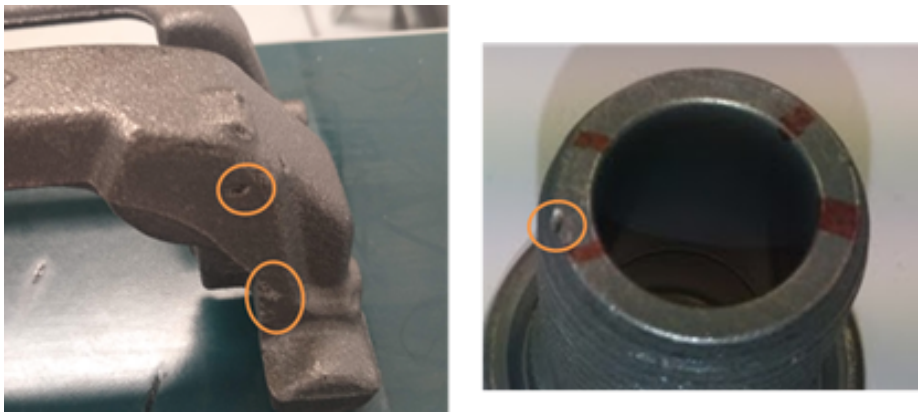
1.1 Motivation of the thesis

In today's competitive market, industries aim to produce higher-quality products in less time. This push for rapid innovation leads to the digitization and interconnection of production systems to improve efficiency, quality, and safety, a change known as Industry 4.0. A key part of Industry 4.0 is optimizing production processes, where inspection is crucial [1], [2].

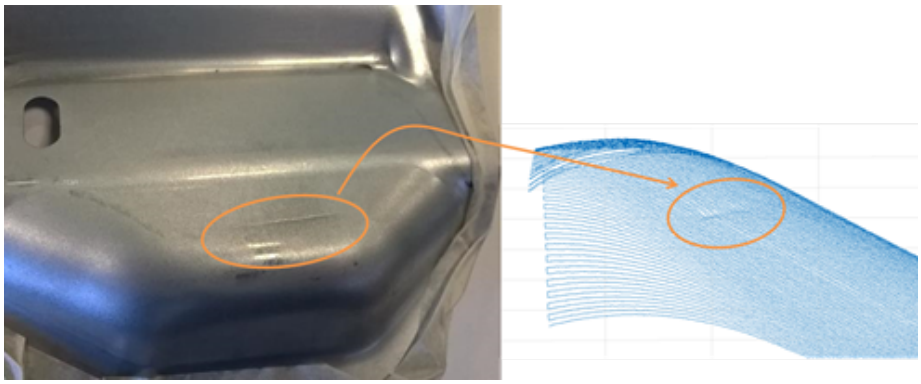
The manufacturing industry faces constant challenges regarding product quality, operational efficiency, and customer satisfaction. In this context, surface inspection emerges as a critical activity to ensure the integrity and quality of manufactured products. Interest in the subject has grown exponentially over the last decade, with a marked increase in publications since 2016. This trend reflects the constant advancement of inspection technologies and their growing importance for industrial applications. [3].

Timely and accurate defect detection is crucial to maintaining product quality. In competitive environments where speed and efficiency are paramount, inspection systems must operate without interruption on production lines. Surface defects,

such as dents, pores, lack of material, scratches, or cracks, not only affect the aesthetic appearance of the final product but can also compromise its functionality and durability. Figure 1.1 shows different types of defects in various parts.



(a)



(b)

Figure 1.1: Examples of surface defects on cast-iron (a) and stamped parts (b). It can be seen that the inspected parts may present complex geometries and that defects may also appear on high curvatura areas of the surface.

Until recently, obtaining high accuracy measurements required the use of contact measurement techniques. These techniques are used off-line in most industries: a sample is taken from the line, measured on a Coordinate Measuring Machine (CMM) [4], and the results are extrapolated to the entire production until a new measurement is obtained. However, for the detection of surface defects (stretching, cracks, bulges, missing material, etc.), this is not a suitable approach, and 100% of the production should be checked in-line. The involvement of human inspectors is undesirable (subjectivity, fatigue, cost), but is now essential in real-world production.

This is why non-contact inspection systems emerged, avoiding the problems associated with impacts between the evaluated part and the measurement system and increasing inspection speed.

Non-contact measurement systems, including optical techniques such as traditional computer vision [5], triangulation [6], interferometry [7], and others, have become increasingly prevalent. While 2D computer vision is commonly used in scenarios where defects cause distinct variations in the reflected light, it faces limitations in capturing subtle surface variations and intricate defects dueto its two-dimensional nature. These limitations become more evident as industries demand higher precision in defect detection, driving the need to transition to 3D inspection methodologies.

3D inspection techniques address these limitations by providing a more detailed view of surface complexities, leading to more accurate defect detection. The research by Jovančević et al. [8] highlights the importance of 3D data analysis for identifying defects such as dents, protrusions, or scratches, in aviation maintenance. This is especially important for detecting small defects, often on the order of tens of micrometers, that are barely visible or invisible to the naked eye. Similarly, in automotive manufacturing, Valentin Borsu et al. [9] underscore the crucial role of 3D inspection techniques, emphasizing the need for effective feature extraction methods to ensure precise defect detection across various manufacturing scenarios.

Different sensors enable the acquisition of 3D information without contact, and they are generally classified based on the type of measurements they capture in each acquisition: Point, profile and volumetric [10], [11]. Point sensors acquire 3D information of a single point with each measurement, making them ideal for precise, localized data collection. Profile sensors acquire 3D information of a line or profile in one go, allowing them to scan along a surface or edge, providing detailed cross-sectional data. Lastly, volumetric sensors capture the 3D information of an entire volume with each acquisition, enabling them to create comprehensive spatial representations of objects or areas in a single scan. Each of these sensor types can utilize different measurement techniques, depending on the specific requirements of the inspection or measurement task.

Typically, scanning a part for inspection with a non-contact sensor relies on either moving the part under the sensor or moving the sensor over the part. This movement must consider the nature of the sensor. In applications involving large-volume parts, point sensors (which are more precise) are discarded due to their time-consuming nature. Volumetric sensors may be unsuitable because of their lack of precision, making profile sensors the preferable choice in many cases. Profile sensors provide high-precision linear measurements of each profile, and the 3D reconstruction is

achieved through the relative movement between the sensor and the part.

Among the profile sensors, profilometric lasers are widely used in the industry, particularly in large-scale manufacturing, accounting for 11% of inspection applications. Their popularity is largely due to their high-speed capabilities, which make them faster than traditional optical cameras and allow for more efficient inspection processes. These sensors rank among the leading inspection tools, following machine vision systems, which dominate at 58%, and methods that utilize existing machine data without additional equipment, at 13% [3].

Within the field of profilometric sensors, laser triangulation sensors are among the most widely used due to their precision, robustness, and rapid data acquisition capabilities. They are crucial for a variety of inspection tasks across different industries. Their effectiveness is demonstrated in applications such as automotive [12], aeronautical [13], gear inspection [14, 15], and other types of surface inspections [16, 17, 18, 19, 20].

These sensors operate by projecting a laser beam onto the surface of an object and detecting the angle of the reflected beam with a sensor. The distance to the object is then calculated based on the geometry of the triangle formed by the laser beam, the reflection point, and the detector. This method provides precise, non-contact measurements of surface geometry, making it particularly valuable for applications that demand high accuracy and quick inspection.

One crucial application of this technology involves detecting superficial defects like dents, bumps, or scratches. However, integrating defect detection into profilometric measurements highlights a critical need for comprehensive datasets. The literature explores various strategies for detecting defects in laser triangulation scans, yet they universally face the challenge of inadequate datasets essential for developing and effectively evaluating these algorithms. Given the complex variations in surface defects influenced by material properties and part geometries, authentic datasets that accurately represent real-world conditions are imperative.

The integration of artificial intelligence (AI) in defect detection methodologies has led to significant advancements, as discussed by Gao et al. [21], who explored the challenges encountered in AI-driven defect detection, particularly with the adoption of deep learning techniques. However, the effectiveness of deep learning heavily depends on the availability of large volumes of labeled training data, as emphasized by Lv et al. [22]. This exacerbates the existing challenges in sample collection and labeling, particularly in industrial settings where labeled data is scarce due to the high costs and time-consuming nature of labeling efforts. Addressing dataset

challenges, Huo et al.[23] emphasized the limited availability of 3D data samples in current research. They pointed out two main challenges: laborious and time-consuming preparation of point cloud training samples, and the lack of diversity in these datasets, often due to the use of artificially generated computer-aided models that may lack authenticity.

While AI techniques, particularly deep learning, focus on improving the accuracy and reliability of defect detection through data-driven approaches, the physical aspect of inspection remains crucial. The successful implementation of any defect detection algorithm for surface inspection also requires precise control over the interaction between the sensor and the part being inspected.

Another critical challenge in surface inspection using profilometric sensors is accurately planning the motion between the part and the sensor. As mentioned earlier, achieving comprehensive scanning of the part requires precise relative movement between them. This is where robotics plays a crucial role in industrial settings. Robotics serves as a versatile platform for automated inspection systems, enabling precise and controlled movement of sensors across the part's surface. By facilitating automated motion, robots ensure consistent and thorough inspection without the need for direct human involvement. This capability is particularly advantageous in manufacturing environments where maintaining high levels of repeatability and precision is paramount to ensuring product quality and operational efficiency.

Within this field, various types of robots have been employed and integrated into a variety of processes. These include robotic arms [24], [25], unmanned aerial vehicles (UAVs) [26], unmanned ground vehicles (UGVs) [27], [28], and even autonomous underwater vehicles (AUVs) [29].

Although the development of robotic systems integrating such sensors has attracted increasing interest, a major gap remains that deserves further exploration. In many cases, the relative motion between the sensor and the part lacks the necessary accuracy, sometimes being one or two orders of magnitude worse than that of the sensor itself. This discrepancy can lead to significant errors during the detection of surface defects, which often occur in the range of a few tens of microns. One of the main challenges is to optimize the scanning distance and optimal orientation, as well as to ensure proper profile separation. Various studies, such as those by Li et al. [30] and Sadaoui et al. [31], have examined the influence of error based on the angle of incidence and the scanning distance. These factors, also specified by sensor manufacturers, underscore the importance of adapting the scanning path to the specific geometry of the part, which is essential for mitigating errors and ensuring accurate and reliable inspection.

The proper integration of robotics and profilometric sensors is critical for the accurate detection of surface defects in industrial environments. However, this process involves a considerable challenge: the optimal configuration of these systems can be costly and time-consuming, especially when dealing with high-precision sensors and complex geometries. This is where simulators emerge as a valuable solution by allowing the evaluation of different inspection configurations in a virtual environment, prior to their practical implementation.

1.2 Problem Statement

In this research, several significant challenges have been identified in the field of surface defect detection using profilometric sensors. One of the main problems is the lack of comprehensive and realistic datasets that cover a wide variety of defect types and accurately simulate real-world industrial conditions. The lack of such data complicates the development and evaluation of defect detection algorithms capable of addressing diverse materials and industrial settings.

Additionally, the integration of artificial intelligence, particularly deep learning, into defect detection methods faces significant obstacles. The success of AI-based approaches relies heavily on access to large, well-labeled datasets. However, acquiring such datasets in industrial environments is challenging due to high costs, labor-intensive labeling requirements, and limited diversity of available data. All these problems complicate the advancement and practical application of advanced AI techniques in defect detection systems.

The variety and complexity of the shapes, together with the nature of the manufacturing process, generate few defective samples for training the algorithms, and frequently biased: once a defect is present in some region of one part, it usually repeats in the next ones; in the other hand, defects in some regions or with certain shapes may never appear in the acquisitions for the training phase. The high quality standards required nowadays demand the inspection system to be able to detect defects that did never appear before in the same configuration

In addition to dataset limitations, ensuring precise relative motion between the inspected part and the sensor is crucial for accurate defect detection. Current methods often struggle to achieve the necessary level of precision, leading to potential inaccuracies, especially in detecting fine surface defects that can be as small as a few micrometers. Addressing these challenges requires advances in motion planning

algorithms that optimize scan paths, minimize errors and improve the reliability of defect inspection processes in industrial applications.

CAD tools are indispensable in addressing the significant challenges encountered in surface defect detection using profilometric sensors. Mohammadikaji et al. [32] highlighted the role of CAD tools in optimizing designs, reducing costs, and facilitating simulation-based inspection planning. These tools simulate the inspection process pre-production, identifying issues and refining methods to enhance product quality efficiently. To be useful, the simulator must accurately replicate the behavior of the inspection system using models that can be easily related to real hardware. Designers will use that tool to test its performance under different conditions to obtain an accurate and cost-effective solution while reducing the development time and costs.

Existing tools often struggle to handle the diverse challenges involved in industrial inspection using profilometric sensors. There is a noticeable lack of a single tool that can integrate comprehensive dataset simulation, accurate motion planning, and realistic replication of sensor behaviors. This gap highlights the clear need for a dedicated tool that can effectively address these various challenges in detecting surface defects with profilometric sensors.

This thesis aims to fill this gap by developing a CAD tool adapted to the inspection of industrial parts using laser triangulation profilometric sensors and robotic systems. The objective is to enhance inspection accuracy and quality, improve the detection and classification of surface defects on manufactured parts, and address specific challenges in industrial inspection to ensure the high quality and reliability of manufactured products. The following section describes the main objectives of the thesis.

1.3 Objectives

The main goal of this thesis is to create an effective and comprehensive system for inspecting industrial parts using laser triangulation profilometric sensors and robotic systems. This project aims to improve the quality and precision of the inspection process and to enhance the detection and classification of surface defects on parts. The system is designed to address specific challenges in industrial inspection to ensure the quality and integrity of manufactured products.

This main goal is divided into more specific objectives, as detailed below:

- **Develop a precise and realistic inspection simulator**: The first goal is to design and create a simulator that accurately reproduces the conditions of the actual inspection system. This simulator uses the CAD model of the parts to be inspected to get a precise representation of their geometry. It also incorporates the operational parameters of the laser triangulation profilometric sensor and simulates speckle noise along with other artifacts that may occur during the scanning process. By faithfully recreating the real scanning conditions, the simulator can generate highly realistic and reproducible data, essential for the rigorous training and validation of defect detection algorithms.

- **Simulate surface defects in CAD models**: The objective is to simulate any type of surface defect in CAD models by directly deforming the model's surface using an elevation map. This method allows for the precise incorporation of a wide range of defects, ensuring highly accurate simulations. Among these, three of the most common defect types in steel products—dents, peaks, and cracks—have been parametrized through detailed mathematical models. These models offer an exact representation of each defect type, enabling more realistic and detailed simulations.

  By generating surface defects in CAD models and using the inspection simulator, the goal is to create a tool to build defects datasets with three-dimensional information. This defect database will provide realistic and varied data to train and validate detection algorithms, ensuring the accuracy of automated inspection systems. It will also be crucial for developing advanced techniques, optimizing scanning paths, and accelerating innovation in industrial inspection.

- **Develop a method for generating optimized scanning paths**: A method based on Reinforcement Learning techniques is designed to generate optimized scanning paths. This method uses information provided by the simulator and the CAD model of the part to dynamically adjust the sensor's position, orientation at each scanning point and the distance between successive captures. The goal is to ensure comprehensive and efficient coverage of the part's surface, maximizing the quality and precision of the inspection. Efforts will focus on optimizing the relative position error between the sensor and the part, ensuring that scanning occurs at the optimal distance and orientation for accurate defect detection, as much as possible.

Together, these objectives create a comprehensive and coherent approach to tackling the challenges associated with inspecting industrial parts using laser triangulation profilometric sensors. The proposed system has the potential to significantly enhance the effectiveness and reliability of inspection processes. This improvement

can positively impact the quality and safety of manufactured products across various industrial sectors.

1.4 Structure of the document

The present thesis is divided into different chapters.. The contents of each chapter are outlined below:

In Chapter 2, the basics of Measurement Techniques for Surface Inspection are discussed. This chapter explores different methods used to check the quality and texture of surfaces, explaining the main ideas behind each method.

Chapter 3 reviews the theoretical foundations of Reinforcement Learning (RL) and presents the most common algorithms in the literature.

In Chapter 4, the development of a simulator for an inspection system using a laser triangulation profilometric sensor is presented. The main objective is to achieve a realistic simulation of readings provided by a commercial sensor. The incorporation of speckle effect, which constitutes the most relevant source of noise affecting the measurement process, is pursued. Additionally, inherent sensor measuring precision is also modeled.

Chapter 5 introduces a novel method for simulating surface defects in 3D models using Free-Form Deformation (FFD) technique. This technique allows generating defects with known dimensions and shapes, essential for developing defect detection algorithms in manufactured products. Furthermore, mathematical models for common defects in sheet metal products, such as dents, peaks, and cracks, are proposed. These models offer flexibility to parameterize defects in terms of depth, size, and orientation, enabling precise customization of simulated defects. This chapter extends the work presented in Chapter 4 by providing a simulator capable of creating databases of labeled defects, crucial for the development of artificial intelligence algorithms.

Chapter 6 presents a novel method for planning inspection trajectories based on reinforcement learning. This technique adjusts the sensor's position and tilt to maintain optimal orientation and distance from the surface, ensuring consistent profile distance for high-quality scanning. A simulated environment based on CAD models replicates real-world conditions, including sensor noise and surface irregularities, enabling effective offline trajectory planning. Key contributions of this chapter include

modeling the state space, action space, and reward function specifically designed for inspection applications using profilometric sensors. The goal is to ensure comprehensive and efficient coverage of the surface of the workpiece, maximizing inspection quality and precision to facilitate defect detection.

Finally, chapter 7 summarizes the final conclusions of the research and possible future lines of work derived from this study. In addition, Appendix A compiles all the journal publications and conference papers arising from this work.

Chapter 2

Measurement Techniques for Surface Inspection

Surface inspection is vital in industries such as manufacturing, aerospace, automotive, and electronics, where the quality and reliability of products are critical. Surface defects can significantly impact the performance, aesthetic appeal, and durability of products. Therefore, precise and efficient measurement techniques are essential for detecting and analyzing these defects.

Over the years, various measurement techniques have been developed and refined to inspect surfaces for defects. These techniques range from simple visual inspections to sophisticated methods utilizing advanced imaging and sensor technologies. Each technique has unique principles, applications, advantages, and limitations, making them suitable for different types of inspections and materials.

In this chapter, we will explore the diverse measurement techniques used for surface inspection, which are crucial for ensuring product quality in modern manufacturing. These techniques can be broadly categorized into two types: contact and non-contact methods.

2.1 Contact Measurement Systems

Until recently, obtaining high accuracy measurements required the use of contact measurement techniques in most industries. Contact measurement systems involve

direct physical interaction with the surface being inspected. These systems typically use mechanical or physical devices to trace or measure the surface profile. Despite their high accuracy, contact methods can be slower and may cause damage to delicate surfaces.

Many industries use off-line contact measurement techniques, where samples are removed from the production line for analysis, and results are extrapolated to the entire run. This method is inadequate for comprehensive inspection as it does not cover the entire production and can miss defects like stretching or cracks. Additionally, relying on human inspectors introduces issues such as subjectivity, fatigue, and cost.

Figure 2.1: Coordinate Measuring Machine. Image extracted from [33]

Among contact profilometers, Coordinate Measuring Machines (CMMs) are particularly noteworthy, see figure 2.1. CMMs are equipped with a mechanical or electronic probe that interacts directly with the part being measured. The probe is mounted on a multi-axis robotic system, which enables it to move precisely in three-dimensional space. This movement is tracked by scales or sensors on each axis,

allowing the machine to determine the exact position of the probe at all times. The primary goal of a CMM is to reconstruct the spatial position of different points on the part, enabling the measurement of various parameters based on this reconstructed model.

These contact systems are recognized for their high accuracy and resolution, making them essential for precise dimensional metrology. Their key advantage lies in their ability to deliver detailed measurements by physically interacting with the surface through a mechanical stylus or probe.

However, this physical contact introduces several challenges. In dynamic environments like production lines, vibrations and changes in surface height can impede precise control. Additionally, contact profilometers may alter or damage delicate surfaces, especially in the presence of contaminants or varying conditions.

Maintaining consistent measurement quality demands careful management of the probe's contact force and movement, which can be difficult outside of controlled laboratory settings. Thus, while these devices offer high precision, their application can be limited by their susceptibility to operational challenges and potential surface impact. Furthermore, these devices are often very slow, typically requiring several seconds to measure a single point.

2.2 Non-Contact Measurement Systems

Non-contact measurement systems are advanced tools that capture surface data without physical contact. These systems address many of the issues associated with contact-based methods, such as potential damage to delicate surfaces and limitations in dynamic environments. However, they introduce their own set of challenges, which vary depending on the specific technology used.

Non-contact measurement systems are generally classified into two main categories: optical and non-optical. Optical systems, which include techniques like laser scanning and interferometry, use light to capture detailed surface and are known for their high precision and versatility. Non-optical systems, such as capacitive, inductive, and ultrasonic sensors, rely on other physical principles to measure surfaces and are often used in environments where optical methods may be impractical.

Different sensors enable the acquisition of 3D information without contact, and they are generally classified based on the type of measurements they capture in each

acquisition: point, profile, and volumetric.

- **Point sensors:** acquire 3D data of a single point with each measurement. This type of sensor is ideal for precise, localized data collection where detailed information about specific spots or features is needed. Point sensors are often used in applications requiring high accuracy at individual points, such as in detailed inspections or quality control where small, precise measurements are crucial.
- **Profile sensors:** capture 3D data along a line or profile in a single acquisition. These sensors are suitable for scanning along surfaces or edges to obtain detailed cross-sectional data. Profile sensors are valuable for inspecting the contours or shapes of objects, where it is important to understand the surface profile or the edge details of a component. They provide a detailed view of the surface's shape and are often used in applications like surface profiling and edge detection.
- **Volumetric sensors:** capture 3D data of an entire volume with each acquisition. They are designed to create comprehensive spatial representations of objects or areas in a single scan. Volumetric sensors are ideal for applications that require full 3D mapping or modeling of objects, capturing the complete structure of a surface or object. This type of sensor is commonly used in applications involving large-scale inspections or detailed 3D modeling where a complete view of the object is necessary. These are, typically, the least precise, as their measurements cover a broader area with less detail per individual point.

In this thesis, we will focus primarily on optical non-contact measurement techniques, given their prominence in modern surface inspection and their ability to provide highly accurate and detailed surface profiles. In the following, the most commonly employed optical methods in the industry are presented.

2.2.1 Time of flight

Time of Flight (ToF) technology measures the distance between the sensor and the surface by calculating the time required for a light pulse to travel from the sensor to the surface and back. This technique is particularly prevalent in laser-based systems, where a laser pulse is emitted towards the surface, reflected back, and then detected

by the sensor. The elapsed time between emission and detection of the reflected pulse is directly proportional to the distance traveled, allowing the system to compute the distance with high accuracy. Figure 2.3 shows its basic principle.

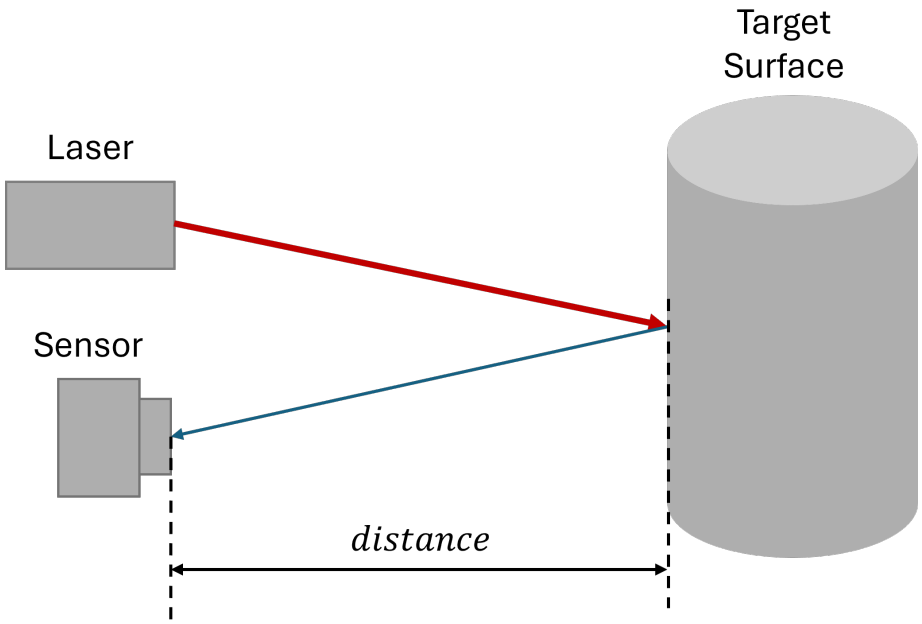


Figure 2.2: Principle of operation of Time of Flight (ToF) systems

ToF systems are known for their rapid data acquisition and ability to cover large areas quickly. However, these sensors usually have a resolution that may fall short when it comes to identifying very fine defects, such as those at the micrometer scale. These sensors generally provide precision within the range of millimeters to sub-millimeters, which may not be detailed enough for detecting micro-scale imperfections.

2.2.2 Photogrammetry

Photogrammetry is a technique that uses photographs captured from multiple angles to create a three-dimensional model of a surface. By analyzing the geometric relationships between corresponding points in different images, photogrammetry accurately reconstructs the 3D structure of the surface. Similar to binocular vision, photogrammetry determines the spatial coordinates of a point by identifying common features in two or more photographs taken from different perspectives. For

each camera viewpoint, a line is drawn connecting the point being digitized to the camera's location. The intersection of these lines from different perspectives allows for the precise calculation of the digitized point's spatial position.

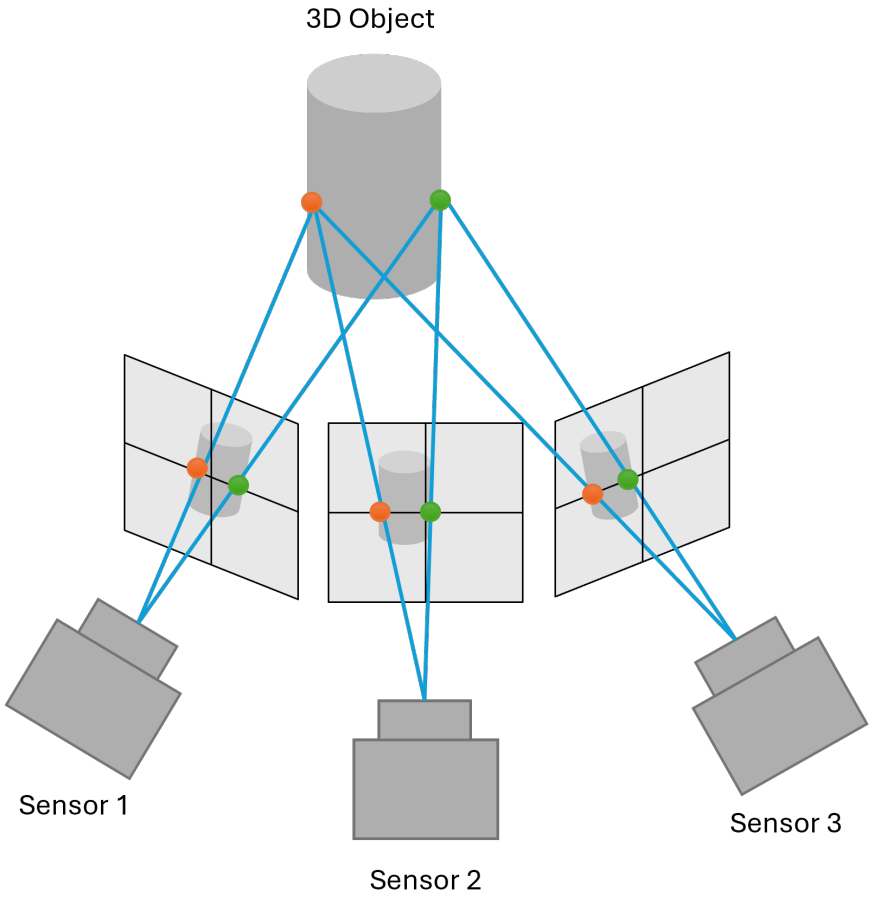


Figure 2.3: Principle of operation of Photogrammetry

This method is particularly useful for large-scale inspections and applications where manual measurement is impractical. Its advantages include the ability to capture complex geometries and its non-invasive nature. However, photogrammetry relies heavily on the quality of the images and the precision of camera calibration. Photogrammetry provides more realistic textures but with less geometric precision, so it is best suited for applications where visual realism is prioritized over exact dimensional accuracy.

2.2.3 Structured light

In a structured light system, a projector emits a predefined pattern of light—such as stripes, grids, or other geometric shapes—onto the surface of the object being measured. This pattern deforms as it interacts with the contours and features of the surface. Multiple cameras positioned at strategic angles capture the reflected pattern, documenting how it has been altered by the surface geometry. By analyzing these distortions and comparing them to the known pattern, the system can accurately reconstruct the 3D profile of the surface, providing detailed spatial information about the object's shape and features.

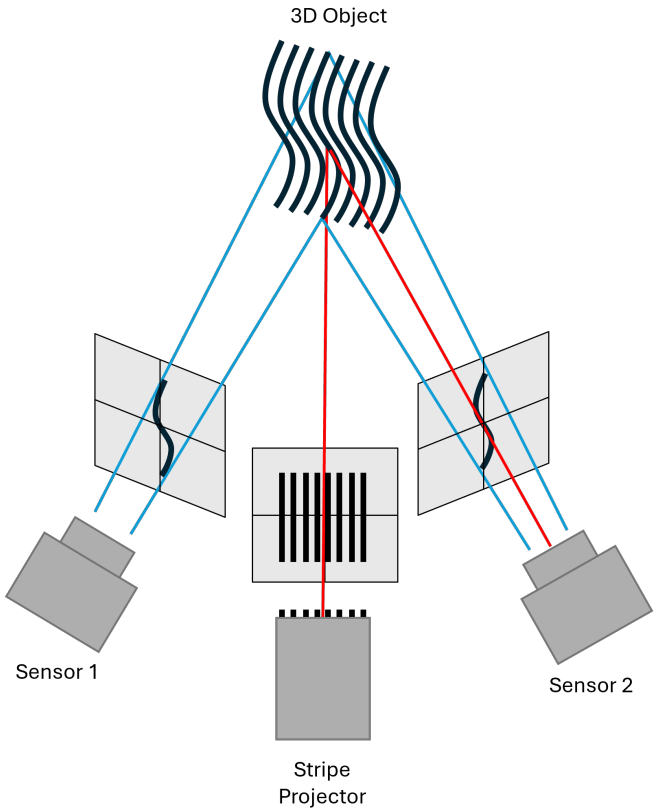


Figure 2.4: Principle of operation of Structured Light systems

Structured light systems are highly effective for capturing detailed surface features with high accuracy and speed, making them widely used in industrial applications. However, they can be sensitive to surface reflectivity and ambient lighting. These systems can achieve sub-millimeter accuracy, but their effectiveness may be

limited when precise measurements of only a few micrometers are required. The accuracy depends on the precision of the projected light or laser, the quality of camera sensors, and the calibration between the camera and projector, making them essential for applications that demand fine detail and high precision.

2.2.4 Confocal Microscopy

Confocal microscopy operates on the principles of point illumination and spatial filtering. In this system, a laser beam is focused on a single point of the sample using an objective lens. As the laser scans across the target, light emitted or reflected from this point travels back through the same objective lens and is directed towards a pinhole aperture located in front of the detector. This pinhole blocks out-of-focus light, ensuring that only light from the focal plane reaches the detector.

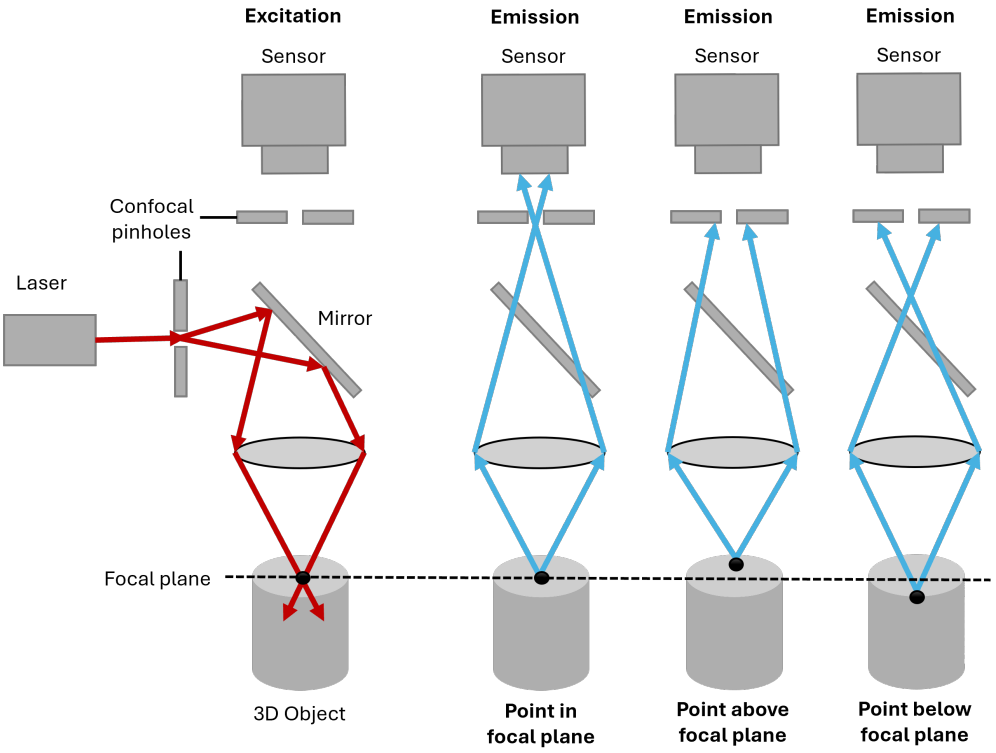


Figure 2.5: Principle of operation of Confocal Microscopy systems

The laser systematically scans the target in a raster pattern, and the detector records the intensity of light from each point. These measurements are then compiled

to create a detailed image of the entire focal plane. To generate a 3D representation of the sample, the focal plane is gradually moved deeper into the sample, repeating the scanning process to capture a series of images at different depths. These images are then stacked to form a complete 3D model of the sample.

The precision of these systems is very high, allowing for accurate measurements at the microscopic level. However, there are some disadvantages. The method involves a vertical scan for each point, which can be slow, making it less suitable for use on an industrial production line.

2.2.5 Interferometry

Interferometry relies on the principle of interference, which occurs when two coherent light waves of the same frequency overlap. This interference can be constructive, where the waves align perfectly, or destructive, where they are out of phase. When the waves are neither perfectly in phase nor completely out of phase, a complex interference pattern of varying intensities is generated. By analyzing this pattern, it is possible to determine the phase difference between the two waves.

In an interferometric system, two beams of light travel along separate optical paths. These paths are defined by a system of mirrors and optical plates. Typically, one beam is directed towards a reference mirror, while the other beam is directed towards a mirror that is located at or on the surface to be measured. After reflecting off their respective mirrors, the two beams are recombined, creating an interference pattern.

The reference mirror is positioned at a known distance, while the mirror on the surface being measured is subject to the variations in the surface profile. The interference pattern generated when the two beams recombine is affected by the differences in the optical paths traveled by each beam. By examining the resulting interference fringes, the phase shift between the beams can be calculated.

This phase shift is directly related to the differences in distance traveled by the beams. Thus, the technique enables precise measurement of surface topographies and variations by converting the phase shift into distance measurements. The accuracy of these measurements hinges on detecting minute changes in the interference pattern, which can be influenced by factors such as vibrations, temperature changes, and other environmental conditions. Interferometric systems are highly sensitive and can achieve nanometer-level precision, making them suitable for applications requiring

extreme accuracy.

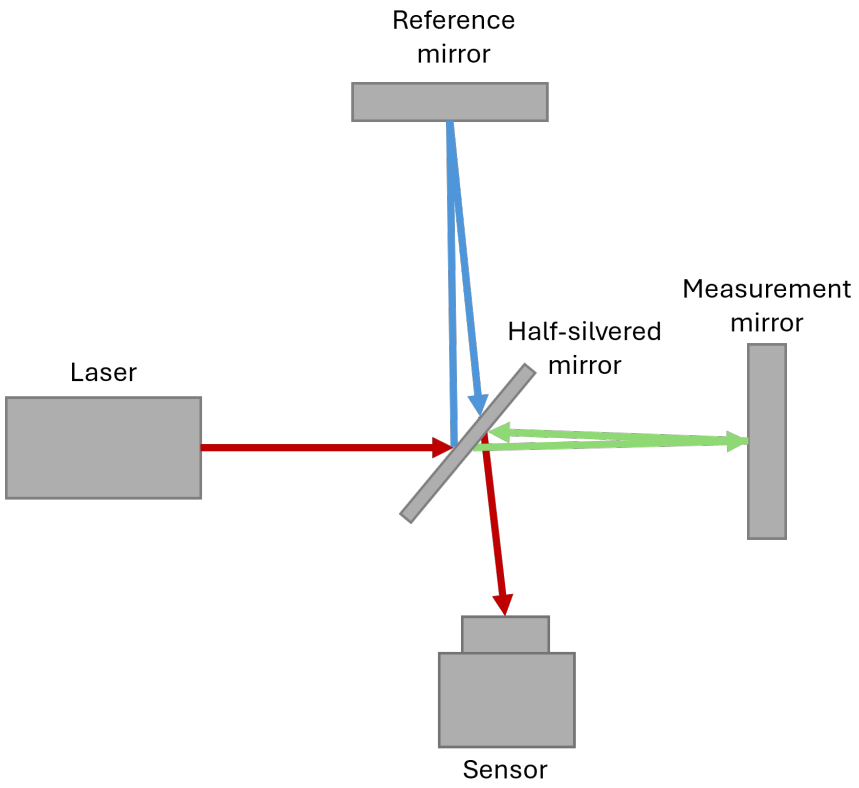


Figure 2.6: Principle of operation of interferometry systems

Interferometry is highly effective for capturing extremely fine surface details with exceptional accuracy and precision, making it valuable in high-precision industrial applications. While it can achieve sub-micrometer accuracy, the technique provides point measurements rather than line or area measurements. This makes it less practical for line-based or large-scale inspections, where continuous or extensive coverage is required.

2.2.6 Conoscopic Holography

Conoscopic holography is an interferometric technique that utilizes polarized light to measure surfaces with high precision. When a beam of monochromatic polarized light enters an uniaxial crystal, it splits into two rays with orthogonal polarizations,

known as the ordinary ray and the extraordinary ray. The ordinary ray travels at a constant speed, whereas the speed of the extraordinary ray varies depending on the angle of incidence, which is directly related to the distance from the observed point.

As the rays exit the crystal, they become out of phase due to their different speeds. By placing a polarizer in the optical path, these two rays can interfere with each other, producing an interference pattern. Analyzing this pattern allows for the precise determination of the distance between the point on the surface and the sensor.

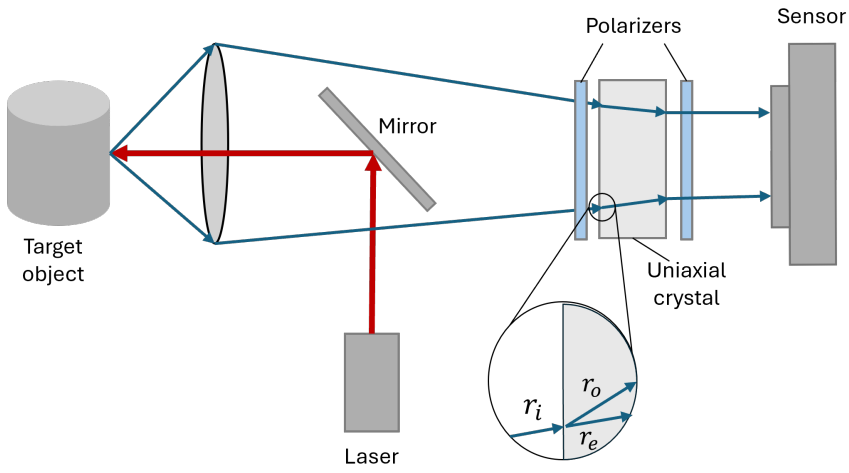


Figure 2.7: Principle of operation of Conoscopic Holography

Conoscopic holography offers exceptional precision and versatility for micron-level surface inspection applications. However, it faces notable drawbacks. A primary disadvantage is the limited availability of commercial sensors, as Optimet, a leading manufacturer in this field, is no longer operational. The technology's point-based sensors were highly precise, but the line-scanning models, which used a moving mirror to capture data, suffered from slower measurement speeds compared to other profilometric methods.

2.3 Laser Triangulation

Laser triangulation is a widely used non-contact measurement technique for capturing precise 3D surface data. This method leverages the geometry of light reflection

to determine the distance from a laser source to a target object, providing high accuracy in surface profiling and dimensional inspection. Here, we explore the fundamental principles, operational mechanism, applications, and advantages of laser triangulation in the context of surface defect detection and quality control.

This technique is based on the geometric principle of triangulation, which involves measuring distances by forming a triangle between a laser source, the target object, and a detector. The basic setup includes (see figure 2.8):

- Illumination Source (Laser): Emits a coherent beam of light onto the surface of the object. Typically, this is a laser diode that projects a narrow, focused beam, often shaped into a line to cover a larger area of the surface for faster scanning.
- Imaging Sensor (Camera): Positioned at a known angle γ relative to the laser source, it captures the reflected laser spot or line. This camera or photodetector is calibrated to accurately record the position of the laser reflection.
- Processor: Computes the distance to the target based on the position of the reflected laser on the detector. This involves analyzing the captured image of the laser line and applying triangulation algorithms to convert the pixel coordinates of the reflection into precise 3D coordinates.

Laser triangulation is a precise measurement technique that projects a laser beam onto the surface of an object from a fixed position. When the laser beam hits the object, it reflects in various directions depending on the surface's topography and material characteristics. An integrated sensor detector captures this reflected light and pinpoints the exact position of the reflection on its array.

To calculate the distance to the object, the system analyzes the angle between the incoming laser beam and the reflected beam—this angle is referred to as the triangulation angle. Additionally, the system considers the position of the reflected light on the detector. By applying trigonometric principles to these measurements, the system can accurately determine the distance from the laser source to the object's surface.

This method enables precise surface profiling and measurement, making it invaluable in applications requiring high accuracy, such as quality control and defect detection.

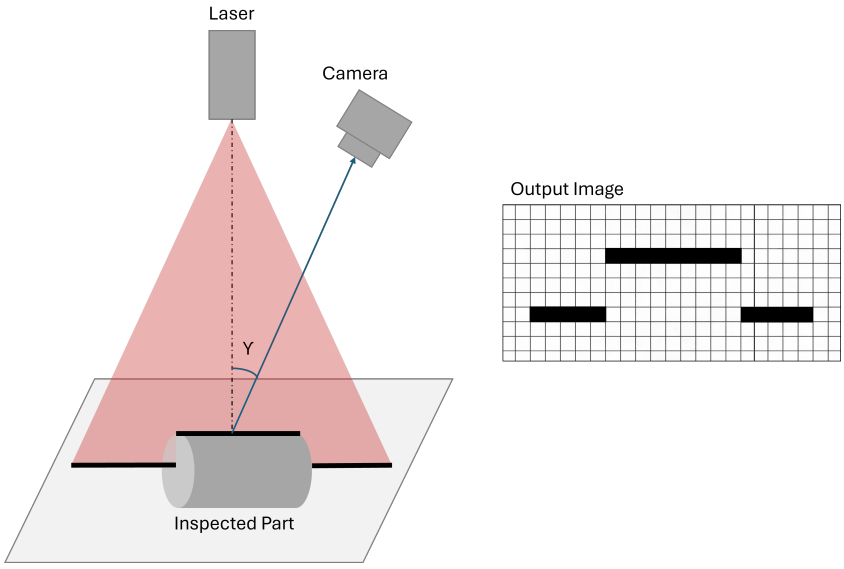


Figure 2.8: Operating principle of laser triangulation.

2.3.1 Sensor Parameters

In laser triangulation systems, understanding the sensor parameters is crucial for optimizing performance and ensuring accurate measurements. These parameters influence the sensor's ability to capture and process data effectively, especially when dealing with fine surface details or small-scale defects.

Commercial laser triangulation sensors typically integrate the laser emitter and the camera into a single unit. They operate with resolutions typically in the micrometer range, ensuring precise measurement capabilities. In this section, we will explore the key parameters that define the capabilities of laser triangulation sensors:

- **Working Distance** (W_d): Optimal distance between the sensor and the surface of the object being measured. This is the distance from the laser source to the reference scanning plane, which is situated at the midpoint of the depth of field.
- **Depth of field (DOF)**: Range of depths or heights within which the sensor can accurately capture points on the surface of the object. Also known as Z-Range.

- **X-FOV**: The width of the laser beam at the working distance, determining the horizontal extent of the area that the sensor can measure in a single scan.
- **Field of view (FOV)**: region within which the sensor can collect points on the digitized surface. It is defined by the Z-Range and the width of the laser beam.
- **Z-Resolution**: Precision with which the sensor can measure the depth of surface features.
- **X-Resolution**: Precision with which the sensor can measure coordinates along the horizontal axis.
- **Points per profile**: Determine the number of points to measure in each profile. This defines the point density in the point cloud generated by the scan and directly affects the quality and accuracy of the captured data.
- **Triangulation angle (γ)**: Angle formed between the laser beam and the camera's optical axis when focused on the reference surface. This angle depends on the sensor's design geometry.
- **Sampling frequency**: Number of measurements that the sensor can perform per second, measured in frames per second (FPS).
- **Intensity**: Measurement of the strength of the reflected laser signal, which indicates the power of the laser light that is reflected back to the sensor.
- **Incidence Angle (α)**: Angle between the laser beam incident ($\overrightarrow{l}$) on a point of the digitized surface and the surface normal at that point ($\overrightarrow{n}$). It depends on the sensor's orientation relative to the object.

In Figure 2.9, the typical parameters of laser triangulation systems are presented, which are crucial for accurately measuring distances and geometries in various industrial and scientific applications.

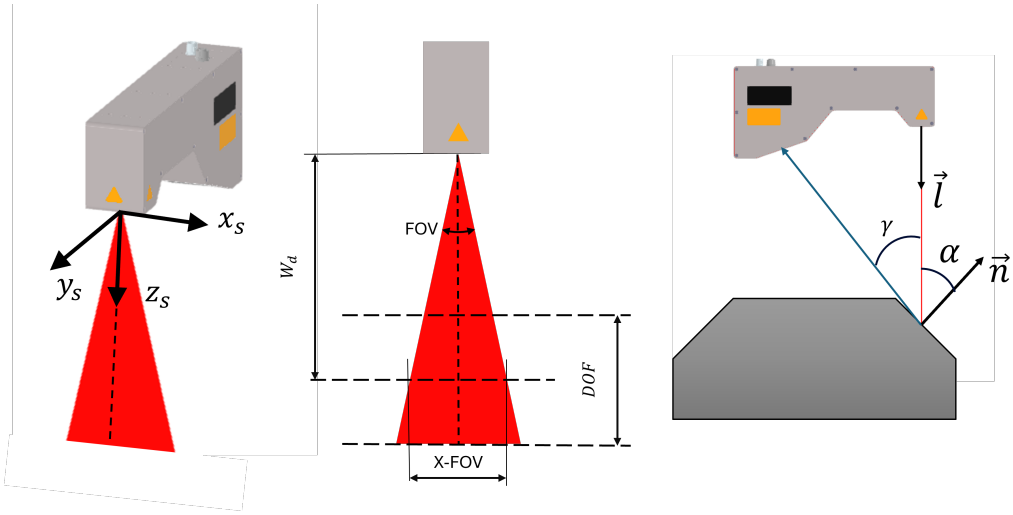


Figure 2.9: Common parameters of a laser triangulation profilometry sensor.

2.3.2 Challenges in Light-Surface Interactions

When evaluating the effectiveness of a triangulation sensor for a specific application, the material of the target plays a crucial role. This information is commonly found in the documentation and specifications provided by manufacturers of laser triangulation sensors, for example in [34].

The interaction of light with the surface—whether it is reflected, transmitted, or absorbed—directly impacts the sensor's ability to obtain accurate measurements. Triangulation sensors rely on detecting the reflected light to generate a digital profile of the surface. If the amount of reflected light is insufficient, the sensor's accuracy can be compromised.

The ideal target is a consistent white, matte surface, which allows the laser spot to diffuse and scatter light effectively, providing a strong and stable reflection back to the sensor. This results in clearer readings and better resolution. Conversely, transparent materials present challenges since the laser light often passes through rather than reflecting, making precise measurements difficult. Shiny surfaces can also be problematic as they reflect light in a narrow beam, which might not provide enough diffused light for accurate detection. However, with proper adjustments, such as altering the angle of the laser, accurate measurements can still be achieved on shiny surfaces.

Materials that transmit light, such as marble or granite, are particularly challenging due to a phenomenon called *subsurface scattering*. In these materials, light penetrates the surface and reflects from within the object, leading to the capture of internal rather than surface reflections. This effect introduces errors in digitalizing translucent surfaces, as the reflected light is not solely from the surface layer.

Glowing or hot surfaces can interfere with measurements due to overlapping wavelengths, especially with red lasers; using lasers of different wavelengths, like blue, can mitigate this issue. Dark materials may reduce the amount of detectable light but can still be measured with adjustments to exposure time or by using more powerful lasers.

Overall, if the laser spot is visible on the target material with the naked eye, the sensor should be able to detect it effectively. Testing sensors with actual materials is recommended to ensure they perform well in specific applications.

In addition to material properties, the roughness of the surface plays a significant role in how well a triangulation sensor can perform. Rough or textured surfaces introduce another challenge known as Speckle, one of the primary sources of noise in laser triangulation systems [35].

Speckle Noise

When a laser beam illuminates a rough or textured surface, it induces a phenomenon known as laser speckle. This effect arises because the coherent light waves arriving at the surface encounter microstructures or irregularities, causing them to scatter in various directions. Consequently, the reflected rays lose their phase coherence, resulting in interference patterns. Regions where surface features create constructive interference appear as bright spots on the sensor, while areas with destructive interference appear dark. This speckled pattern introduces random fluctuations in the intensity of the reflected light captured by the sensor, which adds noise to the measured data. The outcome is a pattern of alternating dark and light spots overlaid on the image, contributing to uncertainty in accurately localizing the laser spot, as illustrated in Figure 2.10.

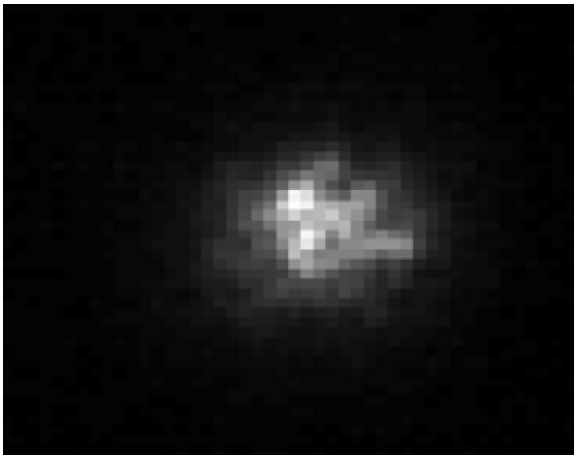


Figure 2.10: Image of a laser spot over a rough surface. The localization of the center cannot be done without certain uncertainty, due to the speckle effect. Image obtained from [35]

2.3.3 Surface Inspection Strategies

For a complete surface inspection, a relative movement between the surface and the triangulation system is necessary. This movement must be precise to accurately reference the captured points within a global coordinate system. To achieve this, the triangulation system can be mounted on various devices such as coordinate measuring machines (CMMs) or robotic arms. These systems enable controlled, repeatable movements and are equipped with encoders to track the exact position, ensuring the necessary precision for detailed and accurate surface scanning.

In these systems, three different types of trajectories are usually performed during the inspection: linear scans, area scans, and volumetric scans. These movements are always in a straight line, without following curves or more complex trajectories.

In linear scans, an initial point and a final point are indicated, and the system performs a linear movement between these points, potentially combining simultaneous movement in the three axes of the machine, see Figure 2.11 (a).

In area scans, three points are specified: the starting point, a point defining the length of the scan, and another point specifying the width. Based on this information and the desired overlap between passes, the system calculates the necessary number of trajectories to cover the entire area and generates the paths for digitizing the surface, see Figure 2.11 (b).

Volumetric scans are similar to area scans, including a third point to indicate the height of the volume to be digitized. As with area scans, the system calculates the number of passes needed for the flat section of the volume according to the overlap value between passes. Additionally, depending on the volume to be digitized, the system calculates the number of planes that need to be scanned. This type of scan is used for surfaces with continuous height changes, where the sensor's depth of field is exceeded, making it necessary to perform multiple passes over the surface while varying the distance between the sensor and the piece.

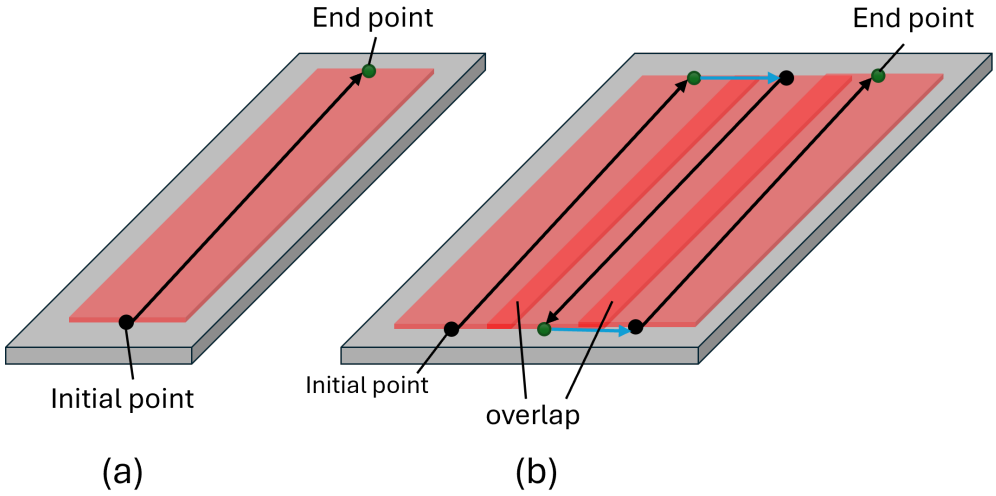


Figure 2.11: Comparison of different scanning techniques: (a) Linear scan, (b) Area scan.

Although non-contact sensors can measure with micrometer-level accuracy, they can still be affected by errors or noise, especially if there's movement in the robot or misalignment between the sensor and the piece. Because of this, it's important to carefully plan and align the scanning path to reduce these issues and ensure accurate results. The accuracy and reliability of laser triangulation sensors are highly dependent on proper sensor-surface alignment.

Key considerations include ensuring that the sensor is oriented perpendicular to the surface of the piece and positioned at the optimal distance for accurate measurement. In general, for optimal results, laser sensors should be mounted so the laser beam is as close to a perfect 90° angle to the surface as possible. If the laser sensors are not mounted at or near this angle, or if the measurable target is not perpendicular to the laser beam, the readings may be affected by cosine error. Additionally,

the sensor should be securely mounted to avoid vibrations, and proper calibration is essential to maintain precision throughout the scanning process. Further details on trajectory generation considerations will be discussed in chapter 6.

Also, laser triangulation systems are not well-suited for measuring surfaces with tight or narrow spaces, such as bore holes, blind holes, or surfaces with significant edges or contours. In such environments, the risk of occlusion between the laser beam and the detector is high, which can result in blocked signals and inaccurate readings. The technique relies on a clear, unobstructed path for the laser light to reach the surface and return to the detector, making it less effective for profiling uneven surfaces or moving targets. As a result, alternative measurement methods may be more appropriate for applications requiring precision in these challenging conditions.

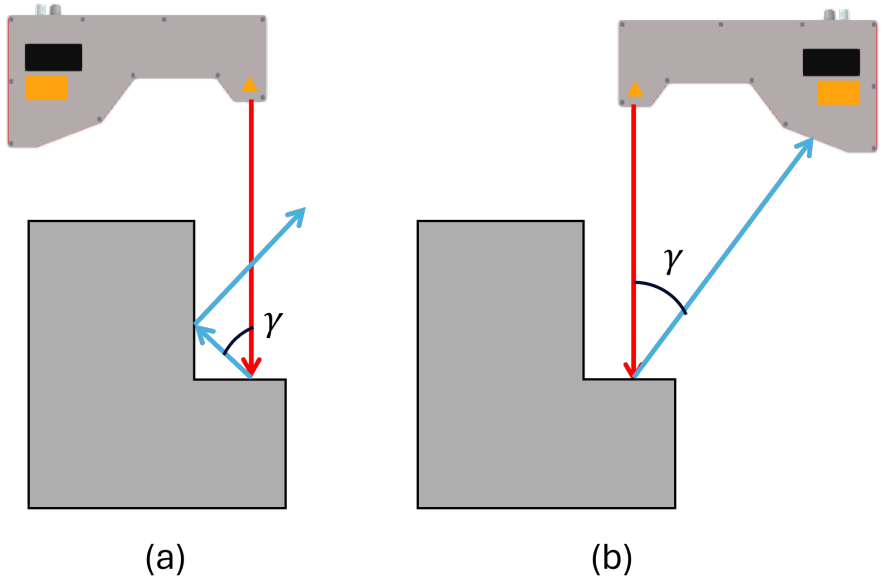


Figure 2.12: The triangulation angle and occlusion.

Chapter 3

Reinforcement Learning: Theoretical Foundations

This chapter provides an overview of the fundamental concepts in Reinforcement Learning (RL). The chapter begins with an introduction to the basic principles of RL, including the roles of agents, environments, rewards, and policies. Following this, the chapter presents a review of commonly used RL algorithms found in the literature.

The objective of this chapter is to establish a solid understanding of the core elements of RL and to familiarize readers with key algorithms that are prevalent in both research and practical applications within the field.

3.1 Basic principles

Reinforcement Learning (RL) is a branch of machine learning inspired by behavioral psychology, focusing on how agents make decisions in an environment to maximize some measure of accumulated reward over time [36]. The fundamental concepts of reinforcement learning include:

- **Agent**: It is the machine learning algorithm (or autonomous system) that interacts with the environment.
- **Environment**: It is the adaptive problem space with attributes such as vari-

ables, limits, rules, and valid actions.

- **Action**: Each action is a step that the RL agent takes to navigate the environment.
- **State**: It represents the environment at a given point in time.
- **Reward**: It is the positive, negative, or zero value that the agent receives as a consequence of an action, evaluating its quality.
- **Accumulated Reward**: It is the sum of all rewards obtained over time.

In this context, the agent dynamically interacts with the environment, observing its current state and selecting actions in response. These actions influence the state of the environment and generate a reward signal that guides the agent's behavior. The primary goal of the agent is to maximize the accumulation of these rewards over time, thus optimizing its performance in the environment. Figure 3.1 illustrates the basic interaction cycle between an agent and the environment.

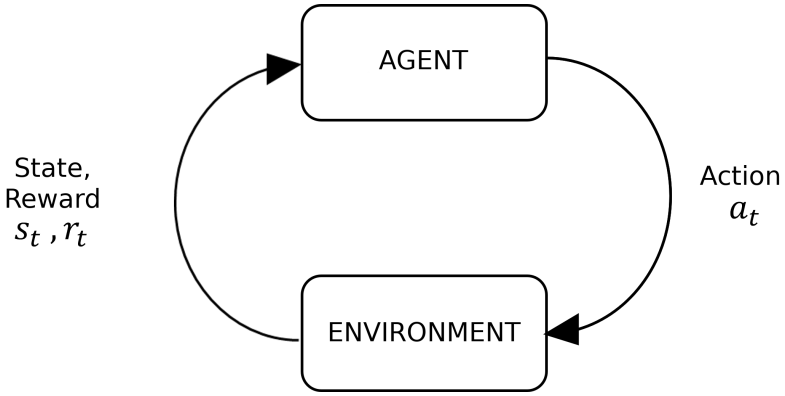


Figure 3.1: Agent-Environment Interaction Cycle

In most cases, the problem to be solved is formally modeled as a **Markov Decision Process** (MDP, *Markov Decision Process*). An MDP can be defined as a tuple of 5 elements $(\mathcal{S}, \mathcal{A}, r, P, \rho_0)$, representing, respectively, the set of all valid states, the set of all valid actions, the reward function, the transition probability function for the action-state set, and the initial state distribution ρ_0. Below, these parameters will be explained in more detail.

3.1.1 States

The set of all possible states in an environment is denoted as $\mathcal{S}$, and mathematically, it is defined according to Equation 3.1.

$$\mathcal{S} = \{s_1, s_2, ..., s_m\} \tag{3.1}$$

Each element $s_i \in \mathcal{S}$ represents a unique configuration of the environment at a specific point in time. States encapsulate all the necessary information that an agent needs to make decisions and determine subsequent actions.

3.1.2 Actions

Different environments allow for different types of actions. The set of all valid actions in a given environment is referred to as the action space $\mathcal{A}$. Mathematically, it is defined according to Equation 3.2, where a_i represents a specific action in the set, and n is the total number of actions.

$$\mathcal{A} = \{a_1, a_2, ..., a_n\} \tag{3.2}$$

The actions are divided into two main categories: continuous and discrete actions. Continuous actions are those where the agent can take an infinite number of possible values within a continuous range. In contrast, discrete actions are those where the agent can only choose from a finite set of options.

3.1.3 Reward and Return

The reward determines how good was the action a_t from state s_t to reach state s_{t+1}. The reward at each time step can be denoted as Equation 3.3, where R is the reward function depending on each state-action pair.

$$r_t = R(s_t, a_t, s_{t+1}) \tag{3.3}$$

Frequently, in the literature, equation 3.3 is simplified to just a dependence on the current state, $r_t = R(s_t)$, or state-action pair $r_t = R(s_t, a_t)$.

The goal of an agent in reinforcement learning is to maximize a measure of cumulative reward over a trajectory, denoted as $R(\tau)$. A trajectory τ represents a sequence of states and actions in the environment:

$$\tau = (s_0, a_0, s_1, a_1, ...) \tag{3.4}$$

The return is the total sum of rewards accumulated from the current state to the goal state. There are two main types of returns: the finite-horizon undiscounted return and the infinite-horizon discounted return.

Finite-horizon undiscounted return is the sum of reward from the current state to goal state which has a fixed timestep or a finite number of timesteps:

$$R(\tau) = \sum_{t=0}^{T} r_t \tag{3.5}$$

Infinite-horizon discounted return is the sum of all rewards ever obtained by the RL agent, but discounting factors determines how far future rewards need to be accounted. This formulation of reward includes a discount factor $\gamma \in (0, 1)$.

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t \tag{3.6}$$

3.1.4 Policy

An essential part of RL is the policy (π), which is a strategy or rule that guides the agent in making decisions within the environment. This policy can be deterministic (equation 3.7) or stochastic (3.8).

A deterministic policy implies that for each state of the environment, the agent selects a specific action in a predictable manner without randomness. Therefore, given a state s, the action a is simply the output of the policy function $\pi(s)$.

$$a_t = \pi(s_t) \tag{3.7}$$

On the other hand, a stochastic policy specifies the probability of selecting each action given a particular state. This means that instead of making completely deterministic decisions, the agent chooses actions with certain probabilities. The probability of choosing a particular action a in a state s is defined by the probability function $P(a_t, s_t)$. This introduces uncertainty into action selection, allowing the agent to explore different options and adapt its behavior based on the uncertainty of the environment.

$$\pi(a_t, s_t) = P(a_t, s_t) \tag{3.8}$$

The parameters of such a policy are frequently denoted by θ, and sometimes is indicated as a subscript on the policy symbol to emphasize the relationship (π_θ).

3.1.5 The RL Problem

The primary objective of any reinforcement learning algorithm is to develop a policy that maximizes the expected cumulative reward when the agent interacts with the environment.

The expected return $J(\pi)$ is represented by Equation 3.9. The expected reward $\mathbb{E}_\pi[R(\tau)]$ is the average of the rewards the agent expects to receive by following policy π in each state s. The objective is to adjust the policy parameters to maximize this reward, using optimization methods to continuously improve the policy and the agent's performance in the specified task.

$$J(\pi) = \mathbb{E}_\pi[R(\tau)] \tag{3.9}$$

Therefore, the final optimization problem in an RL algorithm can be expressed as shown in equation 3.10, where π^* is the optimal policy:

$$\pi^* = argmax_\pi J(\pi) \tag{3.10}$$

3.1.6 Value Functions

In reinforcement learning, the value function is a fundamental tool that evaluates the quality of states or actions based on expected future rewards. This function can be of two main types: the state value function $V(s)$, which estimates the quality of being in a particular state s, and the action value function $Q(s, a)$, which estimates the quality of taking an action a in a state s.

The State-Value $V_\pi(s)$ is the expected total reward, starting from state s and acts according to policy π. If the agent uses a given policy pi to select actions, the corresponding value function is given by:

$$V_\pi(s) = \mathbb{E}_\pi \left[R(\tau) | s_0 = s \right] \tag{3.11}$$

The Action-Value Function is the expected return for an agent starting from state s and taking arbitrary action a then forever after act according to policy π:

$$Q_\pi(s, a) = \mathbb{E}_\pi \left[R(\tau) | s_0 = s, a_0 = a \right] \tag{3.12}$$

From these equations, the optimal versions can be derived. The Optimal State-Value function $V^*(s)$ provides the expected return when starting in state s and always following the optimal policy thereafter. Similarly, the Optimal Action-Value Function, $Q^*(s, a)$, represents the expected return when starting in state s, taking an arbitrary action a, and then following the optimal policy for the remainder of the process.

3.1.7 Bellman Equations

The Bellman equation establishes that the value of a state is equal to the expected immediate reward plus the discounted value of the next state. This relationship can be formulated for the State-Value function $V_\pi(s)$ as follows:

$$V_\pi(s) = \mathbb{E}_\pi [r(s, a) + \gamma V_\pi(s_{t+1})] \tag{3.13}$$

Here, $r(s, a)$ is the reward obtained by taking action a in state s, γ is the discount factor that determines the importance of future rewards, $V_\pi(s_{t+1})$ is the value of the

next state s_{t+1} that the agent transitions to, and $\mathbb{E}$ represents the expectation over possible next states and rewards.

Similarly, the Bellman equation for the action value function $Q_\pi(s, a)$ is defined as follows:

$$Q_\pi(s, a) = \mathbb{E}_\pi[r(s, a) + \gamma Q_\pi(s_{t+1}, a_{t+1})] \tag{3.14}$$

3.1.8 Advantage Functions

In RL, it is often more insightful to evaluate how much better a particular action is compared to others rather than assessing its absolute value. This concept is captured by the Advantage Function, which provides a measure of the relative quality of an action.

The advantage function, denoted as $A_\pi(s, a)$, associated with a policy π, quantifies the relative benefit of choosing a specific action a in a given state s compared to selecting an action randomly based on the policy $\pi(\cdot|s)$. It assumes that the agent will continue to act according to the policy π thereafter. Mathematically, the advantage function is expressed as:

$$A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s). \tag{3.15}$$

With the fundamental concepts of reinforcement learning established, the next section introduces and analyzes the different types of algorithms developed in this field.

3.2 RL Algorithms

It's complex to make a definitive classification of the various existing RL algorithms. Classifications can be made based on different parameters. If we consider the fundamental strategy that algorithms use to learn and improve the environment's performance, three main categories can be identified: Value-based algorithms, Policy-based algorithms, and Actor-Critic algorithms.

Value-based algorithms focus on estimating the value function of a policy or the optimal policy directly. The value function assigns a numerical value to each state or state-action pair, representing the expected reward of following a particular policy. These algorithms use methods like value iteration or temporal difference methods to update and enhance value function estimates over time. Once the value function is estimated, the agent can select actions that maximize the estimated value, thereby improving its performance in the environment. Common algorithms within this classification include Q-learning [37], DQN (Deep Q-Networks) [38], Double DQN [39], or SARSA (State–action–reward–state–action) [40].

On the other hand, **Policy-based algorithms** learn the policy directly without the need for a value function. These algorithms aim to optimize the policy directly to maximize accumulated rewards over time. They employ methods such as policy gradient or proximal policy optimization to gradually improve the policy through experience. Examples of policy-based algorithms include PPO (Proximal Policy Optimization) [41] and TRPO (Trust Region Policy Optimization) [42].

Actor-Critic algorithms are a type of reinforcement learning algorithm that combines elements of both value-based and policy-based approaches. These algorithms use two main components: an actor and a critic.

The actor is a neural network or function that represents the agent's policy, mapping each state directly to a specific action. Its primary objective is to learn an optimal policy that maximizes the expected long-term rewards.

The critic is a neural network or function that evaluates the action taken by the actor, estimating the expected reward from a specific state-action pair. Its goal is to improve the accuracy of these estimations, assisting the actor in refining its policy.

During the learning process, the actor and critic are updated jointly and collaboratively. The critic provides feedback on the quality of actions selected by the actor, while the actor uses this feedback to improve its policy. This feedback is used to compute the policy gradient, indicating how the actor's policy parameters should adjust to maximize expected rewards.

Some prominent examples of actor-critic algorithms include A2C/A3C (Advantage Actor-Critic) [43], SAC (Soft Actor-Critic) [44], DDPG (Deep Deterministic Policy Gradient) [45], or TD3 (Twin Delayed Deep Deterministic Policy Gradient) [46].

Another significant distinction is based on the nature of the agent's action space,

which is categorized into two main types: discrete action spaces and continuous action spaces.

In **discrete action spaces**, the agent can choose from a finite and defined set of actions in each state of the environment. This characteristic simplifies the decision-making process since the agent only needs to evaluate a limited number of options. Algorithms designed to work with discrete action spaces often employ tables or functions to estimate the quality of each action based on the current state. Examples of such algorithms include Q-Learning, DQN, or SARSA.

On the other hand, in **continuous action spaces**, the agent has the ability to take actions in an infinite or continuous range of values. This occurs in situations where actions cannot be enumerated or discretely quantified. Algorithms designed to handle continuous action spaces often require more complex approaches, such as the use of deep neural networks in the case of deep reinforcement learning algorithms. These algorithms use techniques like policy gradient methods, which directly map from states to actions, or actor-critic methods that combine value function and policy learning for environments with continuous actions. Examples of such algorithms include PPO, TRPO, A2C/A3C, SAC, DDPG, or TD3.

Next, the most commonly used RL algorithms in the literature will be presented.

3.2.1 Q-Learning

Q-Learning [37] is a reinforcement learning algorithm designed to find the optimal policy for decision-making problems by learning a Q-function, $Q(s, a)$, which represents the expected future rewards for taking action a in state s and subsequently following the optimal policy. The Q-function is stored in a **Q-table**, a matrix where each entry corresponds to a state-action pair, allowing the agent to keep track of the learned values.

The Q-function is updated using the following formula:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] \tag{3.16}$$

In this equation, α represents the learning rate, and γ is the discount factor that determines the importance of future rewards. The term $\max_a Q(s_{t+1}, a)$ estimates the maximum future reward achievable from the next state s_{t+1}, guiding the agent

towards actions that lead to higher long-term rewards.

The Q-table is initially filled with arbitrary values and is iteratively updated as the agent interacts with the environment, receiving rewards and experiencing state transitions. This process involves balancing exploration (trying new actions) and exploitation (selecting known actions that yield high rewards). A common strategy for this is the ϵ-greedy policy, which chooses actions randomly with probability ϵ and selects the best-known action with probability $1 - \epsilon$.

Over time, the algorithm converges to the optimal Q-values, $Q^*(s, a)$, provided that every state-action pair is visited sufficiently often and the learning rate decreases over time. The optimal policy $\pi^*(s)$ is derived by selecting actions that maximize the Q-value for each state:

$$\pi^*(s) = \arg\max_a Q^*(s, a) \tag{3.17}$$

By following this policy, the agent can achieve the highest expected cumulative reward in the environment. The Q-table thus serves as a crucial component, guiding the agent's decisions and enabling the learning of the optimal strategy.

3.2.2 SARSA

SARSA (State-Action-Reward-State-Action) [47] is a reinforcement learning algorithm that aims to learn a Q-function, $Q(s, a)$, which estimates the expected future rewards of taking action a in state s and then following a specific policy.

The Q-function is updated using the following formula:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right] \tag{3.18}$$

In this equation, s_t and a_t represent the current state and action, r_t is the reward received, s_{t+1} is the next state, and a_{t+1} is the action selected by the policy in the next state. The parameters α and γ are the learning rate and discount factor, respectively. The update uses the action a_{t+1} that is actually taken in the next state, aligning with the current policy.

SARSA employs a **Q-table** to store and iteratively update the estimated Q-

values for each state-action pair. This table starts with arbitrary values and is refined through interactions with the environment. To balance exploration and exploitation, the agent typically uses an ϵ-greedy policy, which selects actions randomly with probability ϵ and follows the current policy with probability $1 - \epsilon$.

As learning progresses, SARSA converges to the optimal Q-values, $Q^*(s, a)$, provided that all state-action pairs are sufficiently explored and the learning rate decays appropriately. The optimal policy $\pi^*(s)$ is derived by selecting the action that maximizes the Q-value for each state. By following this policy, the agent can achieve the highest expected cumulative reward:

$$\pi^*(s) = \arg\max_a Q^*(s, a) \tag{3.19}$$

The main distinction between Q-Learning and SARSA lies in their Q-value update mechanisms. Q-Learning updates Q-values using the maximum future rewards, which generally accelerates convergence to the optimal policy. In contrast, SARSA updates Q-values based on the actual actions taken, which makes it more responsive to the exploration strategy of the current policy.

3.2.3 Deep Q-Network

Deep Q-Network (DQN) [48] is an advancement of Q-learning that uses deep neural networks to approximate the Q-value function, which represents expected future rewards for actions in specific states. Traditional Q-learning uses a table to store Q-values, but this approach becomes impractical for large or continuous state spaces. DQN overcomes this by employing a neural network to estimate Q-values, where the input is the state and the output is the Q-values for all possible actions.

DQN incorporates experience replay and a target network. Experience replay involves storing past experiences (state, action, reward, next state) in a replay buffer and sampling random batches from this buffer during training. This technique helps break correlations between consecutive experiences and improves training efficiency. The target network, which is a periodically updated copy of the main Q-network, helps stabilize the learning process by providing consistent target Q-values.

The Q-values are updated using the Bellman equation:

$$y = r + \gamma \max_{a'} Q'(s', a') \tag{3.20}$$

where r is the reward received, γ is the discount factor, Q' represents the target network, s' is the next state, and a' denotes possible actions in s'.

The loss function used to train the neural network is:

$$L = \frac{1}{N} \sum_{i} [y_i - Q(s_i, a_i)]^2 \tag{3.21}$$

where N is the number of samples in the batch, (s_i, a_i) is the state-action pair at step i, and y_i is the target value for that pair. The network weights are updated using gradient descent to minimize this loss function.

3.2.4 Policy Gradient Methods

Policy gradient methods focus on directly modeling and optimizing the policy. Typically, the policy is represented by a parameterized function in relation to θ. The value of the reward (objective) function is dependent on this policy, and various algorithms can be used to optimize it for the highest reward.

The gradient of the objective function $J(\theta)$ is given by the equation 3.22, where $\pi(a_t|s_t)$ denotes the probability of taking action a_t given state s_t, parameterized by θ. A_π is an estimator of the advantage function for the current policy, and $\mathbb{E}_\pi$ represents the empirical expectation over a batch of samples.

$$\nabla_\theta J(\theta) = \mathbb{E}_\pi \left[\nabla \log \pi(a_t|s_t) A_\pi \right] \tag{3.22}$$

The policy gradient algorithm works by updating policy parameters via stochastic gradient ascent on policy performance:

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta J(\pi_{\theta_t}) \tag{3.23}$$

In this update rule, θ_t represents the policy parameters at iteration k, and α is the learning rate controlling the step size. The new parameters θ_{t+1} are computed by

adding a fraction α of the gradient of the objective function $\nabla_\theta J(\theta)$ to the current parameters θ_t. This iterative process aims to increase the expected return by adjusting the policy parameters in the direction that improves the performance according to the policy gradient estimate.

3.2.5 Trust Region Policy Optimization (TRPO)

Trust Region Policy Optimization (TRPO), introduced by Schulman et al. in [42], is an advanced policy gradient method that improves the stability of policy updates. To ensure stable training, TRPO avoids making large policy changes in a single update by applying a Kullback-Leibler (KL) divergence constraint [49], which limits how much the policy can change at each step.

TRPO seeks to maximize a surrogate objective function, which is given by the equation 3.24, where π_θ represents the new policy, $\pi_{\theta_{old}}$ is the old policy, and A_π is the estimator of the advantage function. This objective function is designed to improve policy performance while controlling the magnitude of policy changes.

$$J(\pi) = \mathbb{E}_\pi \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} A_\pi \right] \tag{3.24}$$

To ensure stability, TRPO includes a constraint on how much the new policy can differ from the old policy. This constraint is expressed using KL-Divergence and ensures that the change between the old and new policies is within a predefined limit δ:

$$\mathbb{E}_\pi[\mathrm{KL}(\pi_{\theta_{old}}(a_t|s_t)|\pi_\theta(a_t|s_t))] \leq \delta \tag{3.25}$$

This constraint helps prevent large, destabilizing updates by keeping the new policy close to the old one, within a "trust region".

3.2.6 Proximal Policy Optimization (PPO)

The Proximal Policy Optimization (PPO) algorithm, introduced by Schulman et al. in [41], is a policy gradient technique designed as an improvement over Trust Region

Policy Optimization (TRPO). It simplifies and accelerates the training process by using first-order gradients and a clipped surrogate objective function that stabilizes policy updates.

The probability ratio $r(\theta)$ is the probability of selecting an action under the new policy divided by the probability of selecting the same action under the old policy, specifically:

$$r(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \tag{3.26}$$

With this definition, the objective function for TRPO simplifies to:

$$J^{TRPO}(\theta) = \mathbb{E}_\pi \left[r(\theta)\hat{A}_\theta \right] \tag{3.27}$$

Proximal Policy Optimization (PPO) improves stability in policy optimization by employing a clipped surrogate loss function, which penalizes excessive policy changes. This approach prevents significant divergence between new and old policies, thereby stabilizing the training process. Without such constraints, maximizing J^{TRPO} could lead to instability due to excessively large parameter updates and significant policy ratios. To address this, PPO introduces a constraint that ensures $r(\theta)$ remains within a narrow interval around 1, specifically $[1-\epsilon, 1+\epsilon]$, where ϵ is a hyperparameter. The clipped objective function of PPO is defined in Equation 3.28.

$$J^{CLIP}(\pi) = \mathbb{E}_\pi \left[\min \left(r(\theta)A_\pi, \text{clip}\left(r(\theta), 1-\epsilon, 1+\epsilon\right) A_\pi \right) \right] \tag{3.28}$$

Where π represents the policy, $\mathbb{E}$ denotes the expectation over time, r is the probability ratio under the new and old policies, respectively, A_π is the estimated advantage, and ϵ is a hyperparameter controlling how much the new policies are allowed to differ from the old policies during the optimization process. It is used to compute a penalty function that limits the policy change in each optimization iteration. Here, $\text{clip}(x, a, b)$ restricts the value of x to be between a and b, as defined in Equation 3.29.

$$\text{clip}(x, a, b) = \begin{cases} a \text{ if } x \leq a \\ b \text{ if } x \geq b \\ x \text{ else} \end{cases} \tag{3.29}$$

When applying PPO to a network architecture where both the policy (actor) and value (critic) functions share parameters, the objective function is extended beyond the clipped reward. It includes an additional value estimation error term J^{VF} and an entropy bonus J^{ENT} to promote adequate exploration.

Specifically, the value estimation error term J^{VF} quantifies the discrepancy between the predicted value function and the target value, typically computed using a squared error loss according equation 3.30, where $V_\pi(s_t)$ is the predicted value of state s_t under the current policy, and V_t is the target value, which is usually derived from rewards or the advantage function.

$$J^{VF}(\pi) = \mathbb{E}_\pi \left[\left(V_\pi(s_t) - V^{target} \right)^2 \right] \tag{3.30}$$

The entropy bonus J^{ENT} is designed to promote exploration by penalizing low-entropy policies, thus helping to avoid premature convergence to suboptimal policies.

The complete objective function is given by Equation 3.31, where c_1 and c_2 are hyperparameter constants that weight the importance of the value error and entropy terms, respectively.

$$J^{PPO}(\pi) = \mathbb{E}[J^{CLIP}(\theta) + c_1 J^{VF} + c_2 J^{ENT}] \tag{3.31}$$

3.2.7 Deep Deterministic Policy Gradient (DDPG)

Deep Deterministic Policy Gradient (DDPG), introduced by Lillicrap et al. [45], is an actor-critic reinforcement learning technique designed for environments with continuous action spaces.

As an actor-critic algorithm, DDPG concurrently learns both a Q-function and a policy. It employs two primary neural networks: Actor Network, that determines the optimal policy by generating a deterministic action $a = \pi(s)$ based on the current state, and Critic Network, that assesses the action-value function $Q(s, a)$ for specific state-action pairs.

DDPG also incorporates target networks, which are delayed replicas of the actor and critic networks. These target networks enhance stability in learning by reducing the risk of errors that can occur when the networks rely too heavily on their own

outputs.

The value network is updated using the Bellman equation to compute the updated Q-value. The Q-values for the next state are obtained through the target networks, and the mean squared loss between the updated Q-value and the original Q-value is minimized:

$$\text{Loss} = \frac{1}{N} \sum_{i} \left(Q(s_t, a_t) - \left(r(s, a) + \gamma \max_{a} Q_{\text{target}}(s_{t+1}, a_{t+1}) \right) \right)^2 \tag{3.32}$$

For the policy function, the goal is to maximize the expected return:

$$J(\pi) = \mathbb{E}_{\pi} \left[Q(s, a) \mid_{s=s_t, a_t=\pi(s_t)} \right] \tag{3.33}$$

DDPG is based on a deterministic policy gradient, where the policy is modeled as a deterministic function that maps states directly to actions. To address the challenge of limited exploration inherent in deterministic approaches, DDPG incorporates additive noise to the deterministic actions. This exploration mechanism is implemented through an Ornstein-Uhlenbeck process [50], which adds temporally correlated noise to the actions, creating more varied and effective training trajectories. The final action executed by the agent is expressed as:

$$\mu'(s) = \mu_\theta(s) + \mathcal{N} \tag{3.34}$$

where $\mathcal{N}$ represents the noise process, characterized by its mean, variance, and correlation factor parameters.

To update the parameters of the actor and critic networks, DDPG utilizes a technique known as a **replay buffer**. During training, all experience tuples (s_t, a_t, r_t, s_{t+1}) are stored in a finite-sized memory. Random batches of experiences are then sampled from this memory to update the networks, enhancing the learning efficiency and stability of the algorithm. This combination of techniques makes DDPG particularly effective for reinforcement learning environments with continuous action spaces.

3.2.8 TD3

Twin Delayed Deep Deterministic Policy Gradient (TD3), introduced by Fujimoto et al. in [46], improves upon the Deep Deterministic Policy Gradient (DDPG) algorithm by addressing its tendency to overestimate Q-values.

A common issue with DDPG is its tendency to overestimate Q-values, leading to policies that exploit these inaccuracies and ultimately break. To overcome these challenges, TD3 introduces three key techniques that effectively reduce overestimations and improve the stability and reliability of the learning process: Clipped Double-Q Learning, Target Policy Smoothing and Delayed Policy Updates.

TD3 used a total of six neural networks, namely, two critics Q_1 and Q_2 with parameters ϕ_1 and ϕ_2, two critic targets Q'_1 and Q'_2 with parameters ϕ'_1 and ϕ'_2 and an actor π and corresponding target π' with parameters θ and θ' respectively.

In addressing overestimation bias, TD3 refines the way the TD-target is computed compared to DDPG. In DDPG, the target actor network predicts the action a' for the next state s', and this action is used to compute the Q-value with the target critic network:

$$y = r + \gamma Q'(s', \pi'(s')) \tag{3.35}$$

In contrast, TD3 builds on concepts introduced in Double Q-learning. Double Q-learning uses two separate value estimates, such that each Q-value is updated using the estimate of the other one, as shown below:

$$\begin{aligned} y_1 &= r + \gamma Q_2(s', Q_1(s', a))) \\ y_2 &= r + \gamma Q_1(s', Q_2(s', a))) \end{aligned} \tag{3.36}$$

The formulation of Double Q-Learning relies on the assumption that Q_1 and Q_2 are completely independent and are updated using separate sets of experiences, which leads to an unbiased estimate. However, in the actor-critic setting of DDPG, a replay buffer is used to sample experiences for learning, and this condition of complete independence cannot be guaranteed. To address this issue, the authors propose a "clipped" version of Double Q-Learning which takes the minimum of the two Q-values to compute the target.

Additionally, to reduce computational cost, the use of two critics Q_1 and Q_2 with

their respective target networks Q'_1 and Q'_2 is suggested. However, there is only a single actor π that is optimized against Q_1. This results in generating a single TD-Target y, which is then used to update both Q_1 and Q_2.

Thus, the final formulation of the learning step for TD3 can be expressed as follows:

$$y = r + \gamma \min(Q'_1(s', \pi'(s')), Q'_2(s', \pi'(s'))) \tag{3.37}$$

In TD3, the learning step for both critics involves regressing to the TD-target. Each critic network aims to minimize a loss function that measures the difference between its predicted Q-value and the TD-target:

$$\begin{aligned} L_{\phi_1} &= \mathbb{E}\left[(y - Q_1)^2\right] \\ L_{\phi_2} &= \mathbb{E}\left[(y - Q_2)^2\right] \end{aligned} \tag{3.38}$$

Here, L_{ϕ_1} and L_{ϕ_2} represent the loss for the first and second critic, respectively. The expectation $\mathbb{E}[\cdot]$ denotes that the average is taken over a batch of experiences sampled from the replay buffer. The learning process involves adjusting the parameters of the critic networks (ϕ_1 and ϕ_2) to minimize the Mean Squared Error between the predicted Q-values and the TD-target y. By minimizing this loss, the critic networks learn to provide more accurate and less biased estimates of expected returns, thereby enhancing the actor's policy.

Lastly, the policy is learned just by maximizing Q_1:

$$\max_{\theta} \mathbb{E}\left[Q_1(s, \pi_\theta(s))\right], \tag{3.39}$$

TD3 addresses the issue of deterministic policies over-fitting to narrow peaks in the value function by incorporating target policy smoothing. This method involves computing the target action using the target policy $\pi_{\theta_{target}}$ with added clipped noise, ensuring the action stays within the valid range.

$$\begin{aligned} a'(s') &= clip(\pi'(s') + \epsilon)) \\ \epsilon &= clip(\mathcal{N}(0, \sigma), -c, +c) \end{aligned} \tag{3.40}$$

In TD3, the actor network is updated less frequently compared to the critic

networks. This is known as Delayed Policy Updates. The rationale behind this approach is to allow the critic networks to converge more accurately to their optimal values before updating the actor network, ensuring more reliable policy improvement steps. Typically, the actor is updated once every two critic updates, enhancing stability and performance.

3.2.9 SAC

Soft Actor-Critic (SAC) [44] is an advanced actor-critic algorithm designed for continuous action spaces. It operates within the maximum entropy reinforcement learning framework, aiming to find the optimal policy that maximizes both the expected long-term reward and the entropy of the policy. This balance between exploitation and exploration is controlled by the parameter α, which adjusts the relative importance of these two objectives.

$$J(\pi) = \sum_t \mathbb{E}_\pi \left[r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t)) \right] \tag{3.41}$$

Here, $\mathcal{H}(\cdot)$ denotes the entropy measure, and α regulates how much emphasis is placed on entropy. A higher α promotes more exploration by increasing the diversity of actions, while $\alpha = 0$ reverts to the conventional goal of maximizing expected rewards alone.

To optimize this objective, SAC uses three distinct neural networks that collaboratively enhance the agent's decision-making: State Value Function, Soft Q-Function and Policy Function.

The state-value function V_ψ, parameterized by ψ, estimates the expected return of a state. The objective is to minimize the mean squared error between this value function's prediction and the expected return from the Q-function, adjusted for the policy entropy. This is formalized as:

$$J_V(\psi) = \mathbb{E}_{s_t \sim \mathcal{D}} \left[\frac{1}{2} \left(V_\psi(s_t) - \mathbb{E}_{a_t \sim \pi_\theta} \left[Q_w(s_t, a_t) - \log \pi_\theta(a_t|s_t) \right] \right)^2 \right] \tag{3.42}$$

This formula signifies that for all states sampled from the experience replay buffer

$\mathcal{D}$, the goal is to minimize the squared difference between the predicted state-value and the adjusted Q-function value, which includes the policy entropy term.

The soft Q-function Q_w, parameterized by w, predicts the expected return of taking a specific action in a given state. It is trained by minimizing the squared difference between the predicted Q-value and the target value $\hat{Q}(s_t, a_t)$:

$$J_Q(w) = \mathbb{E}_{(s_t,a_t)\sim\mathcal{D}}\left[\frac{1}{2}\left(Q_w(s_t, a_t) - \hat{Q}(s_t, a_t)\right)^2\right] \tag{3.43}$$

where the target value is defined as:

$$\hat{Q}(s_t, a_t) = r(s_t, a_t) + \gamma\mathbb{E}\left[V_\psi(s_{t+1})\right] \tag{3.44}$$

This means that for each (state, action) pair in the experience replay buffer, the Q-function is adjusted so that its predictions align with the immediate reward plus the discounted expected value of the next state.

The policy function π_θ, parameterized by θ, determines the probability distribution over actions in a given state. The policy network is trained by minimizing the following objective:

$$J_\pi(\theta) = \mathbb{E}_{s_t\sim\mathcal{D}}\left[D_{KL}\left(\pi_\theta(\cdot|s_t)\,||\,\frac{\exp(Q_w(s_t,\cdot))}{Z_w(s_t)}\right)\right] \tag{3.45}$$

Here, D_{KL} represents the Kullback-Leibler divergence, which measures the difference between the policy distribution π_θ and the distribution derived from the Q-function normalized by a function $Z_w(s_t)$. This objective function aims to align the policy distribution with the optimal action distribution suggested by the Q-function, thereby improving policy performance.

3.2.10 A2C

The Advantage Actor-Critic (A2C) algorithm [51] is an actor-critic method that combines value-based and policy-based strategies. It employs two main components: an actor, which defines the policy $\pi(a|s)$, and a critic, which evaluates this policy by

estimating the value function $V(s)$. The actor adjusts the policy based on feedback from the critic.

The advantage function, $A(s, a)$, is used to measure how much better or worse an action a is compared to the average action in state s. It is defined as:

$$A(s, a) = Q(s, a) - V(s) \tag{3.46}$$

where $Q(s, a)$ is the action-value function and $V(s)$ is the state-value function. The advantage function guides the actor in learning which actions are preferable relative to the average.

The updates for the actor and critic are performed simultaneously. The critic's loss function, see equation 3.47, measures the error between the estimated value function $V(s)$ and the actual return R. Minimizing this loss helps the critic provide a more accurate estimate of the value function, improving feedback to the actor.

$$L_{\text{critic}} = \frac{1}{2} (R - V(s))^2 \tag{3.47}$$

The actor's loss function is designed to adjust the policy to maximize the advantage function $A(s, a)$, see equation 3.48. By minimizing this loss, the actor improves the probability of taking actions with higher advantages, thus refining the policy based on the critic's evaluations.

$$L_{\text{actor}} = -\log \pi(a|s) \cdot A(s, a) \tag{3.48}$$

Chapter 4

Simulation of a Laser Triangulation Profilometric Sensor

For a simulation tool to be truly effective, it must accurately replicate the behavior of the laser triangulation sensor, including its interactions with the inspected surfaces and the influence of environmental factors. This involves creating detailed models of the sensor's geometry, the properties of the surfaces being measured, and the noise effects that can impact the measurements.

In this chapter, we introduce a comprehensive simulation tool designed specifically for laser profilometry using triangulation methods. The primary goal is to achieve a realistic simulation of readings provided by any commercial sensor. The simulation aims to incorporate the speckle effect [52], which is the most relevant source of noise affecting the measurement process.

To achieve realistic simulation, a geometric model is proposed that defines sensor parameters based on characteristics typically found in the datasheet of any commercial sensor. This model allows estimation of distance measurements between the sensor and the inspected object. Furthermore, a noise model is developed that combines speckle noise with sensor uncertainty noise. Speckle noise is simulated using Perlin noise, mimicking its characteristic granular pattern. Conversely, uncertainty noise is modeled as Gaussian noise, considering the sensor's resolution and repeatability.

Additionally, the chapter presents obtained results, including simulations of scans on steel products with various geometries and surface characteristics. A comparative

analysis is conducted between simulated and real scans to assess simulation similarity and accuracy. Quantitative measures and visual comparisons are employed to evaluate the algorithms presented.

The content of this chapter has been published in the journal Sensors under the title "Simulation of Laser Profilometer Measurements in the Presence of Speckle Using Perlin Noise" [53].

4.1 Related work

Simulators have evolved into indispensable tools for designing and evaluating industrial inspection systems by providing virtual environments to model sensor interactions with inspected objects. These simulations offer insights into sensor behavior crucial for optimizing performance and validating designs tailored to specific industrial demands. Early efforts in this field concentrated on creating digital models to visualize and analyze how sensors, like laser triangulation systems, interact with complex geometries.

For instance, Cajal et al. [54] explored the use of CAD models to simulate laser triangulation sensors. Their work allowed for virtual recreation of the sensor's interaction with complex geometries, facilitating the detection of potential collisions and occlusions in a controlled, simulated environment.

The system modeling process involves comprehensive parameterization of various components essential for laser triangulation sensors simulation. This includes defining specific camera operational parameters such as focal length, principal point coordinates, and distortion coefficients, alongside sensor noise characteristics and spatial positioning. Optical components such as aperture settings are also integrated, and laser parameters like positioning and focus specifications are meticulously defined to ensure accurate digitalization of part surfaces.

They also consider various sources of uncertainty, such as laser stripe image misidentification, optical distortion uncertainty, sensor electronic noise, lack of repeatability in part positioning, and unsynchronized position tracking during image triggering. Each source of uncertainty is modeled as Gaussian noise with parameters μ and σ, independently affecting the accuracy of laser-scanned data.

Other work is presented by Abu-Nabah et al. in [55]. They employed commercial animation software to develop virtual models of laser triangulation systems. This

method provided a platform to generate synthetic images, which could be analyzed to extract depth profiles. The ability to simulate sensor behavior and assess its performance in virtual environments proved invaluable, particularly for validating new sensor designs or evaluating existing commercial sensors under industry-specific conditions, such as those found in the oil and gas sector. However, these simulations often lacked detailed noise modeling, focusing primarily on the geometric aspects.

For this tools to be truly effective, the simulator must accurately replicate the behavior of the inspection system using models that closely relate to real hardware. In this way, designers can assess system performance under various conditions, achieving precise and cost-effective solutions while reducing both development time and costs.

In the context of laser triangulation, an important challenge is the speckle effect [52], which constitutes one of the main sources of noise in the measurement process. Speckle is an interference effect caused by the microtopology of the inspected surface, due to the spatial coherence of the illumination source [52]. In [35], the influence of speckle on measurements is analyzed. When lasers are used as light sources on optically rough surfaces, speckle can occur due to diffraction. Optically rough surfaces exhibit spatial variations that cause reflected light rays to travel different distances, resulting in phase shifts. These rays with phase shifts can either reinforce or cancel each other out, generating a pattern of bright and dark areas in the reflected laser light.

Previous research on the simulation of laser triangulation sensors for part scanning has focused primarily on the geometrical aspects of the sensor, often neglecting the impact of material surface roughness on the measurements. Although simulations using CAD models can produce very accurate geometric representations, they typically neglect the speckle effect, a critical source of noise arising from the microstructural properties of the surface being scanned. This can result in significant discrepancies between simulated and actual sensor readings. Focusing solely on the ideal geometric configuration may result in accurate measurements in a theoretical sense, but it misses the main uncertainties encountered in real-world conditions, thereby reducing the practical effectiveness of these simulations. This issue becomes particularly important in applications like surface inspection and defect detection, where it is crucial to determine if variations in surface measurements are significant and suggest defects such as impact marks or surface cracks.

In modern defect detection, many algorithms now use machine learning or artificial intelligence techniques to identify and classify defects accurately [21]. Effective training of these algorithms requires simulating sensor data realistically, including

accounting for major noise sources like the speckle effect. Overlooking this effect in simulations can introduce biases and inaccuracies, which can compromise the performance of this algorithms in real-world scenarios. Therefore, integrating a comprehensive noise model into simulation frameworks not only improves data realism but also supports the development of robust and reliable defect detection systems.

The simulation of speckle has been addressed in some previous works. For instance, Mohammadikaji et al. [56] analyzed various sources of uncertainty associated with 3D measurement processes using laser triangulation profilometers. They highlighted the impact of speckle combined with quantization effects, sensor noise, sampling, and interpolation methods on measurement accuracy. Their approach involved modeling these effects as independent Gaussian random variables with zero mean, affecting the x and y coordinates of the laser line position. Their study proposed a practical approach to estimate the statistics, particularly the covariance matrix of these Gaussian variables. They illuminated a flat target surface with a laser to ensure a straight line formation and conducted image acquisitions from various distances and viewing angles. Within each image column, they interpolated peak intensities and analyzed deviations between detected peak locations and the fitted line to estimate measurement uncertainties. Initially, they used a conservative estimation method based on maximum observed deviations but suggested potential refinements in future research, focusing on sensor distance and orientation parameters.

The same research group proposed in [32] a method to obtain realistic simulated images produced by a laser triangulation profilometer. Their primary objective was to develop a comprehensive tool for evaluating and planning measurement systems during the development phase. This model included detailed simulation of optical components and the effects of wave optics. Specifically, they modeled speckle as a diffraction phenomenon induced by rough surfaces, which can induce significant variations in light phase. To calculate the resulting intensity from coherent light sources in the Fourier domain, they used the product of the estimated amplitude transfer function and the phase of the wave field. They employed precise CAD models of the object under inspection, including its bidirectional reflectance distribution function and corresponding roughness profile.

However, they faced a significant computational challenge due to the detailed modeling of physical phenomena. Simulating all the intricate interactions of light with the object's surface, including diffraction, interference, and scattering, is highly complex and computationally intensive. The accurate representation of speckle patterns and the precise calculation of light phase variations require extensive computational resources. As a result, rendering a single image required approximately 10 hours using eight cores at 3.7 GHz speed. This long processing time is a significant

drawback, limiting the practicality of the simulation tool for real-time or near-real-time applications. The heavy computational load highlights the balance between accurate physical modeling and the feasibility of the simulation process. Therefore, while the method provides highly realistic simulations, its application is limited by the long time needed to produce results, posing challenges for iterative design and testing processes in industrial settings.

Csencsics et al. [57] proposed a different approach to simulate speckle. In their simulation, they define all elements comprising the measurement system, including the geometry, characteristics of the laser source, the measured surface, and the imaging system. The proposed method integrates stochastic laser speckles with deterministic ray tracing for designing optical sensors. It defines system geometry including laser source characteristics, target surface, detector parameters, and imaging optics. Laser spots are approximated by N1 point sources distributed within the spot diameter and varying in height according to surface roughness, with intensity profiles following a Gaussian distribution. N2 rays are uniformly emitted over a solid angle matching the size of the imaging lens and traced through the optical path to compute phase at the detector. Out-of-plane target movements and lens parameters adjust the spot area and point sources accordingly.

Other authors have used complex ray-tracing methods to model laser triangulation profilometers. In their study, Beermann et al. [58] introduced a simulation model for laser triangulation measurement under inhomogeneous refractive index fields (RIF). Employing a virtual camera and a complex multi-step ray tracing optimization, they simulated the complete triangulation process. The research focused on analyzing the impact of RIF on measurement accuracy by numerically calculating RIF through heat transfer simulations and modeling the virtual sensor using a camera pinhole model. They found that object geometry significantly affects laser point displacement and RIF-induced light deflection effects, emphasizing the challenges and benefits of optical inspection in multistage forming of hot workpieces. The study concluded by proposing strategies such as adjusting sensor pose and exploring lateral measurement methods to mitigate RIF effects and enhance measurement precision, providing valuable insights for designing compensation routines aimed at improving measurement accuracy under complex optical conditions.

Physically based simulation methods can be very useful when the objective is to design and validate new sensors. As more steps are physically modeled, it is possible to gain more insight into how the sensor performs and the bottlenecks to achieve the desired performance. It may be possible to measure the impact of each component in the final result. Nevertheless, such insight information is useless when the designers want to use an off-the-shelf sensor. In this case, the sensor is a black box that

provides depth information. The designer will receive from the camera an array of depth values, where each row corresponds to a single line scan. This information must be processed to extract higher-level information, such as a geometric model or a decision about the presence of defects in the inspected part.

In this thesis, a proposal to directly simulate the depth measures produced by an off-the-self laser triangulation profilometric sensor is presented. Using information provided by the sensor's manufacturer, a geometrical model is constructed to accurately represent the surface geometry of the part being inspected using an STL (Standard Tessellation Language) CAD model—a widely adopted format in manufacturing for digital object representation. This model facilitates the simulation of the scanning process, incorporating the trajectory of the sensor relative to the part's surface. Once the simulation is completed, a set of simulated profiles, organized as a bi-dimensional array, is generated to mimic the output format typical of commercial sensors.

A Gaussian noise model is added to the geometrical depth value to take into account the sensor's depth resolution and precision. The main contribution of this work is to propose the use of Perlin noise [59, 60] to model the effect of all the different physical noise sources present in the scanning process, especially the uncertainty due to speckle. This enables the development of a realistic simulation of the sensor that captures its characteristics and the effect of noise sources. This enhances the effectiveness of the simulation for developing, assessing, and optimizing the algorithms that must analyze this information.

Perlin noise is a type of procedural lattice gradient noise initially developed for creating procedural textures in computer graphics. It has been widely employed in film for enhancing the realism of computer-generated images and in video games to generate terrains procedurally. In computer graphics, "noise" denotes a type of random number generator, and "procedural" indicates that the random value at any given point is computed algorithmically. Instead of directly simulating the physical process responsible for speckle effects in images captured by sensor cameras, Perlin noise is utilized to add randomized values to the depth image produced by an ideal sensor. The method also includes techniques for adjusting the parameters of the Perlin noise algorithm. For a comprehensive overview of procedural noise functions commonly used in computer graphics, refer to surveys like [61, 62].

While specific studies on the application of Perlin Noise to simulate certain aspects of sensor noise were not found in the literature, Perlin Noise has been extensively utilized across various domains. Li et al. [63] introduced a method for simulating water surfaces' reflection and refraction using multi-octave Perlin noise

combined with ray tracing to generate random height fields.

Acosta et al. [64] use Perlin Noise to simulate oxide textures for developing a rust detection model. They generate a variety of 2D color images with realistic rust patterns by adjusting Perlin Noise parameters. This method allows them to create synthetic rusted surfaces that closely resemble real-world corrosion conditions. By combining a rust base texture with a metallic map, they produce diverse rust textures using Perlin Noise, optimizing parameters for computational efficiency and texture fidelity.

In the paper [65],Conde-Rodríguez et al. use Perlin Noise to model material microstructures. This method enables precise definition of how primary material phases are distributed within heterogeneous solids. It employs a material distribution function to compute the volume occupied by each primary material and a modified Perlin noise function to specify the shape and size of these material phases.

Additionally, Koutsoudis et al. [66] proposed enhancing photogrammetric 3D reconstruction on featureless surfaces using noise function-based patterns, including Perlin noise as one of their methods.

These studies highlight the versatility of Perlin Noise in simulating and enhancing visual and structural characteristics across various applications, demonstrating its potential for realistically modeling physical noise effects in scanning processes, thereby improving sensor simulations for effective algorithm development and optimization.

4.2 Proposed Method

Our goal is to create a realistic simulation of readings from an off-the-shelf laser profilometric sensor, specifically addressing the influence of speckle noise, which is a significant source of interference in measurement processes. Speckle noise arises due to light scattering interference on rough surfaces, particularly affecting laser triangulation profilometric sensors. These sensors detect fluctuations in the phase and amplitude of reflected light caused by surface irregularities, resulting in a distinctive granular pattern known as speckle in the laser line image. This pattern introduces errors in determining the center point of the laser line, thereby affecting the accuracy of distance estimation to the object.

This study introduces a simulation model comprising three key components: a

geometric model of the scene and two noise models. One model simulates the impact of speckle, while the other accounts for the sensor's inherent accuracy limitations.

$$\hat{r}(x,y) = d(x,y) + PN(x,y) + P_G(0,\sigma) \tag{4.1}$$

Equation 4.1 present the proposed model. Here, $\hat{r}(x,y)$ represents the simulated global reading, where $d(x,y)$ denotes the distance computed from the geometric scan model, $PN(x,y)$ models the effect of speckle noise on the reading, and $P_G(0,\sigma)$ represents the error due to the sensor's accuracy constraints. The variables x and y correspond to the position along the laser line and the scan line, respectively.

4.2.1 Geometrical Model

In this section, we describe the proposed geometrical model. The objective is to provide the distance profile between the sensor and the surface of the object. By modeling the interaction between the laser triangulation sensor and the target surface, we aim to simulate realistic scanning conditions.

The 3D model of the inspected piece is loaded as a triangular mesh, describing a surface using unit normals and triangle vertices. The profile acquired in each measurement corresponds to the intersections between the projected rays of the laser and the triangulated 3D model.

First, the parameters of the laser triangulation sensor, including the working distance (W_d), Z-range, Field of View (FOV), and points per profile (ppp), are set up. These parameters define the geometry and resolution of the scanning process. This was discussed in detail in Chapter 2. In figure 4.1, a scheme with these key parameters is shown.

The working distance denotes the distance from the camera to the center of the depth measurement range and is considered the optimal operating point where the laser achieves its sharpest focal point. The Z-range defines the difference between the sensor's maximum and minimum measurement distances. Consequently, to obtain valid measurements, the scanned surface must lie within the range $W_d \pm \frac{Z_{range}}{2}$.

The Field of View of the camera represents the laser beam's aperture angle and the maximum opening of the sensor, defining the angular limits of the projected beam. It is calculated from the X-FOV and the working distance. X-FOV represents

the total length of the measurement line at the working distance. Therefore, a simple trigonometric calculation leads to:

$$FOV = 2 \arctan\left(\frac{\text{X-FOV}}{2W_d}\right) \tag{4.2}$$

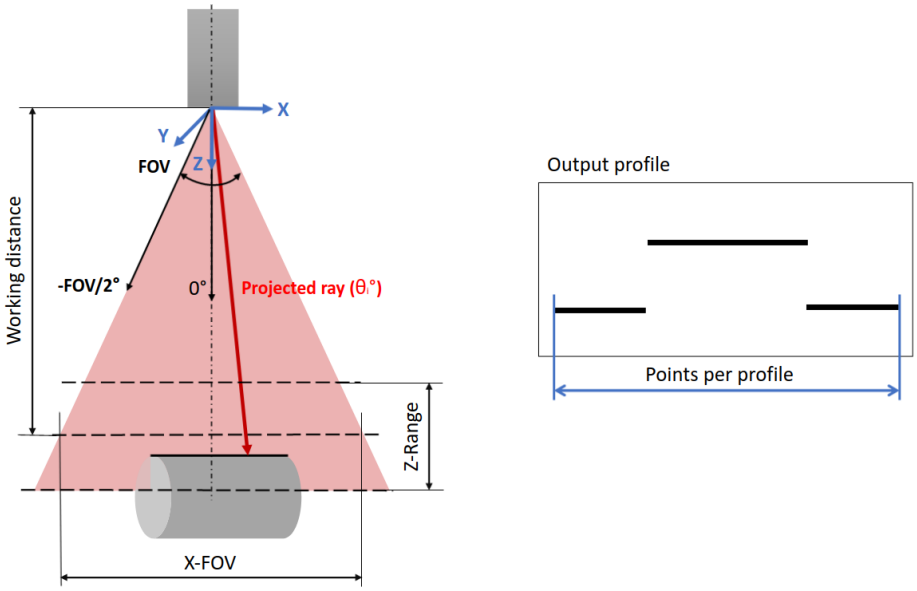


Figure 4.1: Laser triangulation sensor: Key parameters scheme.

The laser beam is divided into multiple rays, each corresponding to a point in the profile and uniformly distributed across the Field of View (FOV). To ensure comprehensive coverage of the scanning area, the angle of projection for each ray is calculated based on the FOV and the total number of points per profile, using the equation 4.3, where $\Delta\theta = \frac{\text{FOV}}{ppp-1}$ is the angular step between each ray. Consequently, each ray is constructed with a specific origin and direction: the origin is determined by the laser source's position, while the direction is precisely calculated based on the Θ_i angle.

$$\theta_i = -\frac{\text{FOV}}{2} + i \cdot \Delta\theta \quad \forall i \in [0, ppp) \tag{4.3}$$

For each projected ray, we use the Möller-Trumbore algorithm [67] to calculate the intersection between the ray and the triangular mesh of the 3D model. This

algorithm is a technique used in computer graphics to determine if a ray intersects a triangle in 3D. It calculates the barycentric coordinates of the intersection point, based on the parametric representation of the ray and the plane equation of the triangle. It solves a system of linear equations using Cramer's rule to obtain these coordinates, which are then used to determine if the intersection point lies inside the triangle.

From this intersection, we derive the distance d from the origin of the projected ray to the point where the ray intersects the surface. This distance provides the basis for simulating the laser triangulation sensor's measurement. In the scanner's coordinate system, a point $(x_s, 0, z_s)$ corresponds to the laser line's position x_s and the distance z_s along the projected ray. The validity of z_s is crucial as it must satisfy Equation 4.4, ensuring it falls within the acceptable range defined by the sensor's working distance W_d and Z-range. If z_s does not meet this criterion, the measurement is considered invalid, and a default value indicating no measurement is assigned to that point.

$$W_d - \frac{Z_{range}}{2} \leq z_s \leq W_d + \frac{Z_{range}}{2} \tag{4.4}$$

However, a single profile is insufficient for comprehensive surface quality analysis using profilometric sensors. Therefore, relative movement between the sensor and the object is typically employed during inspections. This movement results in the generation of a 2D image where each row corresponds to a profile acquired by the sensor. The dimensions of this image depend on the number of points per profile and the total number of profiles scanned during the acquisition. Each pixel in the image represents a distance measurement captured by the profilometric sensor.

To replicate this process in simulation, we use the relative trajectory followed by the sensor with respect to the object. The trajectory is defined by a sequence of sensor position values and sensor velocity between two consecutive poses. We use the maximum profile acquisition rate in Hz to calculate the intermediate position for each individual profile simulation. This parameter is also available in the sensor datasheet and represents the number of complete 3D profiles per second provided by the sensor. Its inverse is the time between two consecutive acquisitions. Using this time and the sensor's traversal speed, we compute each intermediate pose through linear interpolation.

From these simulated sensor positions, the 2D image of distances $d(x, y)$ is generated by computing distance measurements for each profile. Here, x represents the

points per profile, and y corresponds to each profile scanned during the inspection process. Thus, the size of the image is determined by the number of points per profile and the total number of profiles scanned during acquisition. Each pixel in the image corresponds to the distance measured by the profilometric sensor.

4.2.2 Speckle Noise Model: Perlin Noise

The inspected surface will always present a certain degree of optical roughness that will cause speckle when is illuminated with a coherent light source. This effect is the main source of error in the measurement. We model this effect by adding a Perlin Noise term to the value produced by the geometric model.

To realistically simulate speckle noise in the scanning process, we start by exploring the foundational theory of Perlin Noise, a gradient noise function known for generating smooth, pseudo-random variations. Following this theoretical overview, we adapt Perlin Noise to simulate speckle effects in our sensor model by adjusting its parameters based on the surface roughness of the object being scanned. This enables us to replicate the optical disturbances due to surface irregularities and accurately reflect the noise dynamics encountered in real-world laser scanning scenarios.

Perlin Noise

Perlin noise [59] is a type of gradient noise originally used in computer graphics to create procedural textures with a pseudo-random appearance. It is defined by the composition of multiple scaled versions of the same noise function, with different frequencies (f_m) and amplitudes (A_m), known as octaves. It is described according to equation 4.5, where m is the m^{th} noise function being added and $n_{octaves}$ is the number of octaves.

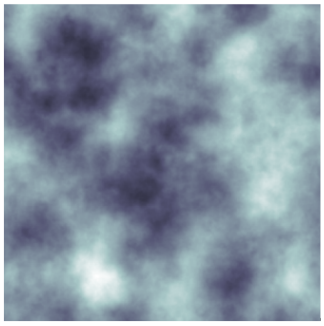


Figure 4.2: Perlin noise image.

$$PN(x,y) = \sum_{m=1}^{n_{octaves}} A_m N_m(f_m x, f_m y) \tag{4.5}$$

The level of detail and complexity in the generated texture or pattern is determined by the frequency and amplitude of each octave. Frequency refers to the number of cycles of the noise function within a given unit of space, and higher frequency values result in more rapid variations, producing a more intricate pattern. The frequency of each octave is typically set by multiplying the base frequency (f_0) by a factor of 2 for each successive octave, expressed as $f_m = f_0 \cdot 2^{m-1}$.

On the other hand, amplitude determines the range of values that the noise function can output. Higher amplitude values produce more pronounced peaks and valleys, resulting in a more dynamic pattern. The amplitude of each octave is usually determined by a persistence factor p, expressed as $A_m = A_0 \cdot p^{m-1}$.

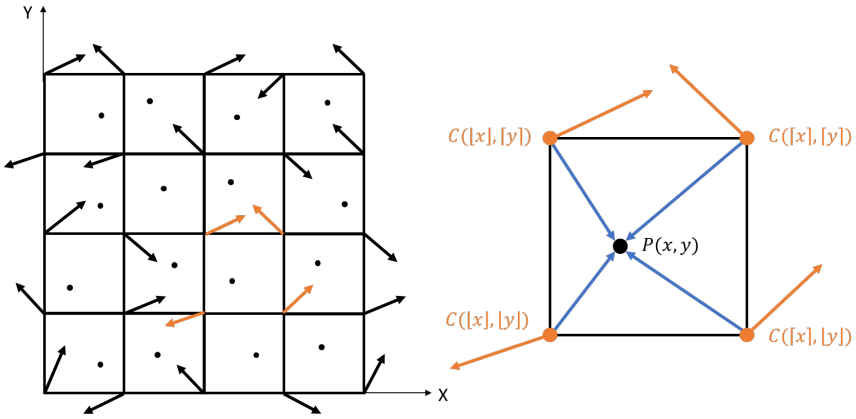


Figure 4.3: 2D Perlin noise generation process. Left image shows the 2D grid with the pseudo-random gradient vectors for each control point. Right image represents the process in one input coordinate $P(x, y)$. $C(x_i, y_j)$ are the corners of the grid, blue arrows represent the displacement vectors $\vec{d}(x_i, y_j)$ and orange arrows the gradient vectors.

To implement a 2D Perlin noise function, the initial step involves defining a 2D grid of control points. For each control point, a fixed pseudo-random gradient vector of unit length is generated. To compute the value at a point $P(x, y)$, it is necessary to determine the control points lying on the corners of the cell where the point falls $C(x_i, y_j)$. Where indices (i, j) denote the 4 corners of the cell, and i and j are either the floor or the ceil of x and y respectively, as shown in Figure 4.3. Then, the position vector of the point $P(x, y)$ with respect to each corner, $(\vec{d}(x_i, y_j))$, is computed. Equation 4.6 shows this calculation.

$$\begin{aligned} \vec{d}(x_i, y_j) &= C(x_i, y_j) - P(x, y) \\ i &\in \{\lfloor x \rfloor, \lceil x \rceil\} \\ j &\in \{\lfloor y \rfloor, \lceil y \rceil\} \end{aligned} \tag{4.6}$$

For each corner, the dot product between its gradient and displacement vectors is calculated to get the influence values $n(x_i, y_j)$ for point $P(x, y)$.

$$n_{ij} = \vec{g}(x_i, y_j) \cdot \vec{d}(x_i, y_i) \tag{4.7}$$

The calculation of the 2D Perlin Noise function at point $P(x, y)$ is done according

to equation 4.9. An interpolation between the 4 dot products is performed to estimate the blending of the noise contribution from the four corners, where n_0 and n_1 are the interpolations between n_{00}, n_{10} and n_{01}, n_{11}, respectively, see equation 4.8.

Interpolation is performed using the improved version of the general form of the *smoothstep* function [60], shown in equation 4.10. This function has zero first and second derivatives in the edges, and this ensures a smooth transition between neighbouring cells, giving Perlin noise its characteristic look.

$$\begin{aligned} n_0 &= n_{00} + smoothstep(P_x - C_{x_0}) \cdot (n_{10} - n_{00}) \\ n_1 &= n_{01} + smoothstep(P_x - C_{x_0}) \cdot (n_{11} - n_{01}) \end{aligned} \tag{4.8}$$

$$N(x, y) = n_0 + smoothstep(P_y - Cy_0) \cdot (n_1 - n_0) \tag{4.9}$$

$$smoothstep(x) = \begin{cases} 0 & x \leq 0 \\ 6x^5 - 15x^4 + 10x^3 & 0 \leq x \leq 1 \\ 1 & x \geq 1 \end{cases} \tag{4.10}$$

Perlin noise modeling with roughness parameters

Based on the properties of Perlin noise, it can be modeled in a way that, by adjusting its parameters appropriately, allows for the simulation of the Speckle noise found in real measurements These parameters are: The size and the number of control points of the 2D grid, the signal amplitude, the number of octaves, and the persistence factor. In this subsection, we will describe how to select an appropriate set of values to achieve a realistic simulation. Through this approach, a 2D Perlin noise function is estimated and subsequently incorporated into the full scan originating from the geometric model.

Given the variability of noise in each real measurement, the aim is not to reproduce point-by-point the exact noise patterns, but to capture the fundamental characteristics of the noise to allow its simulation. In the presence of speckle, the real readings provided by the laser profilometric sensor look like a rough surface. Although they do not necessarily reflect the true surface roughness of the object,

these readings can be adequately characterised using the same parameters that are commonly used to assess the roughness of materials.

Obtaining roughness parameter values can be achieved through analysis of actual measurements or through pre-existing knowledge of the type of material being scan. Using these parameters enables the generation of Perlin noise that replicates the noise introduced by Speckle.

Surface roughness is a feature of the surface texture. It measures the variation between the actual surface and its ideal form in the direction of the normal vector. Roughness values can either be calculated on a profile (line) or on a surface (area). We consider the parameters profile by profile and generalize them to the surface by taking the average of the values of all the profiles. There are many different parameters to characterize roughness [68]. The most important parameters are the amplitude ones, which characterize the surface based on the vertical deviations of the roughness profile from the mean line. Figure 4.4 shows the most important parameters.

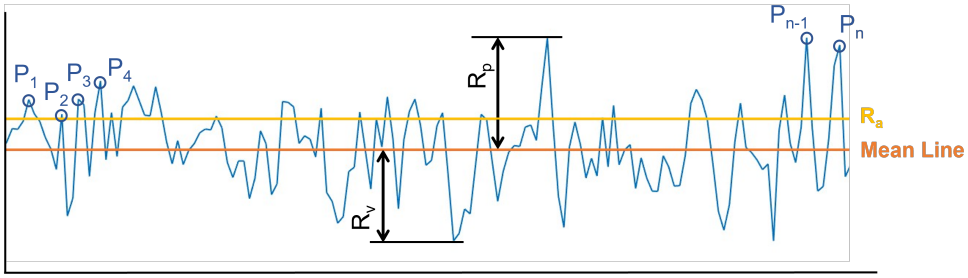


Figure 4.4: Common roughness parameters in a profile. P_i are all the peaks in the profile.

The most common parameter to define roughness is the Arithmetic Average Height (R_a). It is defined as the arithmetic average of profile height deviation from the mean line. It provides a numerical value that quantifies the overall roughness of a surface according equation 4.11, where L is the evaluation length, and $y(x)$ represents the height deviation from the mean line at position x.

$$R_a = \frac{1}{L}\int_0^L |y(x)|dx \tag{4.11}$$

Another important parameter is the Root Mean Square (RMS) Roughness, denoted as R_q. This parameter measures the standard deviation of the surface height

distribution. It provides insight into the overall variance of the surface height profile and is more sensitive to peaks and valleys than R_a. The Rq is calculated as:

$$R_a = \sqrt{\frac{1}{L}\int_0^L y(x)^2 dx} \tag{4.12}$$

Beyond these average measures, surface roughness is also characterized by parameters that focus on the extremities of the profile. The Maximum Height of Peaks R_p represents the height of the tallest peak within the measured profile, while the Maximum Depth of Valleys R_v indicates the depth of the deepest valley. These parameters are critical in applications where extreme values, such as peaks or valleys, could influence the functionality or aesthetic quality of the surface.

To gain a more statistically robust understanding of surface irregularities, the Mean of the Highest Peaks (R_{pm}) and the Mean of the Deepest Valleys (R_{vm}) are used. These parameters compute the average height of the 10 highest peaks and the average depth of the 10 lowest valleys, respectively. By averaging multiple extremities, they provide a more comprehensive picture of the surface roughness, smoothing out the effects of isolated irregularities and offering insights into the general contour of the surface.

Another essential parameter in surface roughness characterization is the Peak Count (P_c). This parameter calculates the number of peaks of the profile per unit length, given in peaks per millimeter. Peaks are only counted when the distance between the current peak and the previous one is greater than 10% of the maximum height of the profile. This parameter helps in assessing the density of surface features and it is calculated accordin equation 4.13.

$$P_c = \frac{N_{peaks}}{L} \tag{4.13}$$

After analyzing the most common roughness parameters, those that align most effectively with the Perlin Noise model can be determined. Equation 4.5, which illustrates the Perlin noise generation algorithm involving multiple scaled versions of a single noise function, reveals the critical parameters that require adjustment to correspond with the roughness parameters from the actual measurements. Subsequently, the chosen roughness parameters for modeling Perlin noise are outlined, along with the reason behind these decision.

First, to generate the 2D Perlin noise function (N_m), as explained in the previous section, we need to adjust the size of the grid and the number of control points. The grid size is set to match the size of the resulting scan in pixels. This corresponds to the number of points per profile (ppf) and the number of scanned profiles ($n_{profiles}$). The number of control points per dimension is then determined based on P_c, which allows us to characterize different surface types. We consider that P_c provides a reliable approximation of the number of intersections with 0 that the first octave of Perlin Noise will have, which corresponds to the number of control points. This parameter is selected because it is more robust than calculating the number of intersections with the mean line of the signal. The number of peaks per mm are estimated by knowing the pixel/mm ratio of a real scan in each direction.

In Perlin noise generation, the frequency represents the number of cycles between two adjacent control points. By aligning the number of control points in the grid with the number of peaks in the desired surface roughness, we can set the frequency to 1. This configuration ensures that there is precisely one complete cycle of the noise pattern between adjacent control points, simulating the spacing between two peaks of the roughness in the resulting noise.

The amplitude is set as the maximum value between R_{pm} and R_{vm}. So, the maximum Perlin amplitude corresponds to the highest value of the 10 sharpest peaks or valleys. In this way, the influence of possible outliers is minimized. The other parameters mentioned above will be used to give an estimate of the similarity between the real and simulated samples.

Experimentally, we have found that a number of octaves equal to 4 and a persistence of 0.5 provides satisfactory results, allowing us to add levels of detail to the noise. Increasing the number of octaves does not produce significant changes in the result. For octave five and above, their amplitude is no longer in the measurement range of the sensor and therefore it is no longer detectable.

Recovering the equation of the generation of Perlin noise algorithm, defined by the composition of multiple scaled versions of the same noise function

$$PN(x,y) = \sum_{m=1}^{n_{octaves}} A_m N_m(f_m x, f_m y) \tag{4.14}$$

Substituting the parameters of the equation 4.14 as explained in this section, equation 4.15 is obtained. This equation presents our proposed model to simulate the speckle of the real measurements from Perlin noise. Remember that the amplitude of

each octave is determined by the persistence factor as $A_m = A \cdot p^m$ and the frequency is typically set by multiplying the base frequency by a factor of 2 for each successive octave. N_m is the 2D Perlin noise function for each octave, with a grid size to match the size of the resulting scan, in pixels and with a number of control points per dimension determined based on P_c and the dimensions of the scan.

$$PN(x,y) = \sum_{m=1}^{4} (0.5^{m-1} \cdot \max(R_{pm}, R_{vm}) \cdot N_m(2^{m-1}x, 2^{m-1}y)), \quad \forall x \in [0, ppf], \forall y \in [0, n_{profiles}] \tag{4.15}$$

In this final equation, $PN(x,y)$ represents the Perlin Noise value for the given coordinates (x,y). The terms x and y span the pixel dimensions from 0 to *points per profile* and 0 to the number of profiles, respectively.

4.2.3 Sensor Uncertainty Model: Gaussian Noise

In the real world, the precision and reliability of any measurement taken by a sensor are influenced by inherent uncertainties and limitations in its performance. For laser profilometric sensors, these uncertainties can significantly impact the accuracy of depth measurements. To create a realistic simulation model, it is essential to account for two critical parameters: Z Resolution (Δ_Z) and Repeatability (σ_{rep}).

Z Resolution defines the smallest incremental change in depth that the sensor can detect with reliability. It essentially captures the sensor's quantization error. In our model, Z Resolution is represented as a uniform distribution spanning the range $\left[-\frac{\Delta_Z}{2}, \frac{\Delta_Z}{2}\right]$. This approach reflects the fact that each depth measurement can deviate by up to half of the sensor's resolution step, thereby providing a realistic portrayal of the sensor's depth measurement granularity.

The repeatability, on the other hand, quantifies the consistency of the sensor's readings when the same measurement is repeated under identical conditions. It is modeled as the standard deviation of these repeated measurements. This parameter captures the random fluctuations inherent in the sensor's performance, effectively reflecting its ability to reproduce the same measurement reliably each time.

To incorporate these parameters into our simulation, we need to model the com-

bined effect of Z Resolution and Repeatability. Since they are independent variables, their combined impact on the measurement can be represented as a Gaussian distribution centered at zero.

Gaussian noise, also known as normal noise or random noise, is one of the most common types of noise encountered in various scientific and technical applications. It is characterized by following a Gaussian distribution, also known as a normal distribution. The Gaussian distribution is a mathematical function that describes a bell-shaped curve, symmetric around its mean value. This curve is defined by two parameters: the mean (μ) and the standard deviation (σ). The mean specifies the central value of the distribution, while the standard deviation determines the spread of the data around the mean. The probability density function of the Gaussian distribution can be expressed by the following equation:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{4.16}$$

In this equation, x represents a random variable that follows a Gaussian distribution, $f(x)$ is the probability that x takes on a specific value, μ is the mean of the distribution, and σ is the standard deviation.

To model the sensor uncertainty, the standard deviation of this distribution is derived using the equation 4.17. This equation accounts for the fact that the uniform distribution's contribution to variance is $\frac{\Delta_Z^2}{12}$, which represents the spread of the Z Resolution over its range, while σ_{rep} directly contributes to the overall variance as the standard deviation of the Gaussian distribution representing Repeatability.

$$\sigma = \sqrt{\left(\frac{\Delta_Z^2}{12} + \sigma_{rep}^2\right)} \tag{4.17}$$

In the simulation, we add a value drawn from this Gaussian distribution (with standard deviation σ) to each measurement to simulate the sensor's uncertainty. Afterward, we truncate the resulting value according to the parameter Δ_Z to ensure it remains within the valid measurement range of the sensor. This process effectively simulates the combined influence of quantization error and random measurement fluctuations, thereby providing a more realistic representation of the sensor's performance in practical scenarios.

4.3 Results

This section presents the results of the laser triangulation profilometry simulator, which integrates both geometrical and noise models. The simulation algorithms were developed on C++ on the Ubuntu 20.04 LTS operating system. For handling the 3D models and simulating the inspection system, the open-source Visualization Toolkit (VTK) [69] was used. MATLAB 2022b was employed for the subsequent processing and analysis of the results.

To evaluate the performance and reliability of the simulation algorithms, experiments were conducted involving the replication of real measurements from three distinct steel products: bearing caps, car doors, and heavy steel plates. Each product features unique geometrical configurations and surface characteristics, influenced by their respective manufacturing processes, despite being composed of the same material.

The analysis of the simulation results involved a detailed comparison between the simulated scans and actual scans of these products. Assessing the accuracy of noise simulation proved challenging due to the inherent variability that noise introduces into the measurements. This variability complicates the straightforward quantification of the simulation's fidelity.

To address this, a quantitative comparison was made, focusing on the distribution of measurements and surface roughness parameters between the simulated and actual scans. In addition to the quantitative analysis, a visual comparison was conducted to examine the resemblance in the appearance of individual profiles and the overall scan area.

These comparisons were instrumental in evaluating how closely the simulation replicates real-world scanning conditions, particularly in terms of both geometric precision and the characteristics of simulated noise. This comprehensive approach provided insights into the effectiveness and realism of the simulation in practical laser scanning applications.

4.3.1 Geometrical Model

Any commercial profilometric sensor can be simulated by modifying its parameters as needed. To accurately replicate the conditions under which the real inspection system operates, a simulation model must carefully account for the specific parameters of

the profilometric sensor used. In these experiments, the AT-C5-4090CS30-495 sensor from Automation Technology was chosen as the reference device. Main parameters extracted from its datasheet [70] are listed in table 5.2.

The selected sensor's parameters are critical in defining the scope and precision of the simulation. These include factors like the working distance, Z-range, field of view, and resolution, which collectively influence how the sensor interacts with the surface being scanned. Additionally, the sensor's travel speed during the experiments, set to 0.1 m/s, plays a significant role in determining the temporal and spatial resolution of the scans. By mirroring these conditions in the simulation, it becomes possible to create a highly accurate model that can be used to evaluate the performance and capabilities of the sensor in various inspection scenarios.

Table 4.1: Parameters of the profilometric sensor from its datasheet and travel speed used in these experiments.

Working distance	400 mm
Z-Range	250 mm
FOV	63.5°
Points per profile	4096 pixels
Z resolution	3.8 μm
Profile speed	500 FPS
Travel speed	0.1 m/s

The simulation environment supports loading any 3D part as an STL file, making it a flexible tool for testing and analysis. For instance, Figure 4.5 shows the simulator's interface with a 3D model of a car door. This setup works with a modeled profilometric sensor, allowing for realistic simulations.

The simulator also enables configuring different scanning paths. In the example shown, a straight-line scan is conducted between two points, P_0 and P_1. This path simulates a typical inspection route that the sensor might take in actual use. The travel speed for this scan is set to match the real system's speed, ensuring that the simulation closely mirrors real-world scanning.

Once the scan is completed, the results are displayed as both a point cloud and a 2D image, as illustrated in Figure 4.6. The point cloud provides a detailed spatial view of the scanned surface, while the 2D image gives a pixel-based depiction of the scan. The size of the 2D image is determined by the total number of profiles captured and the number of points per profile, giving a clear view of the scanned area. The total number of profiles is calculated from the sensor's travel speed and the distance

scanned, reflecting the sensor's capabilities in mapping the surface accurately.

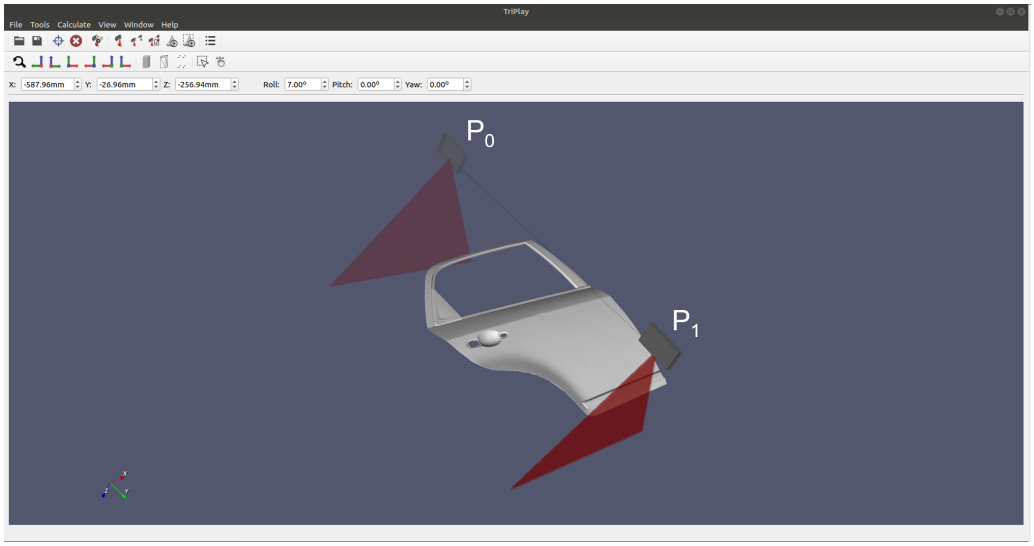
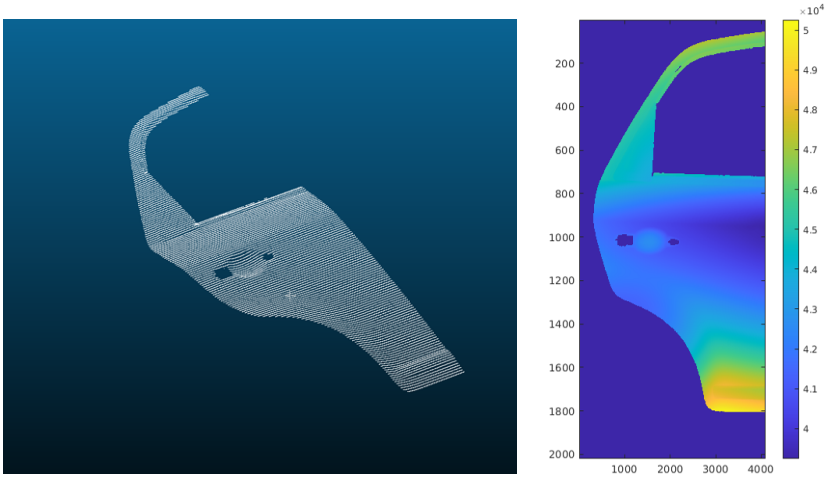


Figure 4.5: Interface of the developed simulator. The CAD model of a loaded car door and a scan path between the marked points P_0 and P_1 is shown.



(a) (b)

Figure 4.6: Result obtained during the simulation of a scan of the section of the CAD model shown in figure 4.5. (a) Result in point cloud form (b) Result as 2D image. Each row corresponds to a scan profile, so the size of the image is determined by the total number of profiles scanned during the acquisition and the number of points per profile. (Size: 2000x4096)

4.3.2 Noise Model

In this study, we explore two distinct sources of noise: sensor uncertainty and speckle noise. The uncertainty of the sensor is represented as Gaussian Noise, with the noise amplitude determined by the sensor's resolution. According to the datasheet, the sensor resolution along the z-axis is specified at 3.8 micrometers. However, empirical evidence suggests that a more practical estimate for this resolution is closer to 15 micrometers, reflecting real-world conditions more accurately.

Speckle noise, on the other hand, is modeled using Perlin noise. This type of noise varies depending on the material properties and manufacturing processes of the inspected product. The characteristics of Perlin noise are influenced by parameters such as R_{pm}, R_{vm}, and P_c, which describe the surface roughness and profile of the scanned object.

In Figure 4.7, a profile extracted from the scan shown in Figure 4.6 can be observed, specifically from the door handle area. The figures on the left provide a more comprehensive view of the profile, while those on the right are zoomed in on a specific part of it.

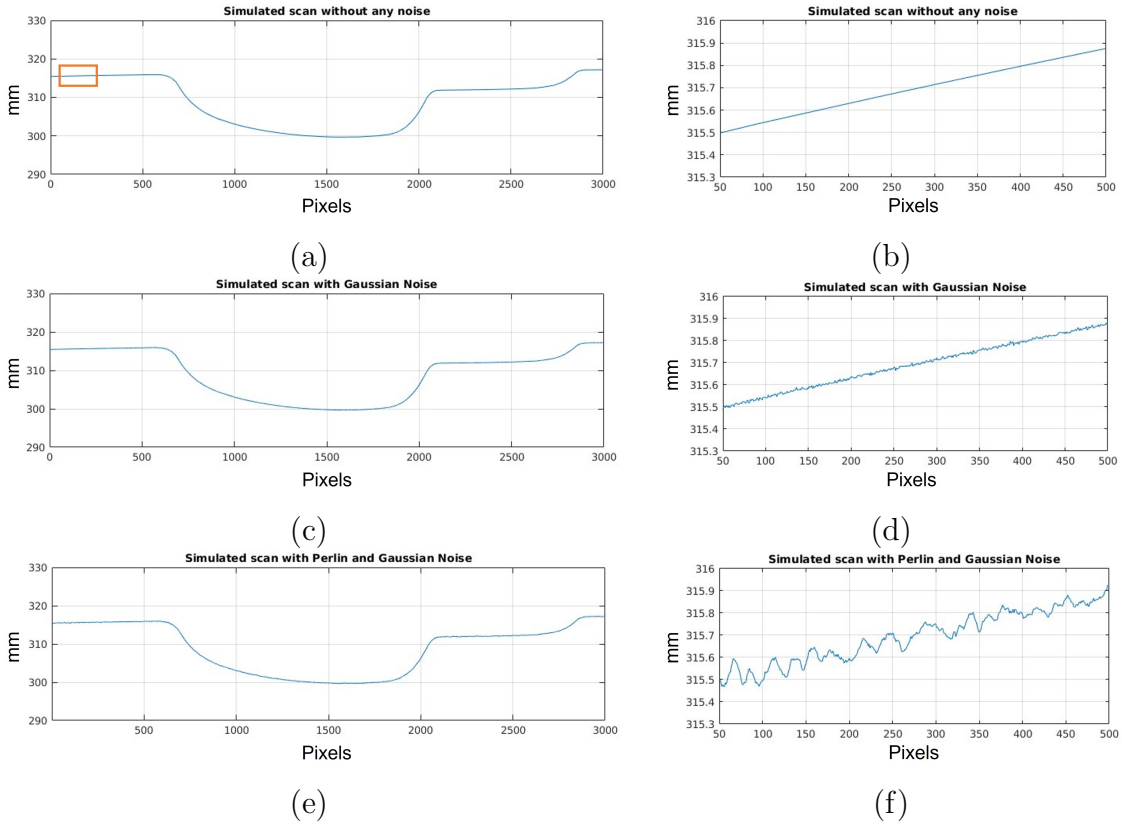


Figure 4.7: Noise simulation results. Images on the left show the complete profile and those on the right show a zoom of the section outlined in orange. (a) y (b) Simulation profile before adding any noise. (c) y (d) Simulation profile after Gaussian noise to simulate sensor uncertainty. (e) y (f) Simulation profile after Perlin and Gaussian noise.

Figures (a) and (b) demonstrate that the scan without the addition of noise yields an ideal profile without any alterations. However, this is not desirable for replicating real scans.

In images (c) and (d), Gaussian noise is applied to simulate sensor measurement inaccuracies. Set at 15 micrometers based on estimated sensor specifications, this noise introduces typical measurement variations. Additionally, Perlin Noise with an amplitude of 80 micrometers is introduced, deliberately higher than the actual surface roughness of the door material. This intentional contrast between Gaussian and Perlin noise helps distinguish their effects in this specific experiment.

Images (e) and (f) display the final profiles after incorporating both types of

noise. Here, it's evident that Perlin noise exhibits a noticeably higher amplitude compared to Gaussian noise, reflecting the chosen parameters that align with the sensor's capabilities and the surface characteristics being simulated.

In the upcoming sections, we will present the results of simulating scans for three distinct steel products: Bearing Caps, car doors, and heavy steel plates. Each product is manufactured using a different process: Bearing caps are cast, car doors are stamped, and heavy steel plates are rolled. Figure 4.8 illustrates these product types.

In these simulations, Gaussian noise with a standard deviation of 15 micrometers has been applied to replicate sensor measurement inaccuracies. Additionally, each piece is configured with specific Perlin Noise characteristics. This approach considers the diverse manufacturing methods and surface textures inherent to each product, enabling a detailed analysis of their impact on scan profiles and noise characteristics.

(a) (b) (c)

Figure 4.8: (a) Bearing cap. Image obtained from [71] (b) Car door (c) Heavy steel plate

Comparing real and simulated scans directly is challenging due to the random nature of noise patterns, which vary with each measurement. Our goal is to realistically model measurements rather than replicate specific instances, simulating scans of various products using any laser triangulation sensor. Nevertheless, we will conduct a comparison of roughness parameters between real and simulated scans to assess their similarity. Additionally, we will evaluate their visual appearance at both profile and surface levels, focusing on scans of relatively flat areas to facilitate analysis without distorting the part's shape.

Emphasizing the importance of simulating both uncertainty-induced noise and speckle noise, we will analyze the same scan under two conditions: with only Perlin noise and with the complete noise model. Roughness parameters in the simulations will be estimated for comparison with real samples, specifically focusing on R_a, R_q,

R_p, R_v, R_{pm}, and R_{vm}. These parameters will be computed as averages across all profiles.

Furthermore, we will use the Two-sample Kolmogorov-Smirnov Test [72] to evaluate the similarity between the distributions of real and simulated scans. The two-sample Kolmogorov-Smirnov (KS) test is a non-parametric statistical method used to compare two independent samples and evaluate if they come from the same distribution. In our context of simulation and scan data analysis, it provides a robust way to quantify the similarity between real-world and simulated measurements.

The KS Test provides a way to compare the cumulative distribution functions (CDFs) of two different datasets. The CDF essentially shows the probability that a random variable is less than or equal to a particular value. By comparing the CDFs of real and simulated scan data, the KS test evaluates how well these two distributions correspond to each other across their entire range. This comparison helps to determine if the variations in the datasets are significant or if they can be considered similar statistically.

The null hypothesis (H_0) in the KS test proposes that the two samples being compared are from the same distribution. To evaluate it, the KS test calculates a statistic, denoted as D, which represents the maximum absolute difference between the CDFs of the two datasets. Mathematically, this is expressed as:

$$D = max|F_1(x) - F_2(x)| \tag{4.18}$$

where $F_1(x)$ and $F_2(x)$ are the empirical CDFs of the first and second samples, respectively. This test statistic captures the largest discrepancy between the two CDFs at any point along their range, providing a measure of how divergent the distributions are.

The decision to reject or not reject the null hypothesis depends on the value of D relative to a critical value D_α, which is determined by the chosen significance level α (commonly set at 0.05 for a 5% significance level). If the test statistic D exceeds D_α, the null hypothesis is rejected, suggesting that the samples are unlikely to come from the same distribution. Conversely, if D is less than or equal to D_α,, the null hypothesis cannot be rejected, implying that there is no significant evidence to conclude that the distributions are different.

Bearing Cap

This section presents the results obtained during the simulation of scans over a bearing cap. Figure 4.9 shows the comparison between the results of a simulation scan and a real one. It presents a profile and a 2D image of the obtained scan. It is possible to visually verify the similarity between simulated and real ones. The displayed areas correspond to a portion of the entire scan performed with a sensor of a 4096 pixels per profile. More precisely, it is an area of 100 profiles by 501 pixels, that represents an area of 22.5x22.5mm. A section of the area is presented to offer a clearer view of the noise details and facilitate a more comprehensive analysis, also to preserve the confidentiality of the data.

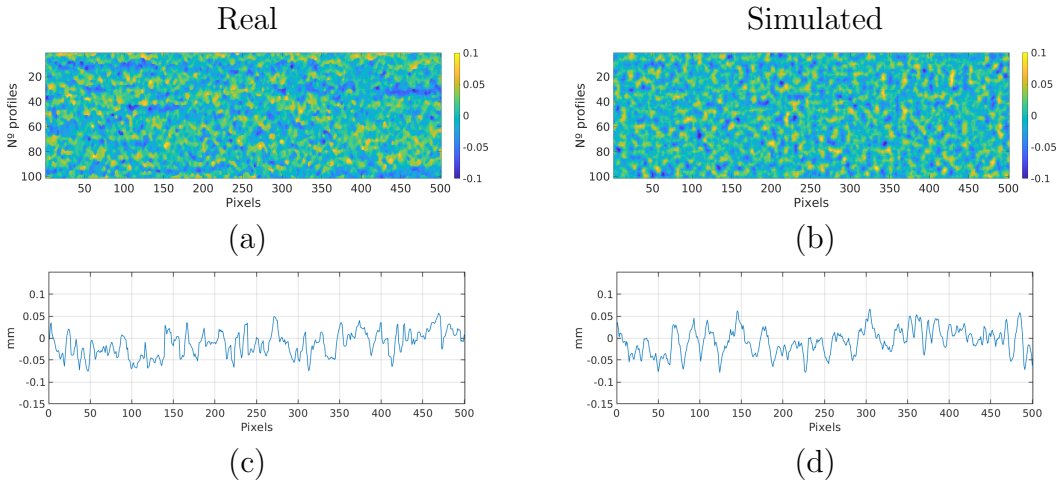


Figure 4.9: Bearing cap experiment: Comparison between real and simulated scan. (a) Real 2D scan image. (b) Simulated 2D scan image. (c) Real scan profile. (d) Simulated scan profile.

Table 4.2 shows the estimated parameters obtained from the analysis of real scans, and compares them with the parameters obtained from the simulation results. The analysis shows that the simulated roughness parameters are quite similar to the real measurements. For instance, the average roughness (R_a) and root mean square roughness (R_q) have relative errors of only 1.29% and 0.11%, respectively, indicating that the simulation effectively captures the general surface texture.

Extremity measures such as maximum peak height (R_p) and maximum valley depth (R_v) show slightly larger discrepancies, with relative errors of 4.47% and 6.55%, respectively. These differences suggest that while the simulation tends to

Table 4.2: Bearing Cap: Comparison of roughness parameters between real and simulated scans. Estimated as the average of the parameters calculated for each profile.

Bearing cap			
	Real	**Simulation**	$\epsilon_r(\%)$
$\mathbf{R_a}$ **(mm)**	0.0211	0.0208	1.29
$\mathbf{R_q}$ **(mm)**	0.0263	0.0263	0.11
$\mathbf{R_p}$ **(mm)**	0.0731	0.0764	4.47
$\mathbf{R_v}$ **(mm)**	0.0742	0.0790	6.55
$\mathbf{R_{pm}}$ **(mm)**	0.0676	0.0702	3.76
$\mathbf{R_{vm}}$ **(mm)**	0.0700	0.0741	5.85
$\mathbf{P_c}$ **(peaks/mm)**	4.36	4.62	5.74

overestimate extreme surface features, it does so within a tolerable range. The mean peak height (R_{pm}) and mean valley depth (R_{vm}) are similarly accurate, with relative errors of 3.76% and 5.85%. These parameters are essential for understanding the average surface profile and confirm that the simulation model captures the surface details effectively.

Finally, the peak count (P_c), which measures the frequency of significant surface peaks, shows a relative error of 5.74%, indicating a slight overestimation by the simulation. Overall, the simulation achieves a high level of precision across all parameters, generally maintaining relative errors around 5 %. This confirms that the simulation model can effectively replicate the surface characteristics of bearing caps, providing valuable insights for quality control in manufacturing.

Figure 4.10 presents the normalized histogram of real and simulated scan distributions for this experiments. Both histograms show comparable shapes, suggesting that the simulation effectively captures the distribution characteristics of the real scans. The CDF comparison further supports this, with closely aligned CDFs indicating that the distributions follow similar patterns across their range of values.

The application of the Two-sample Kolmogorov-Smirnov Test supports the observed similarities between the real and simulated scan distributions. The test calculates the maximum difference between the CDFs of the two datasets. In this case, the calculated difference is small enough that we do not reject the null hypothesis at the 5% significance level. This suggests that there is no significant statistical difference between the distributions of the real and simulated scans, indicating they likely come from the same distribution.

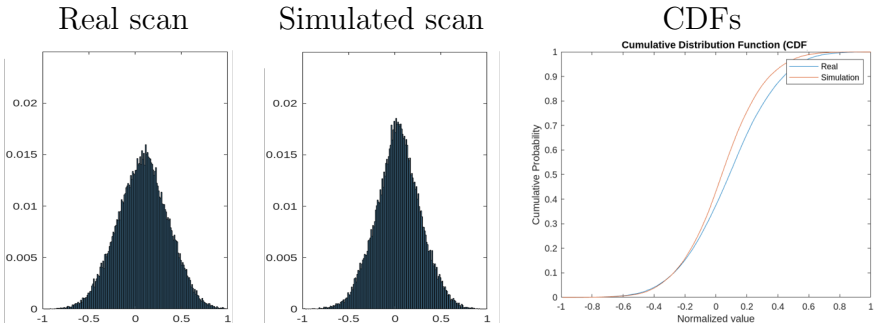


Figure 4.10: Comparison of distributions. (a) and (b) Normalized histogram of each of the captures (real and simulated) in the bearing cap experiment. (c) Comparison of the CFD's of the captures.

Car door

This section presents the results of the simulation measurements on a car door. Similarly to the previous section, table 4.3 presents a comparative between the roughness parameters of the simulated and real scan. Figure 4.11 shows the 2D image of the obtained scans. As the previous experiment, it is an area of 100 profiles by 501 pixels, also corresponding to 22.5x22.5mm.

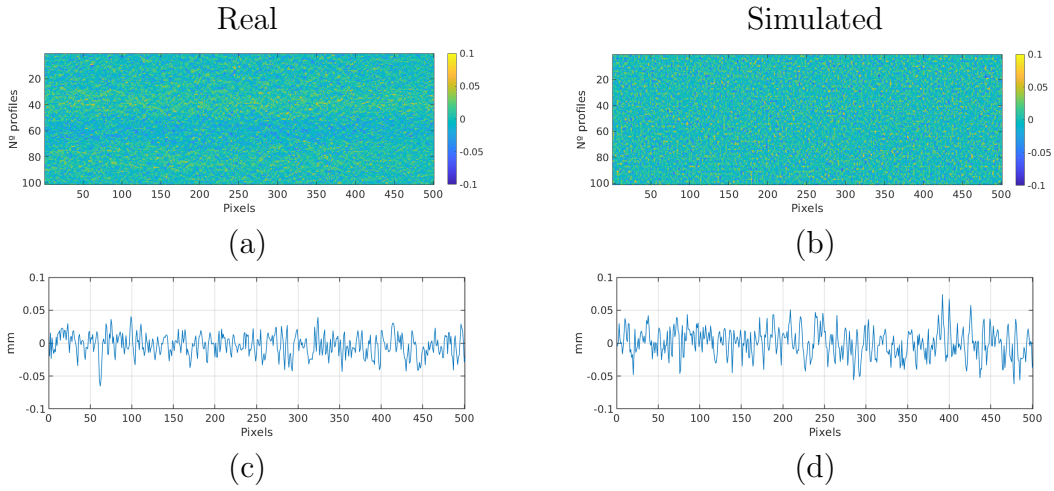


Figure 4.11: Car door experiment: Comparison between real and simulated scan. (a) Real 2D scan image. (b) Simulated 2D scan image. (c) Real scan profile. (d) Simulated scan profile.

The comparison of roughness parameters between real and simulated scans of the car door surface demonstrates a high degree of accuracy in the simulation results. Average roughness (R_a) and root mean square roughness (R_q) exhibit minimal percentage errors of 2.03% and 2.47%, respectively, indicating that the simulated surface texture closely matches the actual measurements. Parameters such as maximum peak height (R_p), valley depth (R_v), mean peak height (R_{pm}), mean valley depth (R_{vm}), and peaks per millimeter (P_c) also show small percentage errors ranging from 3.50% to 6.28%. These findings underscore the simulation's capability to faithfully reproduce both the general surface characteristics and detailed features of the car door, validating its suitability for industrial applications requiring precise surface analysis and quality control.

In this case, it can be seen that the roughness values are lower than in the previous experiment. This is due to the characteristics of this particular surface. Both are steel materials, but the ingot is manufactured by casting and the car door by stamping. In general, stamping tends to result in lower surface roughness compared to steel casting. This is because stamping involves shaping the metal through controlled pressure and force, resulting in a smoother and more uniform surface. On the other hand, casting involves pouring molten metal into a mold, which can lead to surface irregularities and textures due to the solidification process.

Table 4.3: Car door: Comparison of roughness parameters between real and simulated scans. Estimated as the average of the parameters calculated for each profile.

Car door			
	Real	**Simulation**	$\epsilon_r(\%)$
$\mathbf{R_a}$ **(mm)**	0.0134	0.0137	2.03
$\mathbf{R_q}$ **(mm)**	0.0169	0.0173	2.47
$\mathbf{R_p}$ **(mm)**	0.0523	0.0542	3.65
$\mathbf{R_v}$ **(mm)**	0.0522	0.0555	6.28
$\mathbf{R_{pm}}$ **(mm)**	0.0466	0.0482	3.50
$\mathbf{R_{vm}}$ **(mm)**	0.0477	0.0504	5.73
$\mathbf{P_c}$ **(peaks/mm)**	5.9446	6.2403	4.97

The Two-sample Kolmogorov-Smirnov Test also confirms that the null hypothesis holds, indicating that both real and simulated samples originate from the same distribution. Figure 4.12 presents the histograms and the cumulative distribution functions.

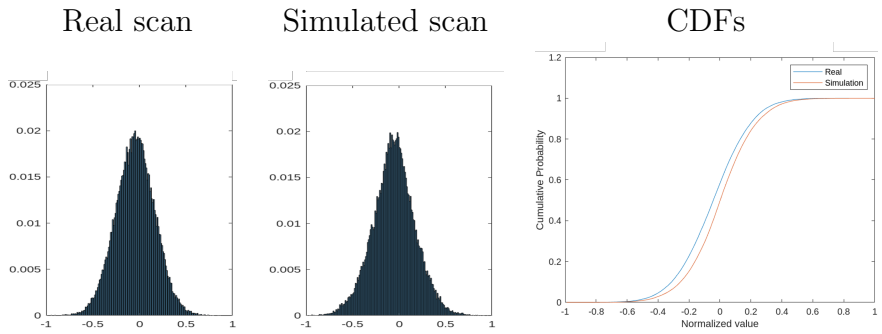


Figure 4.12: Comparison of distributions. (a) and (b) Normalized histogram of each of the captures (real and simulated) in the carDoor experiment. (c) Comparison of the CFD's of the captures.

Heavy steel plate

This section focuses into the simulation scans on a heavy steel plate. The roughness parameters of the simulated and real scans are compared in Table 4.4. Additionally, Figure 4.13 showcases a 2D image of the acquired scans, covering an area of 100 profiles by 501 pixels. In this case, the scanning parameters were different and cover an area of 20x40mm.

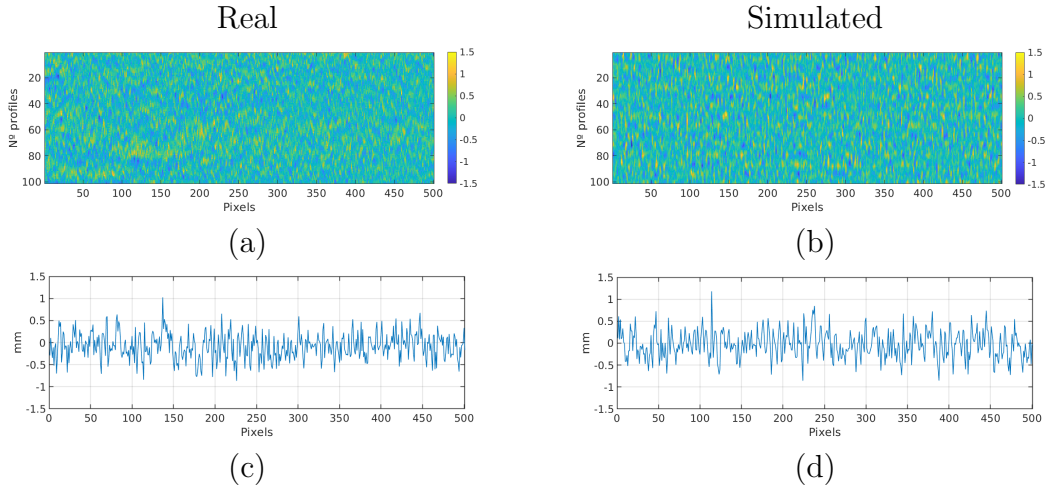


Figure 4.13: Plate experiment: Comparison between real and simulated scan. (a) Real 2D scan image. (b) Simulated 2D scan image. (c) Real scan profile. (d) Simulated scan profile.

As observed in Table 4.4, the comparison between real and simulated scans of the heavy steel plate demonstrates a high degree of alignment in roughness parameters. The average roughness (R_a) shows a minor deviation, with the real value at 0.2484 mm and the simulated value at 0.2403 mm, resulting in a percentage error of 3.27%. Similarly, the root mean square roughness (R_q) exhibits a small difference, with values of 0.3118 mm for real and 0.3058 mm for simulated scans, corresponding to a percentage error of 1.94%.

For parameters reflecting surface irregularities, such as maximum peak height (R_p) and valley depth (R_v), the simulation also closely approximates the real measurements. R_p shows values of 0.9397 mm for real and 0.9633 mm for simulated scans, with a percentage error of 2.52%. R_v has real and simulated values of 0.9834 mm and 0.9642 mm, respectively, resulting in a percentage error of 1.95%.

Moreover, mean peak height (R_{pm}) and mean valley depth (R_{vm}) display percentage errors of 2.65% and 1.22%, respectively, indicating a robust simulation of these specific surface features. Peaks per millimeter (P_c), though exhibiting a larger percentage error of 9.43%, still demonstrates a reasonable alignment between real (6.8758 peaks/mm) and simulated (6.2277 peaks/mm) scans. Compared to the bearing cap and car door, heavy steel plates typically have higher surface roughness parameters. This is mainly due to the presence of scale or surface oxides, which can contribute to a rougher surface texture and irregularities.

Table 4.4: Heavy Steel Plate: Comparison of roughness parameters between real and simulated scans. Estimated as the average of the parameters calculated for each profile.

Heavy steel plate			
	Real	**Simulation**	$\epsilon_r(\%)$
$\mathbf{R_a}$ **(mm)**	0.2484	0.2403	3.27
$\mathbf{R_q}$ **(mm)**	0.3118	0.3058	1.94
$\mathbf{R_p}$ **(mm)**	0.9397	0.9633	2.52
$\mathbf{R_v}$ **(mm)**	0.9834	0.9642	1.95
$\mathbf{R_{pm}}$ **(mm)**	0.8329	0.8550	2.65
$\mathbf{R_{vm}}$ **(mm)**	0.8925	0.8817	1.22
$\mathbf{P_c}$ **(peaks/mm)**	6.8758	6.2277	9.43

As in the previous experiments, we can confirm that both real and simulated samples have similar distribution using the Two-sample Kolmogorov-Smirnov Test. Figure 4.14 presents the histograms and the cumulative distribution functions.

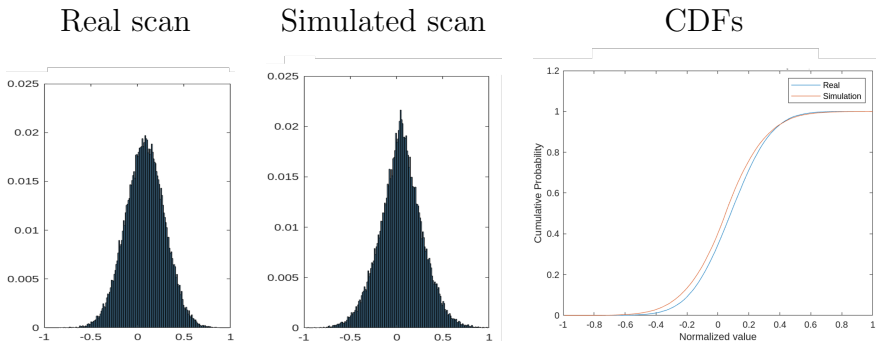


Figure 4.14: Comparison of distributions. (a) and (b) Normalized histogram of each of the captures (real and simulated) in the heavy steel plate experiment. (c) Comparison of the CFD's of the captures.

4.4 Discussion

This study introduces a simulation model designed to simulate the output of a real laser triangulation profilometric sensor, which performs precise depth measurements along a line. The model consists of three key components: a geometrical model that incorporates the physical properties of the sensor, a Gaussian noise model to simulate sensor uncertainty, and a Perlin noise model to replicate speckle-induced noise. Validation experiments demonstrated a high degree of similarity between the generated scans and real sensor scans. Our research provides satisfactory results, enabling a realistic simulation of laser triangulation scans.

We use CAD models in STL format because they are widely accepted for representing industrial part geometry. STL files store data as triangular meshes, where each triangle is defined by its vertices and a unit normal vector. However, when simulating measurements from CAD models using profilometric sensors, a challenge arises: the sensor's resolution often surpasses that of the triangular mesh. As a result, simulated measurements produce profiles composed of small straight segments corresponding to each triangle face in the mesh. This method suffices for rough 3D reconstructions but proves inadequate for simulating sensors that operate with resolutions in the range of a few micrometers.

To address this challenge and achieve realistic scan simulations, we introduce sensor measurement noise and speckle using a combination of Gaussian and Perlin

noise. Quantifying the exact fidelity of the noise simulation is complex due to the variability of the actual scans caused by sensor characteristics and speckle effects. However, our approach aims to validate the similarity between simulated and real-world scans.

The proposal enables simulation of any commercial laser triangulation sensor based on its datasheet, incorporating both measurement noise due to sensor limitations and noise generated by the scanned product surface. Noise simulation is achieved straightforwardly by considering surface roughness parameters. Successful simulation of speckle generated by the roughness of different materials was achieved.

Speckle generated by the roughness of various materials was successfully simulated, focusing specifically on three steel products manufactured using different methods. The study results highlight the accuracy of the modeling and simulation process across a wide range of surfaces. Comparison between real scans and simulated results for bearing cap, car door, and steel plate surfaces consistently showed similar roughness characteristics. By generalizing the noise model using roughness parameters, different surfaces can be simulated based on prior knowledge of material type and surface finish or through straightforward analysis of roughness parameters from a real scan.

Comparative analysis revealed minor differences in roughness parameters between simulated and real scans across the three experiments. These small errors in roughness parameters indicate that the simulation approach effectively captures the general surface roughness characteristics. Thus, the results suggest that the current simulation method adequately reproduces essential roughness characteristics.

The purpose of the proposed model is not to precisely replicate real measurements but to enable simulation of real measurements obtained from commercial sensors. Overall, the results validate the effectiveness of the approach. The similarity between real and simulated scans indicates that the simulation accurately captures the essential roughness characteristics of each surface. Simulating surface roughness opens opportunities for using simulated scans in various algorithms and applications, providing a cost-effective and efficient alternative to extensive experimental measurements.

In this study, the simulator primarily focused on simulating the laser triangulation process to evaluate its capabilities under controlled conditions. The simulation notably addressed the significant noise source of speckle, which is crucial in laser scanning applications. However, it did not explicitly consider other potential challenges such as occlusions due to the angle between the laser and camera, variations

in power levels, specular reflections, and other environmental factors. These factors can significantly impact the accuracy and reliability of real-world laser scanning systems, particularly when scanning highly reflective or dark surfaces, where laser triangulation sensors are not the most suitable and thus are not frequently used. However, for applications where such sensors are appropriate, this simulator proves to be valuable.

Chapter 5

Simulation of surface defects in 3D models

The present chapter introduces a method for simulating surface defects in 3D models using a Free-Form Deformation (FFD) technique. This approach allows for the creation of defects with known dimensions and shapes, crucial for developing defect detection algorithms in manufactured products. Mathematical models are proposed for common defects found in sheet metal products, such as bumps, peaks, and cracks. These models offer flexibility to parameterize defects in terms of depth, size, and orientation, enabling precise customization of simulated defects.

To complement defect simulation, a commercial profilometric sensor simulator discussed in the previous chapter is employed. This simulator generates high-resolution simulated scans, providing labeled 3D information of the defective 3D models generated. This integration between defect simulation and profilometric scans simulates the inspection experience in a controlled virtual environment, facilitating training and validation of defect detection models. An additional advantage is that the generated defects come pre-labeled, eliminating the need for manual labeling by workers.

The main contribution of this chapter lies in generating synthetic defect datasets in 3D models of manufactured products, particularly beneficial for Artificial Intelligence algorithms. This approach addresses the challenge of acquiring real defect data, especially during the developmental phase, due to the occasional nature of defects and associated high costs. By exclusively using synthetically generated data, this approach facilitates the development and optimization of 3D inspection algorithms, regardless of the availability of real defect data.

Furthermore, a detailed analysis of the results obtained from comparing synthetically generated data with real measurements is presented. A comprehensive dataset of simulated defects is generated to train a neural network using solely synthetic data. This network is evaluated using real-world scans, and its performance is compared with a model trained exclusively with real defects to validate the accuracy of defect simulation. It is important to note that the primary objective is not to improve detection algorithms but to evaluate the realism of simulated defects.

The contents of this chapter are under consideration for publication in Journal of Intelligent Manufacturing, by Springer. ¡¡S. Roos-Hoefgeest, M. Roos-Hoefgeest, D. García Peña, I. Álvarez, y R. C. González, «Realistic Defect Simulation in 3D Models for Defect Detection Using Machine Learning», Journal of Intelligent Manufacturing, (Currently in its third revision)

5.1 Related work

As previously mentioned, the lack of 3D defect data is a big challenge in the industry, especially with the increasing use of AI in defect detection. AI algorithms need a lot of well-labeled data to train properly and detect defects accurately. However, collecting this data in industrial environments is often costly and time-consuming, as defects do not occur very often and can vary widely. This lack of data makes it hard to develop and test accurate detection models, showing the need for new ways to create synthetic data to fill this gap.

Different authors have proposed the generation of synthetic data to overcome this problem for industrial applications. Wong et al. [73] employed photogrammetry techniques to generate a synthetic dataset from 3D models across 10 different classes of real-world objects. Their process involved creating detailed 3D scans of these objects, which were then utilized to generate synthetic images for training deep learning models. Despite using just 60 real images per class, they successfully generated 100,000 synthetic images. Their method, which integrates 3D modeling with render-based image synthesis, proved highly effective for training an InceptionV3 CNN, achieving a remarkable 95.8% accuracy in classifying real supermarket products. Furthermore, they leveraged these synthetic images to train a RetinaNet detector for real-time localization and classification of products.

Cohen et al. [74] proposed and developed a dataset of egocentric synthetic images using only CAD models of desired objects, specifically aiming to replicate challenges

often encountered in images from head-mounted cameras during bus seat assembly. Their approach included rendering textureless CAD models from various viewpoints. They introduced variability in viewpoint, illumination, truncation, and blurring to mimic typical issues found in such camera images. Synthetic views of different objects were juxtaposed against diverse backgrounds, manually selected to ensure relevance to industrial scenarios. Their framework incorporated domain randomization techniques for image augmentation, enabling realistic object placement, simulation of truncation, and introducing variability in appearance through blur and motion effects. They employed these synthetic images to effectively train a YOLOv3 network as an object detector, showcasing its performance on real-world images.

Horváth et al. [75] addressed the issue of transferring knowledge from simulation to real-world images for industrial robotics applications, specifically focusing on object detection. They created synthetic images from 3D models of ten selected industrial parts, including objects such as L-brackets, U-brackets, and seat components. These synthetic images were rendered with diverse backgrounds and added postprocessing steps to simulate real-world challenges, such as randomized multicolor salt and pepper noise, gaussian blur, and cutouts. The authors employed automatic labeling and trained a YOLOv4 convolutional neural network using the synthetic data and evaluated it with a dataset consisting of real images.

Recently, Zhu et al. [76] introduced an automated method for assembly quality inspection leveraging synthetic data to train diverse neural networks. They utilized CAD models to generate both 2D images and 3D point clouds of the assembly components. Automatic labeling was integrated into the data generation process, facilitating the creation of synthetic datasets. The study highlighted how these trained networks effectively enhanced the precision and efficacy of automated quality inspection systems, underscoring the value of synthetic data in industrial applications.

Previously described research focused on the generation of synthetic datasets for the detection of objects under different challenging situations commonly found in industrial applications. Variations mainly affected the pose of the object, the background, or the image noise. At most, defects were associated with the relative pose between different objects in the scene.

A different question is how to generate synthetic data of defective objects, from CAD models of non-defective parts. Traditional methods, including those based on digital image processing and computer-aided techniques, have been widely employed in industrial defect image generation. Digital image processing methods leverage specific algorithms to modify original images and create targeted defects, often through techniques such as filtering or adding noise. In parallel, traditional techniques often

rely on computer-aided design (CAD)-based approaches. This entails constructing CAD models or utilizing software like 3DS Max to input defect geometry and descriptive information. CAD models employ advanced techniques such as ray tracing and X-ray imaging, predominantly applied to rigid materials such as castings and steel, where obtaining or constructing 3D models is feasible.

For instance, Domingo Mery and Nancy Hitschfeld [77] proposed a method for simulating defects in aluminum castings by integrating CAD models with real X-ray images. Their approach involved projecting detailed 3D polygonal mesh models onto 2D radioscopic images, enabling the simulation of specific defects like blow holes and cracks. Flaws with intricate geometries were modeled using manifold surfaces and superimposed onto real radioscopic images, adhering to the exponential attenuation law for X-rays.

To achieve this, the simulation process begins with the capture and calibration of a real X-ray digital image to establish parameters like focal length, object dimensions, and digital image scale factors. Using 3D modeling software, a 3D flaw is meticulously designed and positioned within a virtual environment. Subsequently, a depth map is generated where rays are emitted from each pixel of the 2D digital image towards the X-ray source plane. The depth map records the intersection points of these rays with the 3D flaw. Finally, the computed depth map is superimposing onto the original X-ray digital image, adjusting pixel grey values according to the X-ray attenuation properties. Experiments using this method were primarily conducted on cast aluminum wheels to validate the simulation of defects in X-ray images, specifically focusing on defects such as cracks and blow holes.

Aleksei Boikov et al. [78] introduced a method for creating labeled synthetic datasets to detect surface defects in steel slabs using deep learning. Defects are simulated in 2D and applied to the CAD model as textures using a free 3D editor. Procedural shaders play a crucial role in this method, transforming initial noise textures into specific defect shapes for various types. For instance, cracks are generated using a spherical gradient procedural texture, with adjustments made to UV coordinates and symmetry to achieve diverse shapes. Parameters like detorsion and noise control the appearance and variability of these defects. The process also involves creating textures for other defects, like chipping, through manipulation of Perlin noise.

In the synthetic data generation process, a 3D camera simulation captures traditional 2D images of steel slabs under conditions mirroring industrial settings. Cameras are positioned overhead, and lighting replicates typical conditions for computer vision applications. An artificial labeled dataset of 6000 defect images was created to

train and evaluate two neural network architectures: U-Net for defect segmentation and Xception for defect classification.

Additionally, Bosnar et al, [79] introduce a procedural pipeline designed for the generation of synthetic images intended for industrial surface inspection. Defects are generated in 3D by constructing a defect tool specific to each defect type. This tool is modeled with the desired shape and subsequently applied to imprint onto the surface of the inspected object through a boolean difference with the object mesh. After creating defects, further geometry manipulations can enhance the look of rendered images, for example, adding some noise to face normals to simulate a rough surface. Ultimately, realistic 2D images are produced by integrating the defective object with material textures, physical light sources, and a camera model, simulating industrial inspection environments. The results demonstrate the simulation of various types of defects, such as scratches and dents, on models of metal gears through rendered 2D images.

In recent years, deep learning techniques, including Conditional Convolutional Variational Autoencoders (CCVAE) [80] and Generative Adversarial Networks (GANs) [81, 82], have been widely employed in industries for generating defective images, often as part of data augmentation strategies. These methods serve to increase the diversity of datasets by creating synthetic images to enhance surface defect detection using neural networks. Zhong et al. [83] have meticulously compiled existing research on the generation of synthetic defect images categorizing the methods into deep learning and traditional approaches. They analyze different methods based on the use of Generative Adversarial Networks, discuss their shortcomings, and point out possible research directions. In their conclusions, they point out that traditional methods have the advantages of speed and interpretability, so it is interesting to advance research in new methods to generate synthetic data of multiple defects similar to real ones.

The synthesis of research on generating 3D defects in CAD models underscores the need for advancement in simulating micrometer-scale defects and building comprehensive 3D databases. These developments are essential for refining defect detection algorithms, enabling artificial intelligence models to learn from synthetic data that accurately replicates real-world industrial irregularities. Previous research has primarily focused on generating rendered 2D images, which are inadequate for defects at a micrometer scale that are not visible to the naked eye. Therefore, it is important to study this area, given the scarcity of studies that address the simulation of micrometric defects and the generation of databases with 3D information.

In this chapter, we present a novel method to add defects to a surface described

by a triangular mesh generated from a CAD model of the object to be inspected. The defects to be included must be described by an elevation map that may proceed from previously scanned defective parts or be handcrafted according to previous knowledge from experts. We also propose models to automate the generation of three frequent superficial defects: cracks, bumps, and peaks. The pose of each defect on the object can be selected by the user. The proposed method can adapt the defect to the local curvature of the surface. The result is a new triangular mesh including the defects and the necessary information to label the data.

To apply the deformation to the surface of the object we use the Free-Form Deformation (FFD) technique [84]. FFD is a well-known geometric method used to model simple deformations of rigid objects. Recently, FFD has been used in a variety of applications, for example in medicine [85], [86] or aerodynamic shape parameterization [87], [88]. The proposed process involves creating a structured Cartesian mesh of control points around the defect insertion point. Control points are linked to the inner geometry of the shape, directing the deformation of the 3D surface. The magnitude of control point displacement is determined by the elevation map specific to the defect type, and direction is determined by the surface normal computed from the CAD model in each point.

After simulating defects within the 3D model, the next critical step is to obtain a point cloud or a 3D image. These data are crucial for inspecting the product and, ultimately, generating the complete dataset needed for analysis. In Chapter 4, we presented a simulation model to replicate the readings of real laser triangulation profilometric sensors based on their datasheet parameters. This model mimics the sensor's movement over the CAD model of the inspected component, considering the physical characteristics of the sensor and accounting for noise sources originating both from sensor imprecision and speckle noise due to material roughness.

By using this tool and configuring the sensor's path relative to the scanned surface, it is possible to generate a set of simulated profiles organized into a two-dimensional matrix. The matrix size depends on the points per profile and the total number of profiles obtained during acquisition. Each pixel in the resulting image corresponds to a distance measurement recorded by the profilometric sensor, a format commonly used by commercial sensors of this type. This simulation model is used in this work to obtain synthetic 3D images of the simulated surface defects in the CAD model of the product.

Additionally, knowing the area of the CAD model where the defect is inserted and the specific type of defect involved, it is possible to provide a labeled point cloud or a 3D image with the defect precisely marked. This capability enhances the utility

of the approach by offering pre-labeled data, thus eliminating the need for additional time and resources typically required for labeling processes.

As a further contribution, mathematical models are proposed to automatically recreate the geometry of three common defects in manufacturing. Specifically, models are presented for inserting bumps, peaks, and cracks. In the production of steel products, such as plates, car doors, etc., these types of defects commonly appear, either inherent to the manufacturing process or during the handling of parts. These types of defects are well-known to manufacturers, as well as the most common areas where they typically appear depending on the type and material of the product.

5.2 3D model deformation using FFD for surface defect generation

This section introduces the method for simulating defects in 3D models. The defects are characterized by two main attributes: a height map and their position on the 3D model's surface. The height map specifies the size and shape of the defect in the XY plane, while the position defines its location and orientation on the 3D surface. An overview of the process is shown in Figure 5.1.

The surface under inspection is assumed to be a triangular mesh with known vertex positions and unit normals for each triangle. To achieve smooth deformation in the selected area, the triangles must be small enough, which is accomplished through a remeshing approach that subdivides triangles based on a specified maximum area.

For deforming the 3D model according to the defect characteristics, the Free-Form Deformation (FFD) method is employed. FFD works by enclosing the object within a bounding lattice and then transforming the object as the lattice deforms. This method, as outlined by Sederberg and Perry [89], involves moving control points of a parametric curve—like Bézier curves, B-splines, or NURBS—within the lattice to achieve the desired deformation.

In this approach, the displacements of the control points are linked to the geometry inside the bounding lattice using NURBS volumetric parameterization. The deformation is applied only to the part of the geometry enclosed within this 3D hull.

To ensure realistic deformation, the control points are moved according to the local characteristics of the surface. To manage computational efficiency, two strategies

are employed based on the complexity of the surface in the defect-affected region. These regions are classified as either simple or complex: simple areas have a nearly constant normal vector throughout, while complex areas show significant variability in normal directions. This classification allows for a tailored approach to deformation, balancing accuracy and computational load.

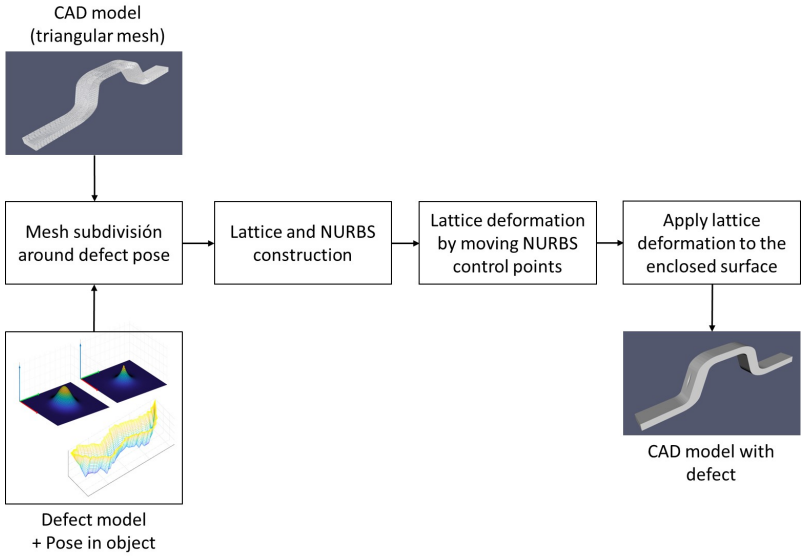


Figure 5.1: Scheme of the process for adding a modeled defect to a CAD model of any product

5.2.1 Mesh remeshing

As previously mentioned, the inspection is performed on a surface modeled as a triangular mesh, with available information about the vertices' positions and the normals of each triangle. To ensure smooth deformation in the area of interest, it is crucial that the triangles are sufficiently small. Therefore, the first step is to perform a remeshing process, which involves subdividing the relevant triangles based on a predetermined maximum area. To optimize computational resources, this subdivision is confined to the specific area where the defect is introduced.

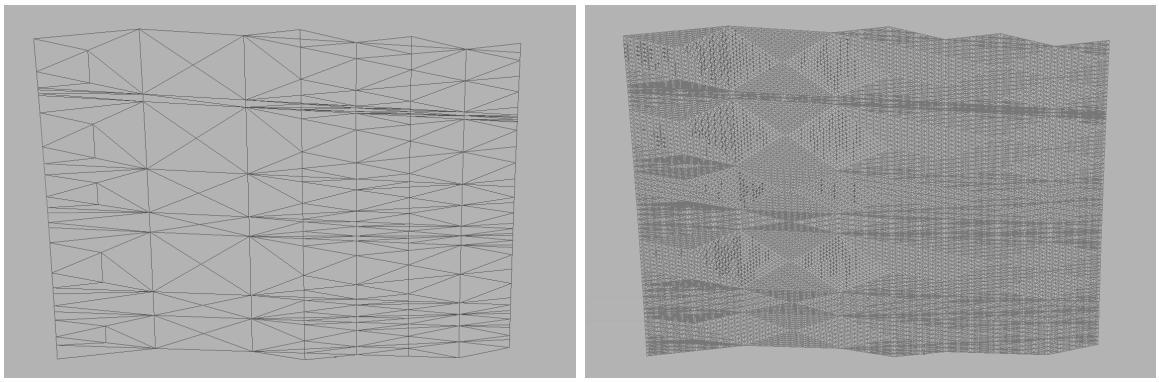
A straightforward, case-based approach with multiple passes is used to repeatedly subdivide the triangles in the selected area to meet the area criterion. The maximum triangle area ($A_{\triangle}$) in the region is defined as a fraction of the area required for the defect. Experimentally, we define this as:

$$A_{\triangle} = \frac{A_{defect}}{100000} \tag{5.1}$$

In this way, the resolution is adjusted to match the defect's size. There is a need to balance between accurately representing the defect and maintaining computational efficiency to ensure that the simulator remains practical and useful.

To further enhance the quality and smoothness of the remeshed area, we apply a smoothing filter to the subdivided mesh. This filter iteratively adjusts the vertex positions, reducing irregularities and distortions.

Figure 5.2 illustrates the outcome of subdividing the triangles in a selected region of a 3D model.



(a) (b)

Figure 5.2: Remeshing of the selected area to be deformed according to the required defect. (a) Before remeshing: The number of triangles is 370. (b) After remeshing: The number of triangles is 123108

Subsequently, FFD [89] based on lattice parametrization is employed to deform the 3D model according to the corresponding defect. FFD is based on the concept of enclosing an object inside a cube, or another hull object, and transforming the object inside it as the lattice is deformed. The deformation of the lattice is achieved by moving the control points of a parametric curve, such as Bézier curves, B-splines, or non-uniform rational B-splines (NURBS). In this work, control point displacements are connected to the object's geometry within the shape using a NURBS volumetric parametrization. Deformation is applied only to the region enclosed by the 3D hull.

First, the general principles of Free-Form Deformation (FFD) and NURBS volume representation are explained.

5.2.2 Generalities of Free-form deformation (FFD) using NURBS parametrization

Free-Form Deformation (FFD) is a versatile technique in computer graphics used to manipulate three-dimensional objects with precision and flexibility [89]. It operates by manipulating a three-dimensional control mesh, often referred to as a lattice, surrounding the target object. This lattice serves as a skeletal structure, influencing how the object deforms. The transformation involves adjusting the positions of control points within this lattice to achieve desired deformations and transformations effectively.

FFD involves a transformation or mapping of the positions of every point P in the original 3D space to its corresponding deformed position P'. This transformation is denoted by $\phi(\cdot)$, see 5.2, where P represents the original position of a point in the object, P' is its deformed position after applying the transformation ϕ at time t.

$$P \rightarrow P' = \phi(P, t) \tag{5.2}$$

The transformation ϕ can be mathematically described by a function $\phi : \mathbb{R}^3 \rightarrow \mathbb{R}^3$, which maps each point P in the original object to its deformed counterpart P' in the transformed object. This function encapsulates the complexity of the deformation process, influenced by factors such as the positions of control points within the lattice and external parameters.

FFD involves adjusting the positions of control points associated with parametric curves such as Bezier curves, B-splines, or NURBS within the lattice structure to achieve the desired deformation effect. By moving these control points, the shape and form of the object can be altered.

FFD involving three fundamental steps to manipulate objects. Initially, an FFD box is set up with control points arranged in a lattice structure. These points serve as markers for subsequent deformations. Following this, the object's surface is integrated into the FFD box, linking its physical coordinates (x, y, z) with the internal parameter space (u, v, w) of the FFD box. This linkage ensures that adjustments made to the control points within the lattice directly influence the object's shape.

The next critical step involves deforming the object's surface by adjusting the positions of these FFD control points. Moving these points enables us to reshape the object while maintaining precise control over its geometry. Mathematically, this process involves the initial vertices of the object represented as $P(u, v, w)$, the FFD control points $Q_{i,j,k}$ on the initial lattice, and the resulting vertices of the deformed object represented as $P'(u, v, w)$ with corresponding new FFD points $Q'_{i,j,k}$. Here, i, j, k range from 0 to l, m, n respectively, indicating the number of control points in each dimension of the lattice. These steps and their associated mathematical formulations guide the transformation from the object's original configuration to its altered state using the principles of FFD.

To map the control points to the deformed geometry, Non-Uniform Rational B-Splines (NURBS) are used [90]. A NURBS curve is defined by three main elements: order, control points, and knot vector. The order determines the number of control points that influence any point on the curve. The curve is represented mathematically by a polynomial. Its degree is equal to the order of the curve minus one. Control points determine the shape of the curve. Typically, each point of the curve is computed by taking a weighted sum of a number of control points. The knot vector is a sequence of parameter values that determine where and how the control points affect the NURBS curve. The number of knots is always equal to the number of control points plus the curve order.

NURBS volumes consist of a regular net of control points Q_{ijk} connected to knot vectors and B-spline basis functions. The volumetric parametrization is defined by a linear combination of these basis functions and controlled by weights w_{ijk}. The new position $P'(u, v, w)$ of the deformed object is determined as:

$$P'(u, v, w) = \frac{\sum_{i,j,k}^{l,m,n} N_{i,p}(u) N_{j,q}(v) N_{k,r}(w) w_{ijk} Q'_{ijk}}{\sum_{i,j,k}^{l,m,n} N_{i,p}(u) N_{j,q}(v) N_{k,r}(w) w_{ijk}} \tag{5.3}$$

where $N_{i,p}(u)$, $N_{j,q}(v)$, and $N_{k,r}(w)$ are the non-rational B-Spline basis functions of orders p, q, and r respectively, and w_{ijk} are the weights of the new control points Q'_{ijk}. The B-Spline basis functions are defined over knot vectors U, V, W, which specify the location and continuity of the NURBS. These knot vectors are defined as:

$$U = \{u_0, u_1, \ldots, u_{l+p+1}\} \tag{5.4}$$

$$V = \{v_0, v_1, \ldots, v_{m+q+1}\} \tag{5.5}$$

$$W = \{w_0, w_1, \ldots, w_{n+r+1}\} \tag{5.6}$$

From Figure 5.3, it can be seen that the control lattice has been deformed, with the initial points for Q_{ijk} (in red) shifting to create a new control lattice Q'_{ijk} (in blue). This figure illustrates the process of deforming the lattice and its control points to achieve a specific defect shape on a surface.

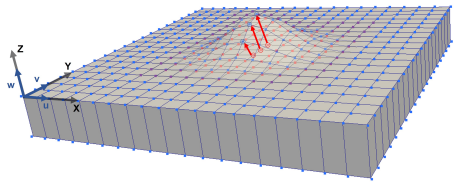
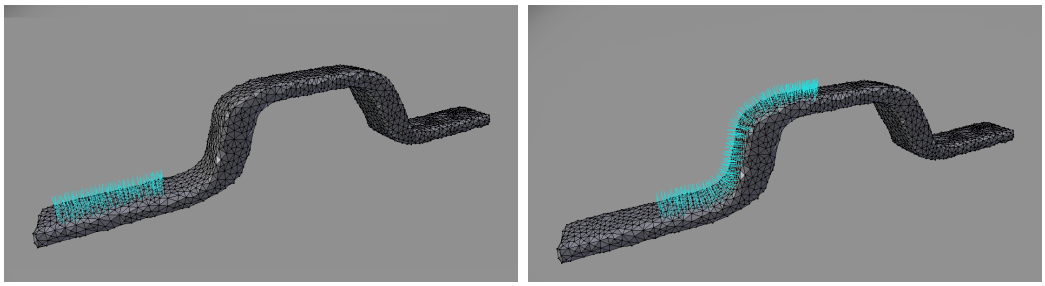


Figure 5.3: Illustration of the lattice and control points displacement process to deform a surface. The lattice structure is shown in the u, v, w space, with initial control points depicted in red and new control points highlighted in blue

5.2.3 Lattice and NURBS parametrization for surface defect simulation

Having established the theoretical framework of FFD parameterization using NURBS volumes, we now discuss how this technique is specifically adapted to the simulation of surface defects. This involves adapting the defect model to the shape characteristics of the surface being deformed. To effectively classify and manage the different surface types, we distinguish between simple and complex surfaces based on their geometrical properties.

To determine whether a surface is simple or complex, we analyze the normal vectors at various points within the area of interest. A surface is classified as simple if it can be closely approximated by a plane. The standard deviation of the normal vector at the surface points within the selected area is computed. If this value approximates zero, it is inferred that all normal directions are similar, thereby indicating a flat surface. If the value of the standard deviation is high, exceeding a predetermined threshold fixed experimentally, the surface is classified as complex, due to the assumption that the direction of the normal changes over the entire surface, as illustrated in Figure 5.4.



(a) (b)

Figure 5.4: Representation of the normal vectors of the faces only in the selected area, indicated by blue triangles. (a) Simple surface. Normals have practically the same direction in the selected area. (b) Complex surface. Normal direction changes over the selected surface

For both simple and complex surfaces, an elemental 3D box that encompasses the entire defect geometry is constructed using a structured Cartesian mesh of control points. These control points are then adjusted according to the defect model and the normal direction specific to each point on the surface. Although this general approach is applied to both simple and complex surfaces, each category has its unique particularities and adjustments. These specific considerations will be discussed in the following sections.

Simple surfaces

For simple surfaces, a lattice is constructed with two parallel planes of control points aligned with the surface: one above and one below.

The lattice dimensions in the uv plane ($L \times W$) are determined by the defect's length and width, while the w-dimension depends on the maximum depth of the defect, ensuring it encloses the selected surface area of the model, as deformation will solely affect the geometry within the lattice.

To position the lattice correctly, the orientation is set so that the base of the lattice is parallel to the chosen region of the surface in the global xyz coordinates. Additionally, the lattice can be rotated around the surface's normal axis to give a different orientation to the defect.

Internally, a NURBS volume consisting of $a \times b \times 2$ control points is established

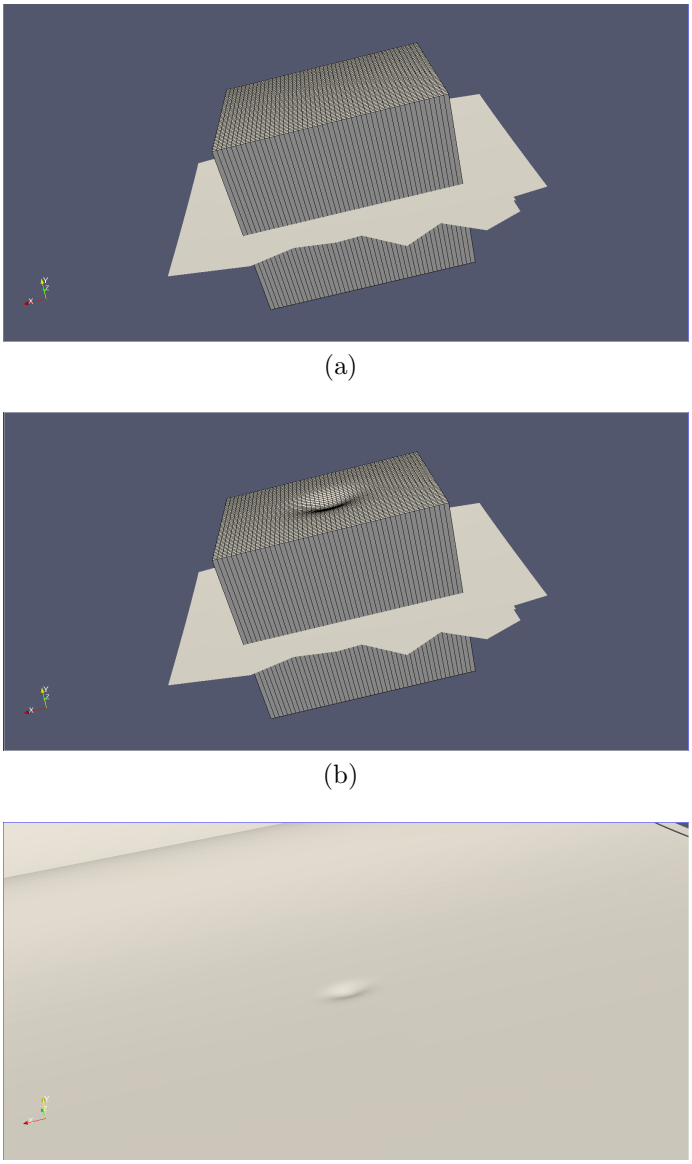
for simple surfaces. The values of a and b are determined based on the defect's size and desired precision using equation 5.7. The parameters δ_u and δ_v denote the maximum spacing between control points in the uv plane.

$$a = \left\lceil \frac{L}{\delta_u} \right\rceil, \quad b = \left\lceil \frac{W}{\delta_v} \right\rceil \tag{5.7}$$

Points in the upper grid are displaced along the w-axis according to the elevation value obtained from the defect's elevation map at the corresponding uv position. The lower grid remains unchanged as it only defines the lattice's size and encloses the surface to be deformed. In section 5.3, mathematical models to compute elevation maps corresponding to the most frequent types of defects are described. So, to simulate the defect on a simple surface, each control point $Q_{i,j,k}$ is adjusted using the equation 5.8, where $Q_{i,j,k}$ represents the original position of the control point in the uvw space, and $Z(u_i, v_j)$ denotes the displacement in the w-direction based on the surface's elevation map at point (u_i, v_j)

$$Q'_{i,j,0} = Q_{i,j,0} + Z(u_i, v_j) \quad \forall i \in [0, a), j \in [0, b) \tag{5.8}$$

Figure 5.5 shows the process of making any defect on a simple surface. First, we build the lattice with the appropriate dimensions, see Figure 5.5a. Then the control nodes are displaced according to the characteristics of the defect, see Figure 5.5b. Finally, the lattice is applied to the vertices of the triangular mesh to transfer the defect to the surface. The surface with the simulated defect is shown in Figure 5.5c.



(a)

(b)

(c)

Figure 5.5: (a) Construction of the lattice around the 3D model area. (b) Result of the lattice after displacement of the control nodes according to the defect. (c) Result of applying the deformation of the lattice to the 3D model. Final defect

Complex surfaces

As in the simple surfaces, a lattice of dimensions according to the size of the defect is constructed, ensuring that the selected surface fits inside it. The lattice dimensions in the uv plane ($L \times W$) are determined by the defect's length and width, while the w-dimension depends on the maximum depth of the defect, ensuring it encloses the selected surface area of the model, as deformation will solely affect the geometry within the lattice.

The lattice is aligned parallel to the global coordinate system of the surface, as depicted in Figure 5.6a(a), ensuring its base matches the chosen region in the xyz coordinates. Furthermore, the lattice can be rotated around the surface's normal axis to provide different orientations for simulating the defect.

For complex surfaces, a NURBS volume is defined using $a \times b \times c$ control points, where a, b, and c are calculated based on the defect size and desired precision according to equation 5.9. The spacing between control points, denoted as δ_u, δ_v, and δ_w, determines the resolution. The dimensions of the lattice, denoted as $L \times W \times H$, guide the overall shape.

$$a = \left\lceil \frac{L}{\delta_u} \right\rceil, \quad b = \left\lceil \frac{W}{\delta_v} \right\rceil, \quad c = \left\lceil \frac{H}{\delta_w} \right\rceil \tag{5.9}$$

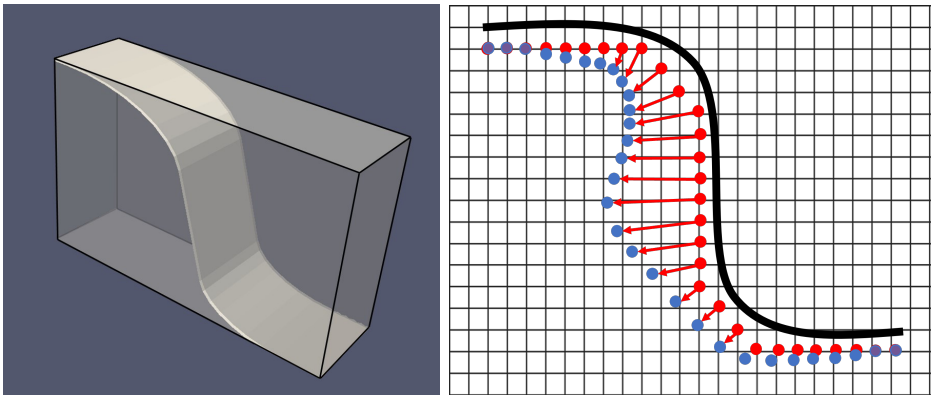
Control points $Q_{i,j,k}$ are displaced along the normal vector $\vec{N}(x_i, y_j, z_k)$ at that point , scaled by a factor $Z(u_i, v_j)$ derived from the surface's elevation map at the corresponding uv position. So, to simulate the defect on a simple surface, each control point $Q_{i,j,k}$ is adjusted using the equation 5.10.

$$\begin{gathered} Q'_{i,j,k} = Q_{i,j,k} + \vec{N}(x_i, y_j, z_k) Z(u_i, v_j) \\ \forall i \in [0, a), j \in [0, b), k \in [0, c) \end{gathered} \tag{5.10}$$

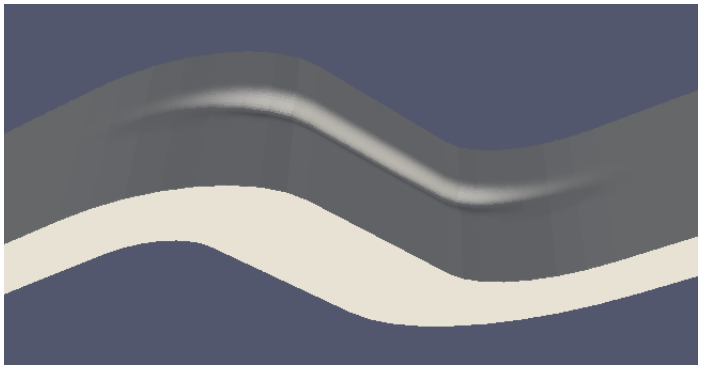
First, the control points closest to the surface are identified based on their proximity to the vertices of the triangular mesh. This is achieved by calculating the Euclidean distance between each control point and each vertex of the mesh. Control points at a distance lower than a threshold are considered. These selected control points are the ones that will be displaced to deform the surface. Once the closest control points are identified, the local orientation of the surface at each control point is calculated considering a small window of neighboring vertices around each control point. The displacement direction and magnitude are then determined from the

elevation map representing the defect.

Finally, the control points are moved according to the displacement direction and magnitude calculated for each one, see Figure 5.6b.



(a) (b)



(c)

Figure 5.6: (a) Construction of the lattice with the complex surface fitting into it. (b) Cross-section of the complex surface. It shows a 2D representation of displacements of the lattice control points to get the defect shape in complex surfaces. In red: control points before displacement. In blue: control points after displacement. (c) Result of applying the deformation of the lattice to the 3D model. Final defect

Figure 5.6 (b) illustrates a cross-sectional view of the complex surface, displaying 2D displacements of lattice control points to simulate the defect. Red points depict control points before displacement, while blue points show them after displacement. Figure 5.6 (c) demonstrates the resulting 3D model after applying the lattice deformation, showcasing the final defect.

5.3 Mathematical model of defects

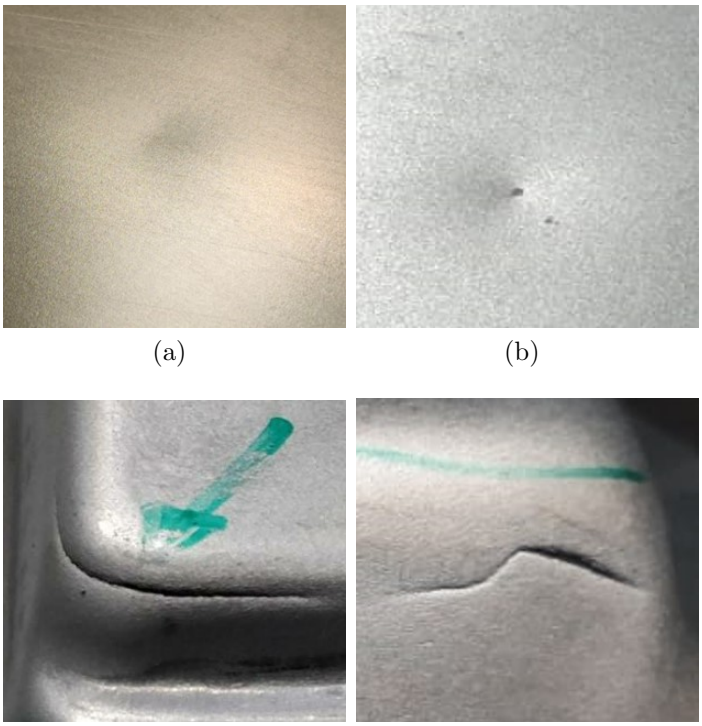
Describing defects as an elevation map defined on a regular grid in the XY plane is a common approach to characterize any type of defect. However, this methodology can be laborious when dealing with defects of different shapes or sizes. Therefore, it is convenient to have parametric mathematical models that can easily describe various families of common defects.

During steel production, surface defects are frequently encountered on products such as sheet metal and car doors. These defects can appear due to different factors, such as inherent flaws in the manufacturing process or improper handling during part manipulation. This paper focuses on three common types of surface defects: bumps, peaks, and cracks. Here, mathematical models are proposed to simulate the geometric characteristics of these specific defect types.

During manufacturing or transportation processes, parts can be susceptible to impacts that result in surface imperfections or small bumps. These defects may vary slightly in shape or depth depending on their causes but typically manifest as changes in curvature with a smooth contour. Bumps are commonly observed as depicted in Figure 5.7a. A Gaussian-shaped surface was selected to simulate this type of defect. The smoothness, symmetry, and adjustable parameters of the Gaussian distribution allow for the replication of the gradual change in curvature and specific characteristics of dents.

Peaks are another typical surface defect observed in manufactured objects, as seen in Figure 5.7b. This type of defect differs significantly from bumps. Instead of having a smooth shape, peaks terminate in a sharp, pointed form, making them much more abrupt. To accurately replicate this sharp geometry, we employ a double-exponential surface model. This model precisely captures the sharpness and abruptness of peaks, enabling a realistic representation of their geometry.

The third type of defects range from small stretches to cracks that propagate through the material, as depicted in Figures 5.7c and 5.7d. These defects share similarities with bumps on an individual profile level, but instead of having a circular area, they extend in a specific direction. Another distinction is that they typically appear in areas of more complex curvatures due to material stress during manufacturing. To capture the characteristics of cracks, Gaussian curves are used along the axis. This model effectively represents the varying extensions and orientations of cracks, providing a flexible approach to simulate their complex curvature patterns.



(a) (b) (c) (d)

Figure 5.7: Images of some common defects in steel products. (a) Bump. (b) Peak. (c)(d) Cracks

5.3.1 Crack model

Stretches and cracks are modeled by a set of sections, distributed along a polygonal curve that represents its medial axis. Each section is a Gaussian-shaped curve with its own parameters to define its particular width and depth. In addition, all of them are uniformly distributed along the medial line.

The medial axis of the crack is modeled as a polygonal curve L with $(n+1)$ vertices. Each vertex, $\mathbf{V}^i = (v_x^i, v_y^i)$, is defined by its coordinates in the X-Y plane. The X-axis is oriented along the main axis associated with the set of sections and the first vertex ($\mathbf{V}^0$) is selected as the origin of the crack.

$$L = \left\{ \mathbf{V}^0, \mathbf{V}^1, ..., \mathbf{V}^n \quad | \quad \mathbf{V}^i \in R^2, \mathbf{V}^0 = \mathbf{0} \right\} \tag{5.11}$$

The medial axis L is sampled at m points, $\{\mathbf{P}^i = (p_x^i, p_y^i)\}$, evenly distributed along the X-axis in the range $[v_x^0, v_x^n]$. Those points are the centers of the different sections that complete the defect model. To add some variability to the model, a zero mean Gaussian random noise is added to the y-coordinate of each center. The maximum variation allowed of the y coordinate is defined by the parameter δ_y. Figure 5.8 shows the model of the crack medial axis in the X-Y plane.

$$p_x^i = \frac{i}{m-1}(v_x^n - v_x^0) \quad \forall \quad i \in \{0, 1, ..., m-1\} \tag{5.12}$$

$$p_y^j = \text{lininterp}(L, x_i) + \frac{\delta_y}{3}\epsilon_i, \quad \{\epsilon_i\} \sim N(0, 1) \tag{5.13}$$

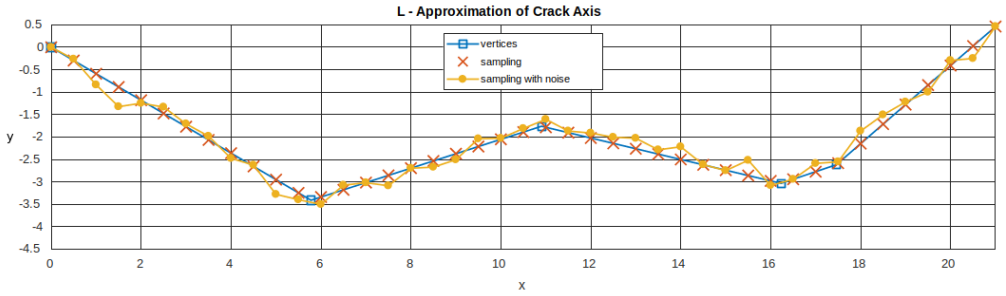


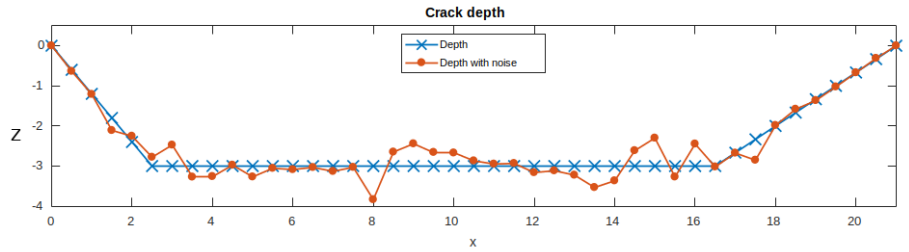
Figure 5.8: Model of the crack medial axis. First, a polygonal approximation is defined using a set of vertices (blue line with square markers). Then, this curve is sampled to obtain the initial position of each section center (red crosses). To finish, random noise in the vertical direction is added to each center (dotted yellow line)

To complete the model, the shape of the crack at each center point ($\mathbf{P}^i$) is described as a Gaussian-shaped curve S^i parallel to the Y-Z plane. Each section is formed by n_s samples. The depth and width values of each section are computed using a trapezoidal profile. Both values increase linearly from zero to their maximum in the m_1 initial sections and they decrease to zero in the final m_2 ones. Nominal width and depth remain constant in the central portion of the crack. This is represented in Figure 5.9. As with the position of the section center, a certain amount of zero mean Gaussian noise is added to the depth and width of each section. The parameters δ_z and δ_w control the maximum random variation of a section. Subsequently, crack width for each section is computed according to equation 5.14, following the trapezoidal profile as previously described. The first and last width values are set

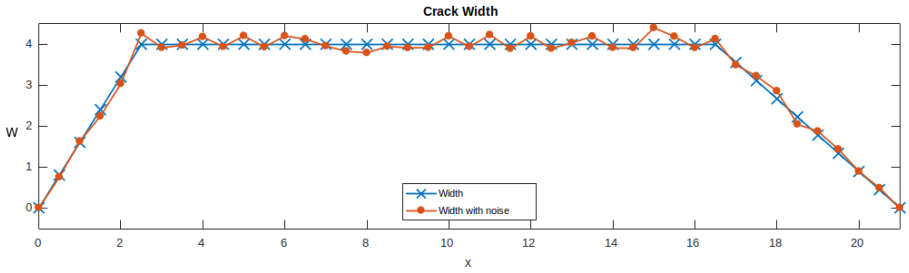
to zero. Additionally, the noise model is applied to the crack width, as outlined in Equation 5.15.

$$w_0 = w_m = 0$$
$$w_i = \begin{cases} i\frac{W_{max}}{m_1} & if \quad 1 \leq i \leq m_1 \\ W_{max} & if \quad m_1 < i < m_2 \\ (m-i)\frac{W_{max}}{m-m_2} & if \quad m_2 \leq i \leq m \end{cases} \tag{5.14}$$

$$w_i = w_i + \frac{\delta_w}{3}\epsilon_i, \quad \{\epsilon_i\} \sim N(0,1), \quad 0 < i < m \tag{5.15}$$



(a)



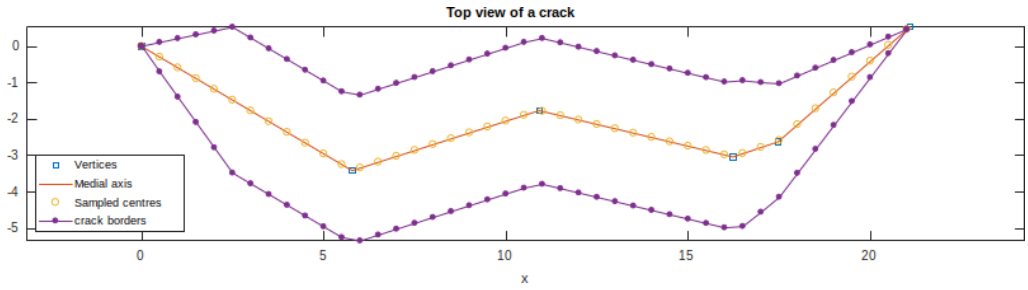
(b)

Figure 5.9: Crack depth (a) and width (b) present a trapezoidal profile with an initial linearly increasing ramp extending over m_1 samples, and a final linearly decreasing ramp that is m_2 samples wide

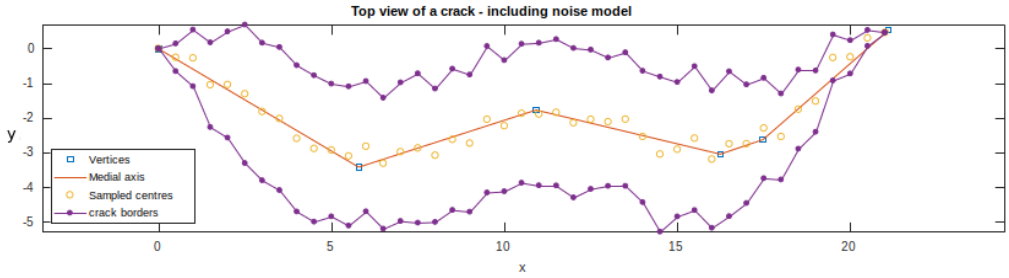
The crack depth for each section is computed in the same way as the width, as detailed in Equation 5.16. The first and last depth values are set to zero and the noise model is incorporated as shown in Equation 5.17.

$$z_0 = z_m = 0$$
$$z_i = \begin{cases} i\frac{Z_{max}}{m_1} & if \quad 1 \leq i \leq m_1 \\ Z_{max} & if \quad m_1 < i < m_2 \\ (m-i)\frac{Z_{max}}{m-m_2} & if \quad m_2 \leq i \leq m \end{cases} \tag{5.16}$$

$$z_i = z_i + \frac{\delta_z}{3}\epsilon_i, \quad \{\epsilon_i\} \sim N(0,1), \quad 0 < i < m \tag{5.17}$$



(a)



(b)

Figure 5.10: Effect of adding a random variation to each section. (a) Medial axis and external contour of the crack without noise. (b) The same information after adding noise to each section. A large amount of noise has been added to show the effect more clearly

Table 5.1 lists all the parameters and variables used to model a crack. The algorithm to compute the elevation model of crack-type defect calculation is described as follows. Initially, the coordinates associated with the center points of each crack section $\mathbf{P}^i = \left(p_x^i, p_y^i\right)$ are computed using Equations 5.12 and 5.13.

Table 5.1: Parameters and variables of a Crack type defect.

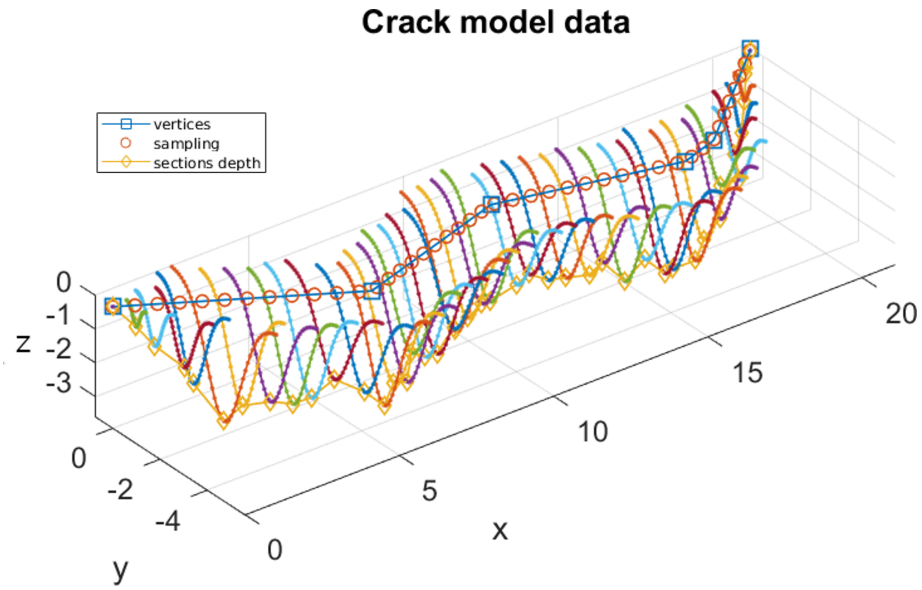
Variables	
L	Polygonal approximation of the medial axis of the crack
$\mathbf{P}^i = (p_x^i, p_y^i)$	Each of the centers sampled from $L, i \in \{0, 1, ..., m-1\}$. $\mathbf{P}^0 = \mathbf{V}^0$ and $\mathbf{P}^{(m-1)} = \mathbf{V}^n$
$(\hat{x_k}, \hat{y_k}, \hat{z_k})$	Normalized crack section. It is used as a template to compute the position of every section.
z_i	Depth of section i.
w_i	Width of section i.
$C = \left\{ \left(c_x^{i,k}, c_y^{i,k}, c_z^{i,k} \right) \right\}$	The set of points that form the crack wall. i represents the section ($i = 0, \ldots, m-1$) and k represents a sample in that section ($k = 0, \ldots, n_s - 1$).
Parameters	
n	L is formed by the origin and n extra vertices. The curve is formed by the segments connecting two consecutive vertices.
$\mathbf{V}^i = (v_x^i, v_y^i)$	Each vertex in $L, i \in \{0, 1, ..., n\}$
m	The number points sampled from L to describe the crack section. The samples are evenly spaced along the x-axis and include the first and last vertex.
Z_{max}	Maximum depth of the crack
W_{max}	Maximum width of the crack
m_1	Number of samples to achieve the maximum depth or width, starting from the first sample.
m_2	Number of samples to descend from the maximum depth or width to zero at the last sample.
$\delta_y, \delta_z, \delta_w$	Maximum random variation of the y-coordinate of section center, depth and width respectively.
n_s	The number of samples to describe the section

To make computations faster, a normalized crack section, $(\hat{x_k}, \hat{y_k}, \hat{z_k})$, is computed. This section is the result of taking n_s equally spaced samples from a Gaussian-shaped curve in Y-Z plane, as shown in equation 5.18:

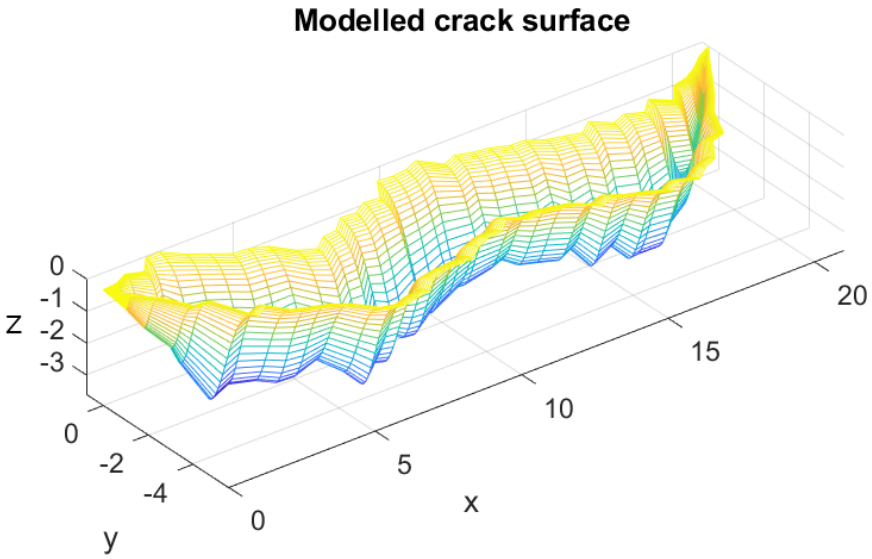
$$\begin{aligned} \hat{x}_k &= 0, & 0 \leq k < n_s \\ \hat{y}_k &= -3 + k\frac{6}{n_s - 1}, & 0 \leq k < n_s \\ \hat{z}_k &= e^{\frac{-y_k^s}{2}}, & 0 \leq k < n_s \end{aligned} \tag{5.18}$$

For each section $i \in \{0, 1, 2, ..., m-1\}$, Equations 5.14, 5.15, 5.16 and 5.17 are used to determine the actual width w_i and depth z_i corresponding to section i. These values, together with the position of its center $\mathbf{P}^i$, are used to compute the final coordinates of the section $\left\{\left(c_x^{i,k}, c_y^{i,k}, c_z^{i,k}\right)\right\}$ using equation 5.19. Figure 5.11 shows the final result.

$$\begin{aligned} C_x^{i,k} &= x_i, & 0 \leq k < n_s \\ C_y^{i,k} &= y_i + \hat{y}_k\frac{w_i}{6}, & 0 \leq k < n_s \\ C_z^{i,k} &= z_i\hat{z}_k, & 0 \leq k < n_s \end{aligned} \tag{5.19}$$



(a)



(b)

Figure 5.11: Volumetric representation of a crack. (a) The medial line that defines the crack and the set of sections. (b) Resulting surface representing the crack

5.3.2 Bump model

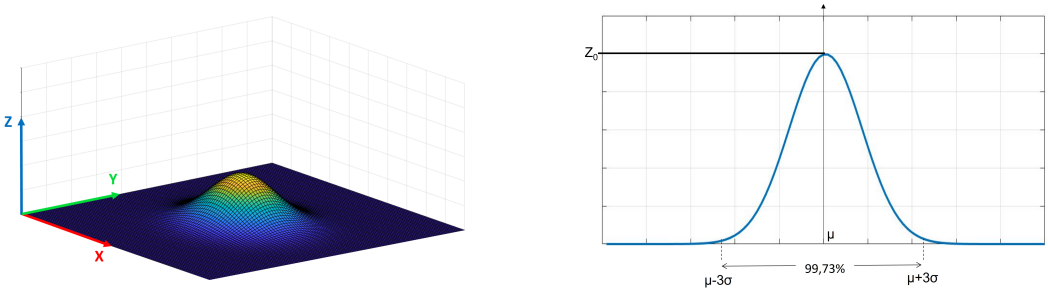
Bumps are surface defects characterized by changes in curvature with a relatively smooth profile. To replicate the gradual change in curvature and specific bump characteristics, a Gaussian-shaped surface model is build according to equation 5.20.

$$Z(\mathbf{P}) = Z_{max} \cdot e^{-\frac{1}{2}(\mathbf{P}-\mathbf{P}_0)^T \sum^{-1} (\mathbf{P}-\mathbf{P}_0)} \tag{5.20}$$

$$\sum = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix} \tag{5.21}$$

In this context, $Z(\mathbf{P})$ represents the surface's elevation at a specific point $\mathbf{P} = (x, y)$. The surface's overall appearance is determined by a gain parameter denoted as Z_{max}, allowing control of the maximum depth of the bump and fine-tuning the defect's prominence. The center of the Gaussian curve, and thus the center of the defect, is defined by $\mathbf{P}_0 = (x_0, y_0)$. The shape and characteristics of the surface are influenced by the covariance matrix $\sum$ as outlined in Equation 5.21. The standard deviations σ_x and σ_y control the shape and characteristics of the surface.

To ensure that the edges have a very small value and that there are no abrupt jumps at the edges of the defect. According to the three-sigma rule of thumb, in a normal distribution, 99.7% of the values are considered to lie within three standard deviations of the mean. Therefore, it is established that the standard deviation of the Gaussian should be at least 6 times smaller than the size of the defect in that direction (Di). The resulting curve is depicted in Figure 5.12.



(a) (b)

Figure 5.12: Result of the bump parametrization. (a) 3D surface representation. (b) 2D section of the surface

5.3.3 Peak model

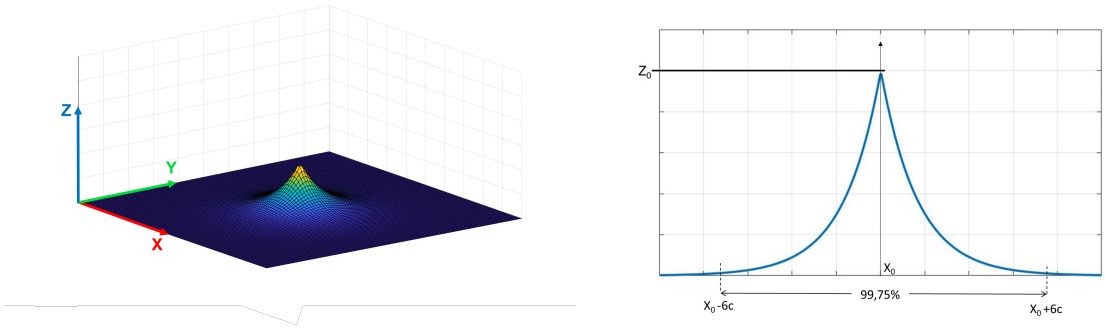
Peaks are characterized by their pointed shape, creating a pronounced and distinctive appearance. This particular defect is parameterized using an exponential decay surface model, outlined in Equation 5.22. It defines the height Z as a function of the point position in the XY-plane $\mathbf{P} = (x, y)$. The surface is characterized by a maximum height Z_{max} and a decay factor c.

The height decreases exponentially as the distance between the point $\mathbf{P}$ and the defect's center point $\mathbf{P}_0 = (x_0, y_0)$ increases. The distance $||\mathbf{P} - \mathbf{P}_0||$ between the two points is calculated using the Euclidean norm.

$$Z(\mathbf{P}) = Z_{max} \cdot e^{-\frac{1}{c}\|\mathbf{P}-\mathbf{P_0}\|} \tag{5.22}$$

Decay is controlled by the factor $\frac{1}{c}$. To ensure that the curve fits within the selected area and exhibits a smooth transition at the defect edges, we introduce the scaling factor c. The value of c is estimated such that 99.73% of the distribution is covered. Using the quantile function of an exponential distribution, as shown in Equation 5.23, we find that 99.73% of the distribution is reached at $x = \frac{5.9143}{\lambda}$. Here, $\lambda = \frac{1}{c}$, and we set c to be at least six times smaller than the defect radius.

$$x = F^{-1}(p|\lambda) = \frac{-ln(1-p)}{\lambda}, \quad 0 < p < 1 \tag{5.23}$$



(a) (b)

Figure 5.13: Result of the peak parametrization. (a) 3D surface representation. (b) 2D section of the surface.

Figure 5.13 presents the result of the peak parametrization, including a 3D surface representation (Figure 5.13a) and a 2D section of the surface (Figure 5.13b). The chosen parameters and equations ensure a smooth and accurate representation of the exponential-shaped defect.

5.4 Resultados

This section outlines the results obtained from our experimental study, which aimed to simulate realistic defects rather than replicate specific existing ones. This approach means that direct quantitative comparisons between simulated and real defects are not feasible. The methodology involved training a neural network, specifically YOLO (You Only Look Once), with simulated defects and assessing its defect-detection performance on real scans. Subsequently, the results were compared with those obtained from another neural network trained exclusively on real defects, both applied to detect defects within the same set of real scans. This comparison aimed to evaluate the reliability of the proposed surface defect simulation on CAD models.

In this study, the focus was placed on the surface inspection of a car's bodywork. A variety of defects were simulated on the CAD model of the car door, encompassing diverse geometries and dimensions. Subsequently, the inspection system simulator was employed to create a comprehensive database with 3D information for these simulated defects. Figure 5.14 shows the CAD model of the car door used in these experiments.

Based on the car door CAD model, a variety of defects were simulated. These simulations encompassed bumps, peaks, and cracks of different sizes and shapes, varying in terms of width, length, depth, and orientations. The defects were strategically placed at multiple locations on the model to simulate a wide range of realistic surface imperfections typically observed in car bodywork. This approach facilitated the creation of a comprehensive dataset that captures the diversity of these defects.

To generate the 3D dataset, simulations of scans were necessary using a 3D profilometric sensor integrated into the simulator. This sensor facilitates data acquisition by loading the CAD model of the product under inspection. The parameters of the simulated profilometric sensor were adjusted to closely match those of the real inspection system used in the experiments, specifically the Automation Technology model AT-C5-4090CS30-495 [91], as detailed in Table 5.2.

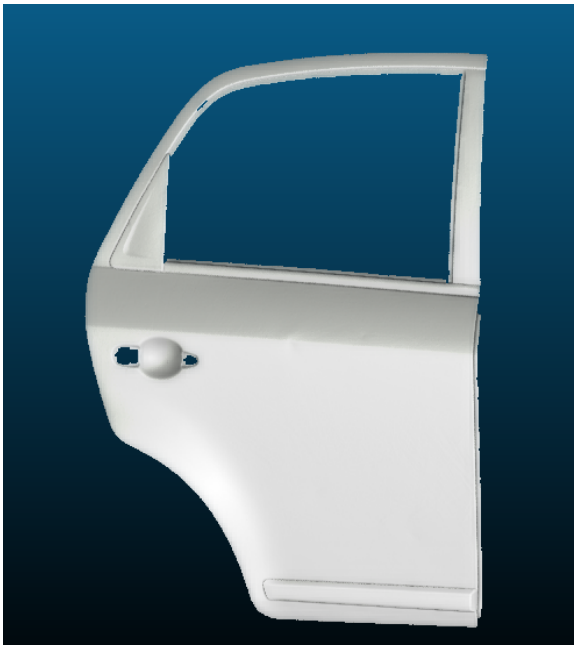


Figure 5.14: CAD model of the car door employed in these experiments.

Additionally, to enhance the realism of the simulated scans, two types of noise can be introduced into the measurements: sensor uncertainty and Speckle noise arising from material roughness. Speckle noise is generated using Perlin noise and is dependent on the specific material properties of the part. In this study, steel parts produced through stamping, such as car doors, were utilized. The Speckle noise is generated based on the roughness information extracted from actual measurements, resulting in an amplitude of 0.0532 mm and a density of 6 peaks/mm. Furthermore, sensor uncertainty is modeled using a zero-mean Gaussian distribution with a magnitude of 15 microns. The details of the simulation process have been previously describe in Chapter 4.

For the sensor's trajectory, the simulated experiments closely mirrored real-world scenarios by following a straight line between two points, covering the scanning area of the defective part. The travel speed of the simulated experiments matched that of the real system, ensuring accurate replication of inspection conditions.

For the implementation, the open-source C++ library called Mimmo [92] is employed for manipulating and morphing surface and volume meshes. This library proved instrumental in handling the NURBS required to construct the defects based on the mathematical models proposed in the previous section. The algorithms were

Table 5.2: Parameters of the profilometric sensor used in the experiments.

AT-C5-4090CS30-495 characteristics	
Working distance	400 mm
Z-Range	250 mm
FOV	63.5°
Points per profile	4096 pixels
Z resolution	3.8 μm
Profile speed	500 FPS
Travel speed	0.1 m/s

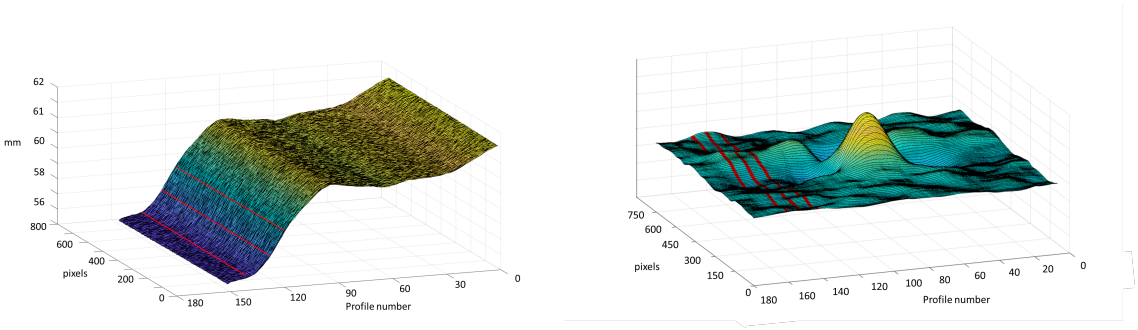
implemented using C++ programming language, running on the Ubuntu 20.04 LTS operating system. To analyze and process the obtained results, we used MATLAB 2022b.

5.4.1 Comparison of Simulated and Real Defects

In this section, a comparison between real measurements and simulated ones for each type of defect, bumps, peaks, and cracks, is presented. Although the simulated dimensions may not be an exact match due to noise in modeling and capturing, a qualitative comparison can be still performed. Several factors influence each defect, including the specific area of the part where it is located, the noise introduced by the inspection system, and the characteristics of the material being inspected.

It's important to note that in the context of a real 3D image, the defect might be challenging to discern due to its small size relative to the overall part, as depicted in Figure 5.15a. To enhance the visibility of peak and bump defects, a preprocessing technique is employed using two Laplacian filters at different scales. To account for resolution changes in each dimension, we utilize two distinct variances. This approach ensures that small deformations stand out from their surroundings, enabling comfortable visualization. It effectively separates the shape of the defect from the actual shape of the part, as illustrated in Figure 5.15b.

In the case of crack-type defects, to decouple the shape of the part, a polynomial profile fitted with the points of each profile is estimated and the difference with the real scan is calculated. This produces an upward effect at the edge of the crack.

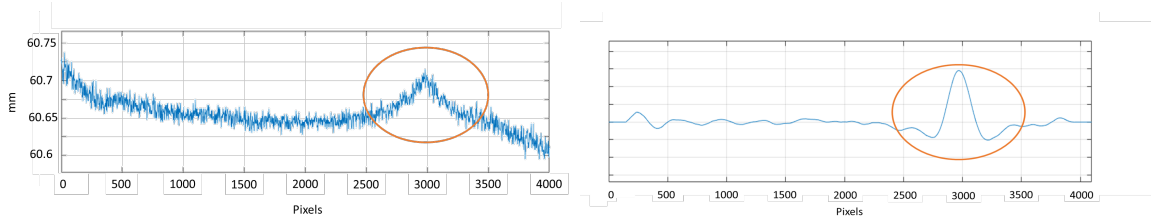


(a) (b)

Figure 5.15: (a) Surface reconstruction from scans without any processing. The defect is not visible due to its small size in relation to the shape of the part. Red lines represent individual profiles from profilometric sensor. (b) Surface reconstruction after preprocessing to emphasize small variations. The defect is now clearly visible.

If only a profile of the scan in the defective area is represented, it may not be necessary to apply the processing explained above. In this case, a raw profile, the geometry of the part does not vary significantly, so the shape of the defect is perfectly visible, see Figure 5.16a.

From now on, the unprocessed profiles will be presented, because they show the real shape of the defect, while the processed profiles are used to look for continuity along with the different profiles of the deformation but distort the real shape of the defect, see 5.16b.



(a) (b)

Figure 5.16: An isolated profile of the scan of figure 5.15 in the defect area. Orange circle enclose the defect. (a) Raw measurement without any processing, corresponding to figure 5.15a. (b) Measurement after processing to emphasize small variations, corresponding to figure 5.15b

Figure 5.17 displays the inspection profiles, while Figure 5.18 showcases a pre-processed reconstruction of the scanned surface within the defective area. Through this analysis, the general shape of the defects and their dimensions can be analized.

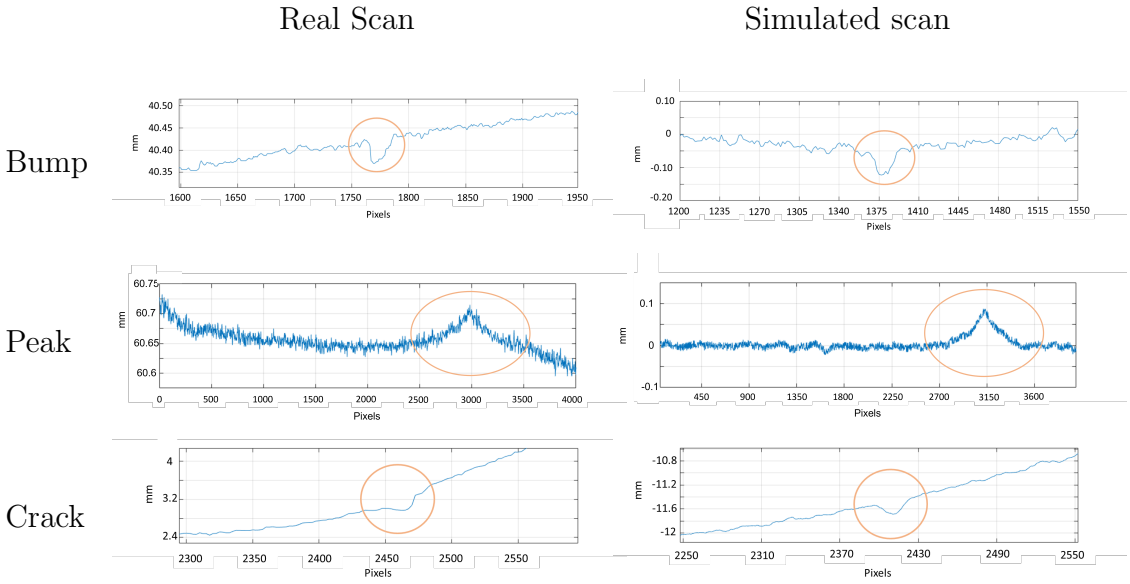


Figure 5.17: Inspection profiles of each type of defect. Orange circles mark the shape of the defect. (a) Real inspection. (b) Simulation.

The first row of the figures represents a simulated bump with a diameter of 1 mm and a depth of 70 μm. The second row presents a peak-type defect with a depth of 70 μm and a diameter of 3 mm. Finally, the third row shows a crack profile with a maximum depth of approximately 0.3 mm, a maximum width of 2 mm, and a length of 2 cm. To closely resemble a real crack obtained from a scan, it was simulated by carefully placing points that define its shape to match the actual crack, as illustrated in Figure 5.19.

The difference between the three types of defects can be easily seen: bumps exhibit a sudden discontinuity in the shape of the part with a relatively smooth contour. Peaks, on the other hand, display a pointed structure. Cracks, at a profile level, are not very different from a bump. However, the surface shows a deformation that extends in the direction of the material stretch, growing as it approaches the center of the defect and decreasing as it moves away from the center and gets closer to a defect-free zone.

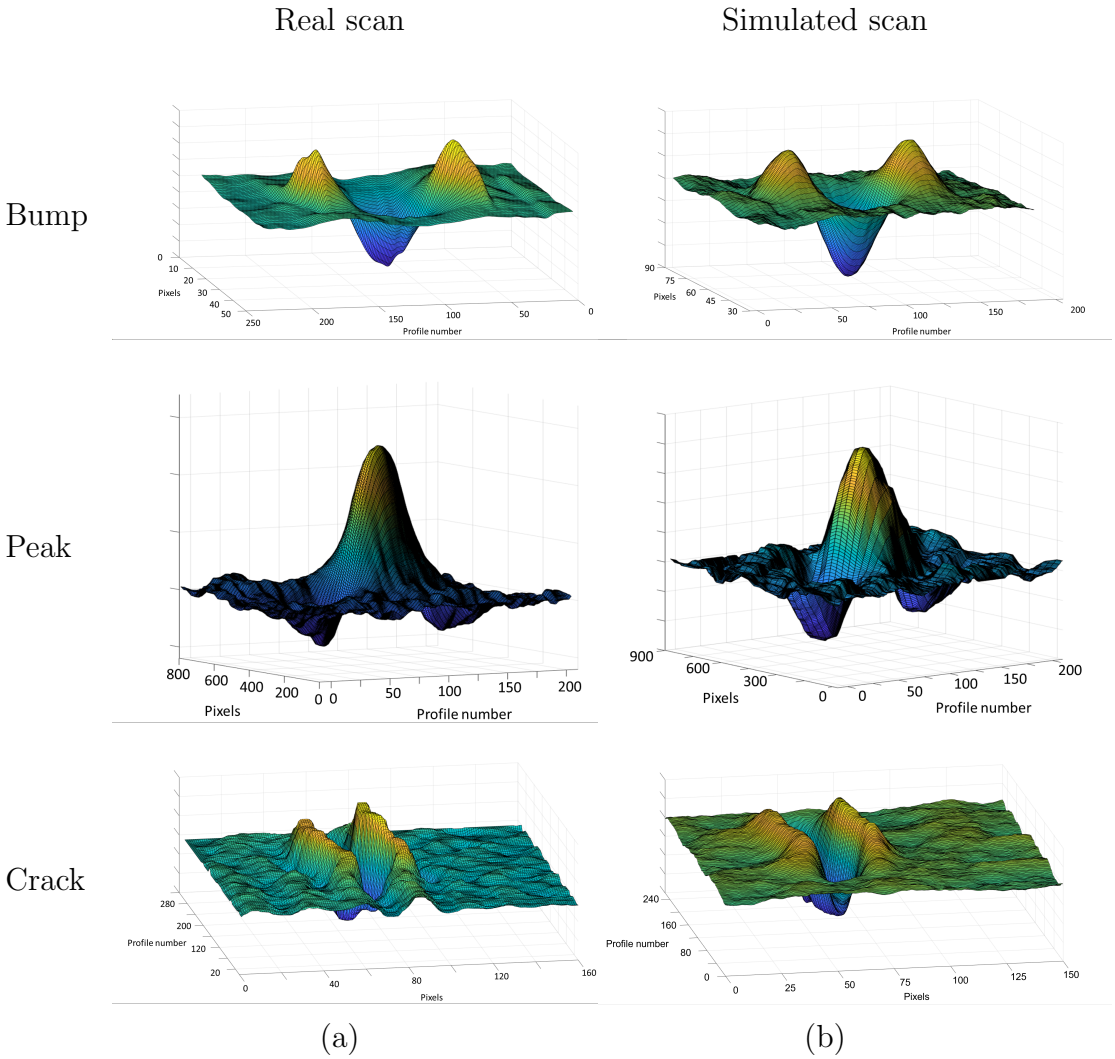
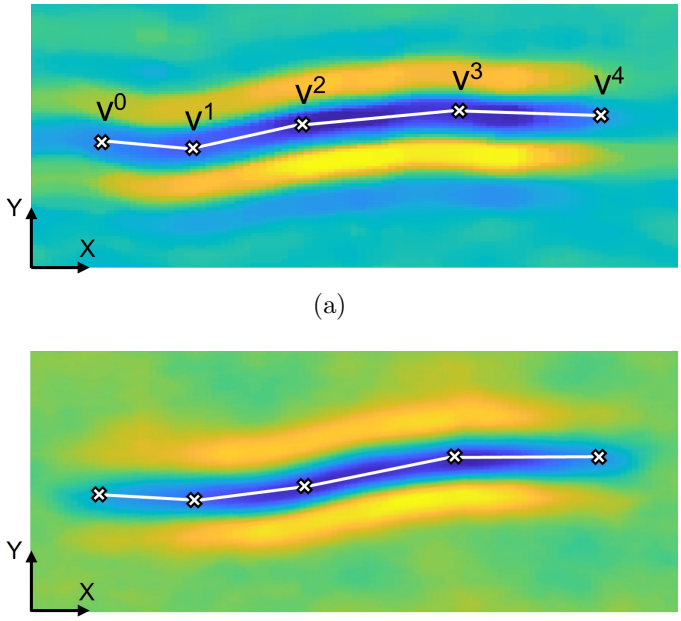


Figure 5.18: Reconstruction of defects from simulated scans after preprocessing to emphasize small variations. (a) Real inspection. (b) Simulation



(a)

(b)

Figure 5.19: (a) Diagram of the layout used to simulate a crack based on a real scan image. (b) Simulated scan of a crack constructed from the real crack shown in (a). White crosses mark the points that define the shape of the crack.

In the simulation of defects, we consider two scenarios based on the geometry of the model in the target deformation area: simple and complex surfaces. While the method developed for complex surfaces is applicable to simple surfaces, processing flat areas can be expedited by reducing the number of control points and simplifying point proximity search. We tested the computational time difference in simulating defects on flat surfaces using both strategies. For a 10x10mm bump with 1mm depth, an 87.8% speed-up was achieved with the specific flat area strategy compared to the generic complex area strategy. Similarly, simulating a 120x90x1mm bump resulted in a 70.6% speed-up. It's essential to note that simulation time varies based on defect characteristics, surface area, and the 3D model.

5.4.2 Machine Learning-based defect detection

This section evaluates simulated defects by comparing them to real-world defects using the YOLO (You Only Look Once) neural network model. A diverse dataset of simulated defects, including bumps, peaks, and cracks of various dimensions and

orientations, was used to train YOLO. Subsequently, the model's performance in detecting real defects was tested on real-world scans to gauge how accurately the simulated defects replicate actual defect characteristics. Additionally, another YOLO model was trained exclusively on a dataset of real defects sourced from real-world scans. Both models were then evaluated on the same set of real scans, enabling a direct comparison of their detection accuracy and generalization ability.

To generate a robust dataset, a diverse collection of labeled defects was created. This dataset captures a wide range of defect types: bumps, peaks, and cracks. Each defect type was varied in dimensions such as width, length, depth, and orientation, and was distributed across different locations on the CAD model of a car door. The aim was to replicate realistic defect scenarios that might occur during the manufacturing and handling of steel parts.

An artificial dataset consisting of 285 diverse defect instances was meticulously crafted for training purposes. This dataset encompassed a comprehensive range of defect shapes, sizes, and positions, ensuring that the models could generalize effectively and detect defects in various scenarios. To further enrich the training data, a systematic data augmentation process was applied. This process involved simple transformations such as flipping the images vertically and horizontally, thereby expanding the dataset to 1140 instances.

In addition to the synthetic dataset, a real defect dataset was prepared. This dataset comprised 506 authentic defect instances collected from real-world scans. Similar to the synthetic data, this real defect dataset underwent the same data augmentation process, increasing the dataset to a total of 2024 samples.

These datasets were crucial for training two different YOLO models: one trained exclusively on the simulated defects and the other on real defects. The comparison between these models helps to determine how closely the simulated defects approximate the characteristics of real defects.

In Figure 5.20, depth images used to train the two models are displayed: one using real samples and the other using simulated samples. These images serve as samples of the defects employed in the training process, showcasing the differences and similarities between actual and simulated data for the purpose of model development and evaluation.

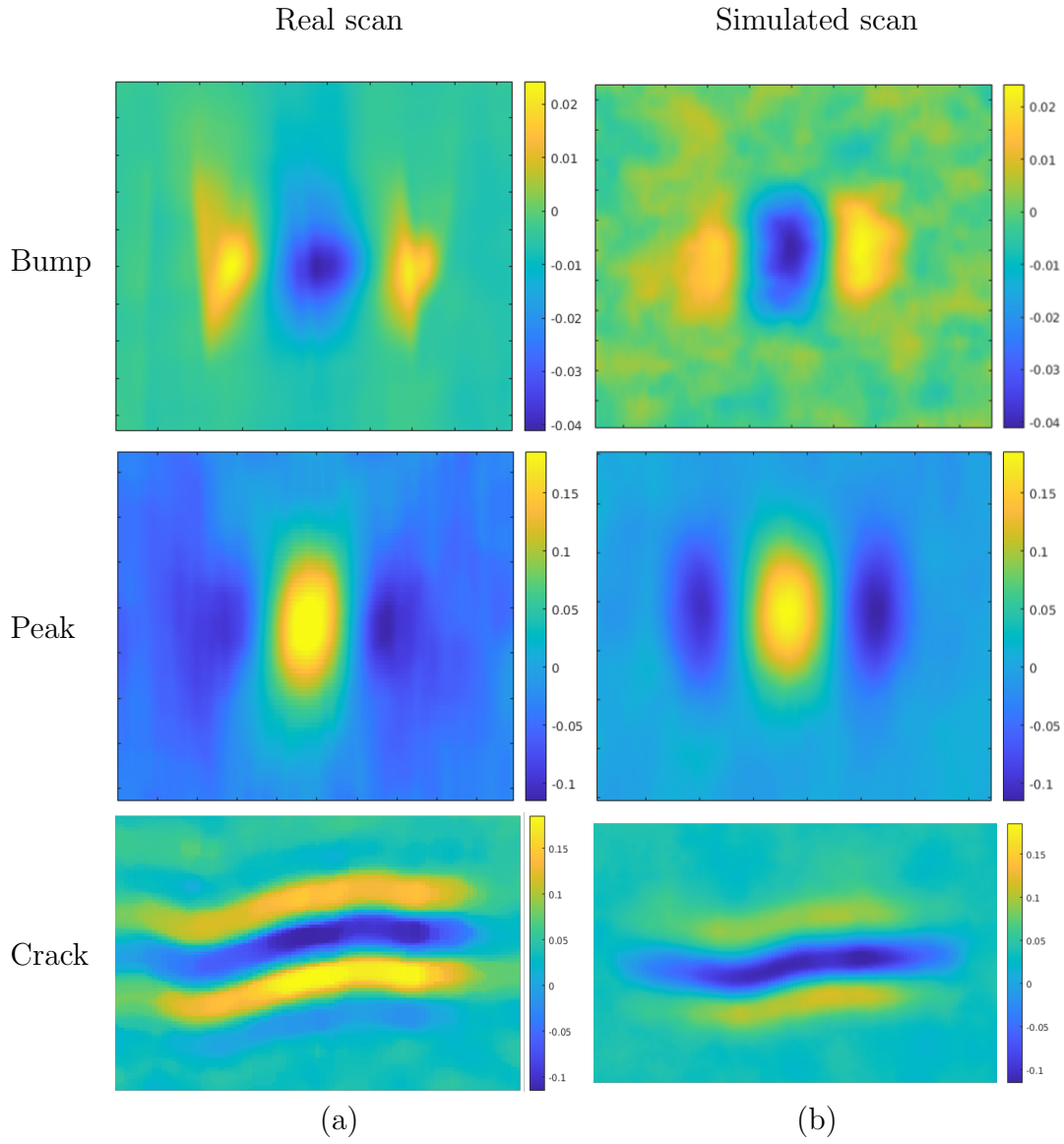


Figure 5.20: Depth image of real versus simulated defect samples used for training the different deep learning models. (a) Real samples. (b) Simulate samples

In the realm of machine learning for image processing, one-stage detectors like YOLO (You Only Look Once) [93] have become highly popular, particularly in defect detection algorithms, as indicated in [23]. Due to its high accuracy and high frame rate detection capability, YOLO is widely adopted in surface defect detection tasks across diverse industries such as steel manufacturing, precision electronic

components, and production equipment. For our study, we opted to use YOLO version 8, specifically opting for its smallest predefined architecture named "nano," to minimize computational latency and enhance the real-time usability of our defect detection system.

YOLOv8 architecture is mainly composed of three main components [94]. First, the "Backbone" consists of a custom CSPDarknet53 convolutional neural network (CNN) designed to extract features from the input image. This architecture uses partial inter-stage connections to improve the flow of information between layers and increase accuracy. Second, the "Neck", or feature extractor, fuses feature maps from different stages of the backbone to capture information at various scales. YOLOv8 employs an innovative module called C2f instead of the traditional Feature Pyramid Network (FPN), allowing the combination of high-level semantic features with low-level spatial information, which improves detection accuracy, especially for small objects. Finally, the "Head" is responsible for making predictions. YOLOv8 uses several detection modules that predict bounding boxes, objectivity scores, and class probabilities for each grid cell in the feature map. These predictions are combined to obtain the final detections.

Two models were trained using the same network architecture: one with simulated data and the other with real data. For now on, we will refer to them as simulated model and real model, respectively. Models were trained to identify various types of defects, and its performance was evaluated on a dataset containing 144 instances of real defects. The results are based on a comprehensive confusion matrix analysis and the calculation of several key performance metrics. Due to the scarcity of real samples, these models were trained with only one category: "Defect," without distinguishing between different types.

Accuracy indicators are commonly used to assess the effectiveness of defect detection systems [95]. These indicators include True Positives (TP), False Positives (FP), False Negatives (FN), Precision, Recall, F1-Score, and Average Precision (AP), each providing insights into different aspects of the model's performance in identifying defects accurately:

- True Positives (TP): Represents cases where the model correctly identifies a defect.
- False Positives (FP): Occurs when the model incorrectly identifies a non-defective sample as a defect.
- False Negatives (FN): Represents cases where the model fails to detect a defect

that is present.

- Precision: evaluates the accuracy of positive predictions made by the classifier by measuring the ratio of true positives to the total number of positive results predicted. It reflects the system's ability to avoid misclassifying positive samples:

$$Precision = \frac{TP}{TP + FP} \tag{5.24}$$

- Recall: assesses the system's ability to capture all relevant positive samples by calculating the ratio of true positives to all relevant samples that should have been identified as positive. It indicates the system's effectiveness in identifying positive samples across the dataset.

$$Recall = \frac{TP}{TP + FN} \tag{5.25}$$

- F1-Score: Harmonic mean of Precision and Recall, providing a single metric to balance these two aspects:

$$\text{F1-Score} = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{5.26}$$

- Average Precision (AP): measures the average precision of the model in identifying defects across all possible confidence thresholds. It quantifies the quality of the model's classification by considering both precision and recall over a range of confidence thresholds. AP is calculated as the area under the precision-recall curve. A higher AP indicates better performance of the model in defect detection, as it reflects high precision and recall at various confidence thresholds.

These metrics are computed for both the real and simulated models across a range of detection thresholds, from 0.1 to 0.5. The detection threshold determines the minimum confidence level required for predictions to be considered valid, crucially balancing the model's precision and recall. Setting a very low threshold can increase detections, including more false positives, whereas a high threshold may overlook valid detections, leading to more false negatives. It's critical to find a balance to optimize detection accuracy. Tables 5.3 and 5.4 present these metrics for the real and simulated models, respectively.

Additionally, Precision-Recall curves are computed to evaluate the models further. Figure 5.21 illustrates these curves for both models, depicting how precision

Table 5.3: Model Performance for the model trained with simulated data.

Threshold	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**
TP	124	112	94	76	60
FN	20	32	50	68	84
FP	64	22	14	10	4
Recall	0.8611	0.7778	0.6528	0.5278	0.4167
Precision	0.6596	0.8358	0.8704	0.8837	0.9375

Table 5.4: Model Performance for the model trained with real data.

Threshold	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**
TP	140	126	110	76	54
FN	4	18	34	68	90
FP	50	24	12	4	0
Recall	0.9722	0.8750	0.7639	0.5278	0.375
Precision	0.7368	0.8400	0.9016	0.9500	1

and recall vary with different detection thresholds from 0.1 to 0.5. The Average Precision (AP) metric, calculated as the area under these curves, provides an estimation of each model's performance across the range of thresholds. A higher AP, closer to 1, indicates superior model performance. For the simulated model, the calculated AP is 0.83, while for the real model, it is 0.92, underscoring their respective efficacy in defect detection.

The AP values indicate a good performance of both models, with the model trained on real defects slightly outperforming for various thresholds. Despite this slight difference in AP scores, both models exhibit comparable trends in their precision and recall metrics across the range of thresholds evaluated. The simulated data model shows respectable precision values ranging from 0.6596 to 0.9375 as the threshold tightens from 0.1 to 0.5. This indicates its ability to reduce false positives while maintaining reasonably high precision, although at the cost of lower recall (0.4167 at threshold 0.5). In contrast, the real data model consistently maintains high precision values above 0.73 across all thresholds, achieving perfect precision (1.0) at the highest threshold of 0.5. However, its recall decreases more significantly as the threshold increases, dropping to 0.375 at threshold 0.5, suggesting a trade-off between precision and recall.

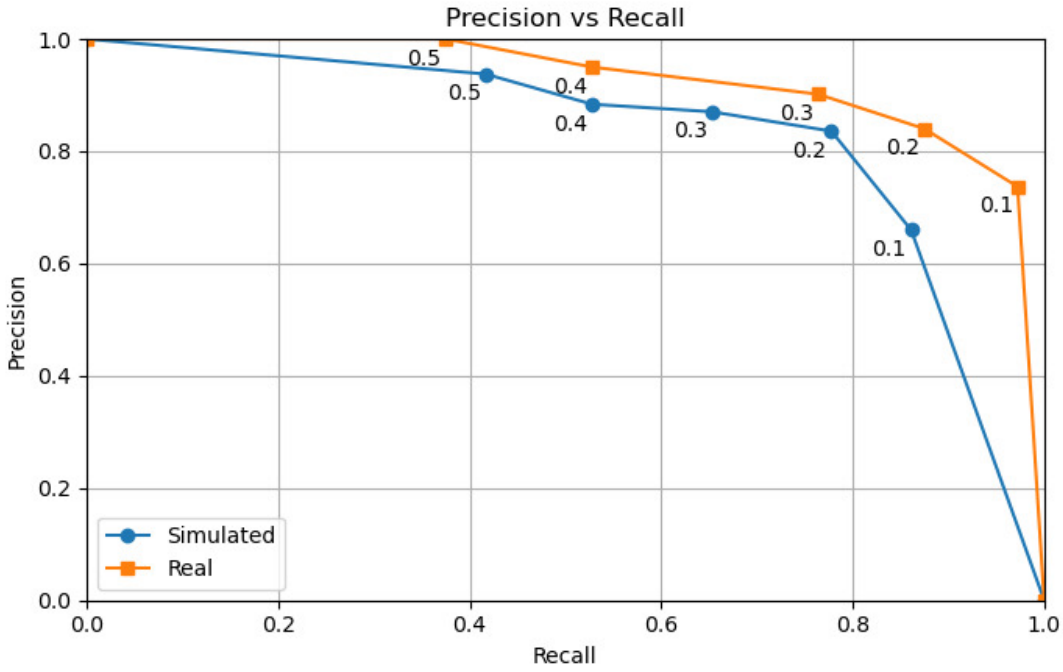


Figure 5.21: Precision-Recall of both models. In blue, the model trained with synthetic data, and in orange, the model trained with real data.

Since the performance of the model trained on simulated defects closely resembles that of the model trained on real defects, it can be concluded that the simulated defects are sufficiently similar to real data to allow for an increase in data quantity when it is scarce or unavailable.

In summary, the results highlight that both models are comparable. Using simulated defects offers an advantage by reducing the extensive time needed for capturing real defects of various types and in numerous locations, potentially spending hours, days, or even months. Furthermore, this method allows for the addition of known types of defects in alternative locations, enhancing its flexibility and applicability.

It's important to mention that the comparison between the two models is mainly focused on evaluating the realism of our synthetic defects. Our aim is to show that these synthetic defect representations can be used similarly to real defects, without any intention of improving defect detection or pursuing additional goals.

5.5 Discussion

This chapter introduces a new approach to address the shortage of 3D data needed for developing surface inspection algorithms with 3D sensors. We describe a method to create surface defects in 3D models using Free-Form Deformation (FFD). This technique is versatile, allowing defects to be inserted anywhere on the model, even in areas with complex geometries.

The process starts by modeling defects as elevation maps from a reference plane, defining their size and shape. By setting the positions and orientations, these defects can be integrated into the CAD model and adjusted using FFD.

Additionally, three common defect types are provided as default options: bumps, peaks, and cracks. These can be customized to change their size, making it possible to include a variety of deformations found in the manufacturing process. This method ensures realistic surface defects, improving the effectiveness and range of 3D surface inspection algorithms. However, the proposed approach also allows for the inclusion of other types of defects beyond these models. By adjusting the control points of the mesh, it is possible to model different deformations, such as those resembling defects obtained from real scans. This capability enables the simulation of defects based on the manufacturer's prior knowledge of potential issues related to their production process or product design.

Although any type of defect can be simulated, mathematical models were proposed for three of the most common types of defects in steel parts: bumps, peaks, and cracks. These models allow for covering a multitude of small or large deformations that appear in the manufacturing of different products. However, this model can be used to simulate other types of defects. For example, a burr can be constructed by creating a crack with negative depth. In metal manufacturing, a burr refers to the formation of rough edges or ridges on the metal piece. By modifying the parameters of the crack model, material loss or other types of holes could be simulated. In the proposed model, both the depth and width gradually increase from 0 at the edges to maximum values at the central parts. This behavior can be altered to grow abruptly, allowing for the simulation of different shapes. Additionally, cracks are constructed as a set of surfaces with Gaussian shape, but other types of shapes can be considered, for instance, the exponential surface used to model peak defects. This would enable the creation of sections or cracks with a less smooth and more abrupt shape.

We consider two possible cases depending on the geometry of the model in the area to be deformed: simple and complex surfaces, to accelerate the process and

reduce computational load. The approach designed for complex surfaces can also be used for simple surfaces. However, handling these flat surfaces can be expedited by reducing the number of control points and simplifying the search for points near the surface. With this strategy, significant improvements in speed were achieved, demonstrating the adaptability and effectiveness of the proposed method.

The experiments show satisfactory results, enabling the simulation of realistic defects with considerable accuracy. However, we did not pursue quantitative comparisons between simulated and real defects because the goal was to simulate defects that resemble real ones, rather than replicate them exactly. The key value of this approach lies in its ability to generate a wide range of defects using mathematical models, eliminating the need for real defect scans for replication. These techniques are crucial for generating data to successfully train AI models.

In our experimental setup, we trained two machine learning models: one using real defects and the other using simulated defects. A comprehensive comparative analysis of their performance was conducted. It's important to note that the same machine learning algorithm was used for both models, focusing not on optimizing the defect detection algorithm but on assessing the performance and utility of synthetic defects for data augmentation. This allowed us to validate the authenticity of synthetic defects in representing real-world defects.

The dataset included 506 real defects compared to 285 synthetic defects. The disparity in numbers stems from the inclusion of all available real defects in the dataset, leading to many similar instances. Conversely, the synthetic dataset was generated to be more balanced, incorporating a variety of defect shapes and positions. Despite being trained on a smaller dataset, the model trained with synthetic defects achieved results comparable to the model trained with real defects in detecting defects.

Given the experimental design, it was anticipated that the model trained with real samples would have a slight advantage. This is because the real samples came from the same car model used to evaluate the results, whereas the defects were simulated only on the car's front door CAD model, without access to the rest of the body. Consequently, the model trained with real defects likely had a better understanding of the defect's environment, including the actual curvature of the car, which should result in fewer false positives due to the car's shape.

These experiments demonstrate that a model trained solely with simulated defects can effectively detect defects in real parts under factory scanning conditions. This success suggests that defect simulations and the profilometric sensor are accurate enough to be used alongside or in place of real defect samples when such samples

are limited.

It is crucial to highlight that the primary goal of this study is not to enhance specific defect detection algorithms but to provide a tool for generating a dataset of sufficiently realistic synthetic defects. This dataset can be integrated into machine learning algorithms designed for defect detection. While we used the YOLO v8 network for this study, the specific choice of this network is secondary. What matters is the consistency of the model used in both cases, allowing for a meaningful comparison. The decision to use YOLO v8 was based on practical experience with this network architecture, and no extensive exploration of different models was conducted.

Chapter 6

Reinforcement Learning-Based Inspection Path Planning

In this chapter, a new strategy is introduced for generating inspection paths for profilometric sensors employing Reinforcement Learning (RL) techniques.

The methodology focuses on optimizing the sensor's position and orientation at each point along the scanning path, using the CAD model of the object as a reference. This allows the system to dynamically adjust the sensor to maintain optimal orientation and a consistent, appropriate distance throughout the scanning process. This improves the capture of high-quality data and reduces noise in measurements. RL techniques facilitate the system's learning and adaptation to the specific geometric characteristics of each object, thereby enhancing inspection efficiency and accuracy.

The main contribution of this section lies in the design of the RL system. The state space, actions, and reward function are carefully defined to guide the system in generating scanning paths. The state space encompasses the possible sensor configurations relative to the object, while actions refer to the movements the system can make to adjust sensor position and orientation. The reward function is designed to encourage optimal behaviors in terms of distance, orientation, and sensor advancement relative to the object.

To achieve this, an RL model is initially trained using a specially designed piece. Once trained, this model is applied to different objects to generate corresponding scanning paths. Trajectories are generated offline, allowing for careful planning and optimization before they are executed in real-world inspection scenarios. Accurately

calibrating the initial scan position is crucial to align the generated path with the physical reality of the inspected object.

This chapter presents the results obtained from applying the proposed strategy for generating scanning paths. The method was evaluated to determine its accuracy and efficiency in creating optimal scanning trajectories. Experiments were conducted using CAD-based simulations and physical tests with real hardware. A range of objects with different geometric shapes and complexities were used to validate the approach, demonstrating the strategy's effectiveness and adaptability across various scenarios.

6.1 Related work

To achieve comprehensive surface scanning of the inspected piece, a relative movement between the piece and the sensor is necessary. Robotic systems, including robotic arms [24], [25], unmanned aerial vehicles (UAVs) [26], unmanned ground vehicles (UGVs) [27], [28], and submarines [29], have increasingly integrated into inspection procedures across various applications to meet this requirement. These systems enable precise and controlled movement between the inspected part and the sensor, facilitating complete surface coverage and efficient inspection processes.

Effective and precise inspection demands meticulous planning of sensor trajectories over the piece's surface. While manual planning suffices for simpler scenarios, more complex geometries or stringent precision standards require automated methods. Generating inspection trajectories for robotic systems presents a significant challenge, necessitating predefined paths that consider surface geometry, defect characteristics, and inspection requirements.

While various studies on automated inspection trajectory planning exist in the literature, particularly focusing on robotic arms, there remains a significant research gap addressing the specific integration of robotics and profilometric sensors for surface inspection tasks.

Chen et al. highlight this gap in their study [96], where they propose an approach to automatically detect surface defects on free-form 3D objects using a 6-degree-of-freedom manipulator with a line scanner and depth sensor. Their method involves defining local paths for precise defect inspection and optimizing global time for efficient scanning. Li et al. [30] propose a method for planning scan paths in automated surface inspection. Their approach employs a path planning algorithm using a tri-

angular mesh model. They divide the surface area of the workpiece into regions, determine scan orientations and points in each region, and then generate scan paths using the minimum bounding rectangle. This method entails developing a section division algorithm to determine scan orientation, followed by generating paths that meet system constraints.

Recently, a new trend has emerged in robotics for trajectory generation using reinforcement learning (RL) methods. These methods have gained popularity due to their ability to learn complex and adaptive behaviors from interactions with the environment.

In robotics, RL has been successfully applied in a variety of applications. For instance, Lobbezoo et al. compile different strategies in [97] from the literature using RL algorithms for pick and place applications. On the other hand, Elguea-Aguinaco et al. provide a comprehensive analysis in [98] of current research on RL usage in tasks involving intensive physical interactions. These tasks refer to activities where object manipulation involves direct and significant contact between the robot and its environment. This study covers research across various domains, including manipulation tasks with rigid objects (e.g., assembly, disassembly, polishing, and grinding) and deformable objects (e.g., folding fabrics, tensioning and cutting, or object manipulation).

Han et al. present an exhaustive research in [99] of different applications of Deep RL in robotic manipulators, highlighting key challenges faced by RL in such applications. One of the major issues is that models often fail to perfectly replicate the real system. For instance, in robots guided by computer vision, simulated RGB images may significantly differ from real images captured by cameras, a well-known issue termed "Sim-to-Real." This discrepancy arises because simplified models do not fully capture the system dynamics, making it challenging to transfer trained models from simulation to real environments.

In the simulation of high-precision laser triangulation profilometric sensors specifically, lack of realism can stem from various factors. These include the surface roughness of scanned materials, which introduces additional complexities in laser reflection, and generated noise such as speckle effect, a granular interference that arises when the laser interacts with textured surfaces. These elements can significantly distort measurements if not accurately replicated in simulation.

To address this issue, we will utilize the realistic simulator presented in Chapter 4, which allows us to faithfully represent measurements obtained by the laser triangulation sensor. This simulator introduces measurement noise and speckle noise

from material roughness, ensuring simulations closely approximate real operational conditions.

Another highlighted issue is that trajectory generation using robotic arms is inherently multidimensional, further complicating the learning and optimization process. Ji et al. also emphasize this problem in [100], pointing out that in robotics, most RL-related works focus on mobile robot navigation due to its well-developed and straightforward theory [101].

Surface inspection using profilometric sensors typically involves straight-line scanning. If the part is too large for a single scan, inspection is performed through parallel passes, often utilizing boustrophedon trajectories [102]. Section 6.2 will delve deeper into the specific characteristics of this type of inspection. During each pass, the sensor advances in a designated direction while adjusting its height and pitch angle over the part, keeping other orientations constant. This feature simplifies the action space to just three dimensions: scanning direction position, height, and pitch orientation of the sensor. Focusing on these three parameters related to the robot's end effector significantly reduces problem complexity, as individual robot axes do not need separate consideration. This inherent feature in inspection applications facilitates sensor trajectory control and planning during the inspection process.

This approach aligns well with reinforcement learning (RL) techniques, which can effectively adapt to problems with a limited action space. RL's ability to learn from environment interactions and enhance control policies based on received rewards makes it a promising tool for addressing challenges in surface inspection. These inspection process characteristics, coupled with recent advancements in RL algorithms, create new opportunities for applying RL-based strategies in this relatively underexplored field.

While various studies have explored RL's potential in diverse robotics applications, few have specifically focused on its application in inspection tasks. This research gap underscores the need for further exploration and investigation in this area.

In the realm of inspection applications, notable research is presented by Xiangshuai Zeng in [103]. This work introduces PIRATE (*Pipe Inspection Robot for Autonomous Exploration*), a robot designed for inspecting the interior of pipes using Reinforcement Learning techniques. PIRATE is equipped with multiple flexible joints and wheels, enabling adaptive and efficient mobility. Reinforcement learning is employed for navigating the robot through pipe interiors, learning to maneuver around sharp corners using algorithms like PPO and deep neural networks. This

approach allows the robot to adjust its behavior in real-time, overcoming obstacles and adapting to various pipe configurations. The system defines specific actions, rewards, and observations. Actions include wheel movements and adjustments in the flexible joints. Rewards are given for collision-free navigation and progress, while collisions are penalized. Observations come from a 2D laser scanner providing distance information about the pipe environment, enabling the robot to perceive and adapt in real-time.

Another focused inspection work is presented by Jing et al. in [104], centered on automatic generation of robotic trajectories for surface inspection in production lines. They use techniques such as Monte Carlo algorithms and greedy search approaches for Coverage Path Planning (CPP). The proposed methodology enables automatic generation of inspection trajectories tailored to objects of different sizes and geometries.

They utilize a structured light 3D scanner to generate a three-dimensional point cloud representing the scanned object's geometry. The primary goal is to minimize the total cycle time by combining View Planning Problem (VPP) and trajectory planning to minimize the total cost of inspection and travel after fulfilling surface coverage requirements. View planning involves determining optimal poses from which to capture images or make measurements of an object or surface.

The trajectory generation process includes: random selection of viewpoints, calculation of robot movements, collision-free path planning, evaluation of visibility for each viewpoint and covered surface area, and application of an RL-based planning algorithm to select inspection actions until completion. Actions include viewpoint selection and robot movement to position the 3D scanner at the selected viewpoint. The proposed RL algorithm automatically generates the online inspection policy, taking as input the robot model, target object model, and sensor specifications.

Continuing on the previous research line, Christian Landgraf, Bernd Meese et al. present in [105] a new solution to address the challenge of automatic view planning in the context of surface inspection. Similar to earlier work, this study proposes a strategy to find optimal sets of viewpoints for the three-dimensional inspection of specific workpieces. Their goal is to automate this process for any industrial robotic arm available in ROS and any 3D sensor specification; in their application, they use a stereo camera.

They employ more advanced reinforcement learning algorithms such as Q-learning, Proximal Policy Optimization (PPO), and Deep Q-Networks (DQN). In these algorithms, each action involves selecting a viewpoint and planning and executing the

robot's movement to this pose. Once the robot reaches its goal, the sensor generates a 3D point cloud at this specific pose. The environment state is constructed using observations obtained from the 3D measurements and the current pose of the robot.

The research by Jing et al. and Christian Landgraf, Bernd Meese et al. focuses on the automatic generation of inspection trajectories using 3D sensors such as structured light 3D scanners and stereo cameras. These sensors provide a detailed representation of the surface but may not be optimal for all applications, especially those requiring high-resolution measurements or in environments with specific constraints.

For case studies like the one in this thesis, which employs laser triangulation profilometric sensors performing measurements along a line, traditional trajectory planning approaches such as the mentioned View Planning Problem (VPP) are not applicable. VPP is designed to find optimal poses from which to capture images or make measurements of an object or surface using wide-area 3D vision sensors. This methodology does not fit the operational characteristics of profilometric sensors, which focus on obtaining precise measurements along a continuous and specific path on the object's surface. Therefore, a trajectory planning approach adapted specifically to the capabilities and operational modalities of these sensors is required, optimizing linear exploration along the surface instead of seeking complete three-dimensional coverage.

To date, no research has been found related to the generation of inspection trajectories using Reinforcement Learning and profilometric sensors. This gap in the literature highlights an unexplored opportunity and a promising area for research in robotics and automation.

This research aims to address this gap by presenting a Reinforcement Learning-based strategy for generating inspection trajectories using profilometric sensors. The main contributions focus on how the state space, action space, and reward function are modeled for each instance. Subsequently, the Proximal Policy Optimization (PPO) algorithm is employed for agent training.

PPO is an algorithm introduced by OpenAI in [41]. The authors emphasize its ability to balance three key aspects in reinforcement learning: ease of implementation, sampling efficiency, and simplicity in hyperparameter tuning. They underscore that PPO not only offers competitive or superior performance compared to more advanced reinforcement learning algorithms but also stands out for its straightforward implementation and parameter adjustment.

The effectiveness of PPO has been demonstrated across a wide range of applications, including robot control such as the PIRATE robot [103] and the development of view planning systems for inspection tasks as presented in the work by Landgraf, Meese et al. [105]. Furthermore, PPO has been successfully applied in pick and place tasks, such as training 7-degree-of-freedom robotic arms for precise object manipulation, as described in [106], and in other pick and place applications discussed in [107].

Comparisons between PPO and other reinforcement learning algorithms like SAC and TD3 reveal interesting patterns in terms of training efficiency, performance, and convergence. For instance, [108] found that PPO tends to perform better in smaller state spaces, while SAC shows advantages in larger state spaces. On the other hand, [107] compared PPO and SAC, where SAC proved more efficient in sampling, but PPO exhibited greater insensitivity to hyperparameters and more stable convergence in complex problems. These findings support the choice of PPO as the primary algorithm for the proposed research.

6.2 Characteristics of surface inspection scanning with profilometric sensors

During the scanning process using laser triangulation profilometric sensors, the quality of the measured data is directly affected by various parameters associated with the relative position between the sensor and the inspected piece, as detailed in [30]. These parameters are critical to ensure accurate and comprehensive surface inspection. Therefore, it is essential to carefully consider these factors during the planning of scanning trajectories to achieve effective results in surface inspection. These parameters were previously explained in Chapter 2, but here, further detail is provided on how they should be applied to generate an effective inspection trajectory.

In Figure 6.1, two views of a profilometric sensor are presented: a 3D representation with its coordinate system and a front view highlighting two key parameters: the optimal working distance W_d and the depth of field Z_r. In addition to W_d and Z_r, other important parameters include the angle of incidence and the distance between consecutive profiles, which also significantly affect the accuracy and quality of the measured data.

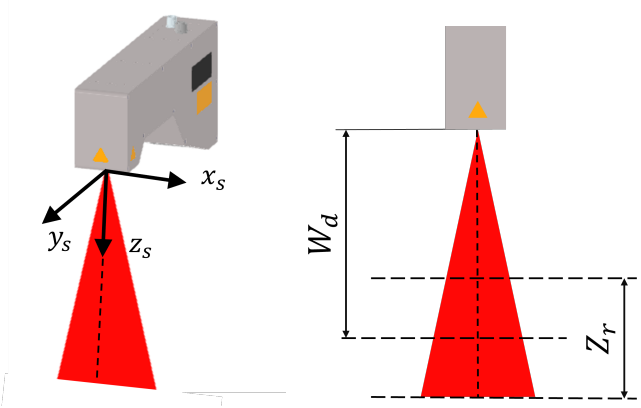


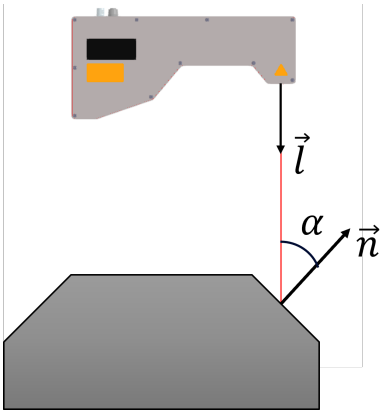
Figure 6.1: Representation of the profilometric sensor and its main parameters. (a) 3D representation of the sensor with its coordinate system. (b) Front view of the sensor. The optimal working distance W_d and depth of field Z_r are depicted.

Here are the parameters mentioned earlier, detailed more in depth:

- **Incidence angle** (α): Refers to the angle between the orientation of the sensor $\overrightarrow{l}$ and the normal vector of the surface of the workpiece $\overrightarrow{n}$, see Figure 6.2. It is calculated according to Equation 6.1. This parameter is critical for determining the accuracy with which the surface is captured. As the incidence angle increases, there is a higher probability of introducing noise into the capture. This phenomenon occurs because the scanner can capture unwanted reflections of laser light and variations in the reflectivity of the surface, which negatively impacts the quality of the obtained data. Research studies such as those conducted in [109] and [31] have experimentally demonstrated how noise in the capture is directly related to the incidence angle in different types of workpieces.

$$\alpha = acos(-\overrightarrow{l} \cdot \overrightarrow{n}) \tag{6.1}$$

  Therefore, it is essential to carefully select the appropriate sensor orientation such that the incidence angle is minimized, in order to minimize the introduction of noise and ensure accurate and reliable capture of the workpiece surface.



(a)

Figure 6.2: Incidence angle (*alpha*): angle between the orientation of the sensor $\overrightarrow{l}$ and the normal vector of the surface of the workpiece $\overrightarrow{n}$

- **Optimal Working Distance (W_d)**: Refers to the optimal distance between the sensor and the surface of the object being measured. This is the distance from the laser source to the scanning reference plane located halfway within the depth of field.

  Maintaining the scanner at this optimal working distance ensures that the captured data is as accurate as possible.

- **Depth of Field (DOF)**: Also known as Z-Range (Z_r). Refers to the range of distance within which the scanner can capture surface data during a single scan, see Figure 6.1 (b). Assuming a point in the scanner's coordinate system is $(x_s, 0, z_s)$, Equation 6.2 must be satisfied.

$$W_d - \frac{Z_r}{2} \leq z_s \leq W_d + \frac{Z_r}{2} \tag{6.2}$$

  In [31], analyses have also been conducted on the noise introduced based on the distance between the sensor and the workpiece, demonstrating that the noise is minimal when working at the optimal working distance and increases as one moves away from it. However, if this is not possible, it is crucial to ensure that the sensor remains within the range of the depth of field, as outside this range the sensor is unable to make accurate measurements.

- **Distance between profiles (Δs)**: The point density between profiles of consecutive scans plays a crucial role in the coverage and completeness of the inspected surface. An adequate point density ensures that important details

are not missed during the scanning process and that an accurate representation of the surface is obtained. This is particularly important in areas with small features or irregular surfaces, where low point density can result in incomplete or inaccurate inspection. See Figure 6.3.

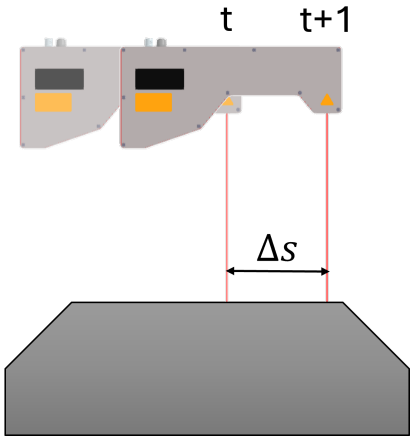


Figure 6.3: Distance between consecutive profiles Δs.

In addition to considering those parameters, it is crucial to choose the appropriate scanning trajectory to achieve comprehensive surface scanning of the part. In laser profilometer inspection applications, one of the most common strategies is to employ a Boustrophedon trajectory [110], [102].

In industry, the boustrophedon method is widely used for efficient surface inspections without the need to define specific trajectories for each part. This approach allows systematic and repetitive scanning of large surfaces, ensuring complete and uniform coverage of the surface to be inspected. In a boustrophedon scan, the sensor initially moves straight along an axis until reaching the edge of the surface to be inspected. Then, it moves laterally by a predetermined distance and changes direction to move back along the initial axis. This pattern of movements is repeated by alternating directions until the entire surface has been covered.

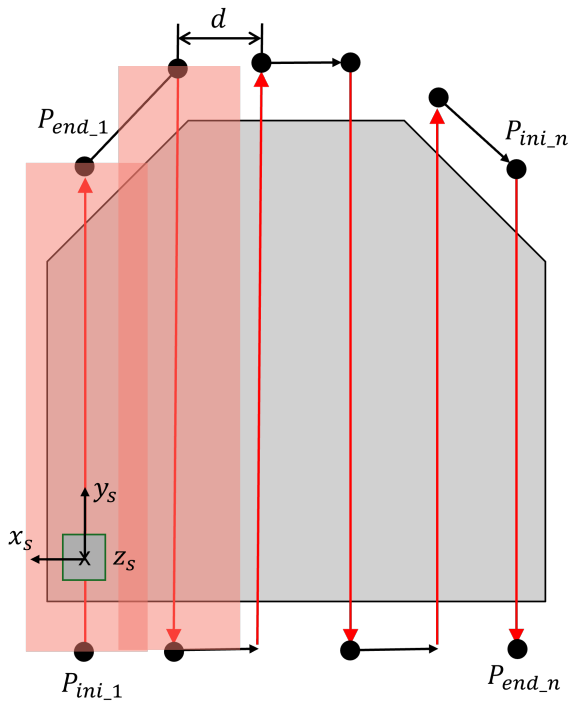


Figure 6.4: Top view of the scanning trajectory, where multiple parallel passes are made, each separated by a distance d. An overlapping area is defined between each pass. The gray square represents the sensor. The red lines show the trajectories where the sensor captures data. The black lines indicate the intermediate movements to position the sensor for the next pass.

Considering these types of trajectories, the profilometric sensor collects data only during the parallel passes along the surface of the piece. In Figure 6.4, these trajectories are shown in red, from the initial point of a pass (P_{ini_i}) to its end point (P_{en_i}), where i denotes the number of parallel passes. The movement of the robot between each pass is shown in black. The distance between passes, d, is carefully adjusted to ensure that the scans overlap adequately, thereby completely covering the piece.

6.3 Proposed Method

In this section, we present the proposed approach for generating inspection trajectories using profilometric sensors and Reinforcement Learning (RL) techniques. Our objective is to improve the inspection trajectories for a workpiece, which are typi-

cally scanned following a straight line between two points or, in the case of larger pieces, through a boustrophedon path.

Each pass along the scanning path is carefully planned to keep the sensor at its optimal orientation and distance from the part at every point. This involves dynamically adjusting the position and tilt (pitch) of the sensor to ensure a consistent pose between the sensor and the surface at all times. The other two sensor orientations will be fixed, allowing for accurate and uniform data capture. In addition, the profile spacing will be taken into account to ensure full scanning coverage.

To train the RL algorithms, a simulated environment replicating the conditions of the real system described in Chapter 4 is used. This simulator emulates the measurements of a laser triangulation profilometric sensor. This setup provides a realistic and controlled training environment.

The state space is constructed using the position and orientation of the robot's end effector. This allows for a generalization of the approach and facilitates the method's transferability to different robotic configurations. Additionally, the state includes other parameters such as the average distance of the profile, the incidence angle, and the spacing between consecutive scans.

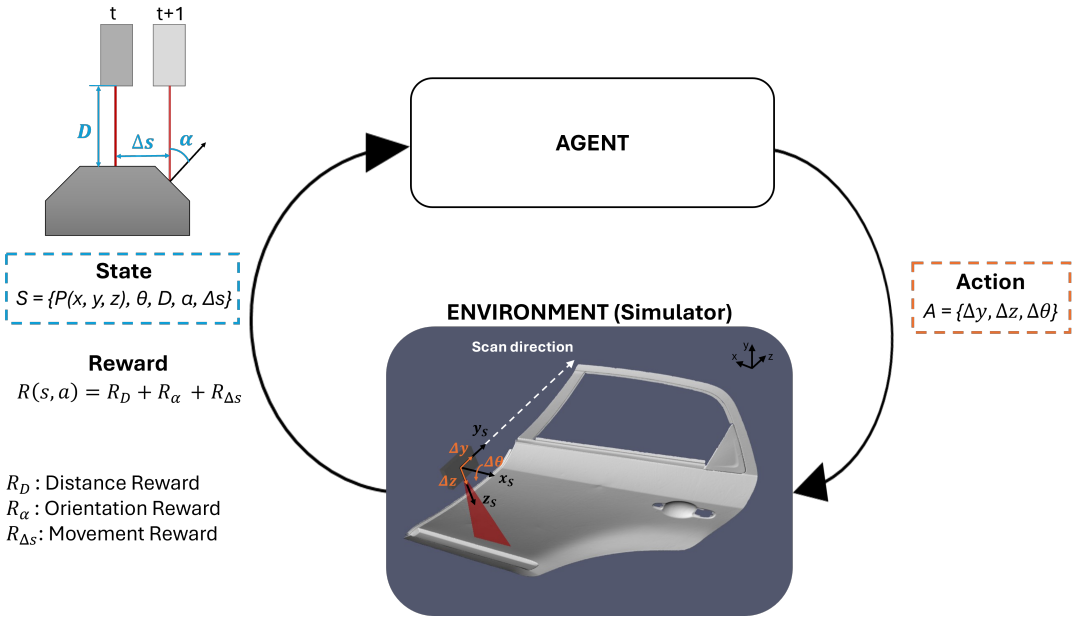


Figure 6.5: Scheme of the process.

The action space is defined by relative increments in the sensor's position and tilt angle, enabling precise adjustments and smooth movements of the sensor. The reward function comprises three key components: the distance between the sensor and the surface, the sensor's alignment with the surface normal, and the spacing between consecutive scans. This comprehensive reward function incentivizes optimal behaviors in terms of the sensor's distance, orientation, and advancement.

Next, we will detail each component of the proposed method, providing an in-depth understanding of its design and implementation. A scheme of the process can be seen in Figure 6.5

6.3.1 Simulated Environment

To effectively train reinforcement learning algorithms, it is crucial to have an environment that closely simulates real-world conditions. Testing directly on the actual system can be costly, hazardous, or impractical in many cases. Hence, simulators are used to create a virtual replica of the environment.

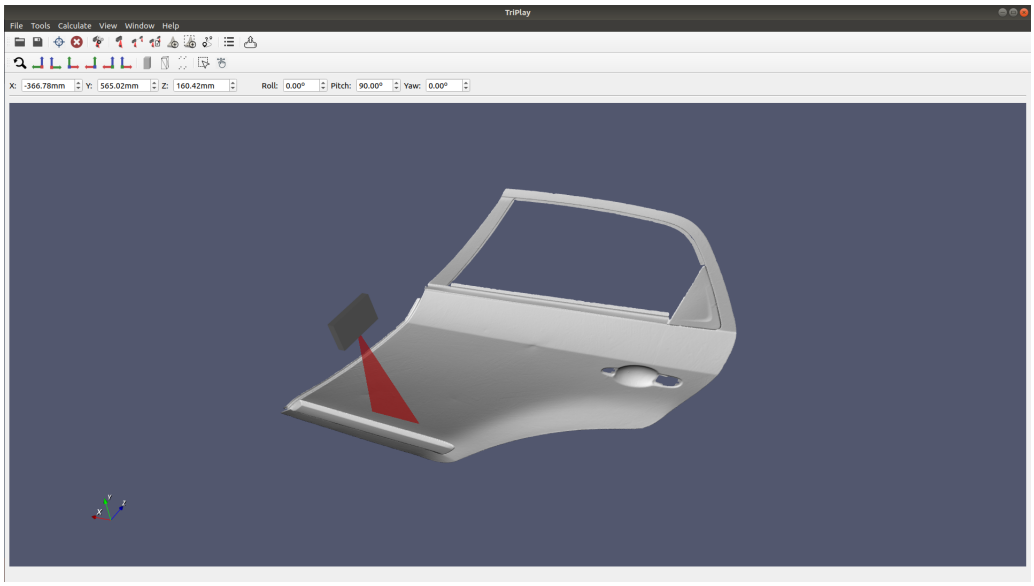


Figure 6.6: View of the simulated environment with the profilometric sensor and a part to be inspected.

In this work, we use a simulator detailed in the previous chapter 4 designed to accurately mimic the conditions of the real system within a virtual setting. This

simulator replicates the measurements of a laser triangulation profilometric sensor and can emulate the parameters of any commercial sensor based on its specification sheet. It allows for the precise measurement of a CAD model of the part to be inspected, including the simulation of inherent sensor noise and speckle noise caused by the object's surface characteristics. Figure 6.6 shows the simulated environment with the profilometric sensor and a CAD model of a part.

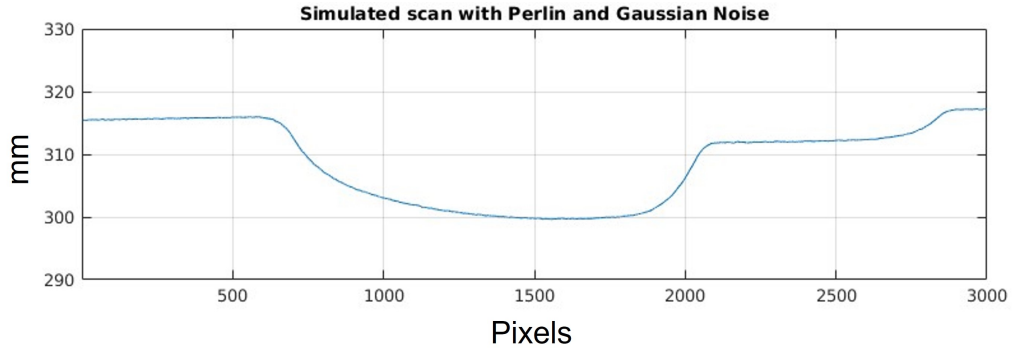


Figure 6.7: Profile obtained during the simulation of a scan of the car door handle section of the CAD model shown in Figure 6.6.

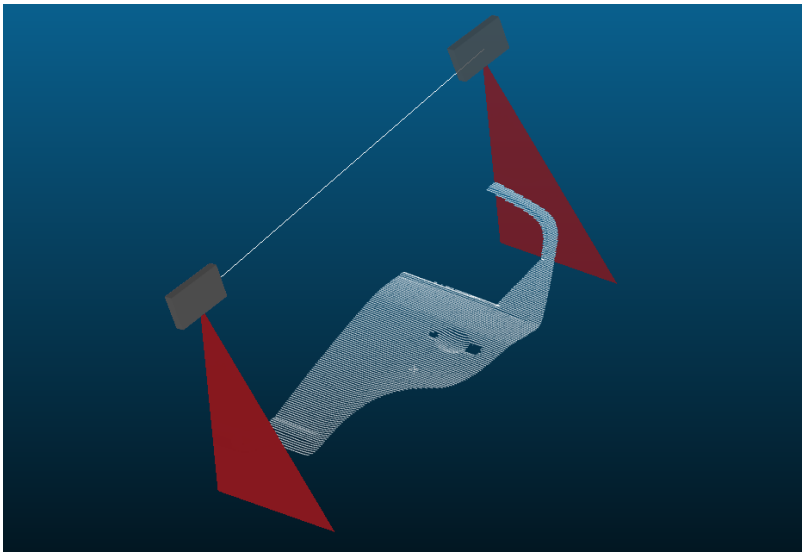


Figure 6.8: Point cloud result obtained during the simulation of a scan of the CAD model section shown in Figure 6.6. The start and end points of the trajectory can be seen.

In each iteration of the simulator, several critical parameters are obtained that will be used later by the RL algorithm. First, the distance profile is captured, a fundamental representation provided by any profilometric sensor, see Figure 6.7. Additionally, the 3D position of the scanned points of the CAD model is collected, providing detailed information about the surface geometry of the object, see Figure 6.8. Furthermore, the simulator also provides data on the normals at those points on the object's surface.

6.3.2 State Space

As previously mentioned the position and orientation of the end-effector are used instead of relying on the positions and velocities of the robot's joints. This choice simplifies the state space and facilitates the transfer of the method to different robotic configurations without the need for specific adjustments in the joints.

Mathematically, the state S is defined as a tuple as follows:

$$S = \{P(x, y, z), \theta, D, \alpha, \Delta s\} \tag{6.3}$$

Here, $P(x, y, z)$ represents the position of the end-effector, while θ denotes its tilt. The parameters D, α, and Δs correspond to the mean profile distance obtained from the scan, the incidence angle, and the advance between consecutive scans in the 3D space, respectively. These values represent the average of the points in each profile.

6.3.3 Action Space

The action space is defined by the increments in the position and tilt angle of the inspection sensor. These increments are defined relative to the sensor's own coordinate system. Mathematically, the action space is represented by equation 6.4.

$$A = \{\Delta y, \Delta z, \Delta \theta\} \tag{6.4}$$

Where Δy represents the increment in position along the predefined scanning direction. Δz refers to the increment in position along the global Z-axis, controlling

the height of the end-effector relative to the part. $\Delta\theta$ denotes the change in the sensor's pitch orientation, which involves rotation around the global X-axis. While the scanning direction initially aligns with the Y-axis, changes in tilt ($\Delta\theta$) will adjust this alignment dynamically. This is illustrated in Figure 6.9.

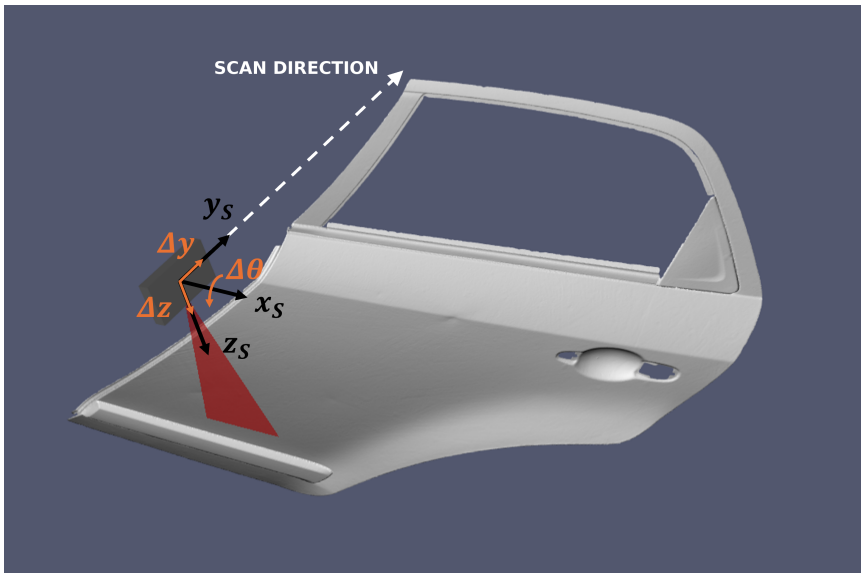


Figure 6.9: The figure shows the simulation environment and represents the action space as unit vectors (in orange). Δy refers to the increment in position in the scanning direction, Δz refers to the increment in the vertical direction (Z), and $\Delta\theta$ denotes the change in the sensor's pitch orientation.

The action space is defined as continuous, meaning that actions span a continuous range of values rather than discrete ones. This approach ensures smooth and controlled sensor movements to avoid abrupt changes that could affect measurement accuracy or cause collisions with the workpiece. Equation 6.5 establishes the limits for each type of action in the continuous space. Here, Δy, Δz, and $\Delta\theta$ are constrained to values between $\pm\Delta y_{\max}$ millimeters, $\pm\Delta z_{\max}$ millimeters, and $\pm\Delta\theta_{\max}$ degrees, respectively.

$$\begin{aligned} \Delta y &\in [-\Delta y_{max}, \Delta y_{max}] \\ \Delta z &\in [-\Delta z_{max}, \Delta z_{max}] \\ \Delta\theta &\in [-\Delta\theta_{max}, \Delta\theta_{max}] \end{aligned} \tag{6.5}$$

Dynamic Action Limitation

To ensure smooth and safe movement of the inspection sensor, the selection of actions is dynamically adjusted based on the environment's observations. This action limitation accelerates the convergence of the reinforcement learning algorithm, enabling the system to learn more efficiently and effectively.

When the sensor is farther from the part surface than the optimal working distance W_d, limits are applied to the sensor's displacement in the Z direction Δz to bring it closer to the surface in a controlled manner. Conversely, if the sensor is too close, displacements in the negative direction are limited, as per equation 6.6.

$$\Delta z = \begin{cases} \text{clip}(\Delta z, 0, \Delta z_{max}) & \text{if } (D - W_d) \geq 0 \\ \text{clip}(\Delta z, -\Delta z_{max}, 0) & \text{if } (D - W_d) < 0 \end{cases} \tag{6.6}$$

Here, $clip(x, a, b)$ limits the value of x between a and b, ensuring that the actions are within the permitted range, according to equation 6.7.

$$\text{clip}(x, a, b) = \begin{cases} a \text{ if } x \leq a \\ b \text{ if } x \geq b \\ x \text{ else} \end{cases} \tag{6.7}$$

Similarly, if the sensor's incidence angle (α) with respect to the surface normal is positive, indicating excessive tilt, limits are applied to the angular displacement $\Delta\theta$ to correct the sensor's orientation. Conversely, if the tilt angle is negative, limits are applied to the angular displacement in the opposite direction to keep the sensor properly aligned with the inspected surface. This is represented in equation 6.8.

$$\Delta\theta = \begin{cases} \text{clip}(\Delta\theta, 0, \Delta\theta_{max}) & \text{if } \alpha \geq 0 \\ \text{clip}(\Delta\theta, -\Delta\theta_{max}, 0) & \text{if } \alpha < 0 \end{cases} \tag{6.8}$$

6.3.4 Reward Function

In reinforcement learning, creating an effective reward model is crucial as it guides the agent toward desirable behaviors within the environment. This model assigns a

value to each state-action pair, reflecting the immediate benefit or cost associated with the agent's decision. This section details the reward strategy designed in this research.

The proposed reward function $R(s, a)$ consists of three distinct components, each capturing different aspects of the inspection process. Mathematically, this function is expressed as shown in equation 6.9.

$$R(s, a) = w_d R_D + w_\alpha R_\alpha + w_{\Delta s} R_{\Delta s} \tag{6.9}$$

R_D represents the reward related to the distance between the sensor and the inspected surface, R_α denotes the reward related to the alignment of the sensor's orientation with the normal of the inspected object's surface, and $R_{\Delta s}$ captures the reward associated with the sensor's movement between consecutive scans in the 3D space corresponding to the point cloud of the inspected piece. $w_d, w_\alpha, w_{\Delta s}$ represent the weights that each component contributes to the overall reward function.

Additionally, to ensure the feasibility of the proposed actions, the inverse kinematics of the robotic manipulator are considered. After calculating a new target position for the sensor, we evaluate whether the robot can reach this position within its kinematic constraints. If the robot is capable of reaching the position, no additional reward or penalty is applied. However, if the robot cannot achieve the specified position due to its kinematic limitations, a negative reward is introduced to discourage the agent from choosing actions that are infeasible in practice.

The proposed rewards are in the range [0, -1], as the reward function aims to incentivize the agent to perform actions that improve the inspection process. The maximum value of 0 is assigned when the optimal goal is reached, while negative values indicate penalties for deviations from the desired behavior.

Distance Reward (R_D)

To ensure that the sensor maintains an optimal distance from the inspected surface, a distance reward function R_D is defined as a continuous penalty function that decreases as the absolute difference between the observed distance and the optimal working distance W_d increases. The reward function is formulated as follows:

$$R_D = -\frac{(W_d - D)^2}{(\frac{Z_r}{2})^2} \tag{6.10}$$

Where W_d represents the optimal working distance, D the observed distance during scanning, and Z_r the specified working range of the sensor. This results in a parabolic function with values between [-1,0], corresponding to 0 when operating at the optimal working distance and -1 at the sensor's range limits, as shown in Figure 6.10. If the distance is outside this range, the penalty is maximum (-1).

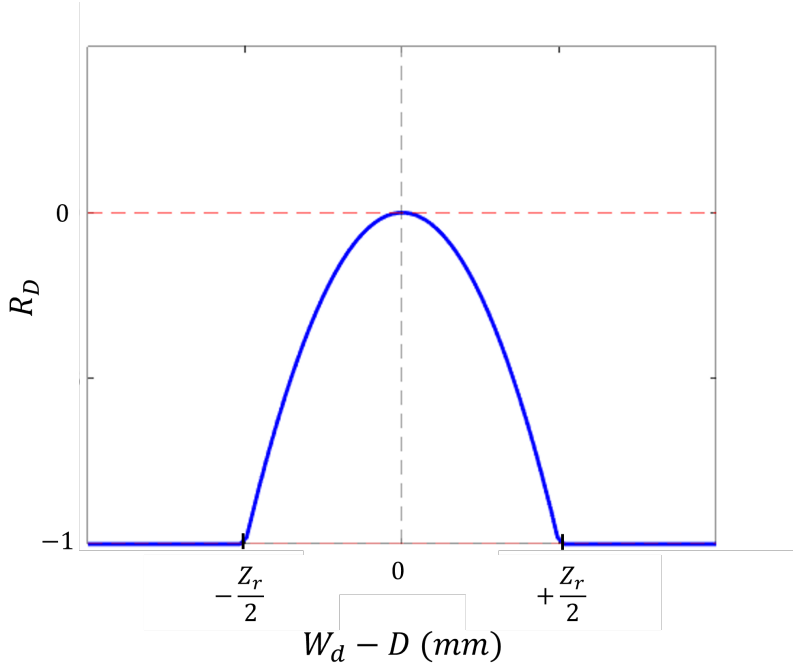


Figure 6.10: Graph of the Distance Reward Function R_D. The graph shows the relationship between the observed distance D and the reward R_D based on the difference from the optimal working distance W_d.

Orientation Reward (R_α)

To induce the agent to align its orientation with the surface normal, we introduce an orientation reward model (R_{alpha}). This model is designed to minimize the angular disparity between the sensor direction and the surface normal vector. The function is defined as a continuous penalty function that approaches 0 as the absolute orientation difference decreases, see Figure 6.11:

$$R_\alpha = \max(-1, -\frac{\alpha^2}{\alpha_{max}^2}) \tag{6.11}$$

Where α is the angular difference between the sensor's orientation and the surface normal, and α_{max} is the maximum allowed angular disparity threshold. This model encourages the agent to maintain close alignment with the surface normal, optimizing the quality of the inspection.

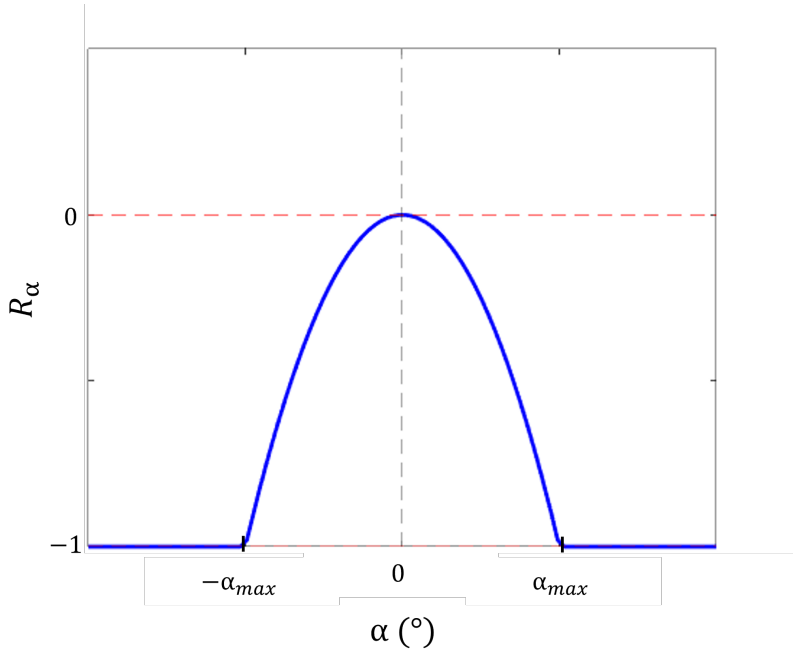


Figure 6.11: Graph of the Orientation Reward Function R_α. The graph shows the relationship between the incidence angleα and the reward R_α according to a maximum incidence angle α_{max}.

Movement Reward ($R_{\Delta s}$)

In addition to optimizing the distance and orientation of the sensor, ensuring smooth forward movement is crucial for comprehensive surface coverage. Forward scanning movement ensures that each scanned 3D profile extends beyond the previous one, facilitating thorough inspection. The reward function $R_{\Delta s}$ is expressed as:

$$R_{\Delta s} = \max(-1, -\frac{(\Delta s - \Delta s_{opt})^2}{\Delta s_{opt}^2}) \tag{6.12}$$

This function penalizes the agent when the scanning spacing Δs is negative, indicating backward movement within the inspection area. Likewise, it behaves parabolically with respect to the scanning spacing Δs. When the spacing is equal to twice the optimal value Δs_{opt}, the reward reaches its minimum value of -1. This indicates a strong penalty for excessively large spacings. As the spacing decreases from this point, the reward gradually increases, reaching a maximum value of 0 when the spacing is exactly equal to the optimal value. Therefore, the reward function motivates the agent to maintain spacing close to the optimal, as both above and below-optimal values result in a decrease in reward, see Figure 6.12.

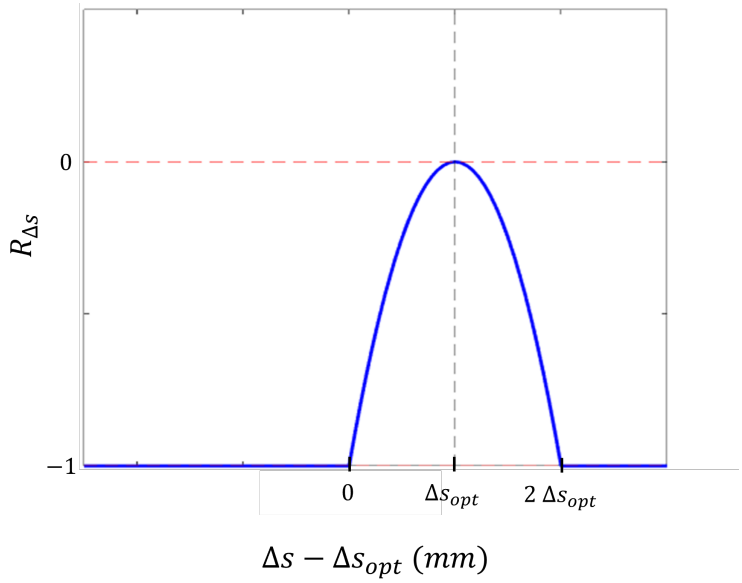


Figure 6.12: Graph of the Movement Reward Function $R_{\Delta s}$. The graph shows the relationship between the scanning spacing between the current and previous profile Δs and the reward $R_{\Delta s}$ according to an optimal spacing Δs_{opt}.

6.3.5 RL Algorithm Configuration

In this study, the Proximal Policy Optimization (PPO) algorithm was employed to train models capable of generating effective inspection trajectories using profilometric sensors. The choice of PPO is driven by its robust performance and ability to handle

continuous action spaces. Specific hyperparameter configurations were fine-tuned to ensure optimal learning performance and convergence.

- **Network Architecture:** This parameter specifies the structure of the neural network used in the RL algorithm, including the number of hidden layers and the number of units in each layer.
- **Activation Function:** It determines the activation function used in the hidden layers of the neural network. The ReLU (Rectified Linear Unit) activation function is commonly employed due to its ability to introduce non-linearities into the model.
- **Learning Rate:** Indicates the step size taken in the opposite direction of the gradient during the update of the neural network weights. This learning rate influences the speed and stability of the learning process.
- **Update Rate:** Represents the frequency with which the model parameters are updated during training. In some algorithms, such as PPO, this rate is measured in terms of steps, while in others, like SAC and TD3, it is measured in complete episodes.
- **Batch Size:** Specifies the number of experience samples used in each update of the neural network. This parameter affects the training efficiency and the stability of the model's convergence.
- **Discount Factor (γ):** Determines the relative importance of future rewards compared to immediate rewards. A higher gamma value places more weight on future rewards, which can influence the model's ability to learn long-term strategies.
- **Clip Ratio:** Limits the magnitude of policy changes between updates to ensure stability in training. This parameter is particularly relevant in algorithms like PPO, where policy-based optimization techniques are used.
- **Epoch:** Indicates the number of times the entire dataset is processed during training. Each epoch corresponds to one complete pass through the training dataset.

6.4 Experiments and results

This section details the experiments conducted to evaluate a new method for automated inspection using profilometric sensors. The primary aim is to assess whether the Reinforcement Learning (RL) approach can generate effective scanning paths that enhance accuracy and coverage for detecting defects across different objects.

The Reinforcement Learning algorithms were implemented using the open-source library stable-baselines3 [111], which provides enhanced implementations of RL algorithms based on OpenAI's frameworks. To analyze and process the results, we used MATLAB 2023b.

Three distinct objects were used in the tests: a car door, a drone body, and a pen holder. See figure 6.13. Each object was selected for its unique shape and the distinct challenges it presents. The car door is mostly flat with some curvature, the drone body features complex contours and tight areas, and the pen holder is small with intricate details.

(a) (b) (c)

Figure 6.13: Parts used for the experiments. (a) Car door. (b) Parrot drone. (c) Pen holder

The experiments compared scanning results from paths generated by RL-optimized trajectories with conventional methods such as straight-line paths between points or Boustrophedon-type paths. The goal was to assess whether the RL method better maintains the sensor's optimal distance and orientation compared to traditional methods. All the initial tests were conducted in simulation.

In addition to simulations, real-world experiments were carried out using a UR3e

robot to follow some of the inspection paths created by the trained RL model. This step aimed to validate how well the simulated solutions transfer to actual conditions.

Additionally, the tests examined how well the system adapts to various shapes. Since real-world objects vary widely in shape and size, it is crucial to determine if the RL approach offers improved performance across different geometries, leading to better accuracy and efficiency in inspections.

To understand the results, it's important to know what the metrics used for evaluating distance and orientation errors mean. The main metrics are Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Maximum Error, Median Error, and Standard Deviation.

MAE shows the average amount by which the measured values differ from the ideal values. It's a simple way to understand accuracy. MSE and RMSE, which square the errors, highlight larger mistakes more clearly. RMSE especially focuses on bigger errors more than MAE does. Maximum Error tells us the biggest single mistake made during scanning. Median Error shows the middle value of all errors when they are arranged from smallest to largest. Standard Deviation shows how spread out the errors are around the average, giving an idea of how consistent or variable the performance is.

6.4.1 Inspection system configuration

The scanning system is composed of a 6 dof UR3e robotic arm equipped with a triangulation laser profilometer model AT-C5-2040-CS-14-100. The complete configuration of the inspection system, which includes the UR3e robotic arm equipped with the profilometric sensor can be seen in Figure 6.31. Main parameters of the sensor are obtained from its datasheet and detailed in Table 6.1.

Table 6.1: Parameters of the profilometric sensor extracted from its datasheet.

Parameters	**Value**
Working Distance	197 mm
Z Range	120 mm
Field of View (X-FOV)	100 mm
Points per Profile	2048 pixels
Z Resolution	3.0 μm

The simulation environment is configured to closely match the actual hardware used in real-world tests. This includes modeling the UR3e robot's 6 degrees of freedom and replicating the AT-C5-2040-CS-14-100 sensor's characteristics, ensuring realistic performance in both environments.

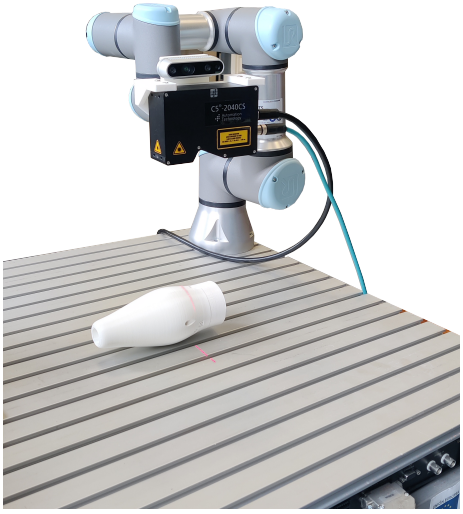


Figure 6.14: Experimental setup: The UR3e robotic arm from Universal Robots, equipped with an AT sensor.

Once the optimized trajectory is generated, the next step involves executing it on the real robot. To achieve this, the RoboDK software [112] is used to calculate the precise movements required for each joint of the robot to reach each pose along the trajectory. This process ensures that the generated motion instructions are accurate and tailored to the kinematic capabilities of the UR3e robot. Once the motion program is generated, it is transferred to the UR3e robot controller, which is configured to execute these motion commands efficiently and precisely in the real-world scenario.

6.4.2 Training process

The training process of the RL model for trajectory optimization in robotic inspection was developed using a detailed simulation environment, the characteristics of which are explained in [113].

In this context, a profilometric sensor was simulated with the same specifications as the Automation Technology model AT-C5-2040-CS-14-100, whose main parameters are detailed in Table 6.1. It is important to note that this setup is designed to generalize based on input parameters, allowing for adjustments to different working distance, for example.

The design of the training piece was aimed at representing a wide variety of conditions that could be encountered in real inspection applications. This piece, created in 3D modeling software, features changes in orientation, height variations, and flat surfaces. Its dimensions are 1050x150x50mm, as shown in Figure 6.15.

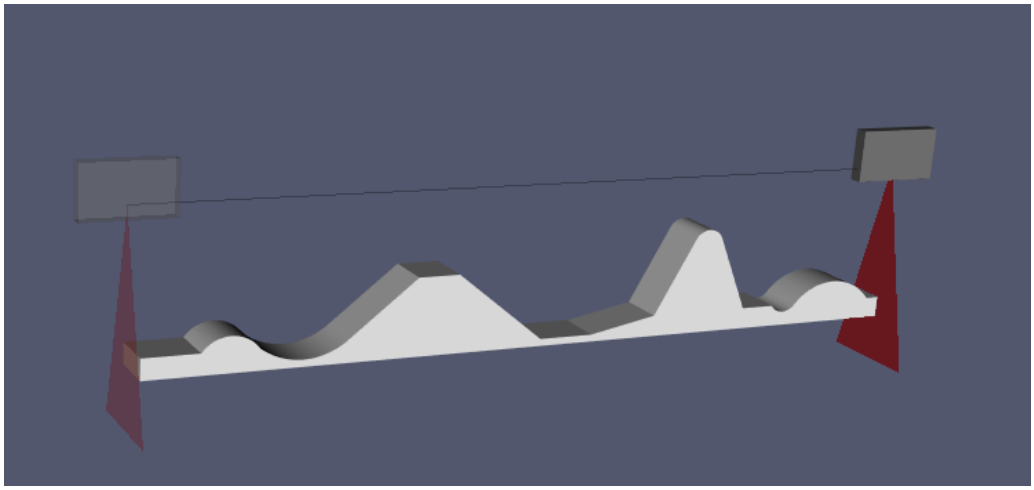


Figure 6.15: Environment used for the reinforcement learning model training. The CAD model of the piece used and the start and end poses of the trajectory to be optimized are shown.

During training, each episode is defined so that it corresponds to a starting point and an ending point, determined by the scanning direction and the piece's dimensions, visually represented in the same figure showing the CAD model of the piece used for training, as illustrated in Figure 6.15.

In the experiments, the action space is continuous, meaning actions are expressed as values within a continuous range rather than discrete values. Specifically, these actions are limited within the interval of [-1, 1], where position increments are measured in millimeters and pitch angles in degrees.

As previously mentioned, the RL algorithm used in this study is Proximal Policy Optimization (PPO). The results presented here focus on the training outcomes of

this specific algorithm. Table 6.2 shows the hyperparameters used for PPO. These parameters were set based on the recommended values from the stable-baselines library, which provided a solid foundation for optimizing the algorithm. The settings were carefully chosen to ensure effective performance in the given application.

Table 6.2: Hyperparameters for the PPO algorithm used.

Hyperparameters	**PPO**
Network Architecture	[64,64]
Activation Function	ReLU
Learning Rate	0.0003
Update Rate	2048 steps
Batch Size	64
Discount Factor (gamma)	0.99
Clip Ratio	0.2
Epoch	10

During the training process of the algorithms, various metrics are employed to assess their performance and convergence capability. These metrics include the reward per episode, the length of episodes, and the number of episodes required to reach a certain performance level. The reward per episode is a crucial metric indicating the total amount of reward accumulated by the agent in each training episode. Generally, a higher reward reflects better performance of the agent in the task.

However, in this specific training context, evaluating solely the accumulated reward per episode might not be the most appropriate approach. This is because the length of episodes can vary significantly depending on the step size, defined as the distance between profiles. Therefore, instead of focusing solely on the accumulated reward, it is preferred to evaluate the globally normalized reward by the length of the episode. This metric provides a more comprehensive assessment of the agent's performance, as it considers both the accuracy of measurements and the efficiency of the trajectory. By doing so, a more precise insight into the overall effectiveness of the model in trajectory optimization and inspection quality is obtained, regardless of the specific length of episodes.

These metrics can be seen in Figure 6.16. Subfigure (a) shows that the mean episodic reward steadily increases as training goes on, which means the agent is getting better at its task. In Subfigure (b), the mean episode length changes in a way that shows how well the agent is learning over time. Subfigure (c) shows the normalized reward, which rises steadily, indicating the agent's performance is

improving. Together, these subfigures show that the RL algorithm starts to stabilize and converge around episode 500, indicating that the learning process is effective.

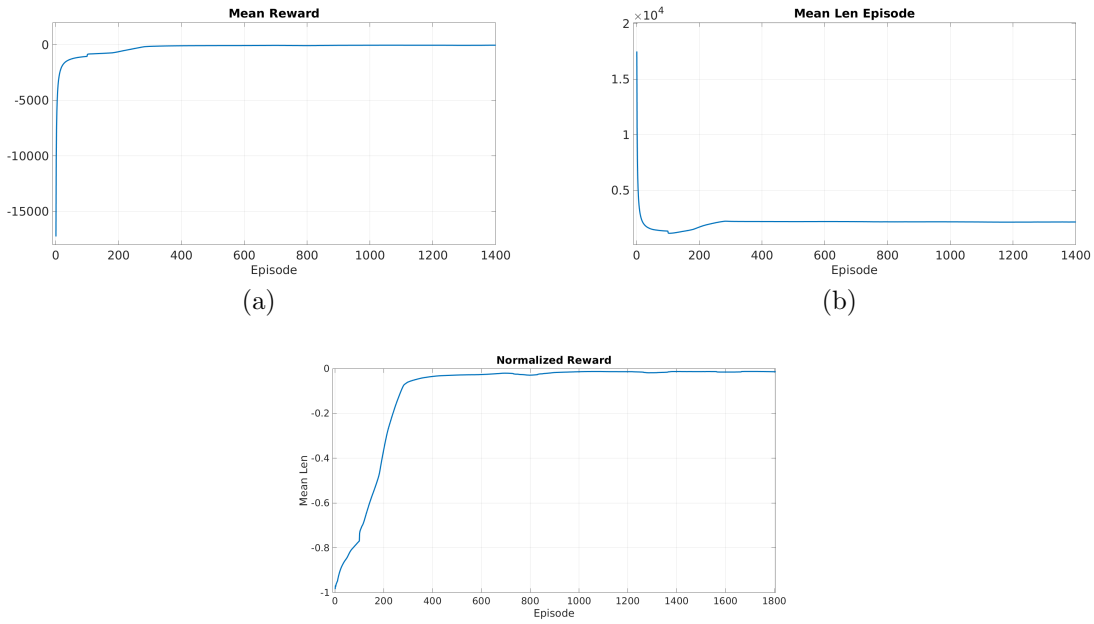


Figure 6.16: Training metrics PPO algorithm: (a) mean episodic reward, (b) mean episode length, and (c) normalized reward over the course of training.

In this experiment, the weight for each partial reward was set according to its significance. Orientation (R_α) and distance (R_D) were considered more important, so their weights were both set to 0.4. The profile separation (R_{Δ_s})received a lower weight of 0.2.

While all parameters are important, deviations in profile separation within acceptable ranges, predefined for reward generation, are considered less critical. Therefore, greater emphasis was placed on ensuring proper forward movement and prioritizing orientation and distance. Additionally, the penalty is at its maximum if the movement goes backward or deviates too far from the desired range.

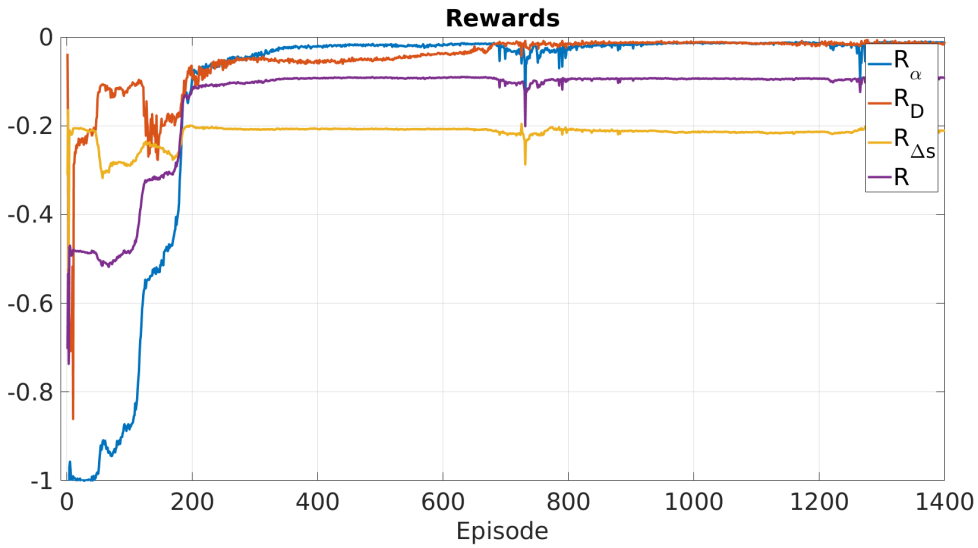


Figure 6.17: Comparison of overall reward (R) and partial rewards ($R_D, R_\alpha, R_{\Delta s}$) during training

In Figure 6.17, the partial and total rewards over time are illustrated. It can be observed that the reward associated with profile separation tends to be somewhat lower than the others, which is likely due to the assigned weights. However, the error remains quite small and within acceptable limits. This balance reflects the design decision to prioritize critical parameters like orientation and distance while maintaining an adequate level of accuracy in profile separation.

6.4.3 Car door

The first part used to evaluate the model is a car door of dimensions 1440x1060x190mm. The initial trajectory, which serves as the basis for the optimization, is shown in figure 6.18. This figure shows the initial Boustrophedon trajectory, with the inspection passes marked in red and the movements between passes in white.

The reinforcement learning model is applied to the different passes, dynamically adjusting the orientation and distance of the profilometric sensor. The model is applied exclusively to the red passes, where the inspection is performed. The optimized trajectories are shown in figure 6.19.

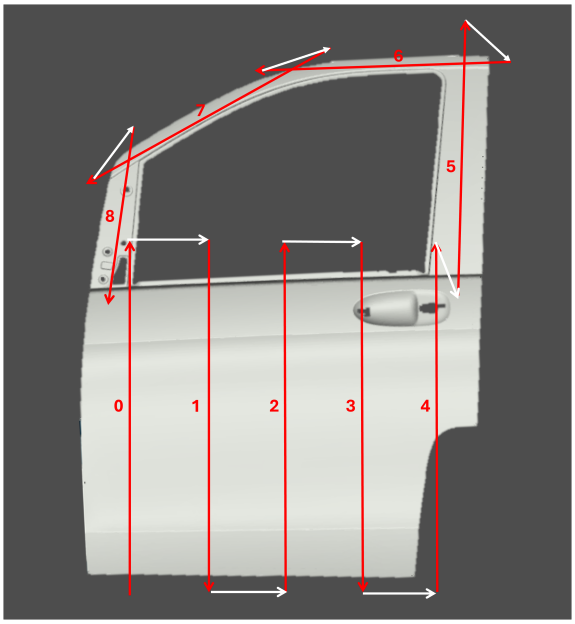
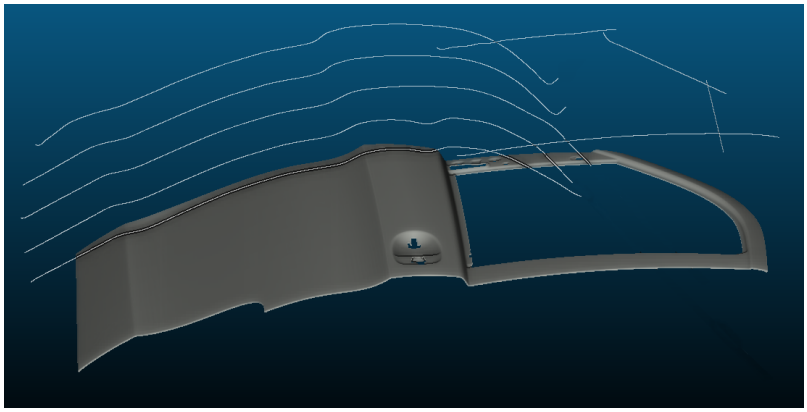


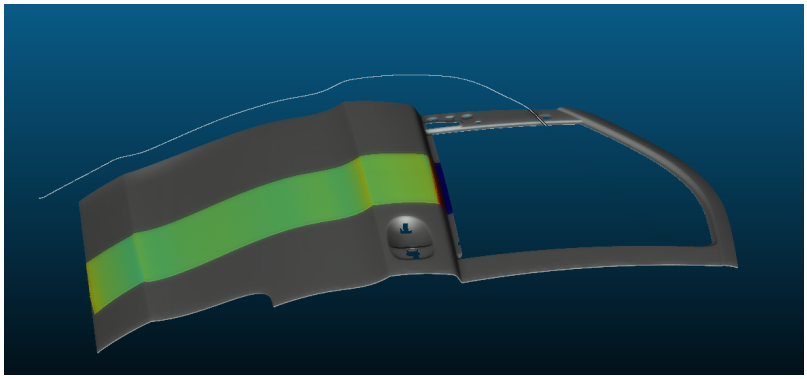
Figure 6.18: Initial Boustrophedon Trajectory for Car Door Scanning. Red lines represent the inspection passes to be optimized, while the white lines indicate movements between passes.

Figure 6.20 (a) shows the point cloud obtained during scanning using the profilometric sensor. The point cloud is represented by a color map indicating the error in the measured distance for each point. This error is defined as the difference between the optimal working distance of the sensor and the actual distance obtained during the measurement at each point.

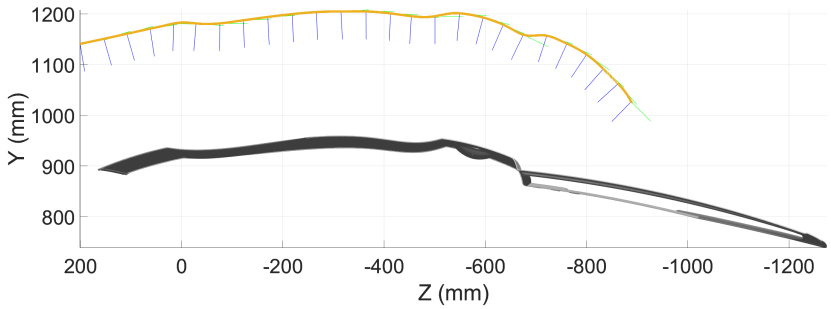
For a meaningful comparison, a scan was also performed using a traditional method. In this case, the boustrophedon trajectory was followed with a fixed configuration of height and orientation, without making adjustments during the process. This approach is commonly used in industrial applications due to its simplicity and ease of implementation. Figure 6.20 (b) shows the distance error map obtained with this method.



(a)



(b)

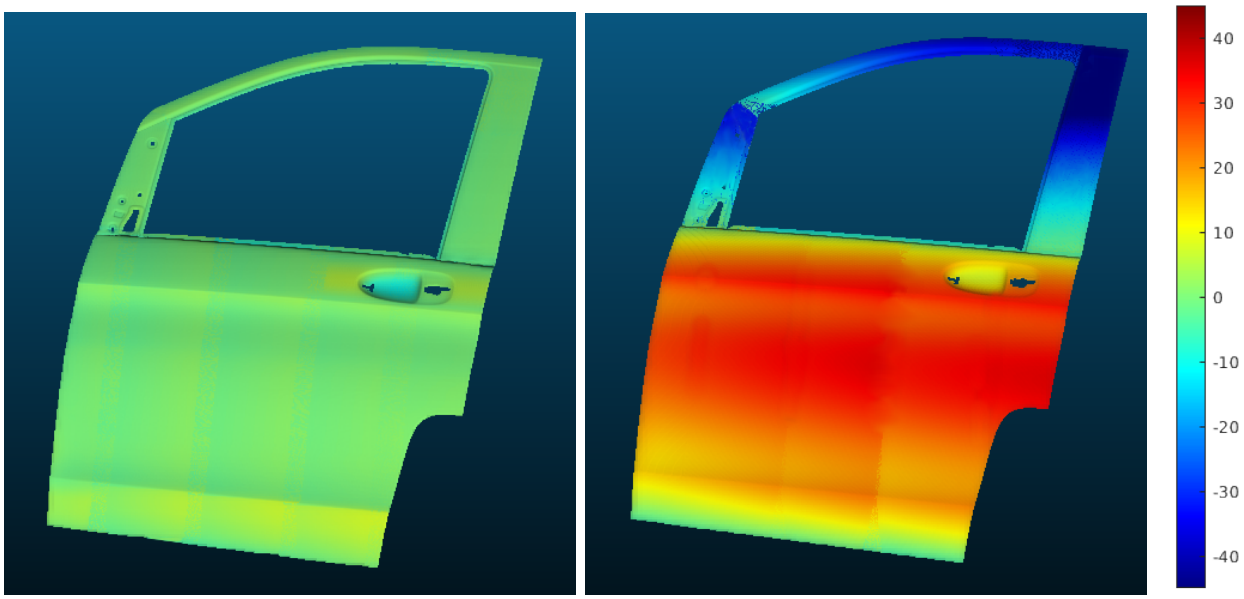


(c)

Figure 6.19: (a) Resulting trajectories after applying the RL model. (b) Area covered during the scanning process for one of the trajectories. (c) Detail of the trajectory shown in (b), highlighting specific orientations.

Table 6.3 presents the distance error metrics for each sub-trajectory using the

RL-optimized approach. These metrics are calculated with all the points of the scan, allowing an evaluation of its performance.



(a) (b)

Figure 6.20: Distance error map obtained during the scanning with profilometric sensor. The colors indicate the difference between the measured distance and the optimal sensor distance, normalized based on defined distance values. (a) Using trajectories that adapt to the surface of the piece, calculated by the RL algorithm. (b) Using straight trajectories defined by a start point and an end point.

The average Mean Absolute Error (MAE) across all sub-trajectories is 2.234 mm. This is substantially lower compared to the straight trajectory's average MAE of 37.862 mm, as shown in Table 6.4. Additionally, the Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) for the RL-optimized trajectories are 12.339 mm^2 and 3.296 mm, respectively, demonstrating a clear improvement over the straight trajectories, which have an MSE of 1936.900 mm^2 and an RMSE of 42.674 mm.

The maximum error for the RL-optimized sub-trajectories is 15.312 mm, while the straight trajectory exhibits a much larger maximum error of 71.279 mm. Median errors for the RL approach are notably smaller, averaging -0.080 mm, in contrast to the straight trajectory's median error of 7.850 mm. Furthermore, the standard deviation of errors is significantly lower for the RL-optimized trajectories, at 3.199 mm, compared to 20.349 mm for the straight trajectories.

This maximum error for the straight trajectory can exceed the sensor's measurement range, which has a Z-range of ±60 mm around the optimal distance. Consequently, in real-world applications, such errors could lead to areas falling outside the scanning zone of the sensor. Although these values are permissible in the simulation, it should be noted that in practical scenarios, the sensor would not be able to measure such extreme errors effectively.

Table 6.3: Distance Error Metrics for Optimized Scanning Paths Using RL (mm).

Scan	**MAE**	**MSE**	**RMSE**	**Max Err**	**Median Err**	**Std Err**
0	1.551	4.948	2.224	10.086	0.018	2.219
1	2.394	9.785	3.128	11.506	0.149	3.095
2	3.034	21.134	4.597	20.917	0.000	4.587
3	2.996	20.945	4.577	15.344	0.056	4.452
4	2.857	29.419	5.424	27.718	-0.522	5.298
5	1.428	6.327	2.515	12.817	-0.609	2.205
6	2.743	10.906	3.303	19.672	-0.193	3.236
7	1.682	3.936	1.984	11.775	1.008	1.897
8	1.424	3.648	1.910	7.969	-0.627	1.800
Average	2.234	12.339	3.296	15.312	-0.080	3.199

Table 6.4: Distance Error Metrics for Scanning Paths Using straight trajectory (mm)

Scan	**MAE**	**MSE**	**RMSE**	**Max Err**	**Median Err**	**Std Err**
0	33.108	1268.500	35.616	50.946	37.276	14.774
1	37.229	1586.300	39.828	56.918	41.156	16.240
2	40.399	1848.600	42.995	60.428	44.327	16.331
3	37.152	1596.500	39.956	57.732	39.699	16.487
4	45.085	2237.200	47.299	59.994	49.447	16.580
5	55.509	4938.800	70.277	149.460	-44.726	43.100
6	27.359	1000.700	31.634	63.074	-27.475	16.675
7	36.534	1787.500	42.279	70.510	-44.177	22.049
8	28.388	1168.300	34.181	72.445	-24.877	20.902
Average	37.862	1936.900	42.674	71.279	7.850	20.349

To give a clearer picture of the errors, we included error histograms with boxplots, as shown in Figure 6.21. This helps to understand the error distribution beyond the average values shown in the tables. The figure compares both types of visualizations for the two trajectory types.

The histograms show how the errors are spread out across the different trajectories, while the boxplots highlight the range and outliers of these errors. For the RL-generated trajectory, most errors are close to zero. On the other hand, the straight-line trajectory has a wider range of errors, with a higher average error of 37.862 mm, as summarized in the tables.

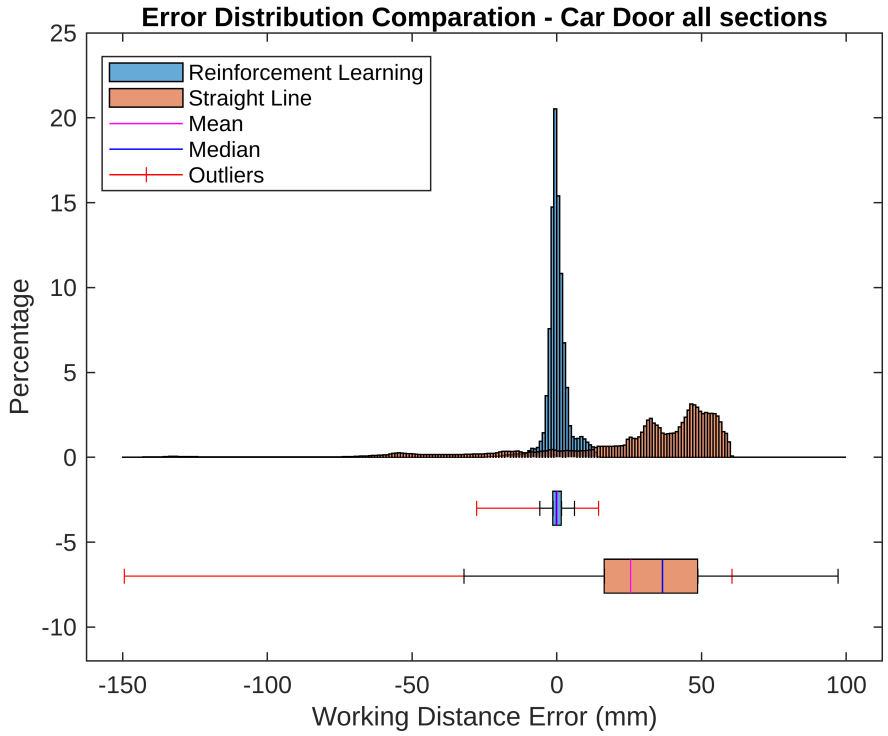
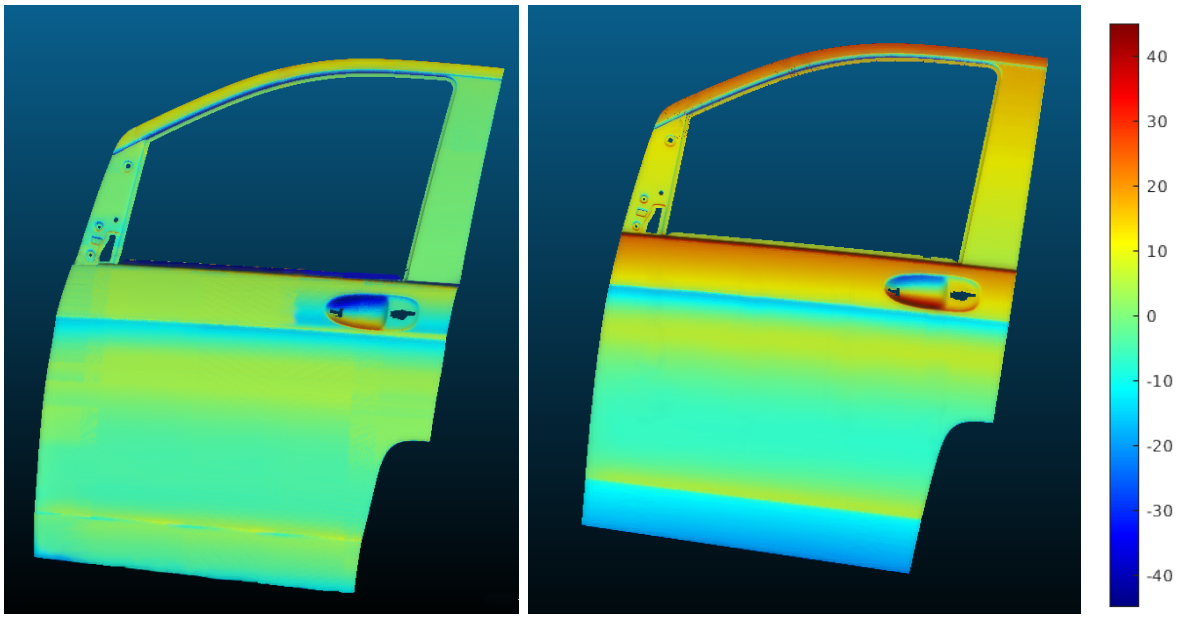


Figure 6.21: Comparation of the distance error distribution for all car door sections.

In addition to the distance error map, an orientation error map was generated, which displays angular deviations from the optimal sensor orientation at each point. This deviation refers to the incidence angle of the sensor on the surface. See Figure 6.22. Figure 6.22(a) shows the angular deviations for the RL-optimized trajectories, and Figure 6.22(b) depicts the deviations for straight trajectories. The color maps normalize the differences between the measured and optimal sensor orientations.



(a) (b)

Figure 6.22: Orientation error map showing angular deviations from the optimal sensor orientation at each point, normalized based on defined orientation values. (a) Using trajectories that adapt to the surface of the piece, calculated by the RL algorithm. (b) Using straight trajectories defined by a start point and an end point.

Orientation error metrics were also assessed. The data presented in Table 6.5 shows that the RL-optimized trajectories have an average Mean Absolute Error (MAE) of 2.256°, with a Root Mean Squared Error (RMSE) of 4.767°. These metrics indicate that the RL approach achieves a reasonably consistent performance across the scanning path, with a maximum error of 82.072° and a median error of 0.636°.

In contrast, Table 6.6 presents the orientation error metrics for the straight trajectories. Here, the average MAE is 13.028°, and the RMSE is 16.337°. While these values are notably higher compared to the RL-optimized approach, the differences are significant, indicating that the straight trajectory approach may not be as effective in minimizing orientation errors.

The maximum errors are reported as 82.072° for the RL trajectory and 89.567° for the straight trajectory. However, these maximum errors may not be representative of the overall performance. As shown in the error maps of orientation (Figure 6.22), these areas of significant error typically occur at edges or detailed features such as

the door handle or window lines. These maps show red points indicating high error values in these specific areas. This suggests that while maximum errors provide some insight, the errors in these specific regions do not fully capture the general accuracy of the scanning methods.

Table 6.5: Orientation Error Metrics for Optimized Scanning Paths Using RL (°)

Scan	MAE	MSE	RMSE	Max Err	Median Err	Std Err
0	1.043	7.291	2.700	89.764	0.286	2.525
1	1.367	6.414	2.533	79.522	0.398	2.304
2	2.488	31.914	5.649	79.740	0.446	5.271
3	3.058	51.104	7.149	80.511	0.310	6.829
4	2.381	20.466	4.524	89.677	0.335	4.145
5	1.393	20.346	4.511	89.371	0.044	4.325
6	4.229	42.648	6.531	77.251	3.374	5.120
7	2.548	21.144	4.598	68.509	0.447	4.093
8	1.795	22.182	4.710	84.303	0.085	4.475
Average	2.256	24.834	4.767	82.072	0.636	4.343

Table 6.6: Orientation Error Metrics for Straight Scanning Paths (°)

Scan	MAE	MSE	RMSE	Max Err	Median Err	Std Err
0	9.298	165.260	12.855	86.481	-3.516	12.840
1	9.979	186.950	13.673	86.480	-2.667	13.672
2	9.963	183.800	13.557	89.567	-2.256	13.557
3	10.273	210.560	14.511	86.480	-2.910	14.507
4	9.956	216.190	14.704	83.513	4.611	13.525
5	13.221	211.620	14.547	88.446	12.115	6.957
6	19.919	488.530	22.103	72.452	17.528	18.840
7	20.297	468.740	21.650	85.701	17.740	14.011
8	14.350	377.630	19.433	84.934	10.698	14.497
Average	13.028	278.810	16.337	84.895	5.705	13.601

Also, histograms and boxplots are presented in Figure 6.23. Similar to the distance errors, for the RL-generated trajectory, most orientation errors are clustered around zero, indicating high precision. In contrast, the straight-line trajectory exhibits a wider spread of errors, with higher variability. Although the mean orientation error is quite close to zero, the boxplot reveal that the majority of errors for the straight-line trajectory are distributed over a broader range. Notably, the outliers for this method show more pronounced deviations from the typical error range, highlighting its greater variability. The median error for this method is 13.028°.

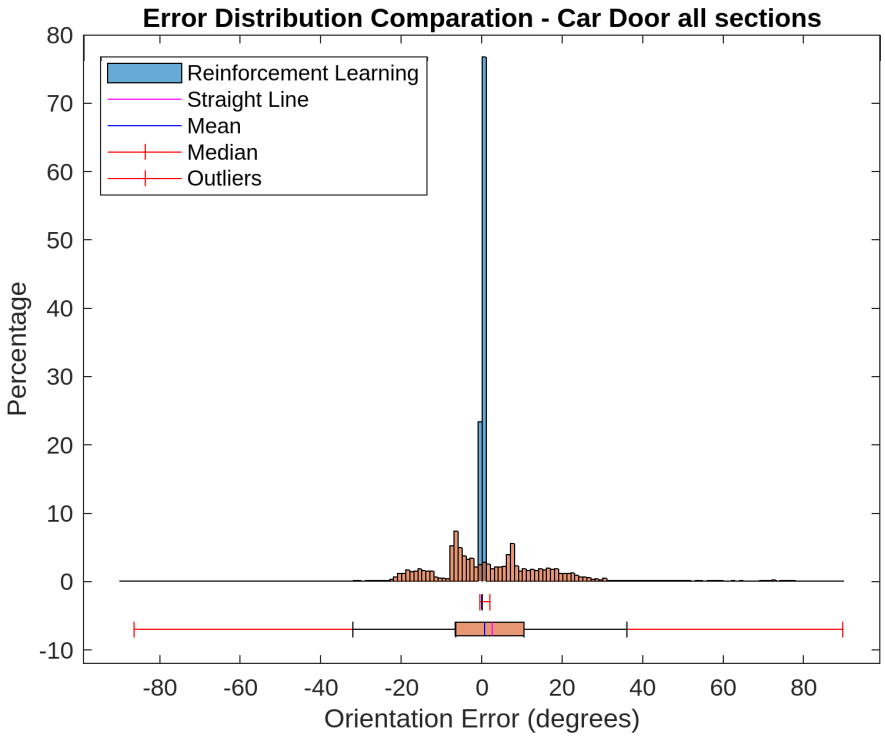


Figure 6.23: Comparation of the orientation error distribution for all car door sections.

The distance measurements show a significant improvement with the RL-optimized trajectories compared to the straight trajectories. The RL approach achieves notably more accurate and consistent distance measurements across the entire scanning area. This enhanced performance is particularly evident in the error metrics, which highlight a substantial reduction in average errors and variability.

In terms of orientation, the RL-optimized trajectories perform better in areas with surface deviations. This suggests that, as expected, the RL approach is more effective at managing variations in sensor orientation in regions where the surface is not perfectly flat.

These findings confirm the superior performance of the RL-optimized trajectories in achieving more precise and consistent distance measurements across all subtrajectories of the door. The straight trajectory, while simpler, results in significantly larger errors and greater variability, underscoring the advantages of using adaptive trajectories optimized by reinforcement learning for accurate scanning.

Moreover, in Figure 6.24, a zoomed-in section of a specific area from the global scan is presented. In this region, a bump-type defect has been intentionally introduced to analyze how the type of trajectory affects defect detection. This area was chosen because it is small and has a very pronounced orientation. The second row of Figure 6.24 shows the results of the defect search in the scans, highlighted by a red rectangle. The third row presents normalized density maps, which offer a detailed visualization of the distribution of points in the scanning trajectories.

The simple trajectory demonstrated a low point density in the defect region due to the scanning angle. In contrast, the RL-optimized trajectory achieved a higher point density at the same scanning speed. This increase in point density facilitates a clearer visualization of the defect, significantly enhancing its detection and characterization during the inspection process.

In summary, while the overall orientation error metrics are similar due to the door's predominantly flat geometry, the RL-optimized trajectories demonstrate their strengths by better managing localized challenges, highlighting their potential benefits in more complex scanning environments.

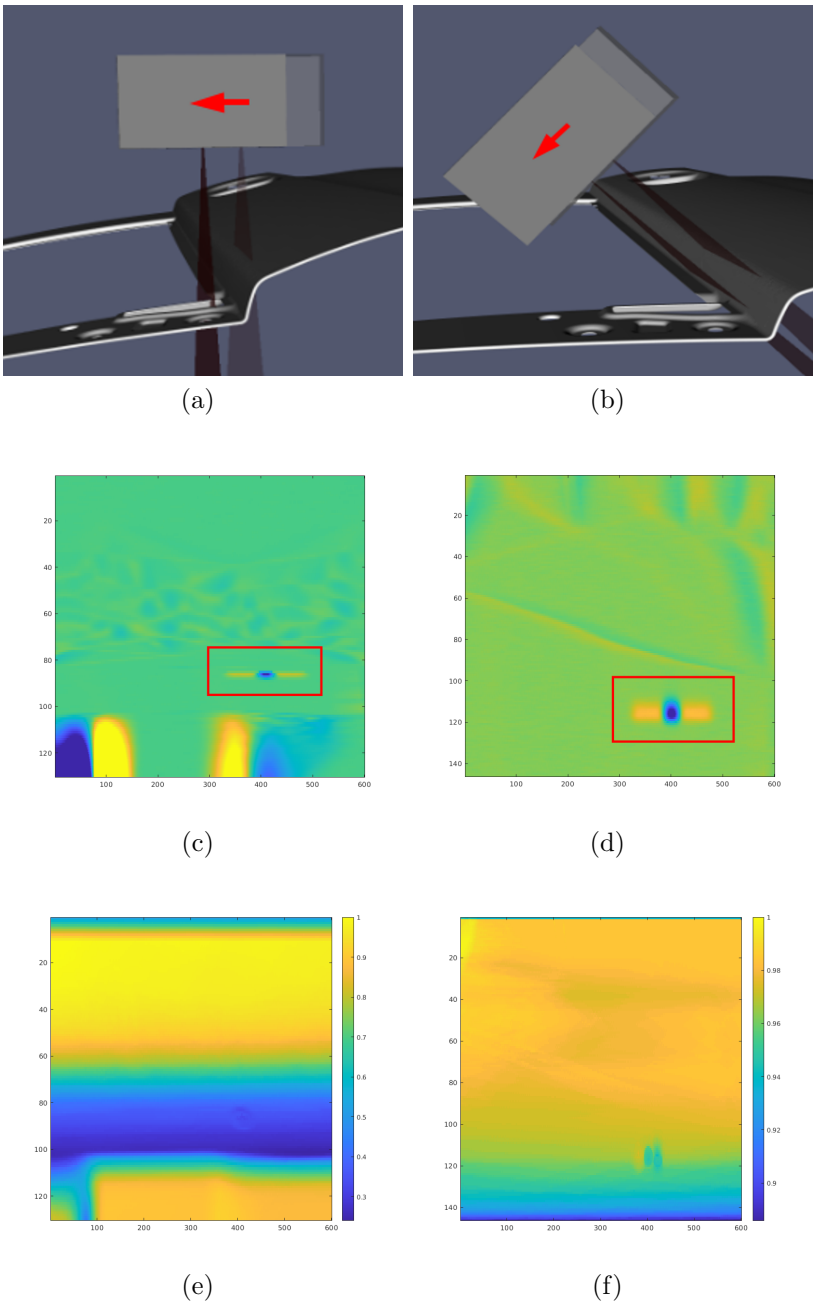


Figure 6.24: Zoom of a specific area of the scan of the car door. First row: Segment of the trajectory followed in the area where the defect was inserted. Second row: Results of defect detection in the scans. Third row: Density maps obtained from the scan trajectories. (The color scale ranges from [0,1]).

6.4.4 Pen Holder

This subsection details the results from the pen holder scanning experiments. The dimensions of the pen holder are 150x75x75mm. In line with previous sections, a comparison is made between two scanning trajectories: a straight trajectory, moving from a start point to an end point as shown in Figure 6.25, and a trajectory optimized by the Reinforcement Learning algorithm, see Figure 6.26.

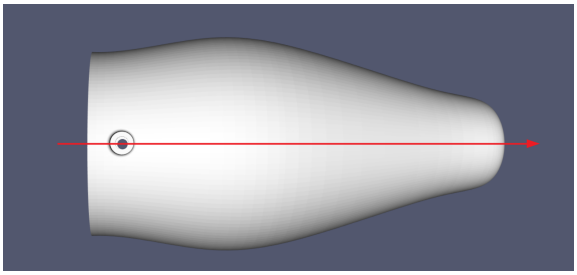


Figure 6.25: Zenital view of the CAD model of the pen holder and the straight trajectory planned for scanning.

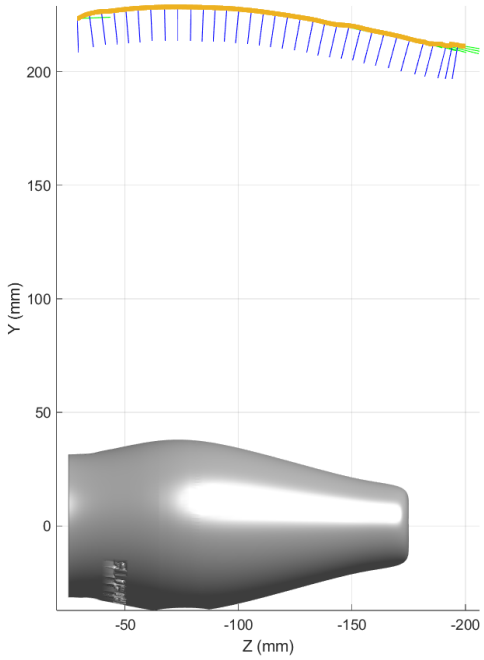
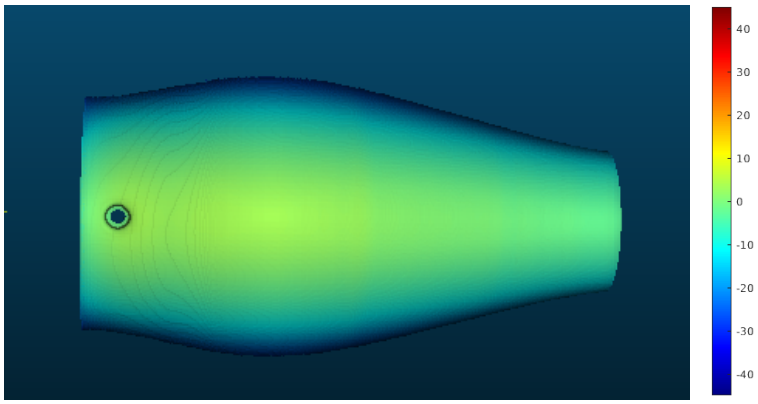


Figure 6.26: Scanning trajectory generated by the RL algorithm for the pen holder.

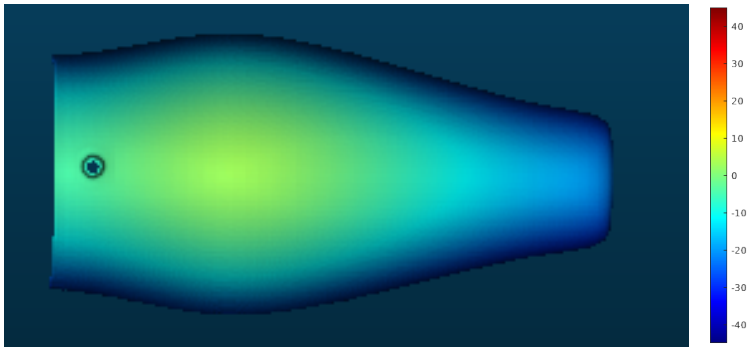
Additionally, results from executing these trajectories are presented for both simulation and real-world conditions, utilizing the previously defined inspection system.

Simulation

Figure 6.27 illustrates the distance error maps obtained from the profilometric sensor during the scanning process. These maps reflect the differences between the measured distances and the optimal sensor distances.



(a)



(b)

Figure 6.27: Distance error map obtained during the scanning with profilometric sensor. The colors indicate the difference between the measured distance and the optimal sensor distance, normalized based on defined distance values. (a) Using trajectory that adapt to the surface of the piece, calculated by the RL algorithm. (b) Using straight trajectory.

The first map, corresponding to the RL-optimized trajectory, shows how the scanning path, adapted by the Reinforcement Learning (RL) algorithm, minimizes deviations from the optimal distance. This trajectory closely follows the surface contours of the pen holder, resulting in a more accurate representation of distance errors. The color map highlights the reduced error across the scanning area, indicating that the RL-optimized path effectively reduces distance discrepancies.

In contrast, the second map depicts the distance errors for the straight trajectory. Here, the sensor moves along a linear path from start to end points, without adapting to the surface contours. This approach results in higher distance errors, as evidenced by the color map, which shows more significant deviations from the optimal values compared to the RL-optimized trajectory.

Table 6.7 provides a quantitative comparison of the distance errors for both trajectories. The RL-optimized trajectory demonstrates a Mean Absolute Error (MAE) of 2.438 mm, a Mean Squared Error (MSE) of 11.299 mm^2, and a Root Mean Squared Error (RMSE) of 3.361 mm. The maximum observed error is 56.720 mm, with a median error of 2.835 mm and a standard deviation of 2.315 mm.

Conversely, the straight trajectory results in a higher MAE of 6.5737 mm, an MSE of 91.605 mm^2, and an RMSE of 9.5711 mm. The maximum error recorded is 61.15 mm, with a median error of 3.415 mm and a standard deviation of 6.9574 mm.

These results indicate that the straight trajectory is less effective, showing larger average and maximum errors compared to the RL-optimized approach. Overall, the data underscores the superior performance of the RL-optimized trajectory in minimizing distance errors during scanning.

Table 6.7: Comparison of Distance Errors (mm)

Type	**MAE**	**MSE**	**RMSE**	**Max Err**	**Median Err**	**Std Err**
RL	2.438	11.299	3.361	56.720	2.835	2.315
Straight	6.574	91.605	9.571	61.150	3.415	6.957

Figure 6.28 shows the distribution of distance errors for the two different trajectories. The figure indicates that the RL-optimized trajectory has smaller distance errors compared to the straight-line trajectory. It's also worth noting that the negative errors shown correspond to the lateral areas of the pen holder. This is expected due to the geometry of the object and is clearly visible in the distance map in Figure 6.27. Ultimately, the goal is to optimize the average distance for each profile. Given

the shape of the pen-holder, it is impossible for all points to have a distance error of zero.

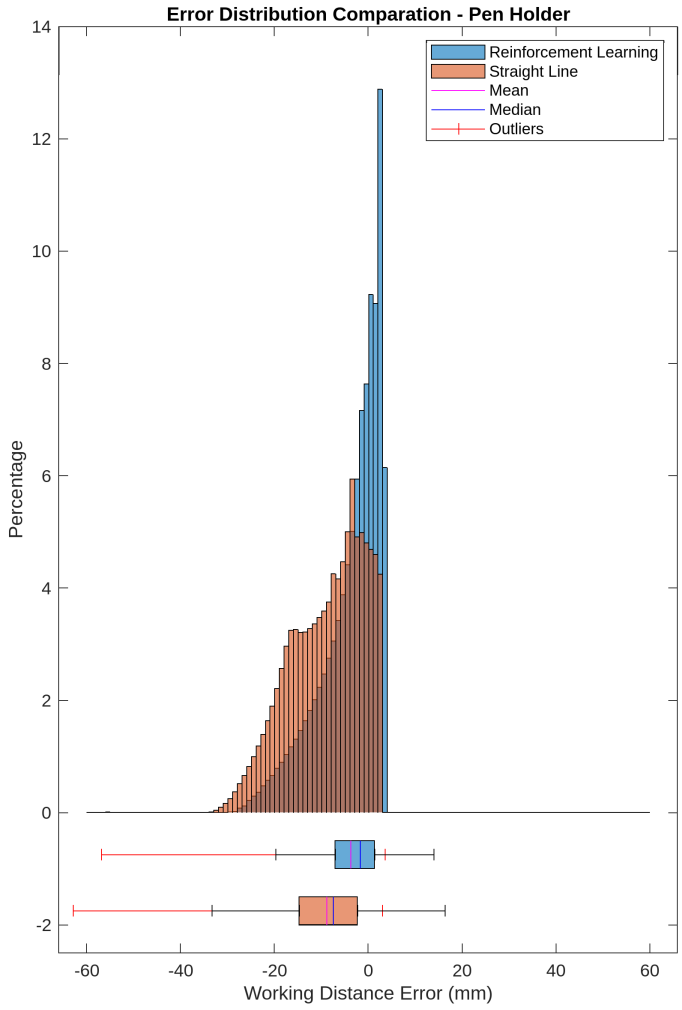
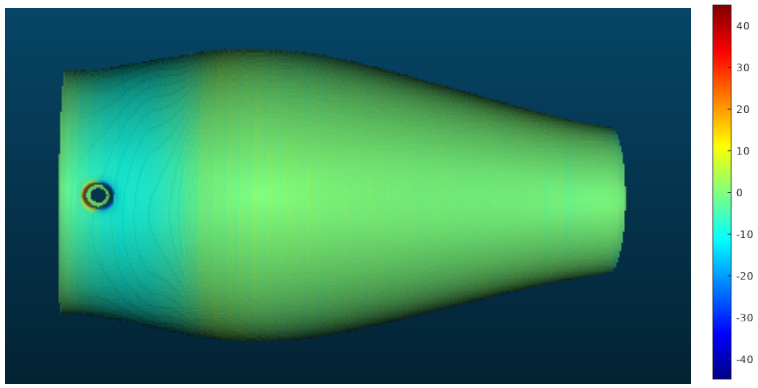
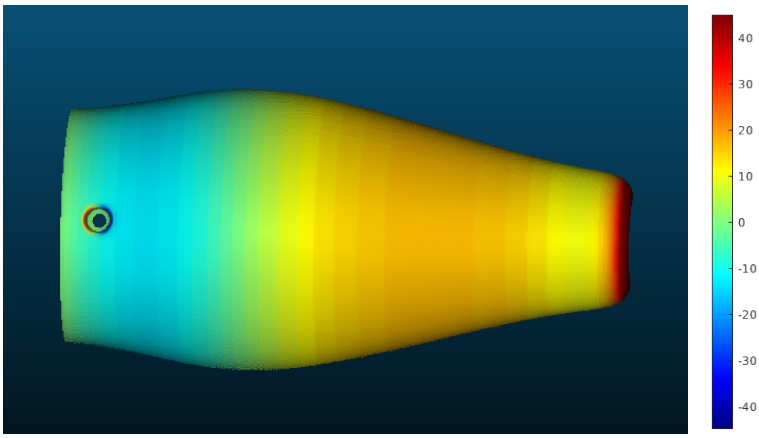


Figure 6.28: Comparison of the distance error distribution for the pen holder experiment.

Figure 6.29 displays the orientation error maps for the pen holder scanning. These maps illustrate the angular deviations from the optimal sensor orientation, with each subfigure representing a different scanning trajectory.



(a)



(b)

Figure 6.29: Orientation error map showing angular deviations from the optimal sensor orientation at each point. (a) Using trajectory that adapt to the surface of the piece, calculated by the RL algorithm. (b) Using straight trajectory.

Subfigure (a) shows the errors for the trajectory optimized by the RL algorithm. This trajectory adjusts dynamically to the surface of the pen holder, leading to more accurate sensor orientations with generally lower angular deviations. The color map indicates relatively small deviations, suggesting that the RL optimization effectively reduces orientation errors.

In contrast, subfigure (b) depicts the errors for the straight trajectory. This method follows a linear path from start to finish, resulting in higher angular deviations. The color map for this trajectory shows more significant deviations, reflecting the less effective alignment of the sensor with the surface contours compared to the RL-optimized path.

Table 6.8 highlights a clear contrast between the orientation errors for the two scanning trajectories. The RL-optimized trajectory demonstrates a significantly lower MAE of 0.707°, compared to the straight trajectory's MAE of 11.460°. This difference indicates that the RL approach achieves much better alignment with the optimal sensor orientation.

The MSE and RMSE further reflect this disparity. The RL trajectory has an MSE of 5.634° and an RMSE of 2.374°, whereas the straight trajectory shows an MSE of 187.33° and an RMSE of 13.687°. However, the maximum errors reported—66.075° for the RL trajectory and 87.304° for the straight trajectory—are seen in areas where the sensor angle changes sharply. The median errors are much lower: 0.373° for the RL and 10.942° for the straight trajectory. This shows that the RL method is more consistent in keeping the sensor well-aligned. The standard deviations also reflect this, with 2.266° for the RL trajectory and 7.484° for the straight trajectory.

Overall, these comparisons underscore the superior performance of the RL optimized trajectory in minimizing both average and peak orientation errors, demonstrating its effectiveness in achieving more precise sensor orientations during scanning.

Table 6.8: Comparison of Orientation Errors (^{o}): RL and Straight trajectory

Type	**MAE**	**MSE**	**RMSE**	**Max Err**	**Median Err**	**Std Err**
RL	0.707	5.634	2.374	66.075	0.373	2.266
Straight	11.460	187.330	13.687	87.304	10.942	7.4844

Figure 6.30 illustrates the distribution of orientation errors for the two trajectory types. It is particularly notable that, for the RL-optimized trajectory, most of the errors are concentrated around zero, whereas the straight-line trajectory shows a wider spread, ranging from -10 to 20°.

The boxplot reveals some high-value outliers for the RL method, which correspond to the region of the object's left side with a hole. This is clearly visible in the map shown in Figure 6.29, where areas of high error, both positive and negative, are highlighted in blue and red. Despite these outliers, the histogram indicates that their occurrence is quite rare. Overall, this reinforces the RL-optimized trajectory's effectiveness, although certain geometric features of the object can still introduce isolated errors.

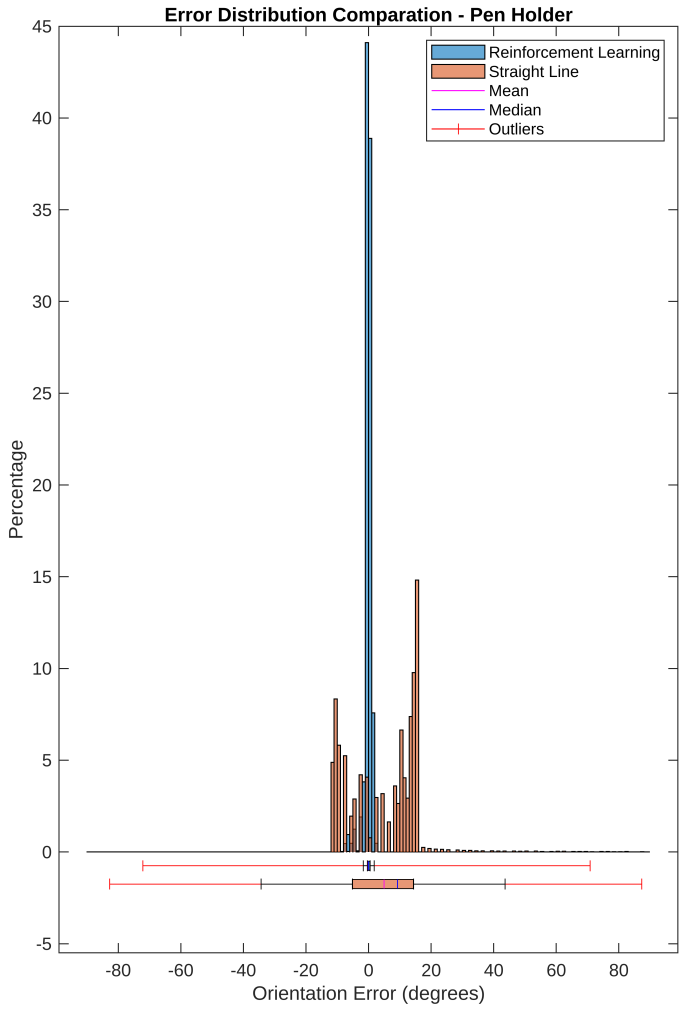


Figure 6.30: Comparison of the orientation error distribution for the pen holder experiment.

Real experiment

Here, results provided by executing the RL trajectory in real world are presented. As previously noted, RoboDK software is employed to compute the exact movements needed for each joint of the robot to achieve every pose along the trajectory. These

computed movements are then executed on a UR3e robot, which is equipped with the sensor mounted at its end effector. The experimental setup is illustrated in the image 6.31, which displays the sensor and pen holder prepared for scanning:

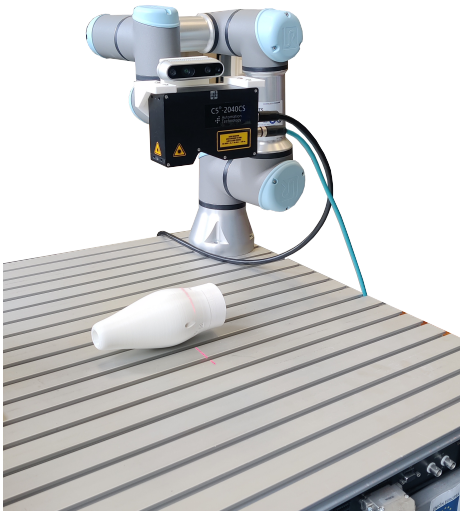


Figure 6.31: Experimental setup: The UR3e robotic arm from Universal Robots, equipped with an AT sensor.

In Figure 6.32, the scanning results are presented as 2D images, comparing the RL and straight trajectories across both real and simulated scenarios. On the X-axis, the points per profile are plotted, while the Y-axis represents the number of profiles obtained during the scan. The color gradient in the images indicates the distance measured relative to the sensor's optimal working distance.

The number of profiles differs between the RL and straight trajectories because the straight trajectory scans from a starting point to an endpoint at a constant speed, which in this case was slower than the RL trajectory. As a result, the straight trajectory collected fewer profiles compared to the RL trajectory.

In the images, the results from the simulation and the real-world experiments show a high degree of similarity. This indicates that the simulation effectively replicates the real-world conditions. However, some differences can be observed, which are likely due to the initial calibration of the piece.

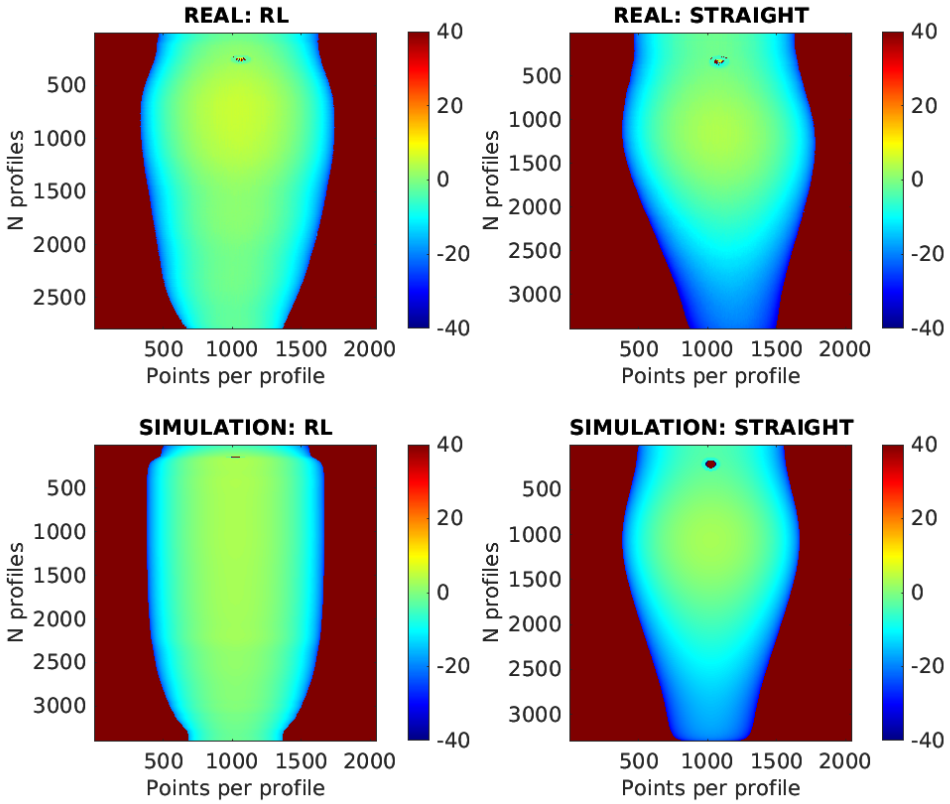


Figure 6.32: Pen Holder Scan: 2D images of scanning results comparing RL and straight trajectories in both real and simulated scenarios.

The metrics obtained during the real experiments are shown in Tables 6.9 and 6.10. The comparison between the simulated and real-world distance error metrics reveals a strong correlation in performance for both RL-generated and straight paths.

Table 6.9: Comparison of Distance Errors for RL Trajectory (mm)

Type	MAE	MSE	RMSE	Max Error	Median Err	Std Err
Real	2.422	9.139	3.023	5.881	1.798	1.809
Simulated	2.438	11.299	3.361	56.720	2.835	2.315

Table 6.10: Comparison of Distance Errors for Straight Trajectory (mm)

Type	MAE	MSE	RMSE	Max Err	Median Err	Std Err
Real	6.965	100.550	10.027	24.364	3.172	7.215
Simulated	6.574	91.605	9.571	61.150	3.415	6.957

In the real-world tests, the RL path demonstrated a Mean Absolute Error (MAE) of 2.422 mm, while the simulated results showed an MAE of 2.438 mm. This close match indicates that the RL model effectively transfers its learned strategies from simulation to practical applications. The Root Mean Squared Error (RMSE) for the RL path was 3.023 mm in real-world tests, compared to 3.361 mm in simulation, further demonstrating consistent accuracy across different environments.

For the straight path, the real-world MAE was 6.965 mm, while the simulated MAE was 6.574 mm. This comparison highlights a slight variance but still aligns with the overall trend, where the RL model consistently reduces errors more effectively than the traditional straight-line approach. The consistency in MSE, RMSE, and Median Error across both real-world and simulated scenarios underscores the robustness of the RL method in diverse settings.

The alignment between the simulated and real-world results validates the use of simulation as a reliable proxy for real-world experiments. This consistency confirms that the insights and optimizations developed in the simulated environment are applicable and beneficial in practical scenarios. As a result, the analyses and conclusions drawn from the simulations can be confidently extrapolated to real-world applications, offering valuable guidance for further industrial implementations.

6.4.5 Parrot Drone

The results obtained from the inspection trajectory of the Parrot drone are presented below. Figure 6.33 shows an image of the drone's body and its CAD model. The dimensions of the part are 350x95x65mm.

A scan of one side of the drone body will be performed, focusing on the flat top surface. Due to its dimensions, the base trajectory used will be a simple straight line from the beginning of the part to the end, see figure 6.34.

Figure 6.33: Image of the Parrot AR. 2.0 drone body and its CAD model.

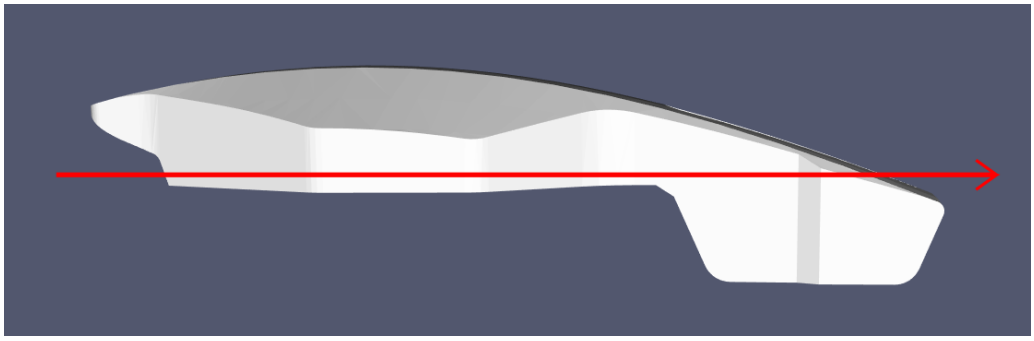


Figure 6.34: Zenital view of the straight path followed during scanning.

Using the trained reinforcement learning (RL) model, the optimized scanning trajectory for the drone is generated, see figure 6.35. In this trajectory, the sensor progresses along the path but occasionally moves backward to adjust its orientation. This backward movement helps to maintain the correct working distance and ensure consistent measurement accuracy throughout the scanning process.

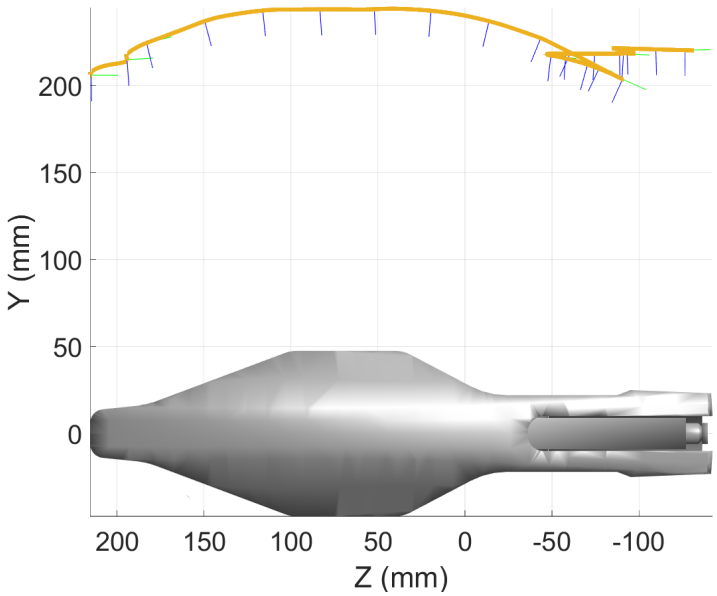


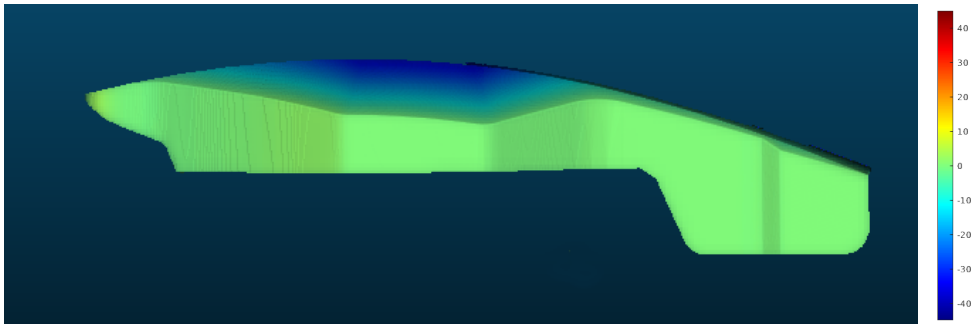
Figure 6.35: Scanning trajectory generated by the RL algorithm.

Simulation

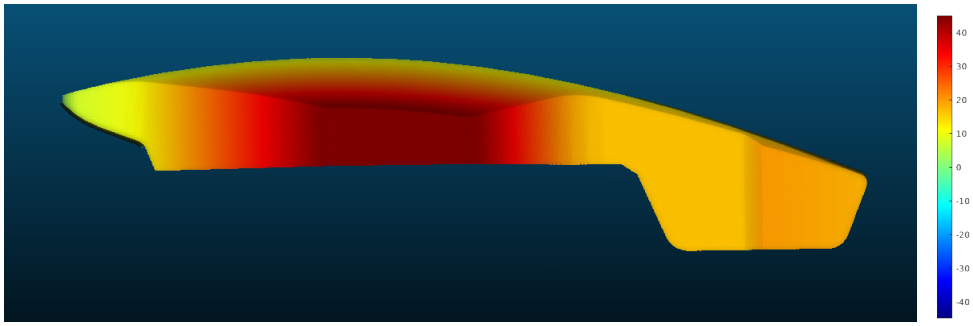
Figure 6.36 displays distance error maps for both trajectories. These maps illustrate how the difference between the measured distance and the optimal distance varies across the scanned surface. The RL-optimized trajectory results in a more consistent and less severe distribution of errors compared to the straight trajectory, especially in areas where the distance to the object changes.

The results in Table 6.11 provide a clear comparison of distance errors between the RL-optimized and straight trajectories. The RL-optimized trajectory shows a Mean Absolute Error of 3.222 mm, which is notably lower than the 22.205 mm for the straight trajectory.

The Root Mean Squared Error also highlights a difference, with the RL trajectory having a value of 7.488 mm compared to 24.393 mm for the straight trajectory. This indicates that while both trajectories have some large errors, the RL-optimized trajectory tends to have fewer large deviations.



(a)



(b)

Figure 6.36: Distance error map obtained during the scanning with profilometric sensor. The colors indicate the difference between the measured distance and the optimal sensor distance, normalized based on defined distance values. (a) Using trajectory that adapt to the surface of the piece, calculated by the RL algorithm. (b) Using straight trajectory defined by a start point and an end point.

Both trajectories have similar maximum errors, 39.666 mm for the RL path and 40.490 mm for the straight path. These maximum errors occur in areas where the distance changes significantly and are less indicative of overall performance. In the case of the RL trajectory, these maximum errors occur outside the flat top surface and are less indicative of overall performance. The error map shows that the areas with the highest errors, highlighted in blue, are located at the transition between the flat surface and other curved sections of the drone. This suggests that the significant errors are associated with the curvature changes rather than the flat scanning area itself.

The Median Error is much lower for the RL trajectory at 0.325 mm, compared to 17.426 mm for the straight trajectory, showing that the RL method tends to be closer to the optimal distance in most measurements. Additionally, the Standard

Deviation for the RL trajectory is 7.005 mm, less than the 10.248 mm for the straight trajectory, reflecting a more consistent performance.

Table 6.11: Comparison of Distance Errors (mm)

Type	MAE	MSE	RMSE	Max Err	Median Err	Std Err
RL	3.222	56.069	7.488	39.666	0.325	7.005
Straight	22.205	595.030	24.393	40.490	17.416	10.248

Figure 6.37 compares the two trajectory types using histograms and boxplots of distance errors for all scanning points. For the RL-optimized trajectory, most errors are clustered around very low values, close to zero. The boxplot also shows some outlier points with higher errors, which, as indicated in the error map of Figure 6.36, are located in the upper part of the drone. This area has a noticeable curvature and deviates from the inspection plane, which leads to these higher errors. As mentioned earlier, these errors are due to the geometry of the piece. Given the shape of the pen-holder, it's expected that not all points will have a distance error of zero.

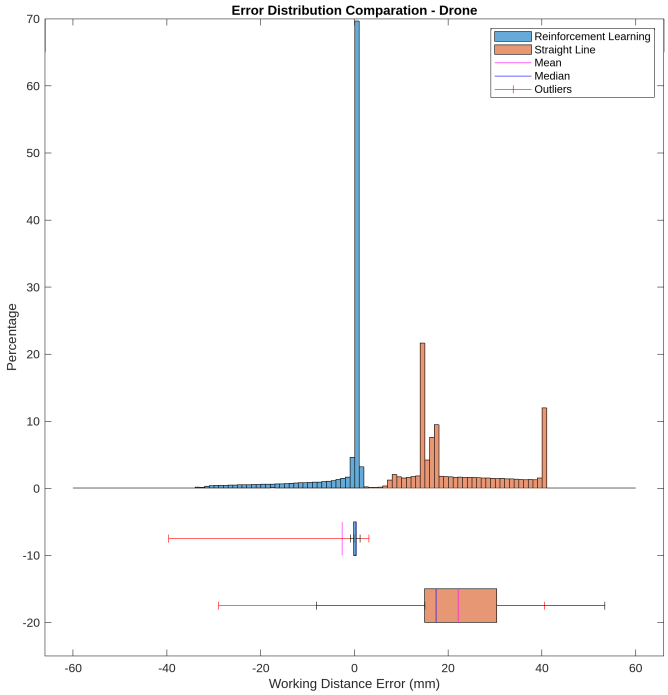
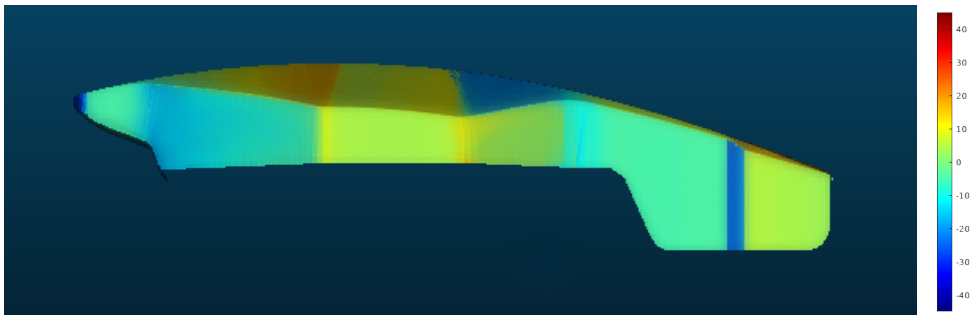


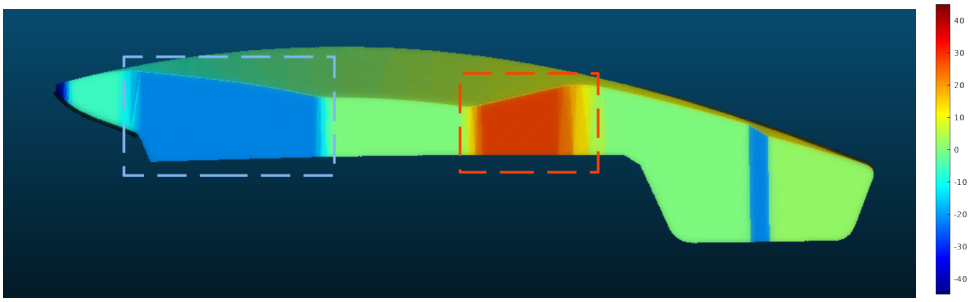
Figure 6.37: Comparison of the distance error distribution for the drone experiment.

In contrast, the straight-line trajectory shows higher errors and more variation. This broader range of errors and greater variability emphasize the RL-optimized trajectory's better accuracy and consistency in reducing distance errors.

The comparison of orientation error metrics between the RL-optimized and straight trajectories shows that both methods yield comparable results. As illustrated in Figure 6.38 and in Table 6.12, the orientation errors are more uniformly distributed with the RL-optimized trajectory, with fewer extreme values. This is evident from the standard deviation metric, which suggests that the errors are more consistent across the scanning path.



(a)



(b)

Figure 6.38: Orientation error map showing angular deviations from the optimal sensor orientation at each point. (a) Using trajectory that adapt to the surface of the piece, calculated by the RL algorithm. (b) Using straight trajectory defined by a start point and an end point. Two problematic areas marked by squares blue and orange.

In particular, the RL-optimized trajectory demonstrates a significant reduction in errors in specific problematic areas, as highlighted in Figure 6.38(b). Notably, in region 1, marked by the blue area on the left, the RL approach reduces errors

Table 6.12: Comparison of Orientation Errors (Degrees)

Type	MAE	MSE	RMSE	Max Err	Median Err	Std Err
RL	7.269	88.751	9.421	63.401	4.881	5.993
Straight	8.977	194.710	13.954	73.328	2.283	10.688

from -20° to -8°. In region 2, represented by the orange area on the right, errors are reduced from 27° to 4°. Although the overall error metrics are comparable, the RL trajectory shows clear improvements in minimizing and standardizing errors in these critical regions.

The histograms and boxplots presented in Figure 6.39 support this observation. In the histogram for the straight-line trajectory, two prominent peaks are evident at the extremes: one at around -20° and another at approximately 27°, which correspond to the problematic areas previously discussed. The remaining errors are concentrated around lower values, with a median error of 8.977°.

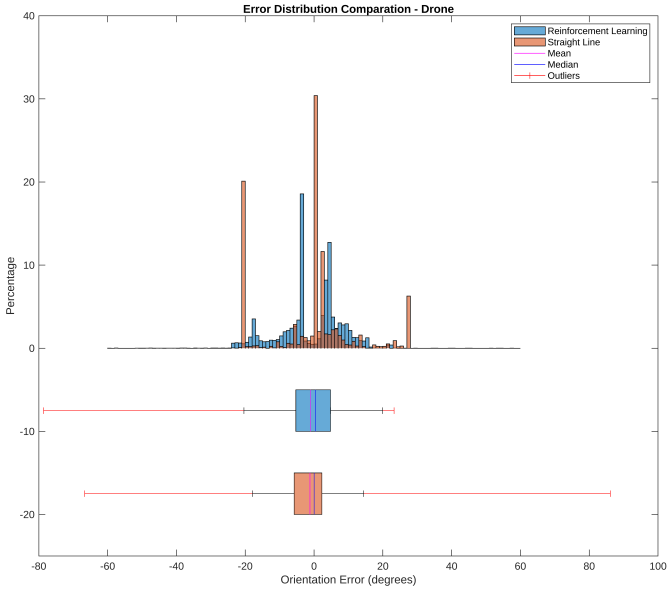


Figure 6.39: Comparison of the orientation error distribution for the drone experiment.

In contrast, the histogram for the RL-optimized trajectory shows a noticeable reduction in these peaks, resulting in a smoother error distribution curve. This reflects

a significant improvement, with the median error for the RL trajectory reduced to 7.269°. This smoother distribution and lower median highlight the RL approach's effectiveness in minimizing orientation errors, particularly in the previously problematic regions.

Real experiment

Here, the results from executing the RL trajectory in real-world drone scanning are presented. Figure 6.40 shows a comparison of scanning results between the RL and straight trajectories in both real and simulated scenarios. The X-axis displays the points per profile, while the Y-axis represents the number of profiles collected during the scan. The color gradient in the images reflects the distance measured relative to the sensor's optimal working distance.

The images reveal a significant similarity between the results from the simulation and the real-world experiments. This suggests that the simulation accurately replicates the real-world conditions.

The metrics obtained during the real experiments are shown in Tables 6.13 and 6.14 The comparison between the simulated and real-world distance error metrics reveals a strong correlation in performance for both RL-generated and straight paths.

Table 6.13: Distance Error Metrics for Real Drone Experiments (RL Trajectory) (mm)

Type	**MAE**	**MSE**	**RMSE**	**Max Err**	**Median Err**	**Std Err**
Real	3.298	14.795	3.847	15.079	3.445	1.979
Simulated	3.222	56.069	7.488	39.666	0.325	7.005

Table 6.14: Distance Error Metrics for Real Drone Experiments (Straight Trajectory) (mm)

Type	**MAE**	**MSE**	**RMSE**	**Max Err**	**Median Err**	**Std Err**
Real	21.570	590.660	24.303	36.574	18.643	11.202
Simulated	22.205	595.030	24.393	40.490	17.416	10.248

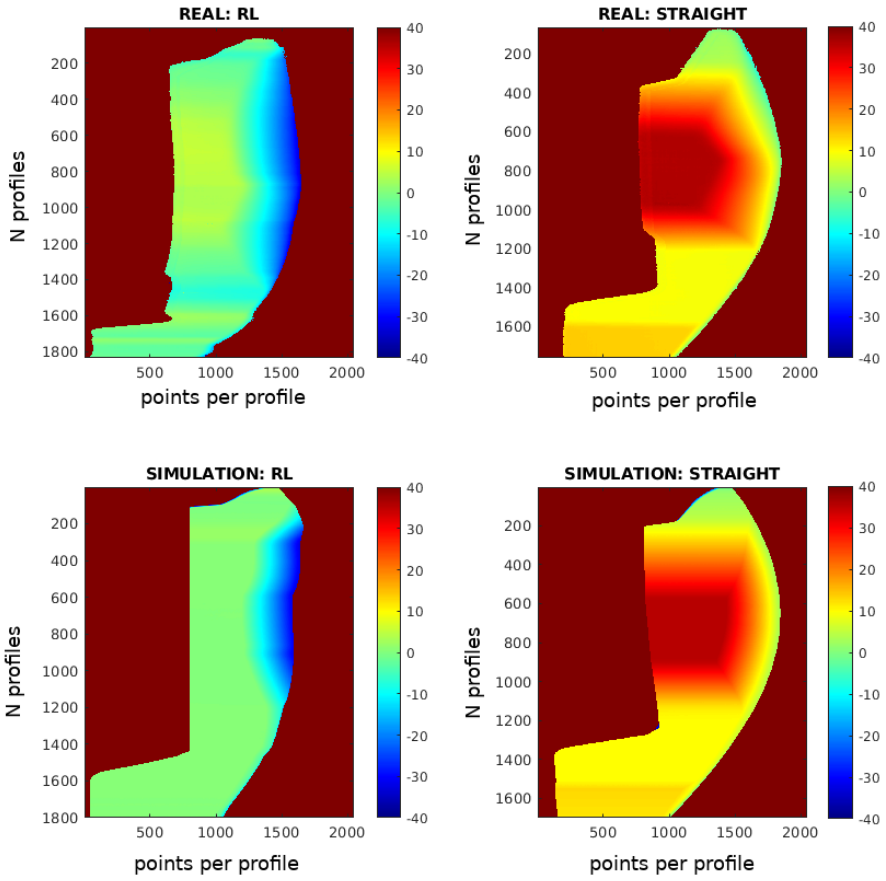


Figure 6.40: Parrot Drone Scan: 2D images of scanning results comparing RL and straight trajectories in both real and simulated scenarios.

The comparison between the real and simulated distance error metrics for the RL trajectory reveals that both sets of results are quite similar. For the RL trajectory, the real-world metrics show an MAE of 3.298 mm, an MSE of 14.795, and an RMSE of 3.847 mm, while the simulated metrics are slightly higher with an MAE of 3.222 mm, an MSE of 56.069, and an RMSE of 7.488 mm. Despite these differences, the closeness of these values suggests that the RL model performs consistently across both simulated and real-world scenarios, validating the simulation as a reliable predictor of real-world performance.

Similarly, for the straight trajectory, the real-world results indicate an MAE of 21.570 mm, an MSE of 590.660, and an RMSE of 24.303 mm, compared to the simu-

lated results which show an MAE of 22.205 mm, an MSE of 595.030, and an RMSE of 24.393 mm. The minimal variations between the real and simulated results for the straight trajectory further confirm that the simulation environment closely approximates real-world conditions. These findings demonstrate that the insights and optimizations derived from simulations are applicable in practical scenarios, reinforcing the reliability of the simulation model for evaluating and predicting performance in real-world applications.

6.5 Discussion

This chapter presents a method to generate inspection trajectories for laser profilometric sensors using Reinforcement Learning (RL) techniques. The objective was to improve the scanning process by dynamically adjusting the position and orientation of the sensor to maintain an optimal pose with respect to the surface of the inspected part.

A simulated environment that reproduces real-world conditions was used, as developed in previous work. The state space was defined by the position and orientation of the sensor (usually the robot end-effector), which allowed generalization and adaptability to various robotic configurations. Additional parameters, such as mean profile distance, steering angle, and spacing between consecutive scans, were incorporated to provide a global understanding of the inspection process.

The action space was designed to include relative increments in both sensor position and tilt angle, allowing for precise adjustments and smooth movements. The reward function was constructed with three key components: the distance between the sensor and the surface, the alignment with the surface normal, and the spacing between scans.

Experiments conducted in the simulated environment validated the ability of the RL model to adapt to various parts and maintain optimal scan trajectories, even those that had not been encountered during training. The method was also tested with a UR3e robotic arm in a real scenario, where an optimized trajectory, generated offline from a CAD model, was successfully executed. This demonstrated that the method can produce accurate, high-quality inspection trajectories that ensure effective surface coverage.

In the real-world validation tests of the scanning trajectories, we used smaller pieces that required only a single linear scanning path, eliminating the need for a

boustrophedon pattern. This choice was dictated by the physical limitations of the UR3e robotic arm, which has a relatively restricted reach. Given the limited range of the UR3e, it was not feasible to scan larger surfaces that would necessitate multiple passes. Consequently, we opted for a smaller and more suitable piece that allowed for effective validation within the operational capabilities of the available equipment. This decision ensures that, although the UR3e cannot cover extensive surfaces, the methodology and trajectory optimization principles we developed are applicable and verifiable in a controlled and representative environment.

Despite their smaller size, the real pieces chosen for the tests have sufficient geometric diversity to validate our proposed methodology. Their varied surface features ensured that the trajectory optimization and scanning techniques could be robustly tested. If we were to work with a larger piece, a boustrophedon scanning pattern would simply be employed. In such cases, our reinforcement learning model would be applied to each pass within the boustrophedon path, just as it was used in the simulated experiment with the car door. This approach ensures that our methods are versatile and can be adapted to both small and large-scale scanning tasks.

The scanning trajectories generated by our approach are designed to be highly versatile and adaptable to any robotic system with sufficient reach. This versatility arises from the fact that the trajectory increments in both position and orientation are small and precise enough to be executed by any industrial robot. When these incremental commands are input into the robot's control software, it calculates the necessary joint movements to achieve the specified end-effector positions.

The dynamic limitation of actions allows for adaptation to different types of applications. For example, in our experiments, backward movement was permitted, but depending on the type of robotic system or the specific part being inspected, this might not always be possible. This flexibility is useful for adjusting the approach to fit various inspection scenarios.

In our experiments, we used the same RL model, originally trained on a generic part, to plan the scanning trajectories. However, this approach can be further refined. The RL model could be retrained with a part that has characteristics more closely aligned with the specific piece being inspected, allowing for more adaptable and precise trajectory planning. By tailoring the model to the particular features of different parts, the system could achieve improved accuracy and efficiency in inspection tasks, making it more versatile across a range of applications.

It is important to note that when inspecting large parts, such as a car door, there is a significant risk of moving outside the sensor's working range. This challenge is

particularly relevant in industries like automotive manufacturing, where large components such as car doors, hoods, and other body panels are routinely scanned. This issue is addressed by adapting the scanning process to the specific geometry of each part, as demonstrated in experiments involving the scanning of a car door.

The advantage of the RL method lies in its ability to adapt the scanning path to complex geometries, which helps to manage errors more effectively and ensures accurate and reliable inspection outcomes. In contrast, traditional scanning methods are heavily reliant on the initial positioning and alignment of the part. If the part is not perfectly aligned or has significant depth variations, traditional approaches may struggle to compensate, leading to an accumulation of errors throughout the process. Additionally, scanning areas that deviate significantly from the horizontal plane can present challenges for these methods. By dynamically adjusting to the actual shape of the part, the RL method minimizes these errors, providing a more robust and accurate inspection.

Figure 6.24 highlights the importance of focusing on areas with pronounced curvature, where defects such as cracks commonly occur. A zoomed-in section of the global scan shows a bump-type defect introduced in such a high-curvature region. The RL-optimized trajectory achieved not only a higher point density but also improved sensor orientation and distance in these critical areas compared to the simple trajectory. This enhancement in both point density and sensor positioning improves defect visualization and detection, demonstrating the RL-based approach's effectiveness in identifying issues in defect-prone regions.

One limitation when transitioning from simulation to reality is the need for accurate initial alignment of the part. For effective scanning, the part must be positioned similarly to how it was in the simulation to ensure consistent results. Although this calibration step can introduce some variability, it is generally manageable. In production environments, part positions are often calibrated relative to the sensors and placed in consistent starting positions. This calibration requirement aligns with standard industry practices and does not significantly impact the overall feasibility of the proposed methodology.

Chapter 7

Conclusions and Future Work

In this final chapter, we present the conclusions of the research conducted in this doctoral thesis and outline potential future work. Additionally, the contributions and key publications resulting from the research carried out during the period of this thesis will also be detailed.

7.1 Discussion and Final Conclusions

This thesis responds to the challenges of industrial inspection using profilometric sensors by introducing a comprehensive framework that addresses critical aspects of surface defect detection and inspection trajectory optimization. Central to this framework is an advanced inspection simulator that accurately simulates scans on CAD models, taking into account sensor parameters and incorporating realistic noise simulations such as speckle.

In addition to simulating scans, our approach involves directly deforming CAD models to replicate defects such as bumps, peaks, and cracks. This methodology enables the creation of diverse and realistic 3D scan databases essential for training and validating advanced detection algorithms.

Moreover, our framework enhances precision by optimizing scanning trajectories through Reinforcement Learning. By dynamically adjusting sensor positions and orientations during scanning, we minimize relative positioning errors and ensure efficient coverage of inspected surfaces.

By integrating detailed scan simulation, realistic defect modeling, and trajectory optimization, our research bridges significant gaps in the literature. This unified approach aims to streamline development processes, reduce costs, and enhance the accuracy and efficiency of industrial inspection systems. Such advancements are crucial for bolstering competitiveness in the context of Industry 4.0, where digitalization and automation are pivotal in improving production processes and meeting stringent quality standards.

Recognizing the intrinsic limitations of our approach is vital, especially where the precision and quality of simulated defects are closely linked to the quality of the 3D model utilized. In our study, we use STL models, which are prevalent in the manufacturing sector. However, STL models consist of triangular meshes, and the size and number of these triangles are critical during defect integration.

Simulating measurements from CAD models using profilometric sensors presents a challenge: the sensor's resolution often exceeds that of the triangular mesh. Consequently, simulated measurements generate profiles composed of small straight segments corresponding to each triangle face in the mesh. While suitable for rough 3D reconstructions, this method proves inadequate for simulating sensors that operate at resolutions in the range of a few micrometers.

Moreover, this challenge extends to defect simulation. Although we implement a process to subdivide and refine these triangles within defect zones, excessively large original triangles relative to the defect size may compromise the detailed representation of the product's geometry. Therefore, careful consideration of the 3D model's quality is essential to ensure precise defect simulation during synthetic data generation.

To overcome this challenge and ensure realistic scan simulations, we introduce sensor measurement noise and speckle using a combination of Gaussian and Perlin noise. This mitigates the limitations posed by low-resolution 3D models and ensures a more authentic representation of surface characteristics. Our results demonstrate that the simulated noise closely resembles real-world sensor noise, further enhancing the fidelity of our simulations.

This high fidelity in simulation plays a crucial role in the development of AI models designed to operate effectively in real-world scenarios. By accurately simulating both defects and scan noise, our approach ensures that the generated data not only enhances the training of inspection algorithms but also extends its utility to other related tasks. These synthetic datasets can be utilized for tasks such as algorithm validation, sensitivity analysis, and optimization of inspection parameters, thereby

contributing to the broader advancement of automated quality control systems in manufacturing and beyond. This comprehensive approach addresses the challenge of realism in 3D datasets, originally highlighted in [23] and discussed in the Introduction, reinforcing the robustness and applicability of our generated data across various practical inspection applications.

Moreover, the tool's capability to produce labeled data stands out as a notable advantage. As emphasized earlier in the introduction, the data labeling process is known to be costly and labor-intensive. However, by automating the labeling process, this approach simplifies this aspect, saving time and resources while facilitating the development of more efficient defect detection algorithms and dataset generation.

Training a machine learning model with synthetic data instead of real data offers significant advantages, particularly in industrial contexts where obtaining real data can be costly and impractical. Firstly, generating synthetic data is much faster and cheaper compared to collecting real data, which requires constant facility operation and incurs high operational costs. For instance, while collecting real data for rare defects in an industrial setting may take a year, artificial defects can be generated within weeks. Secondly, synthetic data allows for simulating a wide variety of scenarios and rare defects, ensuring that the model is well-trained to recognize even the most unusual anomalies, something that may not be feasible with real data due to its scarcity. Additionally, using synthetic data avoids issues related to data privacy and security, which can be a concern in industries with strict regulations. In summary, synthetic data provides a more flexible and controllable path for model training, resulting in faster and less expensive development cycles.

This thesis presents a method for generating inspection trajectories for laser profilometric sensors using Reinforcement Learning (RL) techniques. The goal was to enhance the scanning process by adjusting the sensor's position and orientation to keep it optimally aligned with the surface of the inspected part. The approach was tested in a simulated environment designed to reflect real-world conditions, which helped in adapting the model to various robotic setups and part geometries. Key parameters like mean profile distance, incidence angle, and scan spacing were incorporated to offer a thorough understanding of the inspection process.

The RL model was validated through both simulations and practical tests using a ur3e robotic arm. In both cases, the results were positive, demonstrating that the RL-optimized trajectories effectively improved the scanning process. Additionally, the method's design, which operates in increments of sensor position and orientation (typically the robot's end-effector), allows for generalization across various types of robots. This adaptability makes the approach versatile and applicable to different

robotic systems.

One of the main benefits of the RL-based approach is its flexibility in handling different scanning scenarios. For instance, the method performed well in areas with high curvature, where defects are more likely to appear. The RL trajectory achieved better point density and sensor positioning compared to a straightforward scanning path, improving defect detection in these critical areas.

A practical consideration when applying the method in real-world settings is the need for accurate initial alignment of the part. While this calibration step can introduce some variability, it is a common practice in industry where parts are often calibrated relative to sensors. Overall, the RL-based method has shown to be adaptable and effective, making it a viable option for improving inspection processes in various scenarios.

In conclusion, this thesis presents a comprehensive framework for enhancing industrial inspection processes. By integrating advanced simulation techniques, realistic defect modeling, and trajectory optimization through RL, this work addresses key challenges in defect detection and inspection efficiency. The developed methodologies and tools not only improve scanning precision but also offer valuable insights into adapting inspection processes for various industrial applications.

During the thesis, the company CIN Advanced Systems [114] provided valuable data for experiments, highlighting the practical relevance of the research. CIN Advanced Systems specializes in artificial vision solutions for defect detection in production lines. Their machine vision systems typically combine advanced 3D laser sensors, 2D cameras, and robotic systems to move these sensors across a range of components, including stamped, machined, and cast parts, as well as complex textured surfaces.

The tools and methodologies developed in this thesis have the potential for real-world industrial applications in companies like CIN Systems. By integrating these advanced inspection systems, such companies could enhance their capabilities for inline defect detection, especially in complex applications.

CIN Systems currently rely heavily on applying machine learning techniques to high-resolution scans obtained from laser triangulation sensors to detect defects in production lines. This approach necessitates large, labeled datasets, which is where the contributions of this thesis become particularly valuable. The combination of a laser triangulation sensor simulator that integrates the Speckle effect with a method for generating surface defects in 3D models offers a powerful solution. By using

these tools together, companies can generate extensive labeled datasets that are crucial for training and refining their machine learning models. This capability not only enhances the accuracy and reliability of defect detection but also addresses the significant challenge of sourcing large, high-quality datasets.

Moreover, trajectory planning is often done manually, a process that can be time-consuming and less efficient. The Reinforcement Learning-based trajectory planning method developed in this thesis presents a significant improvement, enabling the automatic design of optimal scanning paths. This method ensures comprehensive and efficient coverage of complex surfaces, reduces inspection time, and ultimately increases the effectiveness of inspection processes.

By adopting these advanced tools, companies like CIN Systems could simplify their development processes, improve the accuracy of defect detection, and make their inspection systems more efficient in practical industrial applications.

7.2 Thesis Contributions

The main contributions of this thesis can be summarized as follows:

1. **Development of a simulator that integrates the Speckle effect**: A laser triangulation sensor simulator has been developed that incorporates the Perlin noise technique to include the Speckle effect in simulations. This simulator provides a useful tool for designing and evaluating real-world applications, allowing the testing of data processing and analysis algorithms before implementing them in physical systems. This can lead to more effective solutions and savings in development time and costs.

2. **Simulation of surface defects in 3D models**: A method for generating surface defects in 3D models using FFD is presented. It is important to note that defects can be inserted in any region of the model and with any orientation, including problematic areas with abrupt changes in geometry. The proposed technique for defect insertion involves modeling them as height maps originating from a plane, defining their size and shape.

   Additionally, three common types of defects are parameterized as default options: bumps, peaks, and cracks, with the flexibility to modify their dimensions, allowing the incorporation of a wide range of small and large deformations typically found during the manufacturing process of various products.

The ability to generate surface defects in 3D models can be employed in different applications: validating different algorithms, training defect detection models, optimizing inspection systems to reduce costs and time, or providing flexibility in controlling defect characteristics.

3. **Generation of labeled surface defect databases**: Labeled defect databases have been generated containing high-resolution simulated scans with 3D information of surface defects. These databases are fundamental for the training and evaluation of defect detection algorithms, as they provide a realistic and diversified dataset that represents a wide range of possible defects in manufactured products, addressing the problem of the scarcity of databases with these characteristics.

4. **Inspection trajectory planning using Reinforcement Learning**: Inspection trajectory planning using Reinforcement Learning allows the design of optimal strategies to guide the sensor movement along the surface of the part to be inspected. Using machine learning techniques, the system can autonomously learn the best actions to take at each point on the surface to maximize scanning efficiency. This ensures complete and uniform coverage of the entire surface of the part, even in hard-to-reach areas or with complex geometries, thus optimizing the inspection process and reducing the time required to complete the task.

5. The development offers an integral solution by bringing together multiple functionalities in a single framework. It allows simulating precise scans, generating surface defects, and planning inspection trajectories, all from the product's CAD model. This integration significantly simplifies the design and evaluation process of automated inspection systems, eliminating the need to use multiple scattered tools. Furthermore, having everything in the same environment facilitates collaboration between different teams and rapid iteration in the development of defect detection algorithms.

Furthermore, during the course of this thesis, two practical studies were conducted in real-world environments, utilizing different robotic systems for the inspection of various products. Although these studies are not the primary contributions of the thesis, they provide valuable insights into the application of inspection technologies in industrial settings:

Use of Mobile Robots for Heavy Steel Plates Inspection: A series of algorithms were developed to enable the inspection and repair of defects in heavy steel plates. Key efforts focused on robot localization in confined environments,

inspection and repair trajectory planning, and reconstructing the inspected sheet to facilitate defect detection.

The inspection process employed RGB-D cameras for 2D surface reconstruction of the metal sheets, capturing detailed surface features and defects. However, this methodology is designed to be flexible and adaptable, allowing for the incorporation of other sensing technologies, such as laser sensors or profilometers. These alternative sensors could enhance the precision of surface measurements or provide specific advantages depending on the application.

For instance, integrating a laser sensor or profilometer could improve the accuracy of surface contour and texture data, enabling more precise defect detection. Moreover, the inspection trajectories can be tailored to meet the specific requirements of the chosen sensing tool, optimizing the process based on the sensor's characteristics and the surface being inspected. This adaptability ensures that the inspection process can be customized to achieve optimal results across various scenarios.

This work was published in [115] and [116].

Use of Drones for Pipeline Inspection: During a 3-month research stay at the University of Napoli Federico II, a vision-based system was developed for the surface inspection of industrial pipelines using drones. The system was validated through both simulations and real-world experiments, demonstrating its capability to accurately follow the central axis of industrial pipelines and detect surface defects.

While the current system utilizes depth sensors for defect detection, future iterations could integrate more advanced sensors to detect a broader range of defects. For example, incorporating the laser triangulation sensors discussed in this thesis could significantly enhance the system's ability to detect smaller defects with greater accuracy. These sensors would provide detailed measurements of surface irregularities and anomalies, complementing the existing vision-based approach.

This work was published in [117].

7.3 Future work

The work developed during the present investigation suggests new lines of research and future work:

In terms of defect simulation, research in developing more advanced mathematical models to simulate a broader variety of defects represents a key opportunity to improve the quality and diversity of databases used in defect detection. These more sophisticated models could enable the simulation of complex and realistic defects found in a wide range of manufactured products, from electronic components to metal structures.

By increasing the diversity and complexity of the databases, more robust and generalizable defect detection models could be trained, significantly enhancing their ability to identify defects across different industrial scenarios. Moreover, the ability to simulate a wider variety of defects would pave the way for combining real and simulated data in the training of artificial intelligence algorithms. This would improve the algorithms' ability to generalize and adapt to new conditions and situations, resulting in more effective and versatile defect detection systems.

Additionally, the development of more advanced mathematical models could facilitate the research and development of new defect detection techniques. For instance, by precisely simulating specific defects, more specialized and effective detection approaches for those particular types of defects could be explored. This could lead to significant advancements in the efficiency and accuracy of automated inspection systems across a wide variety of industrial applications.

Exploring trajectory generation using reinforcement learning offers a promising direction for improving automated inspection systems. Future research could focus on developing more advanced algorithms that integrate additional information to guide the sensor's movement with greater precision and efficiency. For example, deep learning techniques could be utilized to identify complex patterns in the data, enhancing real-time trajectory planning. Incorporating more global rewards that assess the system's overall performance in terms of surface coverage, defect detection, and scanning efficiency could also be beneficial.

Additionally, increasing the dimensionality of the system could allow for the inspection of larger and more complex parts. This might involve expanding the degrees of freedom for sensor movement and adapting trajectory planning techniques to manage data across multiple dimensions. Exploring alternative trajectory types, such as spirals or zigzags, could offer more efficient scanning strategies in certain contexts.

A more detailed investigation into reward design could help improve trajectory optimization. By exploring how different reward structures influence the system's performance, it might be possible to enhance how effectively the reinforcement learn-

ing guides the sensor. Experimenting with various reward functions could help balance factors like surface coverage and defect detection accuracy. Additionally, trying out different reinforcement learning algorithms, such as policy gradient methods or actor-critic models, could offer insights into which techniques work best for specific inspection scenarios.

Chapter 8

Conclusiones y trabajo futuro

En este capítulo final, presentamos las conclusiones de la investigación realizada en esta tesis doctoral y plantemoas posibles líneas de trabajo futuro. Además, se detallarán las contribuciones y publicaciones clave derivadas de la investigación llevada a cabo durante el periodo de esta tesis.

8.1 Discusion y Conclusiones Finales

Esta tesis aborda los desafíos de la inspección industrial mediante sensores profilométricos a través del desarrollo de un *framework* que trata aspectos clave en la detección de defectos en superficies y la optimización de trayectorias de inspección. En el centro de este *framework* se encuentra un simulador avanzado que permite reproducir de forma precisa escaneos sobre modelos CAD, considerando los parámetros del sensor e incorporando simulaciones realistas de ruido, como el *speckle*.

El enfoque planteado también incluye la deformación directa de modelos CAD para imitar defectos como abolladuras, picos y grietas, lo que posibilita la generación de bases de datos de escaneos 3D variadas y realistas, esenciales para el entrenamiento y validación de algoritmos avanzados de detección.

Además, se ha propuesto una metodología para optimizar las trayectorias de escaneo mediante el uso de Aprendizaje por Refuerzo. Esta técnica ajusta de forma dinámica las posiciones y orientaciones del sensor durante el proceso de escaneo, con el objetivo de minimizar los errores de posicionamiento relativo y asegurar una

cobertura eficiente de las superficies inspeccionadas.

La integración de simulaciones detalladas de escaneo, modelado realista de defectos y optimización de trayectorias permite reducir las brechas existentes en la literatura científica. Este enfoque tiene como objetivo agilizar los procesos de desarrollo, reducir costes y mejorar la precisión y eficiencia de los sistemas de inspección industrial, contribuyendo así a la competitividad en el contexto de la Industria 4.0, donde la digitalización y automatización juegan un papel esencial en la mejora de los procesos de producción y en el cumplimiento de los estándares de calidad.

Es fundamental reconocer las limitaciones intrínsecas de este enfoque, especialmente cuando la precisión y calidad de los defectos simulados dependen en gran medida de la calidad del modelo 3D utilizado. En este estudio se emplean modelos STL, que son comunes en el sector manufacturero. Sin embargo, los modelos STL se componen de mallas triangulares, y el tamaño y número de estos triángulos son factores críticos durante la integración de defectos.

Simular mediciones a partir de modelos CAD utilizando sensores profilométricos presenta un desafío, ya que la resolución del sensor suele superar la de la malla triangular. Como resultado, las mediciones simuladas generan perfiles compuestos por pequeños segmentos rectos correspondientes a las caras de los triángulos en la malla. Aunque este método es adecuado para reconstrucciones 3D aproximadas, resulta insuficiente para simular sensores que operan con resoluciones de unos pocos micrómetros.

Este problema también afecta a la simulación de defectos. Aunque se implementa un proceso para subdividir y refinar los triángulos en las zonas defectuosas, si los triángulos originales son excesivamente grandes en relación con el tamaño del defecto, se puede comprometer la representación detallada de la geometría del producto. Por ello, es necesario considerar con cuidado la calidad del modelo 3D para asegurar una simulación precisa de defectos durante la generación de datos sintéticos.

Para mitigar esta limitación y garantizar simulaciones de escaneo realistas, se introduce ruido de medición del sensor y *speckle* mediante una combinación de ruido gaussiano y de Perlin. Esto ayuda a compensar las deficiencias de los modelos 3D de baja resolución y permite una representación más fiel de las características de la superficie. Los resultados obtenidos muestran que el ruido simulado se asemeja al ruido real de los sensores, mejorando así la fidelidad de las simulaciones.

La alta fidelidad en la simulación desempeña un papel clave en el desarrollo de modelos de inteligencia artificial diseñados para funcionar eficazmente en escenarios

del mundo real. Al simular con precisión tanto los defectos como el ruido de los escaneos, se garantiza que los datos generados no solo mejoren el entrenamiento de los algoritmos de inspección, sino que también sean útiles para otras tareas relacionadas. Estos conjuntos de datos sintéticos pueden emplearse en la validación de algoritmos o al análisis de sensibilidad y optimización de parámetros de inspección, contribuyendo así al avance de los sistemas de control de calidad automatizados en el ámbito industrial y más allá. Este enfoque integral aborda el desafío del realismo en los conjuntos de datos 3D, señalado originalmente en [23] y mencionado en la introducción, reforzando la solidez y aplicabilidad de los datos generados en diversas aplicaciones de inspección.

Además, la capacidad de la herramienta para producir datos etiquetados se destaca como una ventaja notable. Como se mencionó anteriormente en la introducción, el proceso de etiquetado de datos es conocido por ser costoso y laborioso. Sin embargo, al automatizar este proceso, se simplifica esta tarea, ahorrando tiempo y recursos, y facilitando el desarrollo de algoritmos más eficientes para la detección de defectos y la generación de bases de datos.

El uso de datos sintéticos para entrenar un modelo de aprendizaje automático tiene grandes ventajas, especialmente en entornos industriales donde conseguir datos reales puede ser caro y complicado. Primero, generar datos sintéticos es mucho más rápido y barato que recoger datos reales, lo que suele implicar mantener las instalaciones en funcionamiento y asumir altos costes operativos. Por ejemplo, mientras que recopilar datos reales sobre defectos poco comunes en un entorno industrial podría llevar un año, los defectos artificiales pueden generarse en solo unas semanas. Además, los datos sintéticos permiten simular una gran variedad de escenarios y defectos raros, garantizando que el modelo esté bien entrenado para detectar incluso las anomalías más inusuales, algo que no siempre es posible con datos reales debido a su escasez. También, el uso de datos sintéticos evita problemas de privacidad y seguridad, que son importantes en industrias con normativas estrictas. En resumen, los datos sintéticos ofrecen una forma más flexible y controlada de entrenar modelos, lo que acelera y abarata los ciclos de desarrollo.

Esta tesis presenta un método para generar trayectorias de inspección para sensores laser profilométricos utilizando técnicas de Aprendizaje por Refuerzo (RL). El objetivo fue mejorar el proceso de escaneo ajustando la posición y orientación del sensor para mantenerlo alineado de manera óptima con la superficie de la pieza inspeccionada. El enfoque se probó en un entorno simulado que reflejaba condiciones del mundo real, lo que ayudó a adaptar el modelo a diferentes configuraciones robóticas y geometrías de piezas. Se incorporaron parámetros clave como la distancia media del perfil, el ángulo de incidencia y el espaciado de los escaneos para ofrecer

una comprensión detallada del proceso de inspección.

El modelo de RL fue validado tanto en simulaciones como en pruebas prácticas utilizando un brazo robótico ur3e. En ambos casos, los resultados fueron positivos, demostrando que las trayectorias optimizadas con RL mejoraron el proceso de escaneo. Además, el diseño del método, que opera en incrementos de posición y orientación del sensor (generalmente el efector final del robot), permite su generalización a distintos tipos de robots. Esta adaptabilidad hace que el enfoque sea versátil y aplicable a diferentes sistemas robóticos.

Uno de los principales beneficios del enfoque basado en RL es su flexibilidad para manejar distintos escenarios de escaneo. Por ejemplo, el método mostró un buen rendimiento en áreas con alta curvatura, donde es más probable que aparezcan defectos. La trayectoria optimizada por RL logró una mayor densidad de puntos y un mejor posicionamiento del sensor en comparación con una ruta de escaneo simple, mejorando la detección de defectos en estas áreas críticas.

Un aspecto práctico a considerar al aplicar este método en entornos reales es la necesidad de una alineación inicial precisa de la pieza. Aunque este paso de calibración puede introducir cierta variabilidad, es una práctica común en la industria, donde las piezas a menudo se calibran en relación con los sensores. En general, el método basado en RL ha demostrado ser adaptable y eficaz, lo que lo convierte en una opción viable para mejorar los procesos de inspección en diversas situaciones.

En conclusión, esta tesis presenta un *framework* integral para mejorar los procesos de inspección industrial. Al integrar técnicas avanzadas de simulación, modelado realista de defectos y optimización de trayectorias mediante Aprendizaje por Refuerzo (RL), este trabajo aborda desafíos clave en la detección de defectos y la eficiencia de la inspección. Las metodologías y herramientas desarrolladas no solo mejoran la precisión del escaneo, sino que también ofrecen información valiosa para adaptar los procesos de inspección a diversas aplicaciones industriales.

Durante la realización de esta tesis, la empresa CIN Advanced Systems [114] proporcionó datos valiosos para los experimentos, lo que resalta la relevancia práctica de la investigación. CIN Advanced Systems se especializa en soluciones de visión artificial para la detección de defectos en líneas de producción. Sus sistemas de visión combinan sensores láser 3D avanzados, cámaras 2D y sistemas robóticos para mover estos sensores en diferentes tipos de componentes, incluidos piezas estampadas, mecanizadas y fundidas, así como superficies con texturas complejas.

Las herramientas y metodologías desarrolladas en esta tesis tienen un potencial

de aplicación real en entornos industriales como los de CIN Systems. La integración de estos sistemas de inspección avanzados podría mejorar las capacidades para la detección de defectos en línea, especialmente en aplicaciones complejas.

Actualmente, CIN Systems depende en gran medida del uso de técnicas de aprendizaje automático aplicadas a escaneos de alta resolución obtenidos mediante sensores de triangulación láser para la detección de defectos en líneas de producción. Este enfoque requiere grandes conjuntos de datos etiquetados, lo que subraya el valor de las contribuciones de esta tesis. La combinación de un simulador de sensor de triangulación láser que integra el efecto *Speckle* con un método para generar defectos en superficies de modelos 3D ofrece una solución potente. El uso conjunto de estas herramientas permite a las empresas generar amplios conjuntos de datos etiquetados, fundamentales para entrenar y mejorar los modelos de aprendizaje automático. Esta capacidad no solo aumenta la precisión y confiabilidad en la detección de defectos, sino que también aborda el reto de obtener grandes conjuntos de datos de alta calidad.

Además, la planificación de trayectorias suele realizarse de manera manual, un proceso que puede ser lento y menos eficiente. El método de planificación de trayectorias basado en RL desarrollado en esta tesis representa una mejora significativa, ya que permite el diseño automático de rutas de escaneo óptimas. Este método asegura una cobertura completa y eficiente de superficies complejas, reduce el tiempo de inspección y, en última instancia, incrementa la efectividad de los procesos de inspección.

La adopción de estas herramientas avanzadas podría simplificar los procesos de desarrollo, mejorar la precisión en la detección de defectos y hacer que los sistemas de inspección sean más eficientes en aplicaciones industriales prácticas para empresas como CIN Systems.

8.2 Contribuciones de la Tesis

Las principales aportaciones de esta tesis pueden resumirse como sigue:

1. **Desarrollo de un simulador que integra el ruido *Speckle***: Se ha desarrollado un simulador de sensores de triangulación láser que incorpora la técnica de ruido Perlin para incluir el efecto *Speckle* en las simulaciones. Este simulador se convierte en una herramienta útil para el diseño y la evaluación de

aplicaciones en el mundo real, permitiendo la prueba de algoritmos de procesamiento y análisis de datos antes de su implementación en sistemas físicos. Esto puede llevar a soluciones más efectivas y a ahorros en tiempo y costos de desarrollo.

2. **Simulación de defectos superficiales en modelos 3D**: Se presenta un método para generar defectos en superficies de modelos 3D utilizando *Free Form Deformation* (FFD). Es importante destacar que los defectos pueden insertarse en cualquier región del modelo y con cualquier orientación, incluyendo áreas problemáticas con cambios abruptos en la geometría. La técnica propuesta para la inserción de defectos implica modelarlos como mapas de altura originados desde un plano, definiendo su tamaño y forma.

   Además, se parametrizan tres tipos comunes de defectos como opciones predeterminadas: bollos, picos y grietas, con la flexibilidad de modificar sus dimensiones, lo que permite la incorporación de una amplia gama de deformaciones pequeñas y grandes, típicas durante el proceso de fabricación de diversos productos.

   La capacidad de generar defectos en superficies de modelos 3D puede emplearse en diferentes aplicaciones: validación de diversos algoritmos, entrenamiento de modelos de detección de defectos, optimización de sistemas de inspección para reducir costes y tiempo, o proporcionando flexibilidad en el control de las características de los defectos.

3. **Generación de bases de datos etiquetadas de defectos en superficies**: Se han generado bases de datos de defectos etiquetados que contienen escaneos simulados de alta resolución con información 3D sobre defectos en superficies. Estas bases de datos son fundamentales para el entrenamiento y la evaluación de algoritmos de detección de defectos, ya que proporcionan un conjunto de datos realista y diversificado que representa una amplia gama de posibles defectos en productos manufacturados, abordando el problema de la escasez de bases de datos con estas características.

4. **Planificación de trayectorias de inspección utilizando Aprendizaje por Refuerzo**: La planificación de trayectorias de inspección mediante Aprendizaje por Refuerzo permite diseñar estrategias óptimas para guiar el movimiento del sensor a lo largo de la superficie de la pieza a inspeccionar. Utilizando técnicas de aprendizaje automático, el sistema puede aprender de manera autónoma las mejores acciones a seguir en cada punto de la superficie para maximizar la eficiencia del escaneo. Esto garantiza una cobertura completa y uniforme de toda la superficie de la pieza, incluso en áreas de difícil acceso o con

geometrías complejas, optimizando así el proceso de inspección y reduciendo el tiempo necesario para completar la tarea.

5. La tesos ofrece una solución integral al reunir múltiples funcionalidades en un solo *framework*. Permite simular escaneos precisos, generar defectos en superficies y planificar trayectorias de inspección, todo a partir del modelo CAD del producto. Esta integración simplifica significativamente el proceso de diseño y evaluación de sistemas de inspección automatizados, eliminando la necesidad de utilizar herramientas dispersas. Además, tener todo en el mismo entorno facilita la colaboración entre diferentes equipos y la iteración rápida en el desarrollo de algoritmos de detección de defectos.

Además, durante el transcurso de esta tesis, se llevaron a cabo dos estudios prácticos en entornos reales, utilizando diferentes sistemas robóticos para la inspección de diversos productos. Aunque estos estudios no constituyen las contribuciones principales de la tesis, proporcionan valiosas perspectivas sobre la aplicación de las tecnologías de inspección en entornos industriales:

Uso de robots móviles para la inspección de chapas de acero: Se desarrollaron una serie de algoritmos para permitir la inspección y reparación de defectos en chapa gruesa. Los esfuerzos clave se centraron en la localización del robot en entornos confinados, la planificación de trayectorias de inspección y reparación, y la reconstrucción de la chapa inspeccionada para facilitar la detección de defectos.

El proceso de inspección empleó cámaras RGB-D para la reconstrucción en 2D de las superficies de las chapas de acero, capturando características y defectos detallados de la superficie. Sin embargo, esta metodología está diseñada para ser flexible y adaptable, permitiendo la incorporación de otras tecnologías de detección, como sensores láser o profilómetros. Estos sensores alternativos podrían mejorar la precisión de las mediciones de superficie o proporcionar ventajas específicas según la aplicación.

Por ejemplo, integrar un sensor láser o un profilómetro podría mejorar la precisión de los datos sobre el contorno y la textura de la superficie, permitiendo una detección de defectos más precisa. Además, las trayectorias de inspección pueden ajustarse para cumplir con los requisitos específicos de la herramienta de detección elegida, optimizando el proceso en función de las características del sensor y la superficie a inspeccionar. Esta adaptabilidad asegura que el proceso de inspección pueda personalizarse para lograr resultados óptimos en diversos escenarios.

Este trabajo fue publicado en [115] y [116].

Uso de drones para la inspección de tuberías: Durante una estancia de investigación de 3 meses en la Universidad de Nápoles Federico II, se desarrolló un sistema basado en visión para la inspección de superficies de tuberías industriales utilizando drones. El sistema fue validado tanto mediante simulaciones como en experimentos reales, demostrando su capacidad para seguir con precisión el eje central de las tuberías industriales y detectar defectos en la superficie.

Aunque el sistema actual utiliza sensores de profundidad para la detección de defectos, futuras iteraciones podrían integrar sensores más avanzados para detectar una gama más amplia de defectos. Por ejemplo, la incorporación de los sensores de triangulación láser discutidos en esta tesis podría mejorar significativamente la capacidad del sistema para detectar defectos más pequeños con mayor precisión. Estos sensores proporcionarían mediciones detalladas de las irregularidades y anomalías de la superficie, complementando el enfoque basado en visión existente.

Este trabajo fue publicado en [117].

8.3 Trabajo Futuro

El trabajo desarrollado en esta investigación sugiere nuevas líneas de investigación y trabajo futuro:

En cuanto a la simulación de defectos, investigar el desarrollo de modelos matemáticos más avanzados para simular una variedad más amplia de defectos representa una oportunidad clave para mejorar la calidad y diversidad de las bases de datos utilizadas en la detección de defectos. Estos modelos más sofisticados podrían permitir la simulación de defectos complejos y realistas encontrados en una amplia gama de productos manufacturados, desde componentes electrónicos hasta estructuras metálicas.

Al aumentar la diversidad y complejidad de las bases de datos, se podrían entrenar modelos de detección de defectos más robustos y generalizables, mejorando significativamente su capacidad para identificar defectos en diferentes escenarios industriales. Además, la capacidad de simular una gama más amplia de defectos allanaría el camino para combinar datos reales y simulados en el entrenamiento de algoritmos de inteligencia artificial. Esto mejoraría la capacidad de los algoritmos para generalizar y adaptarse a nuevas condiciones y situaciones, resultando en sistemas de detección de defectos más efectivos y versátiles.

Además, el desarrollo de modelos matemáticos más avanzados podría facilitar la

investigación y el desarrollo de nuevas técnicas de detección de defectos. Por ejemplo, al simular de manera precisa defectos específicos, se podrían explorar enfoques de detección más especializados y efectivos para esos tipos particulares de defectos. Esto podría llevar a avances significativos en la eficiencia y precisión de los sistemas de inspección automatizados en una variedad de aplicaciones industriales.

Explorar la generación de trayectorias mediante aprendizaje por refuerzo ofrece una dirección prometedora para mejorar los sistemas de inspección automatizados. La investigación futura podría centrarse en desarrollar algoritmos más avanzados que integren información adicional para guiar el movimiento del sensor con mayor precisión y eficiencia. Por ejemplo, se podrían utilizar técnicas de aprendizaje profundo para identificar patrones complejos en los datos, mejorando la planificación de trayectorias en tiempo real. Incorporar recompensas más globales que evalúen el rendimiento general del sistema en términos de cobertura de superficie, detección de defectos y eficiencia del escaneo también podría ser beneficioso.

Además, aumentar la dimensionalidad del sistema podría permitir la inspección de partes más grandes y complejas. Esto podría implicar expandir los grados de libertad para el movimiento del sensor y adaptar las técnicas de planificación de trayectorias para gestionar datos en múltiples dimensiones. Explorar tipos de trayectorias alternativas, como espirales o en zig-zag, podría ofrecer estrategias de escaneo más eficientes en ciertos contextos.

Una investigación más detallada sobre el diseño de recompensas podría ayudar a mejorar la optimización de trayectorias. Al explorar cómo diferentes estructuras de recompensas influyen en el rendimiento del sistema, podría ser posible mejorar la eficacia con la que el aprendizaje por refuerzo guía al sensor. Experimentar con diversas funciones de recompensa podría ayudar a equilibrar factores como la cobertura de superficie y la precisión en la detección de defectos. Además, probar diferentes algoritmos de aprendizaje por refuerzo, como métodos de gradiente de políticas o modelos actor-crítico, podría ofrecer ideas sobre qué técnicas funcionan mejor para escenarios específicos de inspección.

Chapter A

Publications

This annex lists all publications related to this research. They are divided into two categories: 1) Publications in journals; and 2) Communications to Congress.

A part from the listed publications, the content of the chapter 5 is currently being considered for publication in the Journal of Intelligent Manufacturing by Springer. The manuscript, titled "Realistic Defect Simulation in 3D Models for Defect Detection Using Machine Learning", authored by S. Roos-Hoefgeest, M. Roos-Hoefgeest, D. García Peña, I. Álvarez, and R. C. González, is currently in its third revision.

Also, another paper is being written based on the contents of chapter 6.

A.1 Publications in journals

sensors

MDPI

Article

Simulation of Laser Profilometer Measurements in the Presence of Speckle Using Perlin Noise

Sara Roos-Hoefgeest [1,*], Mario Roos-Hoefgeest [2], Ignacio Álvarez [1] and Rafael C. González [1]

[1] Department of Electrical, Computer Electronics and Systems Engineering, University of Oviedo, 33003 Oviedo, Spain; ialvarez@uniovi.es (I.Á); rcgonzalez@uniovi.es (R.C.G.)
[2] Desarrollo de Soluciones Integrales Plus S.L., 33211 Gijón, Spain; mroos@cinsystems.es
* Correspondence: roossara@uniovi.es

Abstract: In the manufacturing industry, inspection systems play a crucial role in ensuring product quality. High-resolution profilometric sensors have become increasingly popular for inspection due to their ability to provide detailed surface information. However, the development and testing of inspection systems can be costly and time-consuming. This paper presents the development of a simulation of an inspection system using a high-resolution profilometric sensor. A geometrical and noise model is proposed to simulate the readings of any actual profilometric sensor. The model replicates the sensor's movement on the CAD model of the inspected part. The model incorporates the physical properties of the sensor and combines noise sources from sensor uncertainty and speckle noise induced by the roughness of the material. Our contribution lies in noise modeling. This work proposes a combination of Perlin noise to simulate the speckle noise and Gaussian noise for the uncertainty-related noise. Perlin noise is generated based on the surface roughness parameters of the inspected part. The accuracy of the simulation system is evaluated by comparing the simulated scans with real scans. The results highlight the ability to simulate real scans of different parts, using commercial sensor specifications and the CAD model of the inspected part.

Keywords: laser triangulation profilometry; Perlin noise; speckle; surface roughness; simulation; computer vision

Citation: Roos-Hoefgeest, S.; Roos-Hoefgeest, M.; Álvarez, I.; González, R.C. Simulation of Laser Profilometer Measurements in the Presence of Speckle Using Perlin Noise. *Sensors* **2023**, *1*, 0. https://doi.org/

Academic Editors: Shichang Du, Yiping Shao and Delin Huang

Received: 14 July 2023
Revised: 23 August 2023
Accepted: 29 August 2023
Published:

1. Introduction

As increasingly higher requirements are demanded from manufactured products, industries rely on inspection systems to guarantee that their products achieve the expected quality level. This is especially relevant in some industrial sectors, such as the aerospace and automotive sectors or precision parts manufacturing. These sectors demand a full inspection of their production that implies the acquisition of metrological information as well as the detection of other possible defects, such as cracks or irregularities on the product surface. This kind of superficial inspection may be carried out at an intermediate stage of the process to discard defective parts as soon as possible.

However, developing inspection systems can be costly and time-consuming, especially when dealing with high-precision sensors and complex geometries. Therefore, there is a need for CAD tools that can facilitate the development of industrial inspection applications [1,2]. These tools can optimize the design and reduce costs by providing initial data for algorithm development and evaluation. These tools allow for a simulation of the inspection process before actually implementing it on the production line. From the simulation data, it is possible to identify potential issues and refine the inspection process, ultimately improving product quality. To be useful, the simulator must accurately replicate the behavior of the inspection system using models that can be easily related to real hardware. Designers will use that tool to test its performance under different conditions to obtain an accurate and cost-effective solution while reducing the development time and costs.

Figure A.1: "Simulation of Laser Profilometer Measurements in the Presence of Speckle Using Perlin Noise". Paper published in MDPI's Sensors magazine [113] with the results of the chapter 4.

Realistic 3D Defect Simulation for Deep Learning-based Industrial Inspection Systems

Sara Roos-Hoefgeest[1], Mario Roos-Hoefgeest[2], Daniel García Peña[2], Ignacio Álvarez[1], Rafael C. González[1*]

[1*]Department of Electrical, Electronic, Communications and Systems Engineering, University of Oviedo, Oviedo, 33003, Spain.
[2]CIN Advanced Systems Group S.L., Gijón, 33211, Spain.

*Corresponding author(s). E-mail(s): rcgonzalez@uniovi.es;
Contributing authors: roossara@uniovi.es; mroos@cinsystems.es; dgarcia@cinsystems.es; ialvarez@uniovi.es;

Abstract

High-precision surface defect detection in manufactured products is a challenge for industrial producers. The current trend is to use dense 3D point clouds obtained with contactless sensors, such as Laser profilometric sensors, in order to achieve the high-precision dimensional requirements demanded; then, from the point clouds, classical machine-vision or modern machine-learning (ML) techniques are applied in order to outline the defects. In both cases, but more intensely for ML, large datasets including different shapes and defects are necessary to train the algorithms. Acquiring and labeling real defect datasets is challenging because of the random and infrequent nature of the defects, and the labor costs associated to manual detection and classification.

We present a method to create synthetic labeled datasets tailored for each application. Defects are modeled by an elevation map that defines their size and shape. The triangular mesh describing the object is modified using Free-Form Deformation techniques. Random noise is added to ensure variety in the dataset. We propose mathematical models to characterize three common types of superficial defects. To validate our proposal, synthetic depth images are obtained by simulating the 3D sensor acquisition process. Real and synthetic depth maps are compared; moreover, the results of basic YOLOv8 networks trained with our synthetic dataset and a real one are compared. Models were tested using a fresh dataset of real samples. Results showed that synthetic data can replace the scarce real data in the training phase, easing the development and tuning of inspection applications for new products.

Figure A.2: "Realistic Defect Simulation in 3D Models for Defect Detection Using Machine Learning", authored by S. Roos-Hoefgeest, M. Roos-Hoefgeest, D. García Peña, I. Álvarez, and R. C. González. Paper **currently in its third revision** with the results of the chapter 5.

A.2 Publications in Congress

2023 International Conference on Unmanned Aircraft Systems (ICUAS)
June 6-9, 2023 | Lazarski University, Warsaw, Poland

A Vision-based Approach for Unmanned Aerial Vehicles to Track Industrial Pipes for Inspection Tasks

Sara Roos-Hoefgeest[1], Jonathan Cacace[2], Vincenzo Scognamiglio[2], Ignacio Álvarez[1], Rafael C. González[1], Fabio Ruggiero[2] and Vincenzo Lippiello[2]

2023 International Conference on Unmanned Aircraft Systems (ICUAS) | 979-8-3503-1037-5/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICUAS57906.2023.10156565

***Abstract*— Inspecting and maintaining industrial plants is an important and emerging field in robotics. A particular case is represented by the inspection of oil and gas refinery facilities consisting of different long pipe racks to be inspected repeatedly. This task is costly in terms of human safety and operation costs due to the high altitude location in which the pipes are placed. In this domain, we propose a visual inspection system for unmanned aerial vehicles (UAVs), allowing the autonomous tracking and navigation of the center line of the industrial pipe. The proposed approach exploits a depth sensor to generate the control data for the aerial platform and, at the same time, highlight possible pipe defects. A set of simulated and real experiments in a GPS-denied environment have been carried out to validate the visual inspection system.**

I. Introduction

The transport of fluids like steam, heating water and oil or liquid chemicals is made through a network of pipelines. Pipelines are typically grouped in a steel-framed structure called pipe racks (see Fig. 1). Typically, pipe racks are laid between different units in any chemical processing or power plant and are placed on elevated locations to preserve the ground space of the plant used for operators' mobility. Pipelines must be regularly inspected to assess their external/internal status. Their damage can be detected as a weakening of the external covering or the corrosion of its structure (i.e., rust on the pipe surface). Besides, damaged pipes can cause dangerous situations like explosions or chemical incidents. Pipeline cracks in oil and gas companies produce financial loss and environmental pollution rather than heavy casualties. For this reason, the early detection of defective pipe sections plays a crucial role in preventing unnecessary loss faced by oil and gas companies, ensuring safe working conditions as well. However, pipe racks often extend for miles and are located on elevated structures. Visually inspecting all the sections of the pipes is an expensive and demanding task. In particular, manual inspection of pipelines can be done regularly, but it is time-consuming and unsafe in hazardous areas. In addition, expensive scaffolding should be assembled to allow operators to reach inspection points.

Fig. 1: Pipe racks structures

In this context, using unmanned aerial vehicles (UAVs) equipped with vision sensors represents a low-cost and reliable solution to perform similar inspection tasks [1], [2]. Aerial systems can follow the surface of the pipelines, processing the information captured from a vision sensor to detect eventual defects. Automating such a task is not trivial, and different challenges must be addressed. First, the drone should be able to see the pipe to inspect and consequently follow its shape, regulating its position and orientation to track the pipeline. At the same time, during the navigation of the UAV, pipe defects must be detected. Different flaws can be present in the pipe structure, both externally and internally. In the latter case, the internal structure of the pipe is corrupted, and its thickness decreases. Here, the defects are detected using conventional non-destructive testing (NDT) with ultrasonic probes in contact with the inspected surface. Differently, our work focuses on external corrosion flaws visible on the pipeline structures.

This work's main contributions are the definition of a computer vision technique to detect and characterize pipeline structures, a UAV navigation strategy to track the pipes' surface and a method to highlight pipe defects based on vision data autonomously. A simulated case study using different pipe shapes has been carried out based on ROS and Gazebo [3], [4] simulator. Preliminary real-world experiments in a GPS-denied environment have also been performed to demonstrate approach effectiveness.

The remainder of the paper is organized as follows. In Section II, a brief overview of related works is presented while, in Section III, the sensor elaboration module to detect and extract salient information on the pipelines is discussed along with the UAV navigation controller strategy. Section IV describes the system architecture and, finally, Section V presents simulated and real-world experiments.

The research leading to these results has been supported by the AERIAL-CORE project, European Union's Horizon 2020 research and innovation programme under Grant Agreement No 871479; the AERO-TRAIN project, European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 953454; the scholarship under the "Severo Ochoa" program for predoctoral research and teaching with Ref: PA-20-PF-BP19-067, financed by Asturias Regional Government. The authors are solely responsible for its content.

[1]Department of Electrical, Computer Electronics and Systems Engineering, University of Oviedo, Asturias, Spain
[2]PRISMA Lab and CREATE Consortium, Department of Electrical Engineering and Information Technology, University of Naples Federico II, Naples, Italy.

Figure A.3: "A Vision-based Approach for Unmanned Aerial Vehicles to Track Industrial Pipes for Inspection Tasks". Article published and presented in the international congress ICUAS 2023 [117]

Mobile robot localization in industrial environments using a ring of cameras and ArUco markers

Sara Roos-Hoefgeest
ISA
Universidad de Oviedo
Gijón, Spain
roossara@uniovi.es

Ignacio Alvarez Garcia
ISA
Universidad de Oviedo
Gijón, Spain
ialvarez@uniovi.es

Rafael C. Gonzalez
ISA
Universidad de Oviedo
Gijón, Spain
rcgonzalez@uniovi.es

IECON 2021 - 47th Annual Conference of the IEEE Industrial Electronics Society | 978-1-6654-3554-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/IECON48115.2021.9589442

Abstract— **Localization of mobile robots in industrial environments is key in an increasingly automated industry. Nowadays, the inspection and repair of heavy steel plates is performed by human workers. Repair work often requires long hours in uncomfortable postures that can cause problems for the worker. We propose a mobile robot placed on top of a steel plate that must move along the plate to inspect and repair it, without leaving the sheet. Robot localization on the plate is key to generate the inspection and repair trajectories.**

There are different methods of localization, the most widely used require the use of expensive laser sensors to create a map using information from the environment and localize from it. This paper proposes a less expensive localization system for a mobile robot based on the installation of ArUco markers in the environment and the use of a ring of 8 calibrated cameras mounted on the robot that allow a 360º vision. This ensures a correct localization regardless of the working area. It is necessary to map the markers with respect to a common coordinate system.

We propose a method to create the map using the ring. We validate the proposal through experiments comparing the localization obtained with the proposed system and a localization using a state-of-the-art SLAM method employing laser sensors.

Keywords— Mobile Robot, Computer Vision, Robot Localization, ArUco Markers, Ring of Cameras, ROS.

I. Introduction

Mobile robots are appearing in an increasingly automated industry in a growing range of applications. One example is the deployment of autonomous mobile robots for inspection of surface defects. Other applications seek to replace human workers in dangerous tasks, such as grinding.

In the production of heavy steel plates, both problems appear together. Currently, the inspection and repair are performed by human operators. The repair work usually involves long working days in uncomfortable positions that can lead to physical problems for the worker.

This paper presents a localization method for a mobile robot performing sheet metal inspection and repair process without leaving the sheet. Therefore, the robot moves in a localized space, the width of the sheet metal varies from 1.4 to 3.3 m and their length may change from 4 to 18 m. Normally, sheet metal inspections are performed in large indoor environments. So, there may not be enough references in the environment for certain types of localization systems.

Localization problem has been discussed from multiple approaches, being common the use of different sensors that provide direct information about the location of the robot at each time, or about the changes that are produced in its environment.

One of the most widely used alternatives in the localization of mobile robots is the use of Simultaneous Localization and Mapping (SLAM) techniques. These systems build a map of the environment that is used to localize the robot. Expensive laser scanners are generally used for the construction of the environment map. In addition, systems based on natural feature extraction and matching could be used but it can be problematic in certain scenarios if there are not enough natural landmarks.

Sensor fusion is also common in robotics, usually using Kalman filters [1] or particle filters [2], [3], which allows combining estimates from different sources to achieve a more robust pose. In [4], information from inertial sensors is fused with a visual odometry method.

GPS systems are another widely used method. However, systems based on satellite signals are not appropriate for this task because GPS signals are not available in indoor scenarios.

Other strategy is by placing beacons at known positions in the environment to facilitate the localization of the robot. For example, visual beacons, such as luminous beacons of different geometries and colors or fiducial markers.

Different fiducial marker systems can be found, composed by a set of defined markers and an algorithm that allows their detection and identification. These systems have been used in multiple vehicle localization applications. For example, researchers from the AVA group at the University of Cordoba developed different applications to localize mobile robots using ArUco markers with a single camera [5]. Other research using computer vision and artificial markers can be seen in [6], [7] and [8]. The paper in [9] proposes the localization of a mobile robot by a particle filter combining the information from an omnidirectional camera and a range sensor, using fiducial markers.

Other strategy is using a robot tracking system. For that, fiducials are also used, as in [10], in which the mobile robot is tracked by arranging an artificial marker on top of it using a system of multiple cameras fixed in the environment. In [11] they use geometric features of the robot itself instead artificial markers.

In this paper, we propose a localization system using an 8-camera ring fixed in the robot and a set of artificial markers placed in the environment, surrounding the plate being inspected by the robot. It is necessary to know the position of the markers with respect to a common coordinate system. Therefore, a way to locate them and create a map of the markers is proposed, so that the robot can move over the sheet as accurately as possible.

Research supported by: Daorje S.L.U inside project code CN-18-011 and a scholarship under the "Severo Ochoa" program for predoctoral research and teaching with Ref: PA-20-PF-BP19-067 financed by Asturias Regional Government.

Figure A.4: "Mobile robot localization in industrial environments using a ring of cameras and ArUco markers". Article published and presented in the international congress IECON 2021 [115]

Block Matching Mosaicing for Surface Inspection Using an Autonomous Mobile Robot

Sara Roos-Hoefgeest Toribio
Systems and Automation Engineering
Universidad de Oviedo
Gijón, Spain
roossara@uniovi.es

Ignacio Alvarez Garcia
Systems and Automation Engineering
Universidad de Oviedo
Gijón, Spain
ialvarez@uniovi.es

Rafael C. Gonzalez de los Reyes
Systems and Automation Engineering
Universidad de Oviedo
Gijón, Spain
rcgonzalez@uniovi.es

IECON 2021 - 47th Annual Conference of the IEEE Industrial Electronics Society | 978-1-6654-3554-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/IECON48115.2021.9589139

***Abstract*—Visual inspection of manufactured products is a field in constant expansion. In this work we present a method to create a high resolution panorama (1mm/pixel) of a large rectangular plate using a mobile robot with two RGB-D cameras. The panorama is intended to analyze the surface in search of possible defects and identify areas of interest that have been encircled using a high contrast mark. Identifying which points belong to the surface plane and estimating the amount of distortion caused by the perspective correction we are able to form a panorama of a 4400 by 2500 mm plate with errors lower than 2% and a resolution of 1 mm/pixel.**

***Index Terms*—visual inspection, image mosaicing, autonomous robot.**

I. Introduction

Surface inspection of steel products using computer vision techniques is a well established practice nowadays. There have been a huge effort to analyze different categories of steel products as they are manufactured [1]–[4]. Some defects occurring in slabs or heavy plates may be repaired. Defects are removed manually using an angle grinder. The worker that conducts the operation usually needs to adopt awkward postures. As a consequence, his exposition occupational hazards increases.

We are designing a mobile robot to automate heavy steel plates reparation. To be successful, the robot must be able to explore the plate surface to identify which parts have been marked as defective, or even to look for defects. The thickness of heavy steel plates ranges from 5 to 150 mm. Their width varies from 1400 to 3300 mm and their length may change from 4 to 18 m. To enforce robot and human safety, the robot must complete the analysis moving exclusively within the plate. To complete the taks, the robot must move on top of the surface and take partial surface images. Before the inspection starts, these images have to be stitched together to form a single view called a panorama. This approach has been used in many different applications [5]–[7].

Image registration and stitching is usually solved through the pairing of image features. Those methods usually provide very good results and are considered to be the state of the art. Works by Szeliski [8] or Zitová and Flusser [9] are a good starting point to have an overview of the full process. A common problem when these methods are applied at a large scale is that concatenation of geometric transforms may cause an increasing amount of image distortion [10]. Block matching methods, although not so common, have been successfully applied to register a sequence of images in industrial environments [6] or to determine the motion of glaciers or landslides [11]. In this paper we describe a method to reconstruct the surface of a large rectangular plate using an omnidirectional mobile robot and RGB-D cameras. The method has been developed to be used on heavy steel plates, although real tests have been done using Oriented Strand Board (OSB) wooden panels. This is because they can be easily handled inside our lab and present a repetitive texture where key points are difficult to detect. The areas to be repaired are expected to be enclosed within a high contrast mark.

Our proposal is based on a classical block matching approach, although several new ideas are introduced to account for image distortion and dynamic selection of the reference block to be matched. The cameras are pointing 30° downward with respect to a horizontal plane. We apply an homography to remove the perspective effect. This introduces a distortion in the re-projected image. We use the transformation Jacobian to get an estimation of the distortion. This value is used to reduce the search area within the block matching algorithm and to control how images are merged to build the final panorama. In addition, we classify pixels as surface/not surface points according to their distance to the plane defined by the inspected plate. This clustering is used to further restrict the candidate positions of the template block. To build the full panorama we use a hierarchical approach similar to the one described by Xie et al. [13]. First, we register images from a single camera to form a stripe. Neighboring stripes are stitched together to build a unique panorama for each camera. We finish by joining both partial panoramas into a single one.

II. Methods

A. Algorithm description

Our algorithm, outlined in Fig. 1, follows a hierarchical structure. First we compute the translation between two consecutive frames. The result is used to register the new image within the current stripe panorama. Those stripe panoramas are

Research supported by: Daorje S.L.U inside project code CN-18-011 and a scholarship under the "Severo Ochoa" program for predoctoral research and teaching with Ref: PA-20-PF-BP19-067, financed by Asturias Regional Government .

Figure A.5: "Block Matching Mosaicing for Surface Inspection Using an Autonomous Mobile Robot". Article published and presented in the international congress IECON 2021 [116]

XL Jornadas de Automática Visión por Computador

LOCALIZACIÓN DE ROBOTS MÓVILES EN ENTORNOS INDUSTRIALES USANDO UN ANILLO DE CÁMARAS

Sara Roos Hoefgeest Toribio, Álvaro Fernández García, Ignacio Álvarez García, Rafael Corsino González de los Reyes

Área de Ingeniería de Sistemas y Automática. Universidad de Oviedo
Campus Universitario de Gijón, s/n.
{roossara, fernandezgalvaro, ialvarez, rcgonzalez}@uniovi.es

Resumen

Se pretende desarrollar un robot móvil capaz de inspeccionar y reparar chapas de acero en un entorno industrial de grandes dimensiones. Uno de los principales problemas a resolver es la localización del mismo mientras navega sobre la chapa.

El presente documento propone un sistema de localización de un robot móvil basado en la instalación de marcadores ArUco y el uso de un anillo de 8 cámaras calibradas dispuesto sobre el robot que permiten una visión de 360º.

Es preciso conocer la posición de los marcadores respecto a un sistema de coordenadas común. Por ello, se propone una forma de localizar los marcadores y crear un mapa de los mismos con una sola cámara, de tal manera que, posteriormente, pueda ser utilizado para desplazarse sobre la chapa con la mayor precisión posible.

La estrategia escogida se desarrolló en forma de paquetes de ROS capaces de proporcionar el estado del robot a otros algoritmos encargados de tareas como la navegación.

Palabras clave: Robótica, Visión 3D, Localización, Marcadores ArUco, Mapeo, Entorno industrial, Anillo cámaras, ROS.

1 INTRODUCCIÓN

Se desea desarrollar un robot capaz de inspeccionar la totalidad de una chapa de acero en busca de defectos y, posteriormente, ser capaz de subsanarlos. Para ello, el robot deberá ser capaz de navegar con precisión en un entorno industrial de grandes dimensiones.

Uno de los principales problemas es la localización del robot dentro del entorno. A priori, no se puede garantizar que existan elementos externos que puedan ser utilizados como balizas. Para resolver este problema, se propone un sistema de localización basado en la instalación de marcadores ArUco y el uso de un anillo de 8 cámaras dispuesto sobre el robot móvil.

No se puede garantizar que la posición exacta de los marcadores con respecto a un sistema de coordenadas común sea conocida. En este artículo se propone una manera de localizar los marcadores y elaborar un mapa de los mismos utilizando una sola cámara, que permita, posteriormente, ser utilizado para que el robot se desplace sobre la chapa con la mayor precisión posible.

El sistema de mapeo utiliza una cámara calibrada para estimar la pose de los marcadores respecto a un sistema de referencia común, elegido en la posición de uno de los marcadores en el entorno. Una vez realizada una primera estimación de sus posiciones se aplica un ajuste *bundle* para minimización del error y optimizar sus localizaciones.

En cuanto a la localización, cada cámara que forma el arco sigue, individualmente, un esquema monocular, estimando la posición del robot resolviendo el problema de *Perspective-n-Point* (PnP) a partir de las poses 3D de marcadores y sus proyecciones en sus imágenes. Posteriormente, se escoge la mejor estimación entre las 8 cámaras y se someten los datos a un filtro de Kalman para eliminar malas estimaciones y conseguir una localización más fiable.

2 ESTADO DEL ARTE

Cada vez resulta más habitual encontrar robots móviles en las industrias. La localización en cada instante de tiempo es vital para estos robots. Este problema se ha tratado desde múltiples perspectivas, siendo común el uso de diferentes sensores que proporcionan información directa sobre la localización del robot en cada instante, o bien, sobre los cambios que se han producido en su entorno.

https://doi.org/10.17979/spudc.9788497497169.849

Figure A.6: "Localización de robots móviles en entornos industriales usando un anillo de cámaras". Article published and presented in the national Spanish congress JJAA 2019 [118]

XL Jornadas de Automática Robótica

ALGORITMO DE GENERACIÓN DE TRAYECTORIAS EN EL INTERIOR DE CHAPAS PARA LA SUBSANACIÓN DE DEFECTOS

Álvaro Fernández García, Sara Roos Hoefgeest Toribio, Ignacio Álvarez García, Rafael Corsino González de los Reyes

Área de Ingeniería de Sistemas y Automática. Universidad de Oviedo
Campus Universitario de Gijón, s/n.
{fernandezgalvaro, roossara, ialvarez, rcgonzalez}@uniovi.es

Resumen

El presente documento describe un algoritmo de generación de trayectorias para un robot móvil encargado de sanear chapas de acero. Las trayectorias generadas deben asegurar la cobertura total de las superficies poligonales que delimitan los defectos con la herramienta acoplada al robot. Además, el robot deberá resolver la tarea sin abandonar el interior de la chapa. Para resolver el problema, la chapa se divide en distintas zonas de trabajo derivadas de las orientaciones seguras que permiten al robot reparar sin abandonar la superficie de la chapa.

Los defectos se recibirán en forma de polígonos que serán, en primer lugar, divididos de acuerdo con las zonas de trabajo y, a continuación, descompuestos en formas más simples teniendo en cuenta la orientación de trabajo de cada zona. El recorrido completo del defecto podrá realizarse calculando trayectorias paralelas para cada uno de los polígonos simples que lo componen. La trayectoria así calculada corresponde a la seguida por la herramienta, por lo que para calcular la trayectoria del robot bastará con aplicar una traslación.

Palabras clave: Robótica, Planificación de trayectorias, Geometría computacional, Descomposición trapezoidal, ROS

1 INTRODUCCIÓN

En una industria cada vez más automatizada los robots móviles desempeñan cada vez más tareas. Una de estas posibles aplicaciones es la inspección y subsanación de chapas.

Frente a otras alternativas, un robot móvil tiene unas dimensiones y un peso moderados que permite su sencillo traslado a otros lugares de trabajo y facilita su posible colaboración con operarios humanos o con otras máquinas. Otra ventaja de la opción escogida es su versatilidad, siendo capaz de reparar desde hojas de acero con una superficie no mucho mayor que la del propio robot hasta grandes chapas de decenas de metros, como las que pueden encontrarse en la industria siderúrgica. El empleo de, por ejemplo, un robot cartesiano para automatizar la inspección y reparación de estas últimas conllevaría la construcción de una gran estructura.

Se supondrá, por tanto, la existencia de un robot omnidireccional dotado de una herramienta en una posición fija respecto al centro del robot. En este planteamiento el actuador se encarga de subsanar un defecto detectado en la chapa, aunque el algoritmo presentado resulta fácilmente adaptable a otras labores tales como inspección, pintado, limpieza, etc.

En la aplicación propuesta el robot se encontrará trabajando sobre una superficie rectangular plana, y que puede estar elevada una distancia considerable respecto al suelo. Si el robot se sale de la superficie de trabajo, podría resultar peligroso y provocar graves daños en el robot e incluso en operarios humanos.

Para resolver el problema, la chapa se divide en 4 zonas, asociada cada una a una posible orientación del robot paralela a los ejes de la chapa. En cada una de las zonas, el robot podrá, manteniendo la orientación asociada, realizar las pertinentes reparaciones sin correr el riesgo de abandonar el espacio de trabajo.

Los defectos, recibidos en forma de polígono, se dividirán por tanto según las zonas de trabajo. De esta manera, cada división se recorrerá con una única orientación. Con el fin de facilitar la reparación completa de los defectos se llevará a cabo también una descomposición trapezoidal que da lugar a formas simples de reparación más sencilla.

1.1 ESTADO DEL ARTE

Tras las investigaciones realizadas no se ha encontrado bibliografía referida a este conjunto de problemas combinados.

https://doi.org/10.17979/spudc.9788497497169.702

Figure A.7: "Algoritmo de generación de trayectorias en el interior de chapas para la subsanación de defectos". Article published and presented in the national Spanish congress JJAA 2019 [119]

Bibliography

[1] M. Javaid, A. Haleem, R. P. Singh, S. Rab, and R. Suman, "Exploring impact and features of machine vision for progressive industry 4.0 culture," *Sensors International*, vol. 3, p. 100132, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S266635112100053X

[2] R. G. Lins, R. E. dos Santos, and R. Gaspar, "Vision-based measurement for quality control inspection in the context of industry 4.0: a comprehensive review and design challenges," 4 2023.

[3] V. Azamfirei, F. Psarommatis, and Y. Lagrosen, "Application of automation for in-line quality inspection, a zero-defect manufacturing approach," *Journal of Manufacturing Systems*, vol. 67, pp. 1–22, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0278612522002291

[4] S. H. Ali, "State-of-the-art of cmm-coordinate metrology in automotive industry," vol. 2017-March. SAE International, 3 2017.

[5] S. Bharathi and S. Sathiyamoorthy, "Non-contact measurement techniques using machine vision," in *International Colloquium on materials, Manufacturing and metrology, ICMMM 2014*, 8 2014.

[6] C. Wu, B. Chen, and C. Ye, "Detecting defects on corrugated plate surfaces using a differential laser triangulation method," *Optics and Lasers in Engineering*, vol. 129, p. 106064, 06 2020.

[7] I. Alvarez, J. Enguita, M. Frade, J. Marina, and G. Ojea, "On-line metrology with conoscopic holography: Beyond triangulation," *Sensors (Basel, Switzerland)*, vol. 9, pp. 7021–37, 09 2009.

[8] I. Jovančević, H. H. Pham, J. J. Orteu, R. Gilblas, J. Harvent, X. Maurice, and L. Brèthes, "3d point cloud analysis for detection and characterization

of defects on airplane exterior surface," *Journal of Nondestructive Evaluation*, vol. 36, 12 2017.

[9] V. Borsu, A. Yogeswaran, and P. Payeur, "Automated surface deformations detection and marking on automotive body panels," 2010, pp. 551–556.

[10] V. Heikkinen, J. Nysten, V. Byman, B. Hemming, and A. Lassila, "Multi-sensor optical profilometer for measurement of large freeforms at nm-level uncertainty," *Surface Topography: Metrology and Properties*, vol. 8, no. 4, p. 045030, dec 2020. [Online]. Available: https://dx.doi.org/10.1088/2051-672X/abd293

[11] L. Technologies, "3 types of smart sensors for in-line quality control," 2024, accessed: 2024-08-27. [Online]. Available: https://lmi3d.com/blog/3-types-smart-sensors-inline-quality-control/

[12] J. Dias, P. Simões, N. Soares, C. M. Costa, M. R. Petry, G. Veiga, and L. F. Rocha, "Comparison of 3d sensors for automating bolt-tightening operations in the automotive industry," *Sensors*, vol. 23, no. 9, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/9/4310

[13] D. Ding, W. Ding, R. Huang, Y. Fu, and F. Xu, "Research progress of laser triangulation on-machine measurement technology for complex surface: A review," *Measurement*, vol. 216, p. 113001, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0263224123005651

[14] M. M. Auerswald, A. von Freyberg, and A. Fischer, "Laser line triangulation for fast 3D measurements on large gears," *The International Journal of Advanced Manufacturing Technology*, vol. 100, no. 9, pp. 2423–2433, 2019. [Online]. Available: https://doi.org/10.1007/s00170-018-2636-z

[15] A. Hosseinpour, Y. Peng, G. Koch, K. Ni, and A. Guenther, "Optical Gear Inspection Using a Triangulation Sensor and an Areal Evaluation," in *35th American Society for Precision Engineering Annual Meeting (ASPE 2020)*. Online: Americal Society for Precision Engineering, Oct. 2020.

[16] A. Schöch, A. Salvadori, I. Germann, S. Balemi, C. Bach, A. Ghiotti, S. Carmignato, A. L. Maurizio, and E. Savio, "High-Speed Measurement of Complex Shaped Parts at Elevated Temperature by Laser Triangulation," *International Journal of Automation Technology*, vol. 9, no. 5, pp. 558–566, 2015.

[17] T. Czimmermann, M. Chiurazzi, M. Milazzo, S. Roccella, M. Barbieri, P. Dario, C. M. Oddo, and G. Ciuti, "An Autonomous Robotic Platform

for Manipulation and Inspection of Metallic Surfaces in Industry 4.0," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 3, pp. 1691–1706, 2022. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9625638

[18] C. Naverschnigg, E. Csencsics, and G. Schitter, "Flexible Robot-Based In-Line Measurement System for High-Precision Optical Surface Inspection," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–9, 2022. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9927464

[19] R. Usamentiaga, D. F. Garcia, and F. J. delaCalle, "Rail flatness measurement based on dual laser triangulation," *IEEE Transactions on Industry Applications*, vol. 60, no. 4, pp. 5849–5860, 2024.

[20] W. Huang and R. Kovacevic, "A laser-based vision system for weld quality inspection," *Sensors*, vol. 11, no. 1, pp. 506–521, 2011. [Online]. Available: https://www.mdpi.com/1424-8220/11/1/506

[21] Y. Gao, X. Li, X. V. Wang, L. Wang, and L. Gao, "A review on recent advances in vision-based defect recognition towards industrial intelligence," *Journal of Manufacturing Systems*, vol. 62, pp. 753–766, 1 2022.

[22] X. Lv, F. Duan, J. J. Jiang, X. Fu, and L. Gan, "Deep active learning for surface defect detection," *Sensors (Switzerland)*, vol. 20, 3 2020.

[23] L. Huo, Y. Liu, Y. Yang, Z. Zhuang, and M. Sun, "Review: Research on product surface quality inspection technology based on 3d point cloud," *Advances in Mechanical Engineering*, vol. 15, no. 3, p. 16878132231159523, 2023. [Online]. Available: https://doi.org/10.1177/16878132231159523

[24] A. Khan, C. Mineo, G. Dobie, C. Macleod, and G. Pierce, "Vision guided robotic inspection for parts in manufacturing and remanufacturing industry," *Journal of Remanufacturing*, vol. 11, no. 1, pp. 49–70, 04 2021. [Online]. Available: https://doi.org/10.1007/s13243-020-00091-x

[25] J. Oaki, N. Sugiyama, Y. Ishihara, J. Ooga, H. Kano, and H. Ohno, "Micro-defect inspection on curved surface using a 6-dof robot arm with one-shot brdf imaging," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 9354–9359, 2023, 22nd IFAC World Congress. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S240589632300575X

[26] P. Nooralishahi, C. Ibarra-Castanedo, S. Deane, F. López, S. Pant, M. Genest, N. P. Avdelidis, and X. P. V. Maldague, "Drone-based non-destructive

inspection of industrial sites: A review and case studies," *Drones*, vol. 5, no. 4, 2021. [Online]. Available: https://www.mdpi.com/2504-446X/5/4/106

[27] S. PHILLIPS, N. Charron, E. MCLAUGHLIN, and S. Narasimhan, "Infrastructure mapping and inspection using mobile ground robotics," 11 2019.

[28] A. Cantieri, M. Ferraz, G. Szekir, M. Antônio Teixeira, J. Lima, A. Schneider Oliveira, and M. Aurélio Wehrmeister, "Cooperative uav–ugv autonomous power pylon inspection: An investigation of cooperative outdoor vehicle positioning architecture," *Sensors*, vol. 20, no. 21, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/21/6384

[29] F. Nauert and P. Kampmann, "Inspection and maintenance of industrial infrastructure with autonomous underwater robots," *Frontiers in Robotics and AI*, vol. 10, 2023. [Online]. Available: https://www.frontiersin.org/articles/10.3389/frobt.2023.1240276

[30] L. Li, D. Xu, L. Niu, Y. Lan, and X. Xiong, "A path planning method for a surface inspection system based on two-dimensional laser profile scanner," *International Journal of Advanced Robotic Systems*, vol. 16, p. 172988141986246, 7 2019.

[31] S. E. Sadaoui, C. Mehdi-Souzani, C. Lartigue, and M. Brahim, "Automatic path planning for high performance measurement by laser plane sensors," *Optics and Lasers in Engineering*, vol. 159, 12 2022.

[32] M. Mohammadikaji, S. Bergmann, J. Beyerer, J. Burke, and C. Dachsbacher, "Sensor-realistic simulations for evaluation and planning of optical measurement systems with an application to laser triangulation," *IEEE Sensors Journal*, vol. 20, pp. 5336–5349, 5 2020.

[33] Vulture19, "Coordinate measuring machine," *Wikimedia Commons*, 2017, cC BY-SA 3.0 https://creativecommons.org/licenses/by-sa/3.0, via Wikimedia Commons. [Online]. Available: https://commons.wikimedia.org/wiki/File:9.12.17_Coordinate_measuring_machine.png

[34] Acuity, "Laser triangulation sensors," 2024, accessed: 2024-08-27. [Online]. Available: https://www.acuitylaser.com/sensor-resources/laser-triangulation-sensors/

[35] I. Álvarez, J. M. Enguita, M. Frade, J. Marina, and G. Ojea, "On-line metrology with conoscopic holography: Beyond triangulation," *Sensors*, vol. 9, no. 9, pp. 7021–7037, 2009. [Online]. Available: https://www.mdpi.com/1424-8220/9/9/7021

[36] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. The MIT Press, 2018. [Online]. Available: http://incompleteideas.net/book/the-book-2nd.html

[37] C. J. C. H. Watkins and P. Dayan, "Q-learning," pp. 279–292, 1992.

[38] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 2013.

[39] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," 2015.

[40] G. A. Rummery and M. Niranjan, "On-line q-learning using connectionist systems," 1994.

[41] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[42] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization," 2017.

[43] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," 2016.

[44] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," 2018.

[45] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2019.

[46] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," 2018.

[47] G. Rummery and M. Niranjan, "On-line q-learning using connectionist systems," *Technical Report CUED/F-INFENG/TR 166*, 11 1994.

[48] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:205242740

[49] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[50] G. E. Uhlenbeck and L. S. Ornstein, "On the theory of the brownian motion," *Phys. Rev.*, vol. 36, pp. 823–841, Sep 1930. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRev.36.823

[51] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," 2016. [Online]. Available: https://arxiv.org/abs/1602.01783

[52] G. Häusler, "Three-dimensional sensors - potentials and limitations," in *Handbook of Computer Vision and Applications, Volume 1: Sensors and Imaging*, B. Jähne, H. Haussecker, and P. Geissler, Eds. Academic Press, 1999, ch. 19, pp. 485–506.

[53] S. Roos-Hoefgeest, M. Roos-Hoefgeest, I. Álvarez, and R. C. González, "Simulation of laser profilometer measurements in the presence of speckle using perlin noise," *Sensors*, vol. 23, no. 17, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/17/7624

[54] C. Cajal, J. Santolaria, D. Samper, and A. Garrido, "Simulation of laser triangulation sensors scanning for design and evaluation purposes," *International Journal of Simulation Modelling*, vol. 14, pp. 250–264, 2015.

[55] B. A. Abu-Nabah, A. O. ElSoussi, and A. E. K. Al Alami, "Virtual laser vision sensor environment assessment for surface profiling applications," *Measurement*, vol. 113, pp. 148–160, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0263224117305572

[56] M. Mohammadikaji, S. Bergmann, S. Irgenfried, J. Beyerer, C. Dachsbacher, and H. Wörn, "A framework for uncertainty propagation in 3D shape measurement using laser triangulation," in *2016 IEEE International Instrumentation and Measurement Technology Conference Proceedings*, 5 2016, pp. 1–6.

[57] E. Csencsics, J. Schlarp, T. Glaser, T. Wolf, and G. Schitter, "Simulation and Reduction of Speckle-induced Uncertainty in Laser Triangulation Sensors," in *2022 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, 2022, pp. 1–6.

[58] R. Beermann, L. Quentin, G. Stein, E. Reithmeier, and M. Kästner, "Full simulation model for laser triangulation measurement in an inhomogeneous refractive index field," *Optical Engineering*, vol. 57, no. 11, p. 114107, 11 2018.

[59] K. Perlin, "An image synthesizer," in *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '85. New York, NY, USA: Association for Computing Machinery, 1985, p. 287–296. [Online]. Available: https://doi.org/10.1145/325334.325247

[60] ——, "Improving noise," *ACM Trans. Graph.*, vol. 21, no. 3, p. 681–682, jul 2002. [Online]. Available: https://doi.org/10.1145/566654.566636

[61] A. Lagae, S. Lefebvre, R. Cook, T. DeRose, G. Drettakis, D. Ebert, J. Lewis, K. Perlin, and M. Zwicker, "A Survey of Procedural Noise Functions," *Computer Graphics Forum*, vol. 29, no. 8, pp. 2579–2600, 2010. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2010.01827.x

[62] J. Dong, J. Liu, K. Yao, M. Chantler, L. Qi, H. Yu, and M. Jian, "Survey of Procedural Methods for Two-Dimensional Texture Generation," *Sensors*, vol. 20, no. 4, p. 1135, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/4/1135

[63] H. Li, H. Yang, C. Xu, and Y. Cao, *Water Surface Simulation Based on Perlin Noise and Secondary Distorted Textures*. Springer Singapore, 01 2017, pp. 236–245.

[64] M. R. G. Acosta, J. C. V. Díaz, and N. S. Castro, "An innovative image-processing model for rust detection using perlin noise to simulate oxide textures," *Corrosion Science*, vol. 88, pp. 141–151, 2014.

[65] F. Conde-Rodríguez, L. García-Fernández, and J. C. Torres, "Modelling material microstructure using the perlin noise function," *Computer Graphics Forum*, vol. 40, pp. 195–208, 2 2021.

[66] A. Koutsoudis, G. Ioannakis, B. Vidmar, F. Arnaoutoglou, and C. Chamzas, "Using noise function-based patterns to enhance photogrammetric 3d reconstruction performance of featureless surfaces," *Journal of Cultural Heritage*, vol. 16, pp. 664–670, 9 2015.

[67] T. Möller and B. Trumbore, "Fast, minimum storage ray/triangle intersection," in *ACM SIGGRAPH 2005 Courses*, ser. SIGGRAPH '05. New York, NY, USA: Association for Computing Machinery, 2005, p. 7–es. [Online]. Available: https://doi.org/10.1145/1198555.1198746

[68] E. Gadelmawla, M. Koura, T. Maksoud, I. Elewa, and H. Soliman, "Roughness parameters," *Journal of Materials Processing Technology*, vol.

123, no. 1, pp. 133–145, 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924013602000602

[69] W. Schroeder, K. Martin, and B. Lorensen, *The Visualization Toolkit–An Object-Oriented Approach To 3D Graphics*, 4th ed. Kitware, Inc., 2006.

[70] *High-Speed 3D Compact Sensors Fast, Accurate 3D Imaging for Industry*, Automation Technology, 10 2019, version 2.5.

[71] "Crankshaft bearing cap [cat crankshaft bearing cap]," https://parts.cat.com/es/catcorp/167-8081, accessed May 9, 2023.

[72] F. J. Massey, "The kolmogorov-smirnov test for goodness of fit," *Journal of the American Statistical Association*, vol. 46, no. 253, pp. 68–78, 1951. [Online]. Available: http://www.jstor.org/stable/2280095

[73] M. Z. Wong, K. Kunii, M. Baylis, W. H. Ong, P. Kroupa, and S. Koller, "Synthetic dataset generation for object-to-model deep learning in industrial applications," *PeerJ Computer Science*, vol. 5, p. e222, 2019. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7924434/

[74] J. Cohen, C. F. Crispim-Junior, C. Grange-Faivre, and L. Tougne, "CAD-based Learning for Egocentric Object Detection in Industrial Context," in *15th International Conference on Computer Vision Theory and Applications*, vol. 5. SCITEPRESS - Science and Technology Publications, 2020, pp. 644–651. [Online]. Available: https://hal.science/hal-02553622

[75] D. Horváth, G. Erdős, Z. Istenes, T. Horváth, and S. Földi, "Object Detection Using Sim2Real Domain Randomization for Robotic Applications," *IEEE Transactions on Robotics*, vol. 39, no. 2, pp. 1225–1243, 2023. [Online]. Available: http://arxiv.org/abs/2208.04171

[76] X. Zhu, P. Mårtensson, L. Hanson, M. Björkman, and A. Maki, "Automated assembly quality inspection by deep learning with 2D and 3D synthetic CAD data," *Journal of Intelligent Manufacturing*, 2024. [Online]. Available: https://doi.org/10.1007/s10845-024-02375-6

[77] D. Mery, D. Hahn, and N. Hitschfeld, "Simulation of defects in aluminum castings using cad models of flaws and real x-ray images," *Insight: Non-Destructive Testing and Condition Monitoring*, vol. 47, pp. 618–624, 10 2005.

[78] A. Boikov, V. Payor, R. Savelev, and A. Kolesnikov, "Synthetic data generation for steel defect detection and classification using deep learning," *Symmetry*, vol. 13, 7 2021.

[79] L. Bosnar, D. Saric, S. Dutta, T. Weibel, M. Rauhut, and P. Gospodnetić, "Image synthesis pipeline for surface inspection," 11 2020.

[80] J. P. Yun, W. C. Shin, G. Koo, M. S. Kim, C. Lee, and S. J. Lee, "Automated defect inspection system for metal surfaces based on deep learning and data augmentation," *Journal of Manufacturing Systems*, vol. 55, pp. 317–324, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S027861252030042X

[81] S. Jain, G. Seth, A. Paruthi, U. Soni, and G. Kumar, "Synthetic data augmentation for surface defect detection and classification using deep learning," *Journal of Intelligent Manufacturing*, vol. 33, pp. 1007–1020, 4 2022.

[82] Z. Du, L. Gao, and X. Li, "A new contrastive gan with data augmentation for surface defect recognition under limited data," *IEEE Transactions on Instrumentation and Measurement*, vol. 72, 2023.

[83] X. Zhong, J. Zhu, W. Liu, C. Hu, Y. Deng, and Z. Wu, "An overview of image generation of industrial surface defects," *Sensors*, vol. 23, no. 19, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/19/8160

[84] T. W. Sederberg and S. R. Parry, "Free-form deformation of solid geometric models," in *IN: SIGGRAPH, 1986. PROCEEDINGS*, 1986, pp. 151–160.

[85] S. Becker, A. Mang, A. Toma, and T. M. Buzug, "Approximating tumor induced brain deformation using directly manipulated free form deformation," in *2010 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, 2010, pp. 85–88.

[86] N. Lin and J. Duncan, "Generalized robust point matching using an extended free-form deformation model: application to cardiac images," in *2004 2nd IEEE International Symposium on Biomedical Imaging: Nano to Macro (IEEE Cat No. 04EX821)*, 2004, pp. 320–323 Vol. 1.

[87] A. Scardigli, R. Arpa, A. Chiarini, and H. Telib, "Enabling of large scale aerodynamic shape optimization through POD-based reduced-order modeling and free form deformation," in *Advances in Evolutionary and Deterministic Methods for Design, Optimization and Control in Engineering and Sciences*, E. Minisci, M. Vasile, J. Periaux, N. R. Gauger, K. C. Giannakoglou, and D. Quagliarella, Eds. Cham: Springer International Publishing, 2019, pp. 49–63.

[88] F. Salmoiraghi, A. Scardigli, and H. Telib, "Free form deformation, mesh morphing and reduced order methods: enablers for efficient aerodynamic shape optimization," *International Journal of Computational Fluid Dynamics*, vol. 32, 03 2018.

[89] T. W. Sederberg and S. R. Parry, "Free-form deformation of solid geometric models," *SIGGRAPH Comput. Graph.*, vol. 20, no. 4, p. 151–160, aug 1986. [Online]. Available: https://doi.org/10.1145/15886.15903

[90] S.-M. Hu, H. Zhang, C.-L. Tai, and J.-G. Sun, "Direct manipulation of ffd: Efficient explicit solutions and decomposible multiple point constraints," *The Visual Computer*, vol. 17, pp. 370–379, 08 2001.

[91] *High-Speed 3D Compact Sensors Fast, Accurate 3D Imaging for Industry*, Automation Technology, 10 2019, version 2.5.

[92] O. engineering srl, "mimmo: Surface manipulation and mesh morphing library," 2015, https://github.com/optimad/mimmo.

[93] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[94] J. Torres and J. T. J. Austen, "Yolov8 architecture: A deep dive into its architecture," Jan 2024. [Online]. Available: https://yolov8.org/yolov8-architecture/

[95] F. Gao, *Interferometry for Online/In-Process Surface Inspection*. Croatia: InTech, Feb. 2017, pp. 41–59.

[96] H. Chen, S. Huo, M. Muddassir, H.-Y. Lee, A. Duan, P. Zheng, H. Pan, and D. Navarro-Alarcon, "Pso-based optimal coverage path planning for surface defect inspection of 3c components with a robotic line scanner," 7 2023. [Online]. Available: http://arxiv.org/abs/2307.04431

[97] A. Lobbezoo, Y. Qian, and H. J. Kwon, "Reinforcement learning for pick and place operations in robotics: A survey," 9 2021.

[98] Íñigo Elguea-Aguinaco, A. Serrano-Muñoz, D. Chrysostomou, I. Inziarte-Hidalgo, S. Bøgh, and N. Arana-Arexolaleiba, "A review on reinforcement learning for contact-rich robotic manipulation tasks," 6 2023.

[99] D. Han, B. Mulyana, V. Stankovic, and S. Cheng, "A survey on deep reinforcement learning algorithms for robotic manipulation," 4 2023.

[100] M. Ji, L. Zhang, and S. Wang, "A path planning approach based on q-learning for robot arm," in *2019 3rd International Conference on Robotics and Automation Sciences (ICRAS)*, 2019, pp. 15–19.

[101] K. Zhu and T. Zhang, "Deep reinforcement learning based mobile robot navigation: A review," *Tsinghua Science and Technology*, vol. 26, no. 5, pp. 674–691, 2021.

[102] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258–1276, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092188901300167X

[103] X. Zeng, "Reinforcement learning based approach for the navigation of a pipe-inspection robot at sharp pipe corners," 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:214183928

[104] W. Jing, C. F. Goh, M. Rajaraman, F. Gao, S. Park, Y. Liu, and K. Shimada, "A computational framework for automatic online path generation of robotic inspection tasks via coverage planning and reinforcement learning," *IEEE Access*, vol. 6, pp. 54 854–54 864, 2018.

[105] C. Landgraf, B. Meese, M. Pabst, G. Martius, and M. F. Huber, "A reinforcement learning approach to view planning for automated inspection tasks," *Sensors*, vol. 21, pp. 1–17, 3 2021.

[106] A. A. Shahid, L. Roveda, D. Piga, and F. Braghin, "Learning continuous control actions for robotic grasping with reinforcement learning," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2020, pp. 4066–4072.

[107] A. Lobbezoo and H.-J. Kwon, "Simulated and real robotic reach, grasp, and pick-and-place using combined reinforcement learning and traditional controls," *Robotics*, vol. 12, no. 1, 2023. [Online]. Available: https://www.mdpi.com/2218-6581/12/1/12

[108] J. W. Mock and S. S. Muknahallipatna, "A comparison of ppo, td3 and sac reinforcement algorithms for quadruped walking gait generation," *Journal of Intelligent Learning Systems and Applications*, vol. 15, pp. 36–56, 2023.

[109] M. Mahmud, D. Joannic, M. Roy, A. Isheil, and J. F. Fontaine, "3d part inspection path planning of a laser scanner with control on the uncertainty," *CAD Computer Aided Design*, 2011.

[110] H. Choset and P. Pignon, "Coverage path planning: The boustrophedon cellular decomposition," in *Field and Service Robotics*, A. Zelinsky, Ed. London: Springer London, 1998, pp. 203–209.

[111] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html

[112] RoboDK. (2024) Robodk: Simulación de robots industriales. Accessed: 2024-08-11. [Online]. Available: https://robodk.com/es/

[113] S. Roos-Hoefgeest, M. Roos-Hoefgeest, I. Álvarez, and R. C. González, "Simulation of laser profilometer measurements in the presence of speckle using perlin noise," *Sensors*, vol. 23, no. 17, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/17/7624

[114] CIN Advanced Systems, "Cin advanced systems homepage," 2024, accessed: 2024-08-15. [Online]. Available: https://cinsystems.es/en/homepage/

[115] S. Roos-Hoefgeest, I. A. Garcia, and R. C. Gonzalez, "Mobile robot localization in industrial environments using a ring of cameras and aruco markers," in *IECON 2021 – 47th Annual Conference of the IEEE Industrial Electronics Society*, 2021, pp. 1–6.

[116] S. R.-H. Toribio, I. A. Garcia, and R. C. Gonzalez de Los Reyes, "Block matching mosaicing for surface inspection using an autonomous mobile robot," in *IECON 2021 – 47th Annual Conference of the IEEE Industrial Electronics Society*, 2021, pp. 1–6.

[117] S. Roos-Hoefgeest, J. Cacace, V. Scognamiglio, I. Álvarez, R. C. González, F. Ruggiero, and V. Lippiello, "A vision-based approach for unmanned aerial vehicles to track industrial pipes for inspection tasks," in *2023 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2023, pp. 1183–1190.

[118] S. Roos Hoefgeest Toribio, Á. Fernández García, I. Álvarez García, and R. C. González de los Reyes, "Localización de robots móviles en entornos industriales usando un anillo de cámaras," in *XL Jornadas de Automática*. Universidade da Coruña, Servizo de Publicacións, 2019, pp. 849–858.

[119] Á. Fernández García, S. Roos Hoefgeest Toribio, I. Álvarez García, and R. C. González de los Reyes, "Algoritmo de generación de trayectorias en el interior de chapas para la subsanación de defectos," in *XL Jornadas de Automática*. Universidade da Coruña, Servizo de Publicacións, 2019, pp. 702–709.