An Investigation of Automated Visual Inspection System Design for Defect Detection

Chenhao Ran

This thesis is submitted to the Department of Design, Manufacturing and Engineering Management, University of Strathclyde, for the degree of Doctor of Philosophy.

AUTHOR'S DECLARATION

This thesis is the result of the original research. It has been composed by the author and has not been previously submitted for examination which has led to the award of a degree.

Signed:  Data: 09/04/2025

Acknowledgement

First and foremost, I would like to express my deepest gratitude to Professor Xiu-Tian Yan, whose unwavering support and guidance have been instrumental throughout my PhD journey. His invaluable advice and the opportunities to engage in diverse research projects have greatly expanded my horizons in the world of research. I am deeply grateful to Professor Yan for providing me the opportunity to pursue my PhD at the University of Strathclyde, and for his invaluable help in securing funding that has supported my stipend and living expenses.

I would also like to thank Dr Andy Wong, who became my supervisor in the fifth year of my PhD. His unwavering support, patience and dedication were invaluable in helping me to navigate the final stages of my thesis with confidence.

I would like to thank my colleagues in my lab and the department of Design, Manufacture and Engineering Management, Dr. Leijian Yu, Dr. Beiya Yang, Dr. Baixiang Zhao, Dr. Muhammad Siddiqui, Dr. Amr Ahmed, Dr. Quang Le, and particularly Dr. Cong Niu, whose assistance in research and life has been immeasurable. I would also like to thank Ms. Youhua Li for her insightful advice, extending beyond academia. I would like to thank Mr. Duncan Lindsay, Mr. Dino Bertolaccini, Mr. Jim Mitchell, and all the technical colleagues who have provided essential support for my research.

I am very grateful to the Scottish Research Partnership in Engineering (SRPe) programme and Stellantis (formerly PSA Group) for their financial support.

Lastly, I express my eternal gratitude to my friends and family, whose love and understanding have been the cornerstone of my academic pursuits.

Publications

Minor Revision

[Chapter 5] **C. Ran,** X. Yan, 'Real-time Zero-Shot Wheat Head Disease Detection Algorithm Based on Domain Transfer and Anomaly Detection', Computers and Electronics in Agriculture, 2024.

Working Paper

[Chapter 4] **C. Ran,** O. Guezet, X. Yan, 'A Novel Hierarchical Knowledge Transfer DE-Net for Real-Time Crack Classification', Automation in Construction, 2025.

[Chapter 6] **C. Ran**, X. Yan, B. Zhao, E. Nougaredeb, 'A Comparative Study of Vision-based Car Engine Surface Anomaly Detection', IEEE Transactions on Instrumentation and Measurement, 2025.

Published

[Chapter 7] X. Yan, N. Oumer, M. Li, **C. Ran** et al., 'Vision enabled smart manipulations for in-space construction' *71st International Astronautical Congress* (IAC), Virtual, PP. 1-10, 2020.

Abstract

Non-destructive testing (NDT) methods are in high demand for defect detection across various industries, as they ensure the integrity and safety of materials and products without causing damage. Among NDT techniques, automated visual inspection (AVI) systems have gained significant attention due to their potential for efficient, accurate, and non-invasive defect detection. AVI systems offer a powerful solution by leveraging advanced imaging and artificial intelligence (AI) technologies to identify defects in real time. Based on defect localisation precision, defect detection tasks can be categorised into four levels: image-level, object-level, pixel-level, and 3D-level. Each of these levels necessitates distinct designs and techniques. The key challenges include coarse localisation and limited interpretability at the image level, object segmentation and ambiguity in defining object boundaries at the object level, fine-grained localisation and computational complexity at the pixel level, and the integration of spatial information with high computational cost at the 3D level. Despite their potential, several challenges hinder their industrial implementation, including the accuracy-efficiency trade-off, limited computational resources in practical applications, and data scarcity for training deep learning models. This thesis investigates the design and development of AVI systems for defect detection across four different localisation precision levels and diverse industrial applications. Through a systematic design methodology, this research addresses key challenges by optimising system architecture, data acquisition methods, and the detection algorithms to meet the specific requirements of each industrial application.

The key components of the AVI system design are comprehensively reviewed, including data acquisition, defect detection, and system design methodologies. Regarding the defect detection methods, a universal perspective on defect detection techniques is provided and the state-of-the-art methods are compared. Based on the literature review findings, several knowledge gaps are identified, including the absence of a comprehensive design methodology for AVI systems, challenges in real-time detection on devices with limited computational resources, and under-exploration of the anomaly detection methods in AVI systems.

The first contribution of the research is to develop a novel comprehensive design methodology to resolve the identified challenges of the AVI system design, with a focus on defect detection method development. Central to this methodology is a design process model, where the design evolves from an abstract and qualitative concept to a specific and quantitative embodiment, and finally a complete AVI system. This design methodology also emphasises the development

of defect detection method through a detailed algorithm selection and optimisation process, which can be tailored according to the availability of target data.

The second contribution is the implementation of the comprehensive design methodology in developing four AVI systems in different application domains. By adopting the proposed design methodology, four AVI systems are designed and developed in four case studies, targeting different defect localisation precision levels and diverse industrial applications. The first case study emphasises the accuracy-efficiency trade-off in computationally resource-constrained devices, while the other three case studies address the challenges posed by data scarcity. In the first case study, an image- and pixel-level concrete wall crack detection system is designed, incorporating a novel training strategy for agile development of AI models along with a novel model architecture. The second case study explores object-level detection of wheat head diseases, employing zero-shot learning and domain adaptation techniques to train an anomaly detection model without utilising any disease data. The third case study shifts to pixel-level car engine surface defect detection. It includes a comparative analysis to examine the impact of data collection configurations, anomaly characteristics, and anomaly detection methods. Based on these findings, the most suitable defect detection algorithm is selected and optimised. The fourth case study presents a 3D-level spacecraft anomaly detection system, illustrating the design of both image acquisition system and software platform for a modular spacecraft inspection system. These four case studies collectively demonstrate how the proposed design methodology can be implemented to address distinct requirements and challenges of each industrial application in a systematic manner.

Table of Contents

List of Figures

List of Tables

List of Acronyms

AI Artificial Intelligence

AVI Automated Visual Inspection

CNN Convolutional Neural Network

DL Deep Learning

DNN Deep Neural Network

DOF Degree of Freedom

GAN Generative Adversarial Network

GPS Global Positioning System

ICP Iterative Closest Point

ML Machine Learning

PID Proportional Integral Derivative

ROS Robot Operating System

1. Introduction

1.1. Motivation

1.1.1. Industry Demand for Non-destructive Testing

A defect is an anomaly, inconsistency, or imperfection in the physical appearance or structure of an object, which deviates from the established design specifications, standards, or quality thresholds [1-3]. Non-destructive testing (NDT) is a group of measurement techniques used to evaluate the integrity and quality of materials, components, or systems without causing any damage [4]. The primary goal of NDT is to detect defects, irregularities, or imperfections such as cracks, voids, corrosion, or internal flaws that may compromise the safety, performance, or longevity of the object being tested. Industries such as aerospace, automotive, manufacturing, construction, and agriculture place a high emphasis on quality assurance, prioritising the safety, reliability, and performance of their products and infrastructure.

The growing demand for NDT techniques is driven by several factors, including increased quality standards, cost efficiency, safety and reliability, automation and digitalisation, and so on [5]. The identification of defects prior to their manifestation as failures can prevent the necessity for costly repairs, product recalls and periods of downtime. In critical industries such as aviation, nuclear power, and infrastructure, it is of paramount importance to ensure the structural integrity of components to guarantee public safety [6]. Non-destructive methods provide a reliable means of detecting flaws that could potentially result in catastrophic failure, thereby enhancing the safety of products in service [7]. The integration of non-destructive testing (NDT) with automated systems is becoming increasingly prevalent in the context of Industry 4.0 [8]. Automated NDT systems are capable of inspecting large volumes of components with high precision, enabling real-time data collection and analysis. Digitalisation of inspection processes, which is enabled by these systems, has the effect of reducing human error and allowing for predictive maintenance and continuous monitoring of product quality. These factors contribute to the increasing adoption of advanced NDT technologies.

1.1.2. Non-destructive Testing Technologies for Defect Detection

NDT is an essential methodology employed across various industries to detect defects by various techniques, such as ultrasonic [9], thermography [10], tomography [11], machine vision [12], etc.

The ultrasonic method tests the defect by analysing the propagation of ultrasonic waves through the target, therefore can detect the internal flaws of the tested parts. The detectable defect size depends on the wavelength of ultrasonic wave, which is typically larger than 600 μm [9]. It usually requires physical contact with the target to obtain precise results, which may slow down the detection speed. Besides, to generate a 2D view of the target, several ultrasonic beams and corresponding receivers are needed, leading to an increased level of the system complexity [13].

The thermography method detects the temperature difference on the surface with an infrared (IR) camera. The temperature of the target can either be natural or heated by laser [10] or eddy current [14]. With the active heating, the internal defect is revealed by the surface temperature and thus can be detected [15]. Similar to the vision-based method, the thermography method is easy and fast to implement with IR camera, and the detectable size depends on the camera resolution. However, the applications are limited to materials and defects that are sensitive to temperature.

The tomography method can obtain a series of 2D images derived from the X-ray absorption signal after penetrating the target. A 3D model can be created from these images with post-processing algorithm [16]. X-ray micro computed tomography (microCT) can achieve resolution as high as 6 μm to detect the internal defect [17]. Although it has the highest defect sensitivity, the X-ray adaption imposes strict requirements on the installation, safety, and also the budget. Generally, it is not suitable for on-site inspection.

The machine vision NDT technique, often referred to as automated visual inspection (AVI), focuses on detecting defects by analysing the surface appearance of the inspected object. This method employs advanced imaging technologies to capture detailed images. The smallest detectable defect size depends on the resolution and focus of camera and distance to the target, and can be as small as 30 μm [12]. By leveraging automated systems, this approach enhances inspection efficiency and accuracy, making it a valuable tool in quality control processes across various industries [18].

Among the four methods, the vision-based method is the most accessible solution for defect detection with any off-the-shelf commercial camera. It also has the least processing time which is efficient for on-site real time inspection with relatively high accuracy. Therefore, this thesis will explore the application of AVI for defect detection.

1.1.3. Automated Visual Inspection for Defect Detection

Originated from manual inspection, visual inspection mimics human vision by scanning the target for visible defects, including surface imperfections such as cracks and dents, as well as shape anomalies such as misalignment and scale errors. AVI systems have a wide range of applications across various industries, significantly improving quality control and defect detection processes. In the manufacturing sector, AVI systems are used to inspect products for surface defects, dimensional accuracy, and assembly integrity, ensuring that only high-quality items reach the market [19]. In the automotive industry, they are used to examine components, such as engine parts and body panels, to detect flaws that could affect vehicle performance and safety [20]. With advances in artificial intelligence and machine learning, AVI systems continue to evolve, allowing real-time analysis and decision-making, ultimately improving efficiency and reducing costs in various applications [21].

Following common practices in computer vision [22-24], the defect detection task can be categorised into different precision levels based on the defect localisation precision, including **image-, object-, pixel-, and 3D-level defect detection**. The image-level detection methods report if defects are presented in the image. It focuses on identifying overall issues such as improper shapes, colour inconsistencies, or significant surface defects [25]. Object-level defect detection focuses on specific objects within an image and evaluates their quality or integrity. Object-level detection identifies the presence of defects related to distinct items, such as identifying flaws in a specific component or part of a product [26]. Pixel-level defect detection involves analysing individual pixels in an image to identify defects. This high-resolution approach enables the detection of very small imperfections, such as scratches, dents, or variations in colour [27]. 3D level detection focuses on capturing and analysing three-dimensional information about an object or surface. This approach allows for the identification of spatial anomalies, surface irregularities, or structural defects that may not be visible in 2D images [28]. In practice, each level of defect localisation precision necessitates the deployment and combination of various technologies.

Development of AVI systems faces several challenges [29, 30], such as accuracy-efficiency trade-off, data scarcity, environment variability, integration with existing infrastructure, etc. A significant challenge in AVI system design is the absence of a comprehensive design methodology that systematically guides the entire development process. Existing approaches often focus on individual components without providing an integrated framework that considers the interdependencies among these components [31-33]. This fragmented approach can lead to suboptimal system performance, inefficient resource allocation, and limited adaptability across different application domains. Therefore, establishing a comprehensive

design methodology is essential to address these issues, enabling a more holistic and efficient development of AVI systems.

Another significant issue is the accuracy-efficiency trade-off. While advanced algorithms, particularly those leveraging deep learning, can achieve high accuracy in defect detection, they often require substantial computational resources and processing time, which can be impractical in real-time applications [34]. This trade-off is particularly essential in many AVI systems with limited computational resources and specific system requirements, which can hinder the complexity and performance of the algorithms used [35].

Furthermore, the lack of defect images for training deep learning models also hampers the model's ability to generalize across various defect types, leading to lower accuracy and robustness in real-world applications. As obtaining diverse and representative datasets is crucial for developing robust algorithms capable of detecting various defects under different conditions [36]. However, the research of anomaly detection techniques is still in a relatively early stages of research and has been focusing on public datasets [37]. Their deployment in real-world AVI applications is still underexplored.

This work is motivated by the previously discussed challenges and focuses on the design and development of AVI systems for a range of defect detection tasks, from image level to 3D level, across distinct application areas. By adopting the proposed design methodology, the unique requirements of each inspection application are systematically addressed through system designs and the development of defect detection methods.

1.2. Research Aims and Objectives

In practice, the industrial applications often impose stringent demands on AVI systems, particularly regarding the accuracy-efficiency trade-off and data scarcity. To meet these demands, the system architecture, data acquisition methods, and defect detection techniques must be tailored and optimised for each specific industrial application. This thesis aims to address these challenges through the systematic design of four AVI systems. Firstly, a novel comprehensive design methodology is proposed, encompassing the entire process from specifications to system implementation. Following the proposed design methodology, the accuracy-efficiency trade-off will be addressed in the first case study, an image- and pixel-level concrete crack inspection system. The data scarcity will be addressed by three other case studies: object-level wheat head disease detection, pixel-level car engine manufacturing defect detection, and 3D-level spacecraft anomaly detection.

To achieve the research aim, the following objectives have been identified:

1. Conduct a detailed literature review on the state-of-the-art AVI systems, covering both the system configurations and the detection algorithms.
2. Develop a comprehensive design methodology for AVI system with a focus on defect detection method development.
3. Validate this design methodology through four case studies, each addressing specific requirements and challenges of various industrial applications. These case studies encompass image-level, object-level, pixel-level, and 3D-level defect detection.

1.3. Research Methodology

The methodology of this research focuses on the design, development, and evaluation of AVI systems, with an emphasis on visual-based defect detection. The research methodology consists of a combination of experimental, computational, and analytical methods to achieve the following objectives:

1.3.1. System Design and Development

The first stage of the methodology involves the conceptualization and design of the AVI system, focusing on the selection of appropriate sensors, cameras, and computational models. The design process includes tasks initialisation and task clarification, conceptual design, embodiment design, and system verification and validation.

1.3.2. Defect Detection Method

This stage focuses on building a robust defect detection method, which covers the following key applications:

1) Image-level Detection: Implementing methods for defect identification at the image level, focusing on coarse localization.
2) Object-level Detection: Developing object detection algorithms to segment the target objects from the background and identify defects within specific regions of interest.
3) Pixel-level Detection: Refining detection methods to localize defects at the pixel level with high precision.
4) 3D-level Detection: Exploring methods to incorporate 3D information from depth images and point cloud for more accurate defect localization in complex geometries.

These detection methods will employ advanced computer vision and deep learning techniques to enhance performance and robustness.

1.3.3. Dataset and Experimental Setup

A diverse set of real-world datasets will be used to train and test the AVI system. These datasets will cover various industries, including manufacturing, construction, and agriculture, with different types of defects such as cracks, surface irregularities, and discoloration. The experimental setup will involve:

1) Data Collection: Capturing high-resolution images of defective and non-defective objects using industrial cameras and 3D scanners.
2) Benchmarking: Comparing the performance of the proposed AVI system against existing state-of-the-art defect detection methods.

1.3.4. Evaluation Metrics

The performance of the AVI system will be evaluated using the following metrics:

1) Detection Accuracy: The proportion of true positives to the total number of defects.
2) Localization Precision: The accuracy of the system in localizing defects.
3) False Positive/Negative Rate: The rate at which the system falsely identifies defects or fails to identify actual defects.
4) Processing Time: The time required for defect detection and classification, which impacts real-time applicability.

1.3.5. Analysis and Optimization

After system evaluation, the results will be analysed to identify potential areas of improvement. Techniques such as hyperparameter tuning, data augmentation, and ensemble methods will be explored to optimize the system's performance. Sensitivity analysis will be performed to evaluate how different factors (e.g., lighting conditions, camera resolution, and defect types) affect the system's accuracy.

1.4. Thesis Structure

The remainder of the thesis is structured as in Figure 1.1.

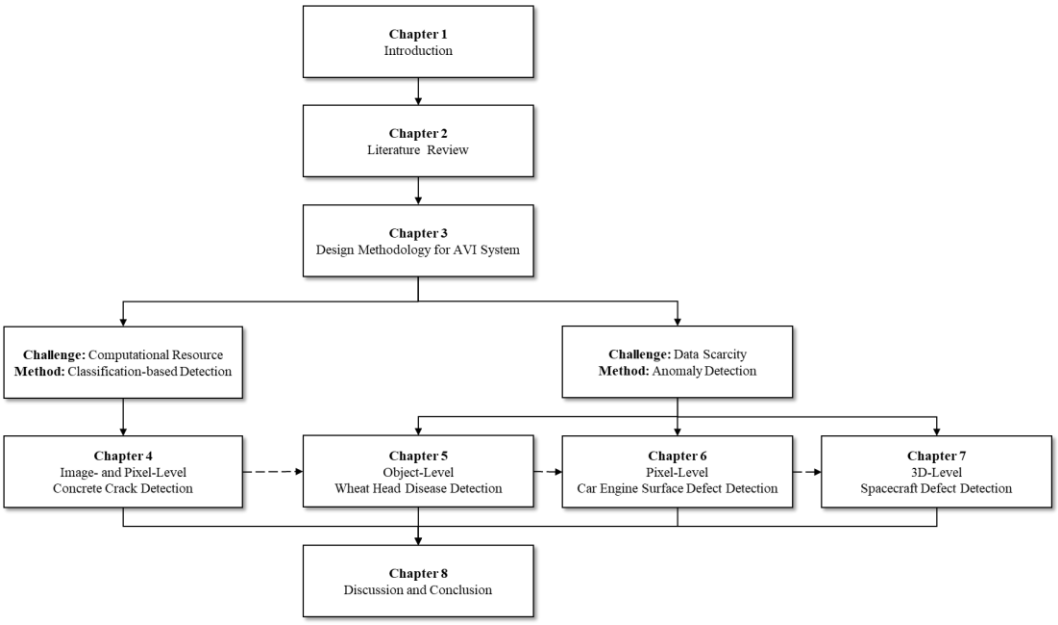


Figure 1.1. Organisation of the thesis.

Chapter 2 examines the current literature of AVI system and defect detection methods to identify the knowledge gaps. Different aspects of the AVI system are reviewed including the data acquisition, defect detection, defect management, and system design methodology. The chapter then delves into a detailed analysis of state-of-the-art defect detection methods, highlighting their effectiveness and application within the field.

Chapter 3 presents a comprehensive design methodology for AVI system design with a focus on defect detection method development. A design process model is proposed covering the four design stages, i.e., analysis of need, concept design, embodiment design, and system validation and verification. Specifically, the defect detection method development is described in detail covering both the dataset establishment and algorithm selection and optimisation. This design methodology is validated through four case studies as described in the following chapter.

Chapter 4 focuses on the design of a head-mounted vision system for concrete wall crack inspection as the first case study. This chapter addresses the challenge of achieving real-time image and pixel-level detection on computationally resource-constrained devices, such as the HoloLens 2 headset. To enhance accuracy and efficiency in crack detection, it introduces a novel training strategy along with a novel network architecture specifically tailored for this application.

Chapter 5 outlines a wheat head disease detection system designed for precision spray

applications as the second case study. Focusing on object-level detection without relying on disease training data, this chapter employs a zero-shot learning method for wheat head disease detection, utilising domain transfer and anomaly detection techniques to identify affected plants effectively.

Chapter 6 details a car engine surface defect detection system developed for manufacturing inspection as the third case study. This chapter focuses on pixel-level detection using only defect-free data for training. Additionally, it explores the use of synthetic datasets to mitigate data scarcity issues, enhancing the system's effectiveness in identifying surface defects.

Chapter 7 describes a vision inspection system designed for modular and reconfigurable spacecraft as the last case study. This chapter emphasises 3D-level detection through advanced 3D reconstruction and six degrees of freedom (DOF) pose estimation. It proposes a processing pipeline built on a developed reconfigurable software framework to enhance the system adaptability and effectiveness in inspection tasks.

Chapter 8 presents the discussion and conclusions of this thesis, highlighting the key contributions of the research. It also addresses the limitations encountered during the study and outlines potential directions for future work.

2. Literature Review of Automated Visual Inspection System and Defect Detection

2.1. Introduction

For automatic defect detection, various non-destructive testing (NDT) methods have been developed for data collection targeting both internal and external defects. For example, the internal defect detection includes the tapping sound, ultrasonic testing, X-ray tomography, etc. The external defect detection contains infrared thermography , automated visual inspection (AVI) [38] including both machine vision and laser scanner. Among these, AVI is one of the most popular methods and has been widely used in industrial applications due to its relatively low price, high efficiency and accuracy, continuous detection, non-contact measurement, etc. [21]. A complete and effective AVI system requires the integration of data acquisition system and defect detection methods [39, 40]. Additionally, a comprehensive design methodology is essential to ensure the seamless integration of these components and the overall performance of the system. Furthermore, as the core component of the AVI system, defect detection methods require careful examination and optimisation to ensure both accuracy and efficiency.

In this chapter, the common academic databases are utilised including Google Scholar, IEEE Xplore, Web of Science, and Elsevier. Specific terms are deployed to narrow down the search results. For AVI system, the general terms are used including ‘automated inspection system’, ‘defect detection system’, ‘defect inspection’, etc. Additionally, the search is further narrowed down through combined keywords for specific defect localisation precision, such as ‘defect classification’, ‘anomaly segmentation’, ‘3D anomaly detection’, etc. For each specific technique such as convolution neural network (CNN), the technique term is used in combination with applications. The screening process typically begins with an initial search using the selected keywords in the chosen databases. Results are filtered based on criteria such as publication date, document type, and subject area to refine the list. Following this, the titles and abstracts of the papers are reviewed to assess their relevance to the research question. Relevant papers are then examined in full to evaluate their contributions to the topic. Additionally, checking the references in relevant papers can reveal further sources that may have been overlooked.

Following the above procedure, this chapter begins by reviewing the key components of the AVI system and the system design methodology. It then focuses specifically on the state-of-

the-art defect detection methods. Finally, the knowledge gaps are identified to provide guidance for the direction of the thesis.

2.2. AVI System

Typically, an AVI system consists of two main components: data acquisition and defect detection [39, 40]. Data acquisition tries to capture and highlight the defects in the data, including the configuration of illumination, camera, and target object. The defect detection covers the data processing, defect detection tasks, and detection evaluation. Specifically, the defect detection tasks are classified based on the defect localisation precision level, ranging from image-level to 3D level. Upon detection of a defect, the details of the defect, such as type, location, severity, can be viewed by expert via a GUI, or decided by an automated decision system. To design a complex system involving the interaction of various components, a systematic design methodology is essential for ensuring the overall quality of the system and meeting all specified requirements.

2.2.1. Data Acquisition

2.2.1.1. Defect Types

Categorisation of defect types is crucial for AVI systems because it guides the selection of data acquisition devices, construction of databases, and development of detection algorithms. For AVI system, the target defect types are mainly the surface and shape defects that can be visually detected. Various defect categorisation methods have been reported and some common defects are classified and summarized in terms of different product and defect characteristics [41-43]. In this review, the defects are categorised based on their 2D and 3D visibility. Selecting appropriate illumination sources and cameras that align with the categorised defect types is essential for ensuring the accuracy and efficiency of the inspection process. By identification of the 2D and 3D visibility, the AVI system can be fine-tuned for each category, ensuring that the illumination conditions and camera settings are optimised to capture the specific features associated with these defects.

Some typical object defects are shown in Table 2.1, which are sorted according to their severity. The visibility is assessed qualitatively based on defect characteristics and experience with four levels, ranging from none, low, medium and high. All the defects are at least visible in 2D images. The identification of surface defects such as appearance and porosity are more suitable to use 2D camera-based methods. While for shape defects such as protrusion and deformation, point cloud-based methods are more feasible and accurate.

Table 2.1. Categorisation of defect types and their 2D and 3D visibility

Defect	Characteristics	2D Visibility	3D Visibility	Related Works
Appearance	Visual quality such as surface finish, colour, texture	High	None	[21, 44]
Scratch	Linear or curvilinear abrasion or marking on the surface	Low	None	[45-47]
Dent	Depression or concavity on the surface of an object, usually caused by an external force or impact	Medium	Medium	[48, 49]
Protrusion	Unwanted or out-of-place element that extends from the surface of the object	Medium	High	[28]
Porosity	Presence of small voids or air pockets within a material	Medium	Low	[50, 51]
Crack	Fissure or fracture in a material, often characterised by a linear separation of the material	High	Medium	[52, 53]
Deformation	Alteration in the object's shape or structure that deviates from its intended design	Medium	High	[53, 54]
Assembly	Issues in the assembly of parts or components, such as misalignments, missing components, or improper fit	High	High	[55, 56]

2.2.1.2. Illumination

High-quality images are essential for defect detection because image clarity and fidelity have a fundamental role in allowing accurate and reliable defect identification, thus ensuring the integrity of quality control processes. Generally, the image quality is affected by the optical illumination and camera. The primary purpose of an optical illumination platform is to highlight significant object features while minimizing undesired aspects. A specially designed optical illumination sometimes can be essential to make the defect visible.

The illumination modes within AVI systems can be categorised as forward or back illuminations, determined by the spatial arrangement of the light source, object, and camera. More specifically, these two techniques can be further divided as bright field forward lighting, dark field forward lighting, coaxial forward lighting, scattering forward lighting of dome structure, and back lighting [57]. These five illumination modes are compared in Figure 2.1. The forward lighting approach entails positioning the light source and the camera on the same side of the object under inspection. This method is widely adopted and particularly effective for detecting surface defects, such as scratches, as well as capturing crucial object details, including surface texture features. Forward lighting can be classified into two primary subtypes: bright field forward lighting, where light reflects directly onto the camera, and dark field forward lighting, where the incident angle of the light is adjusted to create a low-angle dark field. The latter emphasises surface edges and elevations, improving the depiction of topographical structures, and is effective in revealing surface concavities and convexities.

Coaxial forward lighting represents a unique variant within forward lighting modes. In this method, a high-intensity uniform light passes through a half mirror, aligning the light coaxially with the lens. This lighting technique is well-suited for detecting surface defects, cracks, and scratches. For highly reflective objects with smooth surfaces, scattering forward lighting can be adopted. Other illumination techniques include structured light , stroboscopic light , and other auxiliary optical devices such as filter, reflector, polariser, etc [58].

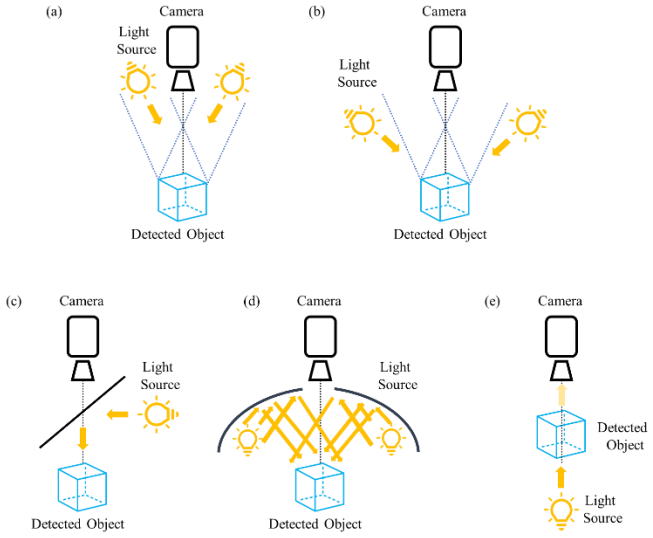


Figure 2.1. Illustration of typical illumination modes including (a) bright field forward lighting, (b) dark field forward lighting, (c) coaxial forward lighting, (d) scattering forward lighting of dome structure, and (e) back lighting. (redrawn from [57])

2.2.1.3. Camera

In the field of visual inspection, the dimension of image data used varies from 1D to 3D. The 1D line profile data is usually captured by linear CCD or line by line in 2D image. The 2D image data is the most commonly used data type usually captured from off-the-shelf CCD camera, due to its cheap price and easy to setup. Besides, data from X-ray tomography and infrared imaging are also regarded as 2D data. The 3D data have several types, including 3D point cloud, depth image and RGB-Depth (RGB-D) image, which contains much more information than 1D and 2D but usually expensive for high accuracy measurement and difficult to setup. In this part, these three data type and corresponding data acquisition equipment and methods will be briefly described.

1) 1D Data

The output format of 1D data is simply grayscale pixels along a line. The line profile is usually obtained from linear CCD or by processing 2D image line by line [59]. The linear CCD camera can usually satisfy high resolution and high-speed requirement. However, since the line

information is low intensity and it relies only on pixel value change along line profile, its application field and detectable defects are limited. For example, defects like scratch would not be detected through this data format.

2) 2D Data

Two-dimensional image is the most popular data format in AVI system and is closer to the way how human eyes work. It is also easier to develop high performance algorithms with 2D data than both 1D and 3D. The most used device for 2D image acquisition is CCD/CMOS camera [60]. Linear CCD camera can also be used to obtain 2D image by stitching the lines to form a complete image [61]. Besides, in some works, auxiliary devices are used for specific objects or defects, such as light control to form bright field and dark field image [62], or polariser to filter out the reflection light [63]. More detailed light source and illumination design can be found in [58]. Radiography (X-ray or gamma-ray) and infrared thermography (IRT) are another two commonly used data type in defect detection, and they all share similar image processing algorithms with CCD/CMOS camera. The comparison of application scenarios of three 2D data collection methods is shown in Table 2.2.

Table 2.2. Comparison of 2D data collection methods

Method	Application Scenarios	Limitations
CCD/CMOS	Surface or deformation defects that cause reflection light intensity variations	Relies on ambient light condition and reflection factor of material
Radiographic	Mechanical or structural internal defects that cause the absorption change of rays	Expensive, low speed, safety consideration, requires two-sided access to test object
Infrared	Electrical or mechanical defects that cause temperature anomaly	Expensive, low accuracy

2) 3D Data

Basically, there are four ways to obtain 3D data according to the device: monocular cues, binocular/multi-view stereo camera, structured light camera, and laser scanner. Monocular cues methods refer to using only one camera to reconstruct 3D object, including shape from shading, photometric stereo, shape from texture, and deep learning. These methods are solving ill-posed inverse problem and therefore requires strong constraints on the target object. For example, shape from shading method requires the surface image to be Lambertian and the directions of the surface normals distributed uniformly in 3D space [64]. While the shape from texture method requires texture uniformly distributed on the surface [65].

Deep learning method for single image 3D reconstruction has become popular in recent years but requires large dataset and the accuracy is still limited [66]. The binocular and multi-view

stereo vision reconstruction is based on the triangulation process, which refers to determine 3D coordinate of a point from its projection onto two or more images [67]. The intrinsic and extrinsic parameters of stereo cameras are usually calibrated before reconstruction, or self-calibrated during reconstruction in some techniques such as structural from motion (SfM) [68]. However, stereo camera methods heavily rely on object texture and ambient environment light. By introducing active and structured light source, the reconstruction can be much more accurate and robust [69]. In addition, with the advance of 3D scanning technology, the commercial laser scanner has become cheaper and more accurate [70]. The comparison of these four 3D reconstruction methods is shown in Table 2.3.

Table 2.3. Comparison of 3D data collection methods

Method	Data	Precision	Price	Limitation
Monocular Cues	RGB-D	Low	Low	Ill-posed inverse problem that needs many constraints
Stereo Camera	RGB-D	Medium	Low/Medium	Heavily relies on object texture and ambient light
Structured Light	RGB-D	High	Medium	Difficult for reflective or transparent surface
Laser Scanner	Point cloud	High	High	Expensive, no RGB information, difficult for reflective or transparent surface

2.2.1.4. Defect Datasets

The availability of diverse and comprehensive datasets is of paramount importance for the development and training of AVI systems, as it enables these systems to recognise and accurately classify a wide range of defects, enhancing their effectiveness in quality control across various industries. Gathering data is typically a resource-intensive and costly endeavour. Contemporary research often capitalizes on publicly available datasets to streamline and enhance the research process. Table 2.4 lists the publicly available datasets identifying defects in various materials and defect types.

Table 2.4. Publicly available datasets for defect detection

Name	Objects	Defects	Label	Task	Resolution	Quantity (train/val) (good/defect)	Illumination	Camera	Image
NEU-DET [71]	Hot rolled steel	Rolled-in scale, patches, crazing, pitted surface, inclusion, scratches	Bounding box with classes	Detection, classification	200x200	1440/360 0/1800	Bright field forward lighting	Area scan CCD	Grayscale
GC10-DET [72]	Metal	Punching, weld line, crescent gap, water spot, oil spot, silk spot, inclusion, rolled pit, crease	Bounding box with classes	Detection, classification	2048*1000	3570	Direct current light source	Linear array CCD	Grayscale
KolektorSDD [27]	Electronic Commutator	Scratch and crack	Pixel-wise mask	Segmentation	500*1240	399 347/52	-	-	Grayscale
KolektorSDD2 [73]	Metal	Scratches and dents	Pixel-wise mask	Segmentation	230*630	2331/1004 2979/356	-	-	Colour
SD-saliency-900 [74]	steel	Inclusion, Patches, and Scratches	Pixel-wise mask	Segmentation	200*200	1800 900/900	-	-	Grayscale
Magnetic Tile [75]	Magnetic Tile	Blowhole, crack, break, fray, unenven	Pixel-wise mask	Segmentation	248*373	1344	-	-	Grayscale
DeepCrack [76]	Wall and Pavement	Crack	Pixel-wise mask	Segmentation	544*384	300/237 537	-	-	Colour
MVTec AD [37]	10 Objects and 5 textures	70 different types of defects such as scratches, dents, contaminations, and various structural changes	Pixel-wise mask	Anomaly classification and segmentation	700*700 1024*1024	3629/1725 4096/1258	-	RGB sensor with two bilateral telecentric lenses	Colour
MVTec 3D-AD [77]	10 objects	41 types of anomalies, such as cracks or uneven areas	Pixel-wise mask	Anomaly classification and segmentation	1920×1200	2656/294 3199/948	Indirect and diffuse light source	Zivid One+ Medium structured light camera	RGB-D, point cloud

2.2.2. Defect Detection

Once the data collection is complete, the defect detection process can be carried out. Usually, the raw data cannot be directly utilised for detecting defects because of the noise from the environment or device, data imbalance, format compatibility, etc. After the data processing, the defect detection tasks can be performed to identify and locate defects. These tasks are classified by the level of localisation precision, ranging from image-level, object-level and pixel-level detection to 3D-level detection. Evaluation is then performed to assess the developed detection model.

2.2.2.1. Data Processing

Though the preprocessing algorithm is highly dependent to the data type, there are some common techniques like denoising, clustering, alignment, and augmentation. The noises may come from the environment and electronic components, which reduce the data quality and the affect the defect detection accuracy. The denoising algorithms can be divided into spatial domain methods and frequency domain methods. The spatial domain methods operate directly on the pixel values of an image or signal. They analyse and process data in its raw form, which is usually represented as a grid of pixel values. Typical spatial domain methods include histogram equalization [78], image sharpening [79], and various filtering algorithms such as Gaussian filtering and biliteral filtering. The frequency domain methods transform data from the spatial domain to the frequency domain using techniques like the Fast Fourier Transform (FFT). These methods are based on the idea that noise often has distinct frequency characteristics that can be separated from the desired signal. Key frequency domain denoising methods include Fourier transform [80], wavelet transform [81], and wiener filtering [82].

The clustering methods are employed to group or segment pixels or regions of an image with similar characteristics. These techniques can serve as either preprocessing, defect detection, or postprocessing methods. For preprocessing, the clustering methods are usually used for segmentation, image enhancement, and feature extraction. For example, clustering methods can be employed for image segmentation, dividing an image into distinct regions or objects based on pixel similarity. Clustering methods can also be applied to enhance image quality. By grouping pixels or regions with similar properties, image enhancement techniques can be applied to each cluster.

Alignment and registration techniques are imperative in the preprocessing of images as they assist in aligning multiple images and ensuring that an image is situated and oriented correctly for subsequent analysis. Image stitching is one of the mostly used alignment method. When

capturing multiple images of a scene, alignment methods match common features or control points in the images and adjust the positions and orientations to create a seamless composite image. This is widely used in applications like landscape photography and medical imaging [83]. Besides, alignment is crucial in tracking moving objects or subjects across a series of images or frames in video sequences [84].

Data augmentation techniques play a significant role in defect detection by increasing the diversity of available data and improving the robustness and generalization of machine learning models. These techniques are used to create variations of the original data, which can help the model learn to recognise defects in different scenarios. The data augmentation techniques can be divided into two main directions: image modification and generation. The image modification includes rotation, flipping, scaling, translation, shearing, brightness and contrast adjustment, noise injection, colour manipulation, cropping, elastic deformation, occlusion, etc [85]. Style transfer is another direction of image modification, which manipulates sequential representations across a neural network to transfer the style of one image to another while maintaining its original content [86].

2.2.2.2. Defect Detection Tasks

From coarse to fine, the detection task of the algorithms can be categorised as image-level, object-level, pixel-level, and 3D-point-level detection. For image level detection, the image is classified according to whether the defect is present. Traditional methods rely on the hand-crafted feature extraction methods, while machine learning based methods improve the accuracy by training based method, such as support vector machine (SVM) [87] and KNN [88]. Modern technology that uses neural networks has proven great efficiency and accuracy in defect classification utilising the advance of computation power and big data. The convolutional neural network is the most popular technology for image classification, such as AlexNet [89] and ResNet [90].

For object level detection, the defect is detected as bounding box in the image. There are two branches in object level defect detection: image patch classification and end-to-end CNN object detection. The image patch classification method divides the whole image into small patches and classify each patch as defective or defect-free. Then the defective patches are grouped together to form a defect region [91]. However, this kind of method is usually time-consuming and only suitable for single class defect detection with uniform background. The object detection method is more popular with the advance of modern CNN architecture such as R-CNN [92], YOLO [93], SDD [94], etc.

Pixel level defect detection is the highest level of defect localisation for 2D image. Important references for evaluating the severity of defects and assessing conditions can be obtained from the results of segmentation. Traditional methods based on the pixel value or frequency information, such as clustering [95] and region growing [96]. Similar to image classification, the CNN-based methods have become one of the most dominant methods in image segmentation, such as fully convolutional network (FCN) [97], U-Net [98], and Mask R-CNN [99]. Recently, the segment anything model (SAM) just appeared with ability of image segmentation according to the prompt from users [100]. This model was trained with 1 billion carefully masked dataset. The usage of SAM has been evaluated in medical image segmentation [101], infrastructure defect assessment [102], fabric defect segmentation [103], etc. However, there is considerable room for performance improvement when using SAM on real-world applications [104, 105].

3D level defect detection involves quantifying the size, depth, or other characteristics of a defect within an image. In vision-based system, the measurements can be done by either RGB camera or reconstructed point cloud. With a pre-calibrated camera and controlled image capture distance, the geometry shape from a single camera can be calculated [106], for example, dimensional measurement of hot parts [107] and crack detection and measurement [108]. When depth information is critical, depth-sensing technologies that use time-of-flight or structured light cameras are utilised to measure the 3D attributes of defects. By outlier remove and model fitting, the defects from point cloud can be detected and measured directly, for example curved surface defect monitoring [109], automated fibre placement defect measurement [110], etc.

2.2.2.3. Evaluation Criteria

When developing defect detection algorithms, there are several important evaluation criteria, such as training time, accuracy, inference time, and model size [22, 23]. Training time is a fundamental factor to consider as it affects the development cycle and resource requirements. The training time depends on the data size, model size, hardware, and optimisation techniques. Improvement of training time usually relies on model architecture optimisation such as depth-wise separable convolution in MobileNet [111], finetune of pre-trained network [112], data cache [113], or simply upgrade the hardware through cloud computation. Similarly, the inference time depends on the model efficiency and hardware. The model architecture optimisation methods are the same as the one in training time reduction. Other way to reduce the inference time is by model size reduction, include model compression through distillation or quantization [114], model pruning [115], or compiler optimisation [116]. The size of the

model impacts various performance aspects, such as training time, accuracy, and inference time. Nonetheless, larger models do not necessarily guarantee greater accuracy and smaller models do not guarantee faster training and inference time [117]. The balance of the three criteria depends on the specific requirements of the application. Accuracy is more crucial in off-line detection, as it prioritizes the quality of results; inference time is essential for real-time applications, where fast decision-making is critical; and model size is particularly important for embedded systems, where both memory and computational resources are limited.

For accuracy requirements, the detection rate is mostly used in defect detection, no matter the original task of algorithms is classification [118], bounding box detection [119], or segmentation [75]. Firstly, the confusion matrix and its terminology are defined in Table 2.5.

Table 2.5. Typical definition of confusion matrix

	Actual Positive	**Actual Negative**
Predicted Positive	True Positive (TP)	False Positive (FP)
Predicted Negative	False Negative (FN)	True Negative (TN)

The detection rate or accuracy is then defined as:

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{2-1}$$

Other criterions such as precision, recall, f1-score [120] are defined as:

$$precision = \frac{TP}{TP + FP} \tag{2-2}$$

$$recall = \frac{TP}{TP + FN} \tag{2-3}$$

$$F1 = \frac{2TP}{2TP + FP + FN} \tag{2-4}$$

Similarly, the Receiver Operating Characteristic (ROC) curve is created by plotting the True Positive Rate (TPR) on the y-axis and the False Positive Rate (FPR) on the x-axis for different threshold values [121]. Each point on the curve represents the TPR and FPR at a specific threshold. The Area Under the ROC Curve (AUROC) quantifies the overall performance of the classification model. It measures the area under the ROC curve. A higher AUROC indicates a better model. An example ROC curve is shown in Figure 2.2.

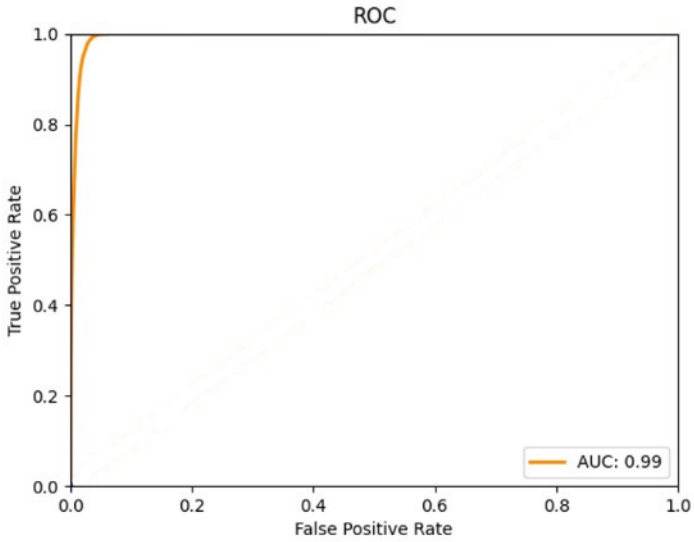


Figure 2.2. Example image of ROC curve with AUCROC of 0.99 for car engine anomaly detection.

Specifically, for bounding box detection or segmentation, a correct prediction is a bounding box or segmentation area that has Intersection of Union (IoU) with ground truth larger than a threshold [122]. The IoU can be calculated as:

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union} \tag{2-5}$$

The dice coefficient is often considered more forgiving than IoU when assessing the similarity between regions, as it doesn't penalize small differences as harshly [123]. It can provide a more robust measure of similarity, particularly when regions are expected to be similar but not perfectly identical. The calculation of dice coefficient is the same as the F1 score.

In bounding box detection, the mean Average Precision (mAP) is more popular criterion than the others [124]. The mAP is calculated by averaging the AUC-PR of each object class of interest. Typically, the mAP is accompanied by a threshold ranging from 50 to 90, indicating that a bounding box is considered a true positive if its overlap area with the ground truth exceeds this threshold. Raising this threshold value signifies a more rigorous evaluation. Formally, the mAP is defined as:

$$\text{mAP} = \frac{1}{N}\sum_{i=1}^{N} \text{AP}_i = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} \text{P}_j \tag{2-6}$$

For 3D defect detection, the per-region overlap (PRO) metric can be used, which is defined as e average relative overlap of the binary prediction P with each connected component C_k of the ground truth, defined as:

$$PRO = \frac{1}{K}\sum_{k=1}^{K}\frac{|P \cap C_k|}{|C_k|} \tag{2-7}$$

where K represents the total number of ground truth components. This process is iterated for different thresholds and a curve is plotted by the resulting PRO values against their corresponding false positive rates. The final performance assessment is obtained by calculating the integral under this curve, up to a specified false positive rate limit, then normalising the resulting area within the [0,1] range. This method serves as a conventional metric for unsupervised anomaly localisation tasks and is especially beneficial when dealing with anomalies that exhibit significant variations in size [77].

In some cases, besides the detection rate, the dimensional error of the detected defects is also important, especially in 3D measurement. The criterion for dimensional error can be expressed as relative error:

$$Relative\ \mathrm{E}rror = \frac{|D_T - D_M|}{D_M} \tag{2-8}$$

where D_T is the true dimension and D_M is the measured dimension.

2.2.3. System Design Methodology

The system design methodology provides a clear framework that helps developers understand the requirements, avoid redundancy and wasted resources, and maintain consistency during development. In the domain of engineering design, the methodological backbone is often structured as design process model that defines the necessary stages and decision points required to transform a requirement into an appropriate product. Based on the design practice observed in industry, Frech proposed a stage-based design process model [125], as shown in Figure 2.3. It outlines the design process as a sequence of interconnected phases, beginning with identifying a requirement that leads to an examination of the issue. During conceptual design phase, concepts are conceived and evaluated for their practicality. As the process evolves, these embodiments are detailed meticulously, resulting in working drawings and other documentation necessary to realise the design in the physical world. The model highlights the significance of feedback as a means for continuous enhancement, emphasising the importance of an agile approach rather than a fixed linear progression.

A more well-known design process model was proposed by Phal and Beitz for mechanical design [126], as shown in Figure 2.3. This model not only depicts a clear pathway from task to solution but also integrates key elements such as market considerations and economic

constraints, recognising the influence of external factors on design decisions. Through its comprehensive structure, the Pahl and Beitz design process model offers a robust framework for the systematic development of mechanical designs.

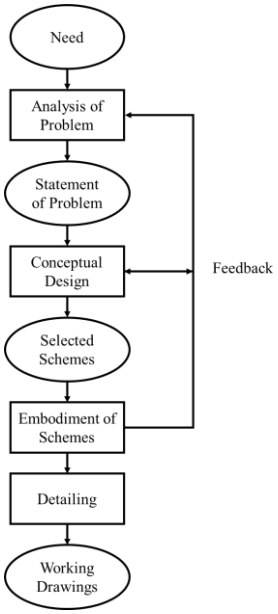


Figure 2.3. French's design process model (redrawn from [125])

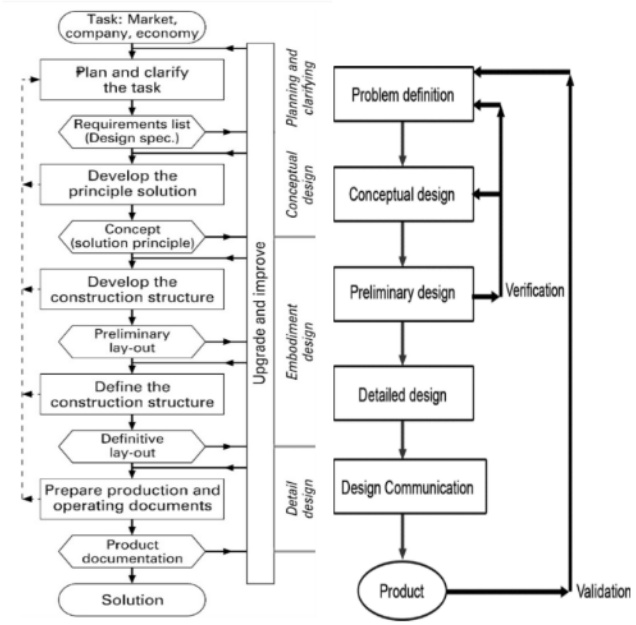


Figure 2.4. Phal-Beitz design process (courtesy of [126])

Yan's mechatronic design process model is based on an enhancement of French's model through integrating proactive design and concurrent engineering principles tailored to mechatronic systems [127], as shown in Figure 2.5. These phases involve generating both qualitative and quantitative models. Concurrently with the development of the mechatronic

system model, the Manufacturing/Assembly Tool Element Model guarantees the integration of manufacturing and assembly processes. This model emphasises a multi-perspective model construction, including system function model, product assembly model, control program model, etc.

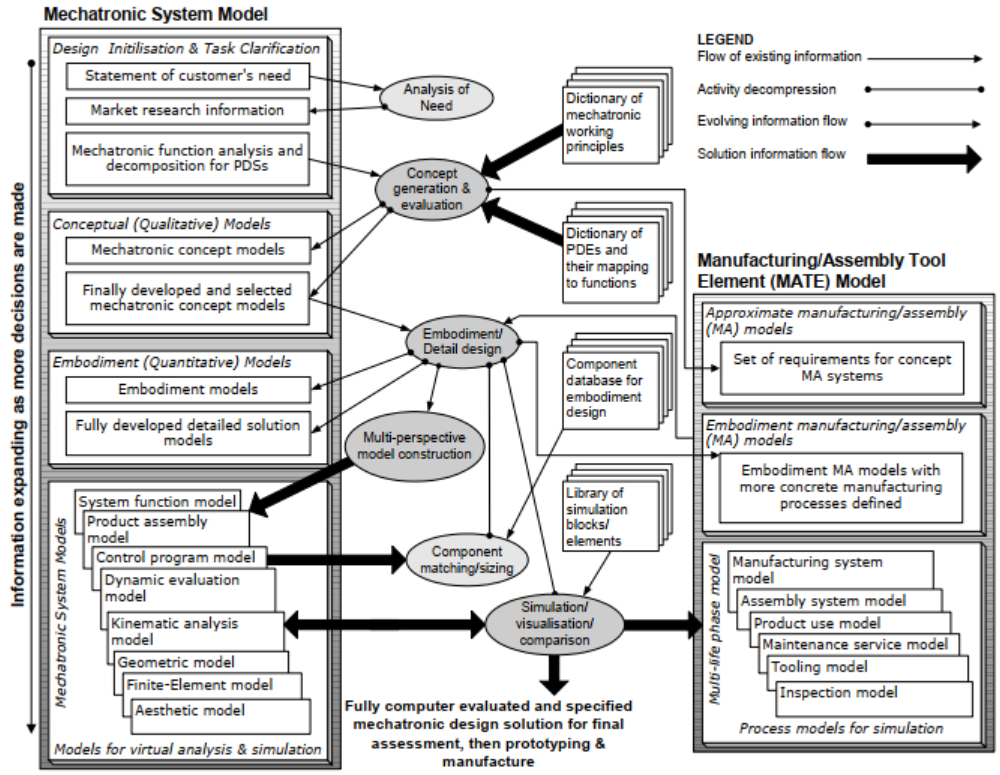


Figure 2.5. Yan's design process model for concurrent mechatronic system and manufacturing/assembly design process model (courtesy of [127])

The V-model is highly regarded for its efficacy in dealing with the complexity of multidisciplinary system design, especially in the realm of mechatronics. As a framework, it has been standardised by VDI committee as the VDI guideline 2206 [128], a practice-oriented guideline for the systematic development of mechatronic systems. This model advocates for the seamless integration of discipline-specific design fields - mechanical engineering, electrical engineering, and information technology - to meet the requirements of each sector with precision, as shown in Figure 2.6.

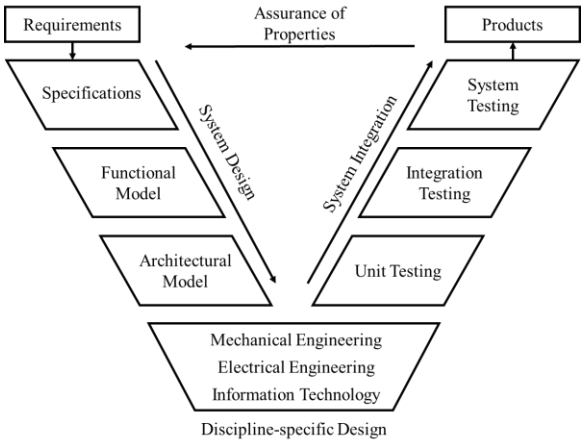


Figure 2.6. V-model in the VDI guideline 2206 (modified from [128])

Designed specifically for AVI system, Batchelor et al. [31] proposed a design methodology for industrial vision system targeting smooth and effortless design. It includes client-involved and in-house activities at different development stages. However, this framework is overly specific to manufacturing industries and lacks the flexibility for broader application. Additionally, the hardware and software design are limited to opto-mechanical systems and image processing algorithms, restricting its use in modern complex AVI systems. Furthermore, it does not include a system validation component, which is critical for ensuring the system's performance and reliability.

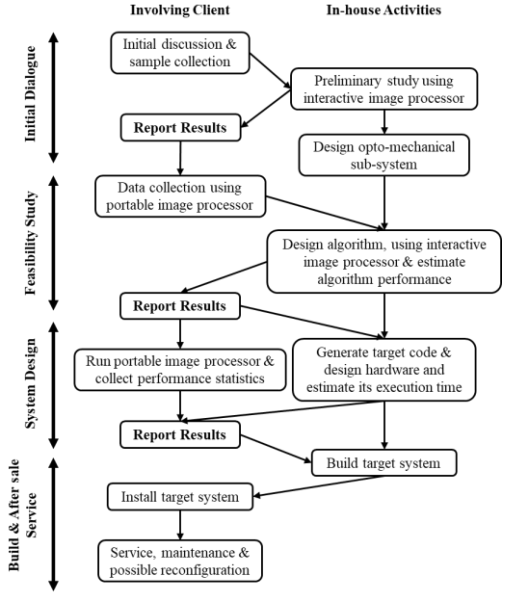


Figure 2.7. Design process of AVI system proposed by [31]

Saha and Bhattacharyya [129] described the design methodology for embedded computer vision system as five subproblems including modelling, specification and transformation, partitioning and mapping, scheduling, design space exploration, and code generation and

verification. Each of the subproblem was discussed separately. While this methodology provides an in-depth analysis of individual components, the lack of a comprehensive design process framework may hinder the ability to understand the project from a holistic perspective.

A comprehensive design refers to a systematic, integrated approach that considers all aspects of the design process in a cohesive manner, ensuring that each component is aligned with the overall project goals and constraints. A comprehensive design methodology would encompass not only the individual subproblems (such as specifications, data collection, defect detection method development, etc.) but also how these subproblems interrelate and influence one another throughout the entire design process. A comprehensive design helps ensure that the system's design is robust, well-coordinated, and adaptable to changing requirements [130].

2.2.4. Summary

From the above literature review, it is evident that designing an AVI system requires a multidisciplinary approach to integrate both the data acquisition and defect detection. The optimal data acquisition method varies depending on the type of defect being inspected, while the defect detection method must align with the specific data type and the required localisation precision. To ensure seamless integration of these components, a comprehensive system design methodology is essential to support a structured development of an AVI system. However, most existing system design methodologies are either too general or too specific to a particular industrial application, hence lack systematic guidelines that can generalise the design process of AVI systems as well as making the process more accessible. Additionally, despite the availability of some public defect datasets, data scarcity has been one of the major challenges hindering the development of defect detection methods for specific applications.

2.3. Defect Detection Methods

This section aims to provide a systematic and unified framework for understanding defect detection. Specifically, the proposed taxonomy categorises detection methods into three main groups based on their approach to define defect: defect similarity detection (DSD) [131], classification-based detection (CBD) [132], and anomaly detection (AD) [133]. The defect similarity detection methods treat defects as distinct features and employs feature extraction methods to detect them. The classification-based methods model the defect detection as a classification task and utilising machine learning models trained on both defect and normal data. The anomaly detection methods try to assess the deviation of test image from some concept of normality to identify defects. With this framework in place, a comprehensive

review of defect detection methods from the last three decades is undertaken, providing reference to the most appropriate methods for different application scenarios.

2.3.1. Method Taxonomy

In this thesis, the term "defect" denotes a broad concept of imperfection resulting from a fault in either the production process or during the usage. It is characterised as something that falls short of being considered perfect. To formally define the defect detection methods, let $X \in \mathbb{R}^D$ be the data space of D dimension, and $N = \{S_N \mid S_N \in X\}$ and $A = \{S_A \mid S_A \in X\}$ be the normal set and defective (abnormal) set, in the data space X. The task of defect detection is to classify a sample S as normal or defective. The most intuitive way to determine if a sample is defective is by how this sample differs from the normal set. Based on this, the defectiveness of sample $S \in X$ can be defined as the distance between this sample and normal data set N after certain space projection:

$$defectiveness = dist\left(f_\theta(S), \varphi\left(f_\theta(N)\right)\right) \tag{2-9}$$

where $f_\theta: X \to F$ is the projection function that maps the samples from original data space to representation space, typically named feature map, differing in algorithms. Function $\varphi: F \to F$ aggregate the representations of the normal data, such as cluster center, Gaussian mixture model, etc. Function $dist: (F, F) \to R$ calculates the distance between two representations, such as Hamming distance or L2 norm. This kind of method that determines the defectiveness based on the distance from target to normal data set is named as anomaly detection method.

Another manner to determine defect is by how close this sample is to the anormal set. Alike to the anomaly detection method, define the defectiveness of sample S as the similarity between this sample and abnormal data set A after certain space projection:

$$defectiveness = simi\left(f_\theta(S), \varphi\left(f_\theta(A)\right)\right) \tag{2-10}$$

where function $simi: (F, F) \to R$ calculates the similarity between two representations. In the representation space, the similarity can be seen as the reciprocal or negative of the distance. This kind of method that determines the defectiveness based on the similarity between target and abnormal data set is named as defect similarity detection method.

The classification-based method stands in between, and it takes into consideration of both the normal dataset and abnormal dataset. The defectiveness of sample S is defined as the distance between both the datasets after certain space projection:

$$defectiveness = cls\left(f_\theta(S), \varphi\left(f_\theta(N)\right), \varphi\left(f_\theta(A)\right)\right) \tag{2-11}$$

where the function cls: $(F, F, F) \rightarrow R$ calculates the probability or score that the sample S belongs to the normal dataset or the abnormal dataset. This kind of method that models the defect detection as a classification task is name as classification-based detection method.

The selection of distance function depends on the feature extractor f_θ and domain modelling φ. In some cases, φ is an identity function that does not change the domain, in which case the distance can be point-to-point distance, such as Euclidean distance. In other cases, the features can be modelled as a distribution, and the distance can be point-to-distribution distance, such as Mahalanobis distance [44].

From the definition, the three kinds of algorithms can be distinguished from both development stage and inference stage. In the development stage, the defect similarity detection method emphasises modelling of the normal data domain, i.e., $\varphi\left(f_\theta(N)\right)$. In opposite, the anomaly detection method emphasises the modelling of the abnormal data domain, i.e., $\varphi\left(f_\theta(A)\right)$. While the classification-based detection method requires both the data domain. During the inference stage, the defects are detected based on the defectiveness described above. The inference speed and model storage depend on the specific implementation of each method. Because the classification-based detection requires the largest number of data, its accuracy is usually the best. The taxonomy of defect detection methods is illustrated in Table 2.6.

Table 2.6 Taxonomy of defect detection method

Category	Defectiveness	Description	Require Normal Data	Require Abnormal Data
Defect Similarity Detection	$simi\left(f_\theta(S), \varphi\left(f_\theta(A)\right)\right)$	Based on the similarity of the data feature to the defect	No	Yes/No
Classification-based Detection	$cls(f_\theta(S), \varphi\left(f_\theta(N)\right), \varphi\left(f_\theta(A)\right))$	Based on the classification of the data feature	Yes	Yes
Anomaly Detection	$dist\left(f_\theta(S), \varphi\left(f_\theta(N)\right)\right)$	Based on the discrepancy of the data feature from the norm	Yes	No

Through this taxonomy, even methods from the same technique family may be grouped into different categories due to their different underlying principle. One example is that both the support vector machine (SVM) method and support vector data description (SVDD) method utilise support vectors to define the boundary. But the former one classifies the defects via the hyperplane constructed by normal and abnormal data. While the hyperplane of the latter one is formed by only the normal data. Therefore, the SVM is classification-based method, but the SVDD is anomaly detection method. This highlights one of the advantages of this taxonomy

that it can reveal the fundamental principles behind these methods and potentially inspire the incorporation of techniques based on different principles. Based on this taxonomy, the remaining part of this section will review the existing defect detection methods, including both conventional and machine learning approaches.

2.3.2. Defect Similarity Detection

In the defect similarity detection methods, the features or characteristics of defects are firstly defined, followed by pattern recognition of these features. Normally used features include grey value, edge, line, etc. This part will review the defect similarity detection methods include edge detection, clustering, Hough transform, Fourier transform, and active contour model.

A. Edge Detection

The edge is defined as curves in image where the brightness changes sharply. For material with smooth surface such as metal plate, the presents of edges are usually caused by defects such as scratch or crack. Therefore, the defect detection is simply modelled as edge detection with the extracted edge treated as defects. The edge extraction methods include Sobel and Canny operator [134], edge drawing [135], etc. One big challenge in edge-based method is that it is sensitive to uneven illumination and camera noise, leading to many false positive detections. The edge detection is a simple yet effective method for defect detection for materials with uniform coloured surface. However, it is prone to output false positive and heavily relies on image preprocessing procedure to remove noises.

B. Clustering

This method is based on the similarity among data or feature values and classification these values into different groups. In defect detection, the pixels of different defect types are clustered, and the classification of defect is by calculation the similarity of clusters. There are different cluster methods leading to different distance calculation. For example, in centroid based clustering such as K-means[136], each cluster is represented by a central point and the similarity is how close between the data point to the central point. While in distribution-based clustering such as Gaussian mixture model, each cluster is represented by a distribution whose parameters are optimised to fit the data in that cluster [137]. The similarity is defined as the probability that a data point is from this distribution. The clustering-based algorithms with defect similarity detection as principle are basically used in defect classification tasks with predefined defect patterns.

C. Hough Transform

The Hough transform is a mathematical and computational technique used in image processing and computer vision for the detection of shapes, lines, curves, and patterns. The Hough transform is based on the idea that specific geometric patterns, such as lines and curves, can be represented in parameterized form. The Hough transform accumulates votes in an accumulator array based on the parameters of the patterns detected. Theoretically, the Hough transform is able to detect any features that can be described in parameterized form. For defect detection, the defects are assumed to be line or curve features which can be detected through Hough transform [138]. However, the Hough transform can be computationally intensive, especially for high-resolution images or complex patterns. It also needs proper selection and configuration of the parameter space for a specific detection task.

D. Fourier Transform

In image processing, the Fourier transform (FT) provides the frequency domain information of the image. On the surface of material with smooth texture, the defects usually appear as high frequency term in the frequency domain. Wang et al. [139] defined the defect on magnetic surface as bright lines in spectral image after Fourier transformation and used Hough transform to detect the bright lines. The spatial image was reconstructed after remove of the lines. The defects were detected by comparison of the original and reconstructed images. In [140], through analysis of the spectral features of simulated defects of fabric, two significant spectrum diagrams were defined for defect classification including double yarn, missing yarn, webs or broken fabric, and yarn densities variation.

E. Active Contour Model

Active Contour Model, often referred to as a "snake," is named after the analogy of a snake seeking to adjust its shape to align with the boundaries of objects or regions within an image [141]. In defect detection, this method is used as edge detector to localise the defects, based on the assumption that defects usually have clear edges in smooth texture background. Yang and Marchant [142] used a two-step segmentation and refinement scheme for fruit blemishes detection. The blemishes were segmented out roughly by flooding algorithm and refined by the snake algorithm. To increase the robustness of the model, Yang et al. [143] improved the active contour without edges model to detect defects with fuzzy boundaries and uneven illumination.

2.3.3. Classification-Based Detection

The classification-based detection methods typically follow a two-step framework consisting of feature extraction and classification. Basically, the representation power of the selected feature space is the most critical part of the classification-based detection. In this part, different feature representations will be reviewed together with the paired classifier and applications in defect detection, including adaptive thresholding, co-occurrence matrix, local binary pattern, Fourier transform, Gabor filter, wavelet transform, convolutional neural network, and vision transformer.

A. Adaptive Thresholding

Thresholding based method separate the data into normal and abnormal sets by a threshold value. In defect detection, adaptive thresholding method is usually used which derives the threshold value adaptively taking into account the spatial variations in illumination and changes dynamically over the image [144]. The feature of this method is pixel value distribution. The classifier determines the threshold based on the grey value distribution among a local image patch or the whole image. The threshold is determined adaptively according to the pixel value distribution, i.e., the histogram. For example, the author detect the defects by finding the valley in the grey value histogram through the proposed valley-emphasis method revised from Otsu method [2]. The thresholding methods are the most intuitive methods and highly explainable. However, unlike the other classification-based method, the dynamic thresholding mechanism adjust the threshold based on the current test image and cannot leverage the advance of big data.

B. Co-occurrence Matrix

A Grey-Level Co-occurrence Matrix (GLCM) is a statistical method used in defect detection and image processing to analyse the spatial relationships between pixels of different grey levels in an image. It is a matrix that quantifies how often different pairs of grey levels occur in close proximity to each other in the image. For an image with N different pixel values, the $N \times N$ co-occurrence matrix $\boldsymbol{C}$ of an image I of size $W \times H$ is defined as:

$$\boldsymbol{C}_{\Delta x,\Delta y}(i,j) = \sum_{x=1}^{W}\sum_{y=1}^{H} \begin{cases} 1, & if\ I(x,y) = i\ and\ I(x+\Delta x, y+\Delta y) = j \\ 0, & otherwise \end{cases} \tag{2-12}$$

where $\Delta x, \Delta y$ are the position offset. The (i, j) value in the matrix is the number of times in the image that pixel value at (i, j) occur in the relation given by the offset. The GLCM is a feature discriminator and cannot be used directly as defect detection method.

The GLCM was first presented by Haralick et al. [145], based on which, 14 texture descriptors

were generated, including angular second moment, contrast, shadow of clustering, etc. To reduce the computational cost of original GLCM, Tsai et al. [146] used the weighted eigenvalue of the GLCM with a small neighbourhood window. The image patch that contains defect will have different eigenvalues. The GLCM is a powerful feature extractor but requires cumbersome features design and choice of parameters.

C. Local Binary Pattern

Local Binary Pattern (LBP) is a local feature extractor for defect classification. LBP functions by comparing the intensity of a central pixel in an image or local region with the intensities of its surrounding neighbouring pixels. Typically, a 3x3 or 5x5 pixel neighbourhood is considered. For each pixel in the neighbourhood, a binary code is generated based on whether the neighbouring pixel's intensity is greater or lesser than the central pixel's intensity. These binary comparisons result in a binary pattern. The binary patterns created for each pixel or region in the image serve as a representation of the local texture. A histogram is then constructed by counting the occurrence of different LBP patterns within the image or a region of interest. This histogram encodes the distribution of texture patterns. The LBP histogram can be used as a feature vector to represent the texture properties of the image or region. The shape of the histogram reveals information about the texture characteristics.

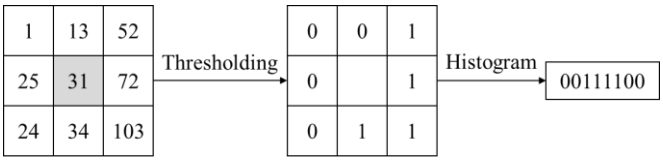


Figure 2.8. An illustration of LBP histogram extraction for a 3x3 image patch

The LBP was first proposed by Ojala et al. [147] for texture classification. To improve the rotation invariance, a completed LBP (CLBP) scheme was proposed by Guo et al. [148]. Three features were combined including the centre pixels CLBP-Centre by threshold, and two features CLBP-Sign and CLBP-Magnitude decomposed by a local difference sign-magnitude transform. However, the threshold scheme is sensitive to noise, which reduces the stability of the algorithm. To solve this problem, the adjacent evaluation CLBP was proposed by Song and Yan [71] by using the adjacent evaluation window from neighbours. Similar to GLCM, the LBP is typically used as feature extractor and extra classifier is needed for defect detection.

D. Fourier Transform

The image features in frequency domain are translation and rotation invariant. In classification problem, the frequency domain was used as feature vectors followed by a classifier. For the

surface defect detection of continuous casting slabs, Ai et al. [149] used the Fourier transform to calculate five kinds of statistical features from spectrum amplitude of sub-band images decomposed by Curvelet transform, followed by dimensional reduction through Kernel Locality Preserving Projection (KLPP) method. Finally, the SVM was used for defect classification. For biscuit tile defect detection, Zorić et al. [150] proposed a Fourier transform based feature extractor. The feature vectors can be obtained from the Fourier power spectrum.

E. Gabor Filter

The Gabor filter is a widely used feature extractor in image processing. It was first proposed by Dennis Gabor [151] as a 1D filter, and extended to 2D by Gösta Granlund [152]. Gabor filters are complex sinusoidal waveforms that are spatially localised and tuned to specific frequencies and orientations. They are defined in both the spatial and frequency domains, making them capable of capturing information at various scales and orientations.

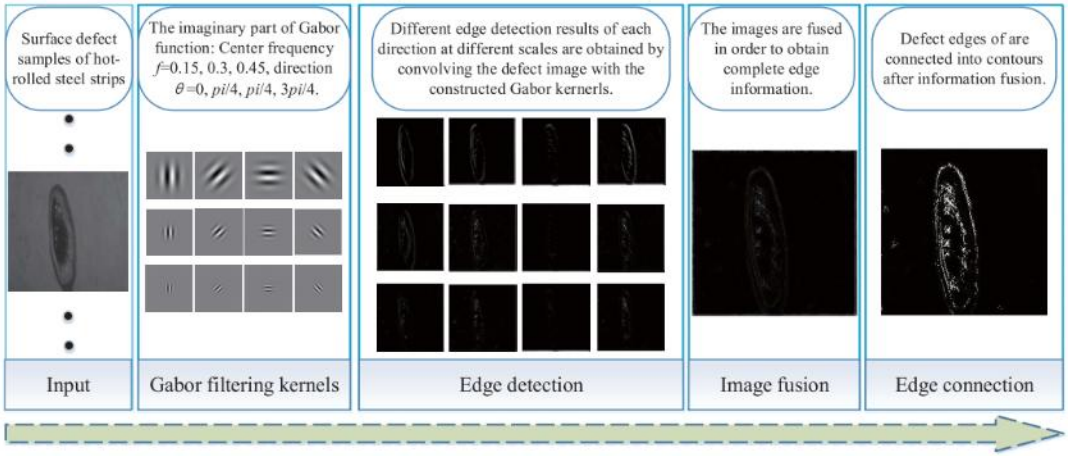


Figure 2.9. Examples of defect edge detection based on Gabor filters (courtesy of [38])

Kumer et al. [153] investigated using the Gabor filter for defect detection in textured materials in both supervised and unsupervised ways. A bank of Gabor filters was selected with a priori knowledge about the orientation and size of a sample. The defect was detected through thresholding method. Raheja et al. [154] compared the performance of GLCM and Gabor filter for fabric defect detection. The results showed that the GLCM performed better than Gabor filter regarding to the detection accuracy and efficiency. By combination of the Gabor filter and PCA, the defect can be identified through Euclidean norm of features with fabric type specific parameters [155].

F. Wavelet Transform

The wavelet transform is a powerful mathematical technique used in various fields, including

signal processing and image analysis. It operates by computing coefficients that are inner products of the input signal and a family of wavelets. These wavelets are generated by scaling and shifting a fundamental "mother" wavelet. What makes wavelets unique is their compact support, which means they are localised in both time and frequency domains. In practical terms, the wavelet transform enables the decomposition of a signal or image into different scales, thereby allowing for the analysis of features at various levels of detail. Compared to the Gabor filter, the wavelet transform offers multi-resolution perception by window adjustment. Through a multiscale windowing technique based on arbitrary wavelet packet decomposition, Lambert and Bock [156] proposed to use the square coefficients or the neighbourhoods coefficients for feature vectors classification. The fast dyadic wavelet transform was used to reduce computational expenses, and the classifier was a general neural net. To solve the nonuniform brightness problem, Zhou et al. [157] proposed a glass bottle bottom defect detection framework. The glass bottle bottom ROI was firstly detected through super-pixel circle detection, followed by frequency-tuned anisotropic diffusion super-pixel segmentation and wavelet transform multiscale filtering for defect localisation.

G. Convolutional Neural Network

A Convolutional Neural Network (CNN) is a deep learning model specifically designed for processing structured grid data, such as images and videos. It has been widely used in common computer vision tasks, such as object classification, detection, and segmentation. CNNs are composed of convolutional layers, which apply convolution operations to the input data. These operations involve sliding small filters over the input to detect patterns from low-level feature, such as edges and textures, to high-level semantic features. After convolution, pooling layers are often used to reduce the spatial dimensions of the feature maps, retaining only the most essential information. CNNs are trained through backpropagation, adjusting the model parameters to minimize the error between predicted and actual outcomes.

The CNN-based method has been widely used for defect detection due to its great power in feature representation. For example, Wang et al. [158] proposed a CNN architecture that can extract features with less prior knowledge and robust to noise. The experimental results show higher accuracy than the traditional hand-crafted optical features. He et al. [159] proposed to fuse the feature maps from multiple stage of the CNN for steel surface defect detection. The network was trained end-to-end to regress the bounding box prediction of defects. To increase the accuracy of the detection for both categories and locations, Cheng and Yu [160]proposed a new architecture with difference channel attention and adaptively spatial feature fusion for steel surface defect detection. Dizaji and Harris [161] proposed a deep 3D CNN network for

concrete columns defect detection with synthetic 3D datasets with defects such as cracks and spalls. For defect detection in laser power bed fusion process, Lee et al. [162] deployed the 3D CNN to classify defects including lack-of-fusion and keyhole-induced defects. The dataset was built by artificially fabricated defects with controlled energy densities. However, the effectiveness of classification-based 3D defect detection is often hindered by the lack of comprehensive datasets.

H. Vision Transformer

Transformer is a deep neural network initially used in natural language processing but has gained more and more attention in computer vision. Compared with CNN, vision transformer offer better global and contextual understanding in cluttered backgrounds with its special attention mechanisms [163]. Vision transformer has been a new trend in defect detection and proven its effectiveness in many applications. For example, based on the Swin Transformer, Gao et al. validated the vision transformer in surface defect detection with a new window shift scheme and more than 4000 annotated images of metal surface defects. The performance surpassed standard CNN-based methods on most tasks. Dang et al. [164] manually collected 47,000 sewer defect images to train a Transformer-based network for severity analysis from the self-attention weights of Transformer. An and Zhang [165] proposed a Transformer-based model named LPViT for PCB defect classification, which exceeds all the other CNN models and became the SOTA. To combine the advantages of CNN and Transformer, Wang et al. [166] incorporated the CNN into the encoder and decoder part of the vision Transformer. The experiments demonstrated higher efficiency compared with CNN- and Transformer-based methods. Nonetheless, Vision Transformers often demand more extensive training datasets to attain performance levels similar to CNNs. In addition, the inherent parallelizability of CNNs contributes to their computational efficiency, rendering them a more feasible choice for real-time applications and resource-limited scenarios. The cooperation of CNN and vision Transformer could be the future trends.

2.3.4. Anomaly Detection

The anomaly detection methods try to model the normal samples and detect the defect by deviation from the model. Based on the modelling of normal samples and deviation detection, a range of anomaly detection methods have been developed, including clustering, grey-level statistics, local binary pattern, Fourier transform, Markov random field, support vector data description, generative CNN, pretrained CNN, and registration-based methods.

A. Clustering

As stated in the defect similarity detection, if the *P* is from the normal dataset, the clustering-based algorithm will belong to the anomaly detection method. For example, the Haar-like feature was extracted from defect-free images and clustered into ten groups with a proposed binary clustering method [167]. The defect is measured by the distance of the test point to the centroid of ten clusters. The computation time is only 0.1 s for 550×550 image of multi-crystal solar cells. The clustering method can serve as segmentation methods for image preprocessing and the defect is detected as parts of the segmented region. Van and Byung proposed a segmentation approach using k-means clustering and graph-based algorithm for fruit defect detection. The Euclidean colour distance was used to cluster the image into subregions as initial segmentation, followed by a merge produce based on graph representation. However, this method only separates the defect region and normal region but does not determine each region is defective or not. The clustering method can also be used in 3D segmentation. Kaiyi et al. used the minimal spanning tree clustering algorithm to segment the defect and intact regions [168]. The defect is detected by that the defective regions do not obey Fourier's law. Therefore, the segmentation-oriented clustering algorithms usually cannot detect the defect directly but require extra classification methods.

B. Grey-Level Statistics

Grey-level statistical methods in defect detection involve analysing and assessing anomalies or defects in images or data through the statistical characteristics of pixel or grey-level values. The statistics can be probability distribution or median values of pixel intensity. By calculation of pixel mean value and distribution from the steel surface background, the bright and dark defects can be simultaneously detected through out of distribution test [169]. For general surface defect detection, Ma et al. [170] proposed a neighbourhood grey difference method based on multi-directional grey-level fluctuation for various surface defects. Both global and local characteristics were used leading to improved generalization ability and detection accuracy. In 3D applications, the grey-level statistics concept can be extended as point cloud statistics. For example, Jovancevic et al. [171] fitted the 3D points of airplane surface as smooth surface in second order polynomial according to the normal and curvature. The defects were detected as outliers of the fitted surface.

C. Local Binary Pattern

Like all the other feature extractors, the local binary pattern can also be used in anomaly detection algorithm. For example, Tajeripour et al. [172] computed a reference feature vector

from defect free fabric image using LBP and determine the threshold during the training stage. Then the reference feature vector is used to detect the defective windows for fabric defect detection. Following the idea that the defects usually have nonuniform patterns on steel surface, Luo et al. [173] proposed a generalized completed local binary patterns (GCLBP) framework that exploit the descriptive information in nonuniform patterns. The defect is detected by histogram matching based nearest-neighbour classifier. Not only for image data, the LBP have been widely used in data anomaly detection such as structure health monitoring [174], streaming data anomaly pattern detection [175], congestive heart failure and arrhythmia classification [176].

D. Fourier Transform

When the Fourier transform used in anomaly detection method, the frequency domain features extracted from the samples are compared with the ones with the normal samples. For example in [177] for weaving defect detection, the Fourier image of good fabric shows well defined spots corresponding to the spatial frequencies of the tissue. While the pattern will change significantly if defect is presented during production of the loom. By define the features of normal directional textures as line patterns, Tsai et al. [178] used 1D Hough transform to remove the high-energy frequency components in the Fourier image and back-transform to spatial domain image. Through this method, only the defective region will be reserved in the restored image.

E. Markov Random Field

In image processing, a Markov Random Field (MRF) is a probabilistic graphical model defined by probability distributions that are used for modelling and understanding the spatial dependencies and interactions among neighbouring elements. These distributions capture both local and global contextual information. In defect detection, the anomalies can be identified based on their deviations from the surrounding context. Noiboar and Cohen [179] proposed a 3D anomaly detection algorithm based on Gaussian Markov random field (GMRF). Each layer of the 3D data was modelled as a GMRF and anomaly detection was implemented by inverse covariance matrix. Goldman and Cohan [180] proposed a multi-scale GMRF model and a corresponding anomaly subspace detection algorithm. The natural clutter images were modelled by the multi-scale GMRF and the anomaly is detected by a matched subspace detector. For defect detection of textured material, Hu et al. [181] proposed to use the hidden Markov tree (HMT) in wavelet domain to model the defect-free images through expectation-maximization algorithm. Then the log-likelihood map was used to classify the defects from

regular texture with the HMT model.

F. SVDD

The support vector data description (SVDD) is one-class classification algorithm that aims to create a model that describes and encloses normal data instances, allowing it to identify anomalies or outliers that fall outside this description. The SVDD algorithm finds the centre and the radius of the hypersphere in such a way that it minimizes the distance between the centre and the normal data points. During testing, SVDD computes the distance from each data point to the centre of the hypersphere. Anomalies are identified by data points located beyond the hypersphere or with a distance exceeding a predefined threshold. Bu et al. [182] used SVDD in combination of multiple fractal features for fabric defect defection with proposed optimal parameter selection for Gaussian kernel function. To address the speed problem of SVDD, Liu et al. [183] proposed to use only one kernel term so that the time complexity during test is O(1).

G. Reconstruction by Generative CNN

For industrial anomaly detection, there are two main branches of methods using CNN, including feature embeddings from pretrained CNN and image reconstruction from generative CNN. For generative CNN method, or reconstructed-based method, an encoder-decoder shaped model is trained to reconstruct the normal samples. It is assumed that through feature compression by encoder, only essential features are remained to reconstruct normal samples only, which does not have enough information and failed to reconstruct the anomalies. The anomaly detection is simply by comparison of the original and reconstructed images. There are two subcategories of generative CNN method, including variational autoencoders (VAE) [184] and generative adversarial network (GAN) [185]. The generative CNN based methods are intuitive and highly interpretable. However, the reconstruction quality can sometimes be too good for anomaly images to detect the anomaly [186]. Besides, the comparison of input and reconstructed image in pixel space can be sensitive to noise and imperfect reconstruction.

H. Feature Embeddings from Pretrained CNN

The pretrained CNN method is based on assumption that the feature embedding from CNN model trained on large and diverse dataset, such as ImageNet, has distinctive distribution for normal and anomaly images. There are two subcategories of pretrained CNN based method including feature embedding modelling (FEM) method and student-teacher (T-S) framework method.

The FEM method tries to model the distribution of feature embeddings of normal samples from pretrained CNN model. The defect is detected by out of distribution test of the feature embeddings from the tested sample. For example, Cohen and Hoshen [187] proposed a semantic pyramid anomaly detection (SPADE) method that aggregated the multi-resolution feature pyramid from pretrained ResNet by k-nearest-neighbour (kNN) method. The anomaly score can be calculated by the closest distance of the feature embedding using Euclidean metric.

The student-teacher based method uses two CNN models for anomaly detection, namely teacher network and student network. The teacher network is a pretrained model on large dataset named teacher, while a smaller student network is trained on the normal dataset to imitate the feature response of the teacher. The assumption of student teacher method is that student network does not have enough representation ability to generalize outside the distribution of normal dataset. Based on this principle, Bergmann et al. [188] proposed the S-T framework for unsupervised anomaly detection. The anomaly score will be the deviation of the student features from the teacher features for test image.

I. Registration-based

Registration-based methods for anomaly detection typically rely on aligning a test image or 3D data to a reference or template. If the test data does not align well with the reference, or if there are significant differences between them after registration, those differences can be treated as anomalies. The registration can be rigid (involving only translations and rotations), affine (which can also involve scaling and shearing), or non-rigid/deformable (where more complex transformations are allowed).

Image registration is an effective way for anomaly detection by aligning two or more images of the same scene taken at different times, from different viewpoints. Through extraction of feature points with CNN, Lu et al. [189] registered the test image with template image via neural best-buddies feature matching module. The anomaly was detected by simply calculation of pixel-wise distances. Zhou et al. [190] proposed a registration and fusion framework for power thermal anomaly detection. The registration parameter including translation and rotation was obtained by low-frequency decomposition and joint histogram segmentation.

Registration-based method has also been widely used for 3D data anomaly detection. Hong-Seok and Mani [191] developed a pressed parts inspection system using laser scanned data. The 3D features were firstly extracted for initial pose estimation, followed by a modified iterative closest point (ICP) algorithm for registration. The 3D point cloud was compared with the CAD model for anomaly detection. For targets with large variations, the deformable

registration method can be employed. Chen et al. [192] proposed a 3-D hierarchical deformable matching algorithm for brain data anomaly detection from MRI.

2.3.5. Comparison and Discussion

A unified view of the defect detection methods based on defectiveness definition is derived and compared in Table 2.7. The advantage of this unified view is that it can reveal the fundamental principles behind these methods and potentially inspire the incorporation of techniques based on different principles. For example, the combination of classification-based method and anomaly detection method may enhance the overall performance of defect detection. Additionally, the feature extractor typically used for classification-based method could be adapted for anomaly detection method.

Table 2.7. A unifying view and comparison of defect detection methods

Method	Ref.	Technique	Principle	f_θ	φ	Distance	Strengths	Limitations
Adaptive Thresholding	[144]	Statistical	Classification	Grey value	Threshold determination	Grey value difference	Easy to understand and implement	Sensitive to noise, fail with complex background
Edge Detection	[135]	Statistical, Structural	Similarity	Gradients calculation	Edge points detection and connection	Gradients intensity and direction	Intuitive and easy to implement	Sensitive to noise and only suitable for low resolution image
Clustering	[167]	Statistical, ML	Similarity, Anomaly	Grey value or image features extraction	Clustering	Distance to cluster centroids	High flexibility and scalability	Sensitive to cluster initialization and data outliers
Grey-Level Statistics	[169]	Statistical	Anomaly	Grey value	Mean, median, distribution	Distance to median or distribution	Easy to explain and implement	Difficult to choose threshold, sensitive to noise
Co-occurrence Matrix	[145]	Statistical	Classification	Co-occurrence matrix	Feature aggregation and reduction	Class probability	Powerful feature extractor, noise robust, rotation and scale invariant	Only for local feature analysis, high requirement of computation and memory
Local Binary Pattern	[172]	Statistical	Classification, Anomaly	Local binary pattern	Feature aggregation, histogram	Class probability, feature vector distance	Powerful feature extractor, rotation and grey scale invariant, robust to illumination, efficient	Require careful parameter selection including neighbourhood size and the number of neighbours
Registration	[189]	Statistical	Anomaly	Grey value or feature	Template matching, Registration	Grey value or feature difference	Robust to variations, high accuracy defect localisation	Sensitive to noise, computationally intensive
Hough Transform	[138]	Statistical, Spectral	Similarity	Image binarization	Local maxima in accumulator space	Log-likelihood	Robust to noise, spatial invariance, suitable for multiple defect shapes	Computational expensive, sensitive to parameters
Fourier Transform	[177]	Spectral	Similarity, Classification, Anomaly	Transform to frequency domain	Spectral diagram	Frequency difference, class probability	Shift-invariant, denoising, computational efficient	Only for periodical defect pattern, lack of spatial information
Gabor Filter	[151]	Spectral	Classification	Bank of Gabor filters	Feature aggregation, edge detection	Class probability, Euclidean distance, thresholding	Multi-scale and multi-orientation feature extraction, computational efficient	Configuration of filter parameters can be cumbersome
Wavelet Transform	[156]	Spectral	Classification	Wavelet coefficients	Feature aggregation and reduction	Class probability, thresholding	Multi-scale feature extraction, denoising, image compression	Sensitive to noise and selection of basis functions and the decomposition levels

Method	Ref.	Technique	Principle	f_θ	φ	Distance	Strengths	Limitations
Active Contour Model	[141]	Model	Similarity	Image energy	Energy minimizing	Energy level	Sub-pixel accuracy of object boundary	Sensitive to initial contour and constraints
Markov Random Field	[179]	Model	Anomaly	Grey level	Expectation maximation	Log-likelihood	Contextual information from neighbours,	Design of energy function and potentials is crucial, computationally intensive
Support Vector Data Description	[182]	ML	Anomaly	Feature space transform	Hypersphere centre and radius estimation	Distance to hypersphere	Specially designed for one-class classification, nonlinear mapping	Sensitive to hyperparameter selection and data imbalance
Convolutional Neural Network	[158]	ML	Classification	Stacked CNN layers	Feature learning	Class probability	Hierarchical feature learning leverage big data, reserve global and local features	Require high quality labelled data and large computational power for model training
Vision Transformer	[163]	ML	Classification	Stacked Transformer blocks	Feature learning	Class probability	Global context understanding, multi-modal integration, attention mechanism, high scalability	Require large amount of data and computation resources, larger model size than CNN, struggle to detect small defects
Generative CNN	[184]	ML	Anomaly	Grey value	Reconstruction	Grey value difference	Capture complexity and variability of data, interpretable	Sensitive to noise and imperfect reconstruction, sometimes good reconstruction even for anomaly
Pretrained CNN	[187]	ML	Anomaly	Feature embeddings from CNN	Feature distribution modelling	Distance in feature space	Effective feature extraction from pretrained model, high generalization ability	High memory requirement, less interpretable

2.3.5.1. Defect Similarity Detection

The defect similarity detection methods are naturally intuitive and easy to implement, such as edge detection and Hough transform. For example, Edge Detection operates on the principle of gradient calculation to identify discontinuities in intensity, which often correspond to the edges of defects. On the other hand, the Hough Transform is used to detect regular curves and lines, making it ideal for identifying specific shapes indicative of defects like circles or cracks.

However, the application of this kind of method is limited to a certain range of defects that can be generally characterised, such as scratches and dents. Besides, these methods are intuitive and easy to implement, their efficacy is closely tied to the resolution of the image and the nature of the noise present within the data. In modern industrial applications, the variations in defects often present with irregular shapes and appearances. These conventional techniques typically rely on predefined patterns or models, which may not adequately capture the complex and variable nature of manufacturing defects, thereby constraining their practical utility.

Indeed, while these methods have their limitations, they retain their utility when integrated into broader defect detection workflows. They can be effectively employed as part of preprocessing or postprocessing steps. In preprocessing, they can help in normalizing the data, enhancing features, or reducing noise, thus improving the performance of more sophisticated algorithms downstream. In postprocessing, they can refine the outputs of advanced models, for instance, by removing false positives or further delineating the detected defects. By situating these methods within a larger pipeline, they can contribute to more robust and accurate AVI systems.

2.3.5.2. Classification-based Detection

Classification-based methods involve categorising image pixels or regions into predefined classes. Adaptive Thresholding, for example, operates by comparing grey values to a threshold to classify pixels. The Co-occurrence Matrix and Local Binary Pattern methods extract more complex features that can then be classified. With the advent of deep learning, CNNs and Vision Transformers offer a sophisticated approach by learning feature representations directly from the data, allowing for more nuanced and robust classification. These methods are typically more robust than similarity-based approaches but come with their own limitations such as sensitivity to the training data quality and the computational resources required for training large-scale models.

A list of classification-based methods using CNN is described in Table 2.8. Historically, before 2019, defect detection predominantly revolved around tasks like classification and bounding box detection. This was largely due to the constraints imposed by the accuracy and processing speed of CNNs at that time which usually requires expensive GPU. Segmentation, during this period, was primarily utilised for visual interpretation rather than as a core detection strategy. With the advent of 2020, there was a noticeable paradigm shift as segmentation took centre stage, propelled by deeper and more intricate research into CNN architectures. After 2022, the introduction of transformer-based models marked another notable shift, as they excel in capturing global contextual information. This feature proved advantageous for identifying defects within intricate backgrounds.

Despite these advancements, these methods still face limitations in terms of computational efficiency, segmentation accuracy, and availability of comprehensive datasets. Even though the overall accuracy of image-level classification has surpassed the 99% threshold, the trade-off between accuracy and speed remains a challenge. For example, real-time defect localisation applications predominantly cantered around bounding box detection due to their less demanding computational requirements. While the image segmentation task is less explored in real-time applications due to insufficient accuracy and speed. Moreover, the scarcity of comprehensive datasets poses another significant challenge in the field of defect detection. The most widely used defect datasets are dated, often more than a decade old, and they fall short on several fronts. They typically contain limited data volumes, lack diversity in defect types, and sometimes suffer from inaccuracies in labelling. This inadequacy impedes the development and training of advanced models that require large and varied datasets to achieve high generalization and robustness. Balancing these competing demands is a key challenge for current and future AVI systems.

Table 2.8. Comparison of CNN classification-based defect detection methods

Ref.	Year	Algorithm	Defects	Dataset	Task	Accuracy (%)	Frame Rate	Novelty	Limitation
[193]	2017	DCNN	Defects of hot-rolled steel strip, welds and wood	NEU, Weld Defect Database, Wood Defect Database	Classification, segmentation	99.27 (NEU) 88.00 (Weld) 94.28 (Wood)	2 fps on 24 cores CPU 820x820	Small dataset training by transfer learning	Low resolution and speed for segmentation
[118]	2018	DCNN	Synthetic defect on texture background	DAGM	Classification, segmentation	98.50	-	Compact CNN architecture design for simultaneous classification and segmentation	Focused on classification accuracy, no segmentation and efficiency study
[194]	2018	Fast RCNN	Sewer pipe defects	CCTV sewer inspection videos	Bounding box	mAP 79.80	20.4 fps on GTX1070 224x224	New application of CNN in defect detection with model hyperparameters study	Bounding box output has lower localisation precision
[195]	2019	CNN + LSTM	Hot-rolled steel plates	-	Bounding box	86.20	-	Novel CNN architecture for actual industrial scenarios	No time-cost analysis Bounding box output has lower localisation precision. Limited accuracy
[196]	2019	DCNN	Hot-rolled steel strip	NEU	Classification, segmentation	99.44	-	Class activation map for visual decision making	No time-cost analysis and segmentation accuracy
[197]	2019	Yolov3	scratches, cratering, oil spots, fisheye-like crawling and humps	-	Bounding box	88.00	0.77 fps on GTX1080 1632x1224	Image acquisition system design	Bounding box output has lower localisation precision. Limited accuracy
[27]	2020	DCNN	Electronic Commutator	KolektorSDD	Classification, segmentation	99.90	9.1 fps on TitanX 500x1240	New dataset proposed and novel CNN architecture	Segmentation result not analysed
[198]	2020	MobileNetv2	Defect of hot-rolled strip and wood	DAGM Wood defect	Classification, segmentation	100 (NEU) 99.90 (DAGM)	4.6 fps on GTX1080 512x512	Light weight depth wise pyramid block	Small dataset size. Limited inference speed

Ref.	Year	Algorithm	Defects	Dataset	Task	Accuracy (%)	Frame Rate	Novelty	Limitation
[199]	2020	Concurrent CNN	Hot-rolled steel strip	NEU	Classification	98.89	179 fps on GTX 1080Ti 400x400	20% labelled data for training real-time	Small dataset size. No localisation information"
[200]	2020	mask-RCNN	Oil leak defect	-	Segmentation	92.90	4.11 fps GTX1060MaxQ 512x448	Refined Mask-RCNN with pyramid feature fusion	Relatively low framerate
[75]	2020	U-Net	Defect on magnetic tile: blowhole, crack, break and fray	Magnetic tile	Segmentation	96.6	14.28 fps on GTX 1050ti 196x196	Reformat the input with MCue saliency image, multi-task learning with detection branch. New dataset proposed	Low input resolution
[201]	2020	DCNN	Three typical industrial scratch data sets	WSCRATCH512 CrackDataSet magnetic tile	Segmentation	mIoU 80.05 (WSC) mIoU 81.22 (crack) mIoU 92.86 (magnetic)	2.14 fps on GTX 1070 512x512	Novel CNN architecture design with attention feature fusion and context fusion block	Relatively low framerate
[202]	2021	U-Net	16 types of defects on steel surface	-	Segmentation	mIoU 40 (type) mIoU 43 (severity)"	-	Multi-task model that performs both pixel-based defect segmentation and severity estimation	No time-cost analysis relatively low accuracy
[203]	2021	RCN	1124 defect categories of FPCB	-	Bounding box	mAP 94.15	-	Novel CNN architecture design with multi-task learning	No time-cost analysis Bounding box output has lower localisation precision
[204]	2021	GAN+U-Net	Synthetic defect on texture background	DAGM	Classification	95.00	-	Use GAN to produce more training data	No time-cost analysis segmentation accuracy not analysed
[205]	2021	SSD	Three public datasets	NEU DAGM magnetic tile	Bounding box	mAP 88.8 (NEU) mAP 99.1 (DAGM) mAP 93.4 (magnetic)	31.0 fps on GTX 1080Ti 512x512	Novel CNN architecture design with feature retaining block and skip densely connected module	Bounding box output has lower localisation precision

Ref.	Year	Algorithm	Defects	Dataset	Task	Accuracy (%)	Frame Rate	Novelty	Limitation
[206]	2022	DCNN	defect on rail surface	UAV collected dataset	Segmentation	F1 96.7	-	Novel CNN architecture design with boundary guidance network	No time-cost analysis
[207]	2022	Faster R-CNN	Stains, holes, floats, and yarn defects of fabric	Dataset collected from factories and online	Bounding box	mAP 94.57	-	Incorporation of Gabor filter in CNN	No time-cost analysis
[208]	2022	DCNN	Multiple defects of texture and industrial part	MvTec AD	Segmentation	F1 86.51	52.26 fps on GTX Titan X	New model architecture design with multi-scale feature enhancement and fusion	Apply classification-based method on anomaly detection dataset
[209]	2022	DCNN	13 types of rail surface defect	Rail surface defect dataset	Segmentation	85.51	32.26 fps 1282 ×160	New model architecture design with pyramid feature extraction	GPU not specified
[210]	2022	Swin Transformer	pitting, fatigue cracking, spall, and shallow spall on metal surface	Self-collected dataset	Segmentation	AP 80.2	-	Validated the Transformer network for defect detection	Large model size, not time-cost analysis
[211]	2023	YOLOv4	undercut lack of fusion in Wire and arc additive manufacturing	Self-collected dataset	Bounding box	mAP 94.5	44 fps on GTX 1080Ti 416 × 416	Improved the model with attention mechanism and applied in real-time additive manufacturing	Bounding box output has lower localisation precision
[212]	2023	YOLOv5s	Three public datasets	NEU MT DAGM	Bounding box	mAP 86.80 (NEU) mAP 92.6 (MT) mAP 99.5 (DAGM)	51 fps on GTX 1080Ti 640 × 640	Improved the model for multi-size defect detection	Bounding box output has lower localisation precision
[166]	2023	CNN Transformer	Three public datasets	SD-saliency-900 Fabric defect dataset NRSD-MN dataset	Segmentation	97.91 (SD) 98.08 (fabric) 97.15 (NRSD)	5.88 fps on NVIDIA P5000 256 × 256	Combined CNN with transformer for defect detection	Slow inference speed

2.3.5.3. Anomaly Detection

The third category is anomaly detection method, where the goal is to identify outliers that do not conform to the expected distribution of the data. Clustering algorithms, for example, group similar data points together and identify defects as those that fall outside of the main clusters. Grey-level statistics measure how much a pixel intensity deviates from statistical measures like the mean or median, and registration method aligns images to a reference to spot anomalies. SVDD encapsulates normal data within a boundary in feature space and regards points outside this boundary as anomalies. Machine learning methods such as generative CNNs and pretrained CNNs detect anomalies by learning a model of normality from the data and then identifying deviations from this model. These methods can be very powerful as they do not require explicit defect modelling. However, they may be challenged by complex data distributions and require careful tuning of their parameters.

A list of anomaly detection methods using CNN is described in Table 2.9. These methods are compared on the MVTec AD dataset which is a widely recognised dataset that contains a variety defect of textures and objects. The pixel AUROC is used to evaluate the accuracy of defect localisation and the results of textures and objects are compared separately. The two SVDD-based methods show both high performance with larger than 95% accuracy. For generative CNN method, the VAE and GAN based methods have achieved well enough accuracy but surpassed by the normalizing flow based method including CFlow and Fastflow. As for pretrained CNN, the accuracy increased gradually from SPADE to PatchCore for feature embedding modelling, and S-T to EfficientAD. The registration-based methods also achieved comparable accuracy as the others that higher than 98%. Most algorithms predicted better for objects than textures, except Cutpaste, S-T, and STPM. This may suggest a subtle complexity in texture anomaly detection.

In terms of inference speed, there appears to be a bottleneck when processing on CPUs, with most methods unable to exceed 10 fps. EfficientAD marks a significant leap forward, achieving a remarkable 269 fps on the RTX A6000, which is the most advanced GPU currently commercially accessible, highlighting its potential for real-time applications. Unfortunately, comparable performance metrics for CPU processing are absent, which would have been invaluable for understanding its practicality in less resource-intensive environments. This gap underscores a common challenge in anomaly detection research: balancing accuracy with computational efficiency, especially outside high-end GPU contexts.

Table 2.9. Comparison of anomaly detection method on MVTec AD dataset regarding to pixel AUROC (%)

Category	Ref.	Year	Method	Textures	Objects	Mean	Frame Rate
SVDD	[213]	2020	PatchSVDD	93.7	96.7	95.7	-
	[214]	2021	SE-SVDD	96.4	98.0	97.5	97 fps on (TITAN XP 4 fps on Intel Xeon Silver 4116
Generative CNN	[215]	2020	Vanilla VAE	-	-	82.3	-
	[216]	2020	VE-VAE	80.6	88.8	86.1	-
	[217]	2022	MPAD	97.7	98.4	98.1	-
	[218]	2021	CutPaste	96.3	95.8	96.0	-
	[219]	2017	AnoGAN	57.6	82.6	74.3	-
	[220]	2022	CFlow-AD	98.5	98.7	98.6	9 fps on Intel i7
	[221]	2021	Fastflow	98.1	98.7	98.5	21.8 fps on GeForce GTX 1080Ti
Pretrained CNN	[187]	2020	SPADE	92.9	97.6	96.5	1.51 fps on Nvidia Tesla V4
	[222]	2021	Padim	96.9	97.8	97.5	5.26 fps on Nvidia Tesla V4 1.05 fps on Intel i7-4710HQ
	[223]	2022	PatchCore	97.6	98.3	98.1	5.88 fps on Nvidia Tesla V4
	[188]	2020	S-T	92.7	90.77	91.4	-
	[224]	2021	STPM	98.3	96.33	97.0	-
	[225]	2022	Reverse Distillation	97.7	97.9	97.8	3.22 fps on Intel i7
	[226]	2023	EfficientAD	-	-	98.2	269 fps on NVIDIA RTX A6000
Registration	[227]	2022	RegAD	-	-	96.9	-
	[228]	2022	FYD	98.2	98.3	98.2	-

2.3.6. Summary

In summary, the choice of defect detection method depends on several factors, including the type of defect, the availability of target data, the computational resources available, and the specific requirements of the application such as localisation precision, efficiency, and accuracy. Each method has its strengths and limitations, and in practice, a combination of these methods could be used to achieve the desired level of accuracy and reliability in defect detection. This underscores the necessity for a comprehensive design methodology to guide the development of AVI system design. This highlights the importance of adopting a comprehensive design methodology to systematically address the various factors involved in AVI system development, such as defect detection methods, data processing, computational resources, and application-specific requirements. A structured approach ensures that all elements of the system are properly integrated, optimised, and aligned with the overall goals, leading to more efficient and reliable defect detection.

The literature review also highlights two major challenges in defect detection methods: the accuracy-efficiency trade-off and data scarcity. While most research has focused on improving

detection accuracy, fewer studies have addressed efficiency, particularly on computation resource constrained platforms. To tackle data scarcity, the anomaly detection methods are often employed. However, current early-stage research primarily centres on public datasets, leaving real-world applications underexplored.

2.4. Knowledge Gaps

Based on the literature review findings, the following knowledge gaps were identified:

1) Absence of a well-defined and comprehensive design methodology

Despite significant progress in the development of algorithms and systems for defect detection, it is evident that there is an absence of a design methodology to help design, develop and assessing the complete AVI systems. The challenge of developing effective AVI systems for defect detection lies in managing complex factors such as data availability, defect variability, and environmental conditions. Addressing these challenges during system development requires a well-defined design methodology. This methodology should accommodate detection tasks across various localisation precision levels, including image, object, pixel, and 3D levels.

2) Ability to achieve real-time detection on resource- constrained devices

As shown in Table 2.8 and Table 2.9, though some model can achieve real-time inference GPU, there is no model achieved more than 10 fps on CPU. Their practical applications in real-time scenarios still present a significant challenge, especially on mobile devices. This requires the design of novel training strategy and model architectures to meet stringent time constraints.

3) Under-exploration of the application of anomaly detection in AVI systems

Despite advances in machine learning, the accuracy and generalisability of robust defect detection algorithms still depend heavily on the availability of sufficient high-quality training data. The anomaly detection method is a popular topic for addressing the shortage of high-quality, diverse, and annotated data. However, the research of anomaly detection techniques is still in relatively early stages of research in some specific areas, such as wheat head disease detection or engine surface detection, and has focused on public datasets. Their deployment in real-world AVI applications is underexplored.

3. Design Methodology for AVI System

3.1. Introduction

This chapter proposes a comprehensive design methodology for AVI system development with a focus on defect detection. Inspired by Yan's model, the life-cycle design model [229] and concurrent design principle and multi-perspective modelling [230] are adapted for the design methodology. It features a design process model for developing a complete AVI system, as shown in Figure 3.1.

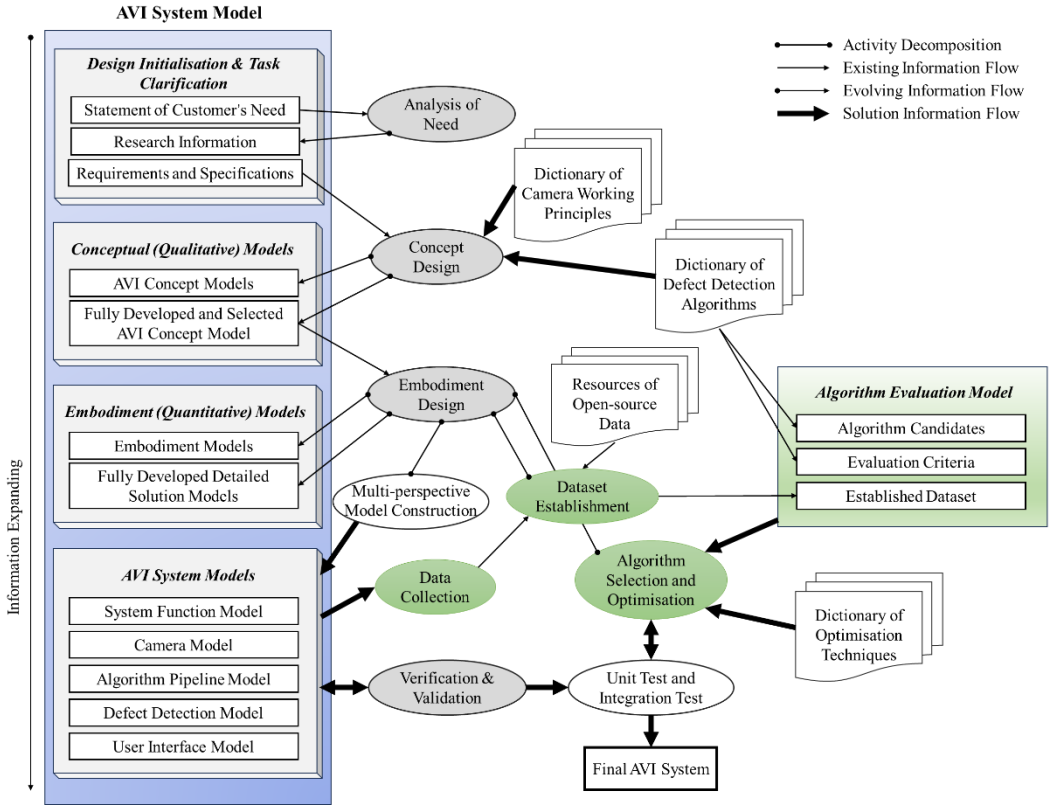


Figure 3.1. Design process model for concurrent hardware and software development for AVI system

Following the structure of Yan's model, it consists of four design stages, i.e., task clarification, concept design, embodiment design, and system verification and validation. The components of the design process and the associated flow of information are adapted for the AVI system. Compared with Batchelor's design methodology, the proposed design methodology is more comprehensive, covering all the three components of the AVI system, especially the defect detection method development. Besides, the proposed methodology provides a more detailed representation of the information flow. In contrast to Saha's design methodology, the proposed design methodology includes a process model to present the interactions between the system

components, rather than merely describing each component separately.

The objective of the task clarification stage is to identify the customer need, collect academic and industrial information, and define the requirements and specifications for the AVI system. Following the task classification, the essential problem should be abstracted during the concept design phase, after which fully developed and selected concept models should be generated. To support the concept generation, a collection of dictionaries and libraries are used to assist designers in producing embodiment models and solution models. This collection includes the fundamental principles of camera and defect detection algorithms.

The embodiment design includes multi-perspective model construction, dataset establishment, and algorithm selection and optimisation. The multi-perspective model covers the system function model, camera model, algorithm pipeline model, defect detection model, and user interface model. The dataset establishment relies on either self-collected or open-source data to ensure diversity and the accurate representation of target defects. The development of the defect detection algorithm involves selecting and optimising algorithms based on an evaluation model, which comprises algorithm candidates, evaluation criteria, and the established dataset.

The system validation should be conducted in parallel with the system integration process. The unit test is carried out to ensure validation of all system models, such as image acquisition and defect detection. Finally, the entire system is subjected to an integration test to verify the performance of the final AVI system.

Throughout this progression, the design information becomes increasingly complex and comprehensive. The design evolves from an abstract and qualitative concept to a specific and quantitative embodiment, and finally a complete AVI system. The subsequent sections will provide a detailed description of the design process and the tasks at each design stage.

3.2. AVI System Design Process

3.2.1. Task Clarification

The initial phase of system design is crucial and involves in-depth and continuous communication with the customer to define the task clearly. This includes the understanding of the operational environment, identification of the specific target object, and the detailed requirements for defect detection. Thorough research should be conducted to collect information from both industry and academia to identify the knowledge gap. From the customer need and research information, the requirements and specifications of the system can

be derived.

Specifications should cover all aspects of the target inspection objects and the environment in which the system will operate. This involves defining the type and material of the object to be inspected, as well as the specific kinds of defects that need to be identified. In terms of the environment, it is vital to specify the conditions of illumination and the characteristics of the background against which inspections will be carried out.

The set of dictionaries and libraries should be established at this stage including the hardware working principle, simulation blocks and elements, defect detection algorithms, and open-source data. The dictionary of hardware working principle should include the mechanical, electrical, and optical hardware working principles, such as the pinhole camera working principle for image collection. These working principles will be used for hardware concept models generation. The simulation blocks and elements, like those available in MATLAB/Simulink, are used to model and test the physical processes within the system before actual hardware is developed. The defect detection algorithms can be collected through research, including the algorithms from the three categories summarized in the literature review chapter. The open-source data is critical for algorithm development and testing, including both publicly available datasets and dispersed data from internet.

Based on the requirements, and research information, a set of specifications can be defined, including the detailed information on the target object, the operational environment, specific system requirements, and available datasets. The specifications will be used for concept design which will be described in the following section.

3.2.2. Concept Design

Concept design is an early phase of the design process, where the broad outlines of function are abstracted and generated, including the design of interactions, experiences, processes, and strategies. The system concept includes an initial design of system architecture, data acquisition system, and defect detection method, based on the following considerations derived from the task clarification stage:

1) Target Object: Understanding the characteristics of the target object is critical for the system design. It includes identification of the size, shape, material, and other properties that could affect how the object should be inspected and treated if defect presented.

2) Environment: The environment where the AVI system will be deployed affects the choices of system components significantly. Factors include lighting conditions, presence of vibrations,

temperature, and humidity. For instance, embedded systems are generally used for portable, on-site inspections where space and power may be limited, while desktop systems might be used in a controlled environment where more room is available for powerful computing resources. Additionally, for object in clear background, it can be easily segmented out. But for cluttered background, extra step may be needed to extract the region of interest.

3) Dataset: The requirements of the target dataset can influence the data acquisition subsystem. For example, the required number of images, the diversity of the objects, and variations in illumination configurations impact how data is collected. The type and size of the dataset available for training and testing the algorithms are critical to algorithm selection. A larger and more varied dataset might be needed for more complex objects or tasks, which can affect the choice of software platform based on its data processing capabilities.

4) Requirements: The requirements regarding to speed and precision can influence the selection of operating system and programming language. As for the programming language, C++ is often favoured in environments where speed and efficiency are critical, due to its performance advantages over languages like Python. However, Python's simplicity and the vast ecosystem of libraries make it a popular choice for rapid development and prototyping, especially in the research phase or when the performance is less critical. TensorFlow and PyTorch are two of the most popular platforms for AI algorithm development. TensorFlow is known for its powerful production-ready tools and broad community support, making it a common choice for deploying AI applications. PyTorch, on the other hand, is often praised for its ease of use and dynamic computational graph, which can be more intuitive for research and development purposes.

3.2.3. Embodiment Design

Embodiment design is a phase where the conceptual models of the system defined in earlier stage are transformed into detailed embodiment models. In this stage, a set of multi-perspective design models are developed to collectively define the behaviour of both the hardware and software systems. The hardware models cover the camera model, illumination model, and sensing model for data acquisition, control and execution model for defect management, and the geometric layout model for the whole system. Based on these models, a series of tasks can be conducted to develop the subsystems, including the dataset establishment, hardware simulation and comparison, and algorithm selection and optimisation.

3.2.3.1. Multi-perspective Model Construction

During the embodiment design, the models of AVI system define the properties of the system in terms of its physical definition and its intentional purpose. It includes the system function model, camera model, algorithm pipeline model, defect detection model, and user interface model, as defined in detail below:

1) System Function Model: This model provides an overview of the system architecture, outlining how the system operates to achieve its goals. It defines the relationships between different components and shows the detection is realised through various functional processes.

2) Camera Model: The camera model is essential for understanding how the appearance or shape of the target object is captured by the designed vision system. For example, the pinhole or fisheye camera model has different image distortions and perspectives. This choice also influences the camera calibration method for image alignment and 3D reconstruction.

3) Algorithm Pipeline Model: The pipeline for defect detection includes the data processing, defect detection, and decision making. The input, output, and available algorithms of each component should be defined.

4) Defect Detection Model: This is core of the AVI system and should be developed and optimised to meet the system requirements. It involves developing algorithm capable of accurately and efficiently identifying the defects based on the acquired data.

5) User Interface Model: The user interface model provides an interface for users to control and inspect the working status of the AVI system. It includes the control buttons, visualization tools, real-time information updating, providing users with an interactive window into the AVI system operations.

6) Algorithm Evaluation Model: This model contains essential components for algorithm selection optimisation, including the defect detection algorithm candidates, evaluation criteria, and established datasets.

3.2.3.2. Dataset Establishment

Data is one of the most important components of modern defect detection systems. The dataset must contain adequate amount of data and sufficient object diversity to ensure robustness and accuracy of the defect detection algorithm. The dataset can be sourced from open-source data, self-collected data, or a combination of both. For self-collected data, camera settings and illumination conditions should be optimised to highlight defects effectively, ensuring high-

quality data for training and validation. Open-source data can be drawn from publicly available datasets or gathered from scattered sources across the internet. Regardless of the source, it is essential to meet the requirements for data volume and diversity as specified during the task clarification phase.

Additionally, the labelling of the data should align with the localisation precision required by the task. For instance, labelling could be at the image level or pixel level based on the specific defect detection goals. This precision will directly influence the model's ability to localise and identify defects accurately in real-world applications.

3.2.3.3. Algorithm Selection and Optimisation

Development of defect detection algorithm tailored for a specific application involves careful selection and optimisation of algorithm to ensure all requirements are met. The process is based on an algorithm evaluation model that includes the algorithm candidates, evaluation criteria and the established datasets. The flowchart of the algorithm selection and optimisation is shown in Figure 3.2. During the design process, the requirements, algorithm candidates, and dataset can be derived from the task clarification, conceptual design, and embodiment design, respectively.

In some applications, the dataset for the target object could be small and does not cover most defect types likely to occur in real-world applications, which can hinder the development of the defect detection algorithms. To address this issue, the supplementary dataset is needed for algorithm development. To ensure the consistency between the selection and optimisation, the supplementary dataset should cover similar objects and defects as those in the target dataset. The best model is selected further optimised on the target dataset to refine its performance. This systematic approach ensures that the defect detection algorithm is both robust and aligned with the specific needs of the application.

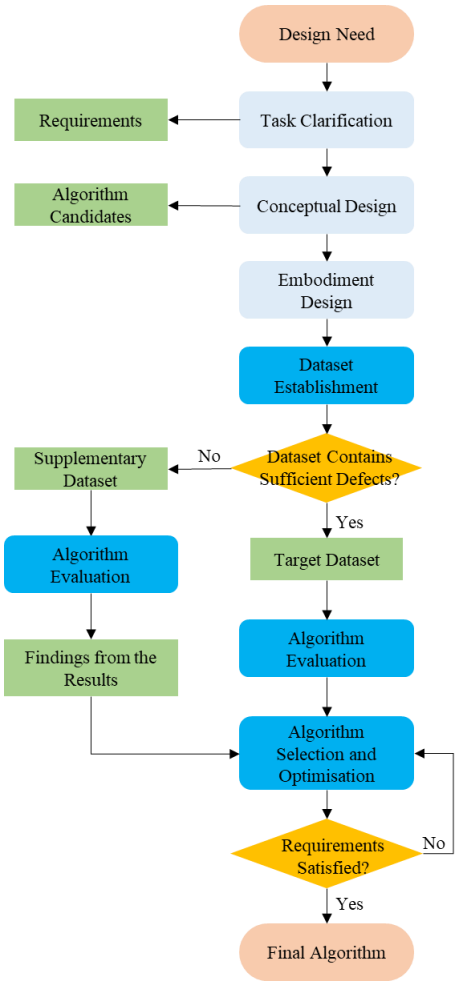


Figure 3.2. Flowchart for basic defect detection algorithm selection and optimisation.

3.2.4. System Verification and Validation

Typically, the embodiment design phase operates alongside system verification and validation, creating a continuous feedback loop. As in the V-model [128], verification of each subsystem ensures compliance with detailed specifications and design requirements. Simultaneously, validation procedures confirm that these subsystems collectively meet the intended use and user requirements. During the development of subsystems such as data collection subsystem and defect detection subsystem, unit tests are carried out for to verify individual functionalities. Then a thorough evaluation of the fully integrated system should be conducted to examine the cooperation amongst all subsystems. If any mismatches from the requirements emerge during the tests, iterative refinements of the concept and embodiment designs are undertaken. The resulting improvements are then incorporated into the development cycle, thereby ensuring that the final product is aligned with its intended objectives.

3.3. Methodology Validation

To validate the proposed methodology, four case studies were conducted in this research covering different applications, defect detection tasks, detection methods, etc. A comparison of the four case studies is listed in Table 3.1.

Table 3.1. Comparison of case studies for methodology validation

	The First Case Study	**The Second Case Study**	**The Third Case Study**	**The Fourth Case Study**
Target Object	Concrete Wall	Wheat Head	Car Engine	Modular Satellite
Detection Task	Image and pixel-level	Object-level	Pixel-level	3D-level
Defect	Crack	Head blight	Surplus	Extrusion, protrusion
Detection Method Category	Classification-based detection	Classification-based and anomaly detection	Anomaly detection	Anomaly detection
Camera	RGBD	Monocular	Monocular	RGBD
Emphasise	Algorithm selection and optimisation	Algorithm selection and optimisation	Dataset Establishment, Algorithm selection and optimisation	Dataset Establishment, Algorithm selection and optimisation

The selection of the four case studies is driven by their ability to comprehensively demonstrate the versatility and applicability of the proposed design methodology for AVI systems across diverse sectors. Firstly, the case studies span industrial manufacturing, civil infrastructure, space exploration, and smart agriculture, which represent key sectors where AVI systems play a critical role in quality assurance, safety, and operational efficiency. In addition, the selected cases require context-specific solutions with varying challenges, such as surface anomaly detection in manufacturing, real-time inspection in AR environments, pose estimation in modular spacecraft, and disease detection in crops. This variety validates the methodology's capability to tailor system configurations based on domain knowledge. Furthermore, The case studies leverage state-of-the-art technologies, including deep learning, augmented reality, and domain adaptation. Their inclusion demonstrates the methodology's suitability for integrating emerging technologies into AVI systems. Together, these four case studies provide comprehensive evidence of the methodology's robustness, flexibility, and applicability across different industrial and non-industrial domains, supporting its generalization and practical value in AVI system design.

The first case study examines an AR-based concrete wall crack inspection system on a commercially available AR headset. This study emphasises the image- and pixel-level classification-based defect detection method development. This project will highlight how the basic algorithm selection and optimisation process is applied for real-time crack detection

algorithm on computation resource restricted device.

The second case study delves into the wheat disease detection for a precision spray system. The classification-based and anomaly detection method are combined for object-level wheat head disease detection. It underscores a feasibility and comparative study of anomaly detection methods with both labelled data and disease data are limited.

The third case study focuses on the design of an AVI system for car engine manufacturing. The primary aim is to develop a pixel-level anomaly detection algorithm in car engine surface detect detection. It will specifically detail the refined process of algorithms selection and optimisation using a large synthetic anomaly dataset and a real anomaly dataset.

In the fourth case study, a visual inspection and anomaly detection system for modular satellites is developed. This study illustrates the design of the image acquisition system with three camera sets and the software platform for reconfigurable algorithm pipeline. It also showcases the refined process of algorithms selection and optimisation for object 6-DoF pose estimation and anomaly detection.

Each case study targets a unique application area and uses a combination of AVI system components and techniques. They collectively demonstrate the versatility and breadth of the proposed design methodology across diverse environments and challenges. The evaluation of these case studies would include analysing the performance of the AVI system in accurately detecting and classifying defects, the integration of system components, and the overall effectiveness of the system in operational settings.

3.4. Summary

In this chapter, a comprehensive design methodology is proposed to provide a systematic guidance for AVI system design. Through the process model, the design evolves from an abstract and qualitative concept to a specific and quantitative embodiment, and finally a complete AVI system. The development of defect detection method includes data collection, dataset establishment, and algorithm selection and optimisation. Specifically, the process of algorithm selection and optimisation can be tailored depending on the availability of target data. To validate the proposed design methodology, it will be adopted in each of the four case studies to demonstrate its effectiveness and usefulness.

The proposed design methodology for AVI systems introduces several key novelties and advancements compared to existing design methodologies:

Context-Driven Design Framework: Unlike traditional methodologies that primarily focus on technical specifications, the proposed approach prioritizes context knowledge—including target object characteristics, environmental conditions, and operational requirements—as the foundation for system design. This ensures that the resulting systems are tailored to specific application demands.

Holistic Design Process: The methodology integrates hardware, software, and algorithm design into a unified framework, guiding the development process from concept design to embodiment design and verification and validation. This comprehensive approach contrasts with existing methodologies that often address these aspects in isolation.

Modular and Scalable Structure: The framework adopts a modular structure that facilitates system customization and scalability, making it adaptable to various inspection applications across different sectors, including manufacturing, infrastructure, space exploration, and agriculture.

Practical Validation through Case Studies: The methodology is validated through four diverse case studies, demonstrating its applicability and effectiveness in both industrial and non-industrial domains. This contrasts with many existing methodologies that lack comprehensive practical validation.

These features collectively position the proposed design methodology as a versatile, context-aware, and future-proof framework that bridges the gap between theoretical design principles and practical AVI system implementation.

4. The First Case Study: Image- and Pixel-Level Concrete Crack Inspection

This chapter reports the first case study of head mounted vision based concrete wall crack inspection system using Artificial Intelligence (AI) and Augmented Reality (AR) technologies. This case study focuses on the algorithm selection and optimisation part of the design methodology. It aims to develop tiny AI models for real-time on-site AR applications using HoloLens 2. The classification-based detection method is applied for image- and pixel-level crack detection. Targeting the accuracy and efficiency requirements on computational resource restricted device, a novel hierarchical knowledge transfer training strategy was proposed to significantly reduce the training time while maintaining similar model accuracy. A new model architecture based on dilated expansion block was designed to improve the parameter efficiency. The developed model and application were deployed and validated on HoloLens 2 for crack inspection and measurement.

4.1. Introduction

Concrete structures are often designed to bear extreme load during their lifetime usage. Overloading or degradation with age may lead to complete collapse of a concrete structure. Physical deterioration damage in the form of cracks, mostly emerges on the surface [231]. Deteriorating concrete structures are a major cause for concern according to the American Society of Civil Engineers (ASCE) as cracks are a primary indicator of structural health [232]. Timely detection and analysis of these cracks are critical to prevent accidents, costly repairs, and structural failures that can jeopardize public safety and disrupt vital services. Furthermore, the Federal Highway Administration (FHWA) estimates that billions of pounds are expended annually on repairing and maintaining concrete infrastructure, underscoring the financial repercussions of suboptimal inspection practices [233].

During the last decades, image-based automatic crack detection methods has gained popularity because they are fast, less-expensive, and robust [234-236]. This method harnesses the power of digital imaging technology to identify and analyse cracks on the surface of concrete walls and other structures. The traditional image-based crack detection algorithm relies on texture information of the image, such as greyscale value thresholding [237] or edge detection [238]. Nonetheless, the complex backgrounds in some cases hinder the determination of threshold or the accurate extraction of crack edges. Recently, the artificial intelligence (AI), especially the

convolutional neural network (CNN) has gradually shown its power in computer vision related processing tasks and gained most of the attention in crack detection area than any other techniques [239]. CNNs work by learning hierarchical and semantic features from image data, automatically identifying patterns and features in the images without the need for explicit feature engineering. This capability is especially beneficial in the context of concrete wall crack detection, where cracks can vary in size, shape, and orientation. CNNs for concrete wall crack detection have demonstrated impressive accuracy and efficiency in identifying even subtle cracks, reducing the need for time-consuming manual inspections [240]. At the same time, more and more crack image datasets have been collected and made public by research bodies both in industry and academia [241-244].

In practice, the images are usually collected by hand-held cameras [245], unmanned aerial vehicles (UAVs) [246], or head mounted device [247]. Hand-held cameras are the most basic tools for image-based crack detection. However, its effectiveness is compromised by its reliance on operator skill, difficulty of stabilisation, image capture speed, and limited real-time analysis. UAV has been widely used for crack image collection, but it requires complicated UAV hardware and control algorithm development. It also suffers from limitations of battery life, payload, regulatory considerations, mobility and cost. Combined the image sensors together with human expert experience, head-mounted device for concrete wall crack detection is a portable technology designed to assist engineers, inspectors and maintenance personnel in the on-site assessment of concrete structures. These innovative devices provide a hands-free, augmented reality (AR) or mixed reality (MR) experience that improves the efficiency and accuracy of concrete wall crack detection.

In this section, a head mounted vision-based crack inspection system deployed on HoloLens 2 was developed. Following the design methodology, the task clarification is conducted to derive the requirements and specifications for the system. The related works from the research information provide the state-of-the-art solutions to crack detection. In the conceptual design stage, the hardware specifications are analysed, the available software are compared and selected, and the conceptual detection procedure is designed. To meet the real-time requirement on AR device with restricted computational resource, the detection process is divided into crack classification, segmentation, and measurement. The crack classification and segmentation models were designed to share the same feature extractor. Only image patches with cracks detected will be passed to the segmentation model to save time.

During the embodiment design, the datasets for crack classification and segmentation are established using the open-source data. Given that the established dataset contains sufficient

crack images, the basic algorithm selection and optimisation process is adapted. The accuracy and efficiency are ensured through a proposed hierarchical knowledge transfer (HKT) strategy and new network architecture, the DE-Net. For segmentation, four different segmentation heads were implemented, including U-Net, DEU-Net, DeepLabv3+, and LR-ASPP, using the DE-Net 0.1 as backbone. They were tested on three different datasets, including Mendeley, CrackForest, and DeepCrack. This process aims to choose the best segmentation architecture and training dataset for best segmentation accuracy and model generalization ability. The GUI was designed using the MRTK APK and integrated in Unity with the selected model. By utilisation of the RGB and depth camera on HoloLens 2, the crack can be measured by combination of crack mask and point cloud. The final system is integrated and tested for crack measurement at millimetre precision.

4.2. Task Clarification

4.2.1. System Requirements and Specifications

The system requirements and specifications were established through collaborative discussions between the project members from University of Strathclyde and the stakeholders.

4.2.1.1. Crack Inspection

• The crack inspection is composed of crack classification and crack segmentation. The crack segmentation model should be called only when classification result is positive.
• The crack classification accuracy should be higher than 95%.
• The crack segmentation accuracy should be higher than 70%
• The crack classification inference speed should be real-time and faster than 20 fps.
• The crack segmentation inference speed should be real-time and faster than 5 fps.
• The crack width and length can be measured in real-world unit by depth camera.

4.2.1.2. GUI

• The GUI should include buttons for user to start and stop inspection program. The buttons should be hologram in front of user.
• The GUI should include information box showing the current progress of the program.
• The GUI should display the crack classification results showing the position of cracks.
• The GUI should display the crack segmentation results as a mask.
• The interaction with the hologram should be fluent.
• The holograms should be within the field of view of HoloLens.
• The interaction of holograms should include several basic hand actions and gestures,

including touch, hand ray and air tap.

4.2.2. Related Works

4.2.2.1. Crack Inspection

Vision-based crack inspection uses computer vision and image processing techniques to automatically detect, categorise and measure cracks in concrete surfaces. Recently, the CNN models have been widely used for crack detection. The CNN-based crack inspection can be divided into three categories: image patch crack classification, crack bounding box detection, and direct crack segmentation.

Classification method divides the whole image into image patches and determine if each patch contains crack using CNN classifier. For example, Cha and Choi [248] trained a model with images of 256×256 resolution and used sliding window technique to scan larger image. The classification-based method always needs to process a large number of image patches, which is impossible for real-time on-site crack inspection.

Crack bounding box detection can provide a more precise location information than classification and can use the entire image as input. Targeting crack detection in images with the presence of handwriting scripts, Deng et al. proposed a faster R-CNN based model for crack bounding box detection [249]. This type of method only works well when the crack is linear, in which case the bounding box can provide a relatively accurate location.

Crack segmentation is a more popular direction than the other two, which can provide pixel wise prediction for crack inspection. For example, Yang et al. implemented a fully convolutional network (FCN) for to measure diverse cracks at pixel level [250]. The single-pixel width skeletons were generated from the segmentation results for crack measurements, such as topology, length, and width. However, segmentation networks tend to be large and time-consuming, which is not suitable for head-mounted device application. For applications where the background is complex, some researchers utilise a classification model to identify the presence of cracks before implementing a segmentation model [251-254]. The feature extractor employed in both classification and segmentation models can be shared, thereby reducing the overall size of the model. This technique is adapted in this study.

As for real-time crack detection, Liu et al. proposed a deep hierarchical CNN network for crack segmentation, called DeepCrack, based on FCN and deeply-supervised nets [243]. The inference time is 0.1 s on NVIDIA TITAN X. for a single image of size 544×384. Based on Faster R-CNN and Bayesian fusion, Fang et al. achieved 0.06 s inference time on TITAN Xp

GPU, but the image resolution was not specified [255]. Chen and Jahanshahi proposed naïve Bayes based FCN and achieved 0.0276 s inference time with image size of 720×540 on TITAN X Pascal GPU [256]. To balance the speed and accuracy, Pang et al. proposed Deep Crack Segmentation Network (DcsNet) with two feature extraction branches, which can reach minimal 51.7 fps for image size of 640×544 on TITAN XP [257]. However, these methods are all tested on high-performance GPUs but not on any mobile devices. Jiang and Zhang achieved real-time crack detection on mobile phone embedded on a wall-climbing UAV, running on 6 fps [258]. The output of CNN is bounding box, followed by image processing algorithms to calculate the crack width. There are few end-to-end segmentation CNN networks that have fewer than 1 M parameters, making them unsuitable for head-mounted devices with restricted computational capacity.

4.2.2.2. Efficient AI Model

Some researchers have focused on improving the crack image classification accuracy by proposing different model architectures or loss functions. While some of other researchers have focused on reducing the inference time of CNN model when maintaining the accuracy, targeting the real-time crack detection scenario [259-262]. Focusing on the parameter efficiency, several model architectures have been developed in literature. MobileNet [263, 264] series use depth separable convolution layer which simulate the standard convolution by a depth-wise convolution and point-wise convolution with less parameter. EfficientNet [265, 266] series use progressive training and neural architecture search (NAS) to improve the parameter efficiency. Dilated convolution [267] is another effective operator that can increase the receptive field without increasing number of parameters. However, these methods need modification of original model architecture leading to time-expensive model development. Therefore, exploiting the parameter efficiency of a given model without changing its architecture is still attractive and challenging for real time crack monitoring. Transfer learning and knowledge distillation are two commonly used methods to address the above issue.

Transfer learning in deep learning refers to pretrain the model on a large dataset, and then fine-tune it on the target dataset. This technique has been widely and has proven its effectiveness in many applications [268-270]. The feature extractor part of the pretrained network contains the contextural knowledge from a larger dataset, make the model handier with a simpler task. For example, Kucuksubasi et al. used pretrained weights of InceptionV3 on ImageNet for UAVs crack detection [246]. However, ImageNet is a fairly large dataset contains more than 1 million images of 1000 classes [271], leading to days of training time on a regular GPU. This is sometimes unacceptable when fast delivery is required, and the budget is not enough to

cover a high-performance hardware platform. Though many deep learning platform such as TensorFlow [272] or PyTorch [273] provide many pretrained weights of some popular models, one may still need to train from scratch if modification is made or a completely new model is proposed. Alternatively, we use CIFAR-100 dataset containing 60k 32x32 images of 100 classes, which is another rich dataset good for transfer learning [274], and only takes several hours to pretrain.

Knowledge distillation is another model compression technique widely used for transferring knowledge from a large model to a smaller model, introduced by Hinton in 2015 [275]. The large model typically has higher knowledge capacity with very deep and wide architecture and this capacity might not be fully utilised. According to the knowledge type, there are three types of knowledge distillation [276], including response-based [275], feature-based [277], and relation-based knowledge [278]. Franco et al. used the knowledge distillation to improve the spped of neural network in a ranking task [279]. In [280], the model is pretrained on ImageNet 1k by semi-supervised label distillation (SSLD), and then finetuned on crack dataset. Our method differs mainly from theirs that the teacher model pretrained on ImageNet is used to distill its knowledge on CIFAR-100 before finetuning on crack dataset through a hierarchical knowledge transfer strategy, which greatly reduce the training time and computation power requirement as well as increase the accuracy. The contextual knowledge from ImageNet will facilitate the training on CIFAR-100, and the contextual knowledge from CIFAR-100 will then again facilitate the training on crack dataset.

4.3. Concept Design

From the requirements and research information, the HoloLens 2 is selected as the head mounted device due to its powerful on-device computation units. The available software components are compared and selected, including the SDK, 3D engine, and AI platform. To ensure the real-time inspection, the detection procedure is designed as patch-level classification, segmentation, and measurement.

4.3.1. System Architecture

4.3.1.1. Hardware and Software Selection

The head mounted device used in this project is Microsoft HoloLens 2, released on 7th November 2019. It introduced a number of improvements compared with the first generation, including a dedicated Deep Neural Network (DNN) core, custom-built Holographic Processing Unit (HPU 2.0) for computer vision, articulated hand tracking and eye gaze

tracking. For image capturing, this device has a depth camera and an RGB camera, as well as an IR eye tracking camera and four head tracking grayscale cameras.

Augmented Reality SDK facilitates many components within the AR application such as AR recognition, AR tracking and AR content rendering. In this study, the AR techniques is provided by Microsoft Mixed Reality Toolkit (MRTK) for hologram generation, interaction logic design, environment perception, etc.

3D Engine is a software-development environment for AR application, including rendering engine for 3D graphics, collision detection engine (and collision response), sound, scripting, animation, artificial intelligence, networking, streaming, memory management, threading, localisation support, scene graph, and porting applications to multiple platforms. Microsoft HoloLens 2 support two 3D engines: Unity and Unreal. Unity engine is more recommended for this project due to its support documents and language difficulty. In addition, the project team has more knowledge about Unity, which could reduce the development time.

To deploy customised application onto HoloLens, the Development Mode need to be enabled through system inside HoloLens [281]. To access the hardware sensors of the device, such as VLC cameras, depth cameras, inertial measurement unit, etc., the Research Mode needs to be enabled. The research mode API is written in C++. To integrate it with the Unity project, a C++ plugin will need to be generated. Using Windows Runtime, the API is wrapped into a C++ class and compiled as 'winmd' format, then it can be imported into the Unity project.

4.3.1.2. Detection Procedure

The detection procedure contains three steps: crack classification, precise segmentation, and crack measurement, as shown in Figure 4.1.

1) Classification

The camera image is fed into a lightweight classifier to detect the presence of cracks. The image of each frame produces 9 Boolean values by dividing the entire image into 3x3 grids and identifying the presence of cracks within each grid. The use of gridding enables the user to identify the area of attention, reduces the input size for the next stage, and is more time-efficient than using a bounding box. The more the grid, the higher the localisation precision, and also the longer the inference time. As a trade-off, the 3x3 is selected to meet the inference time requirement.

2) Segmentation

If the cracks are detected, the user can opt to calculate the segmentation by clicking the appropriate button. Each grid that contains crack will be processed individually by the segmentation model for deducing the mask of the crack. Both the classification and segmentation models utilise the identical feature extractor, thereby reducing the requirement for encoder part computation during crack segmentation. The segmentation results are overlapped onto the original image and displayed on a holographic panel.

3) Measurement

From the crack segmentation outcome, measurements for length and width can be calculated. The mask is skeletonized first. The process begins with skeletonizing the mask, followed by the designation of the crack skeleton as its length. The average width of the crack can then be determined by dividing the number of pixels in the mask by the length obtained through the skeletonization process. To convert from pixel to real-world unit, the depth camera is used. The point cloud of wall can be fitted as a plane equation. Then the length in pixel can be converted to millimetre by back projection from image points to 3D points.

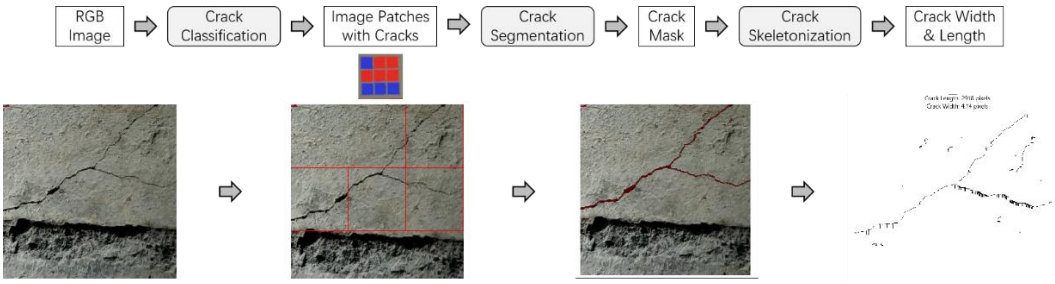


Figure 4.1. Head mounted based crack inspection procedure.

Based on the proposed procedure, the HoloLens 2 application can be developed using the embedded cameras and AI models. The application framework is shown in Figure 4.2. The AI models are firstly trained using PyTorch and tested on PC. Then the models are transferred to ONNX format for compatibility of HoloLens. The HoloLens research model API provide the access for sensor usage, such as left and right VLC camera and depth camera. A script is written in C++ using this API and compiled as Windows Runtime (Winrt) format for integration in Unity. Unity is a develop environment for AR applications. The GUI is designed inside Unity, such as buttons, information panel and other interaction and display units. The AI model and API plugin are also integrated into Unity by C# script. Finally, the Unity compile this project and deploy it onto HoloLens.

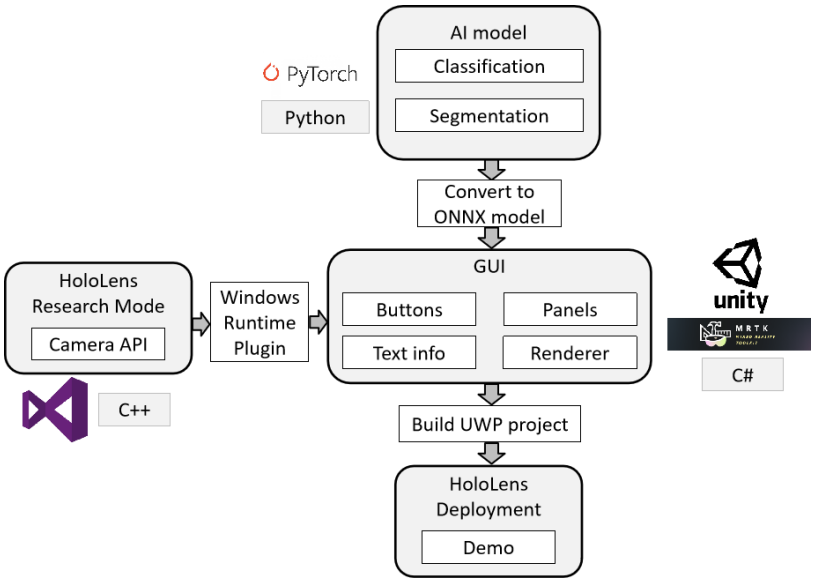


Figure 4.2. The application framework for crack inspection on HoloLens 2

4.3.2. GUI Design

For feasibility study, the GUI has been designed to only include necessary functions for testing, such as buttons, information area, detection results display area, and diagnostic tool, as shown in Figure 4.3. The information area shows the current program progress such as taking picture, the duration of classification or segmentation, and any other debugging information. The function buttons comprise initiating and halting the image capture function and executing the classification and segmentation model with the acquired image. The hand tracking is provided by the MRTK showing the current position of the hand. As designed, the image will be divided into 3x3 grid for classification. The results of each grid will be displayed in the classification results area with blue grid indicating non-crack and red grid showing crack founded. If the crack is detected in any grid, the image patch of that grid will be fed into segmentation model and the segmentation result will be overlapped on the image and displayed. The diagnostic tool provides real-time information about the application performance, including current FPS and memory usage. All of these components are displayed as holograms in HoloLens 2. The buttons and image display area are interactable for the user.

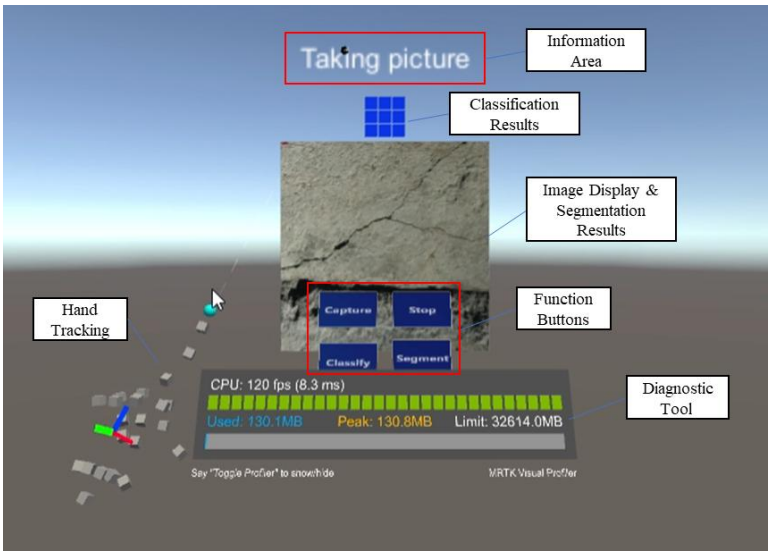


Figure 4.3. GUI design for crack inspection.

4.3.3. Evaluation Criteria

The evaluation criteria for the crack inspection system includes different subsystems:

1) Crack Classification

For crack classification model, the evaluation criterion is the image-wise accuracy defined by Eq. (2-1) on the test dataset.

2) Crack Segmentation

For crack segmentation model, the F1 score, also named as dice score, is used as evaluation criterion, as defined in Eq. (2-4).

3) Crack Measurement

The crack should be measured in millimetre level. The relative error of the crack length and width should be used.

4) Real-time Inspection

The final integrated system should be able to perform the real-time inspection according to the framerate proposed in the task clarification.

4.4. Crack Classification

For crack classification model design, a novel hierarchical training strategy is proposed that combines transfer learning and knowledge distillation, targeting fast model development for real-time crack classification. Additionally, based on dilated convolution and MobileNetV3, a

new convolution block architecture is proposed, and a model named DE-Net is built from it.

4.4.1. Hierarchical Knowledge Transfer Training Strategy

4.4.1.1. Transfer Learning

In supervised transfer learning for classification task, basically two labelled datasets are presented, namely the origin dataset D_O and destination dataset D_D, defined as:

$$D_O = \{(x_{O,i}, y_{O,i}) \mid i \in [1, M]\} \tag{4-1}$$

$$D_D = \{(x_{D,j}, y_{D,j}) \mid j \in [1, N]\} \tag{4-2}$$

where (x_i, y_i) is an image-label pair and M, N are the total number of images in the two datasets. The images are noted as $X = \{x_1, x_2, \dots, x_n\} \in \mathbb{R}^{W \times H \times C}$, where W, H and C are image width, height, and channel. The labels are integers noted as $Y = \{y_1, y_2, \dots, y_n\} \in [1, C_l]$, where C_l is the number of classes of the dataset. The classification training task T on the dataset $D = \{(X, Y)\}$are defined as:

$$T = \{f_\theta \mid D\} \tag{4-3}$$

where f_θ is the neural network which maps the input image to probabilities of each label values with parameter set θ, expressed as:

$$f_\theta = P_\theta(Y|X): \mathbb{R}^{W \times H \times C} \to \mathbb{R}^{C_l} \tag{4-4}$$

and the estimation of parameter set θ can be treated as an optimisation problem:

$$\hat{\theta} = \arg\max_{\theta \in \Theta} P_\theta(Y|X) = \arg\min_{\theta \in \Theta} L(f_\theta(X), Y) \tag{4-5}$$

where Θ is the space containing all possible parameter values and $L(f_\theta(X), Y)$ is the loss function identifying the difference between the prediction and label. In classification task, the loss function is usually cross-entropy loss defined as:

$$L_{CE}(p, Y) = -\sum_{n=1}^{C_l} y_n \log(p_n) \tag{4-6}$$

where $p = f_\theta(X)$ is the prediction of input X, and p_n and y_n are the prediction and label of class n respectively.

The transfer learning is then defined as a method that improves the destination task performance $T_D = \{f_\theta \mid D_D)\}$ on destination dataset $D_D = \{(X_D, Y_D)\}$ with the help of the

origin task $T_O = \{f_\theta \mid D_O)\}$ and origin dataset $D_O = \{(X_O, Y_O)\}$. It is usually assumed for the training task T_O and the destination task T_D that $D_O \neq D_D$ and $P(Y_O|X_O) \neq P(Y_D|X_D)$. This assumption is true either when the two datasets contain different classes or the same classes but different image distribution. Usually, a classification network can be treated as two parts, i.e., feature extractor f_F and classifier f_C:

$$f_\theta(X, \theta) = f_c(f_F(X, \theta_F), \theta_C) \tag{4-7}$$

and $\theta = \{\theta_F, \theta_C\}$ is the parameter set of feature extractor and classifier. The simplest yet effective transfer learning method is reusing the feature extractor f_F pretrained on origin dataset and finetuning on destination dataset, that is, using the feature extractor parameter set from origin dataset $\theta_{O,F}$ as initial value of the one from destination dataset $\theta_{D,F}$. A typical implementation is fixing $\theta_{D,F}$ for the first several epochs and finetuning the whole parameter set $\theta_D = \{\theta_{D,F}, \theta_{D,C}\}$ with a lower learning rate.

4.4.1.2. Knowledge Distillation

As a model compression technique, knowledge distillation tries to train a smaller student model from a larger teacher model. For a specific task T, the teacher model, represented as f_{θ_T}, has higher knowledge capacity than the student model f_{θ_S}. The knowledge distillation is defined as a method that improves the student task performance $T_S = \{f_{\theta_S} \mid D)\}$ with the help of the teacher task $T_T = \{f_{\theta_T} \mid D)\}$ on the same dataset $D = \{(X, Y)\}$. The key difference between knowledge distillation and transfer learning is that transfer learning trains the same model on different datasets but knowledge distillation trains different models on the same dataset. The knowledge distillation can be formulated as an optimisation problem:

$$\hat{\theta}_S = \arg\min_{\theta_S \in \Theta} L\big(\varphi_S(o_S), \varphi_T(o_T)\big) \tag{4-8}$$

where o_S and o_T are the outputs of the student and teacher models, that can be either the final digits output of classifier before softmax function or mid-layer feature maps. φ_S and φ_T are transformation functions mapping the outputs to the same dimensional space. L is the loss function measuring the difference between the transformed outputs from the teacher and student models.

The initial knowledge distillation method proposed by Hinton [275] using soft target (ST) combined with attention transfer (AT) [277] has achieved relatively high distillation effectiveness and is easy to be implemented. Therefore, the distillation method combining ST and AT was adopted to train the student model on CIFAR 100 dataset. The soft targets are

outputs of modified softmax function:

$$\varphi(o^i, T) = \exp\left(\frac{z^i}{T}\right) / \sum_j \exp\left(\frac{z^j}{T}\right) \tag{4-9}$$

where the output o^i is defined as the logit output z^i from model classifier for the i-th class and temperature T is to control the weight of each class. The ST method uses KullbackLeibler (KL) divergence loss between soft targets of teacher and student to train the student model:

$$L_{ST} = \frac{1}{C_D} \sum_i \varphi(o_T^i, T) \log\left(\frac{\varphi(o_T^i, T)}{\varphi(o_S^i, T)}\right) \tag{4-10}$$

In AT method, the feature maps of mid-layer are used as output with L2 norm as transformation function:

$$\varphi(o^j) = L_2(m^j) = \frac{m^j}{\|m^j\|_2} \tag{4-11}$$

where m^j is the j-th feature map. The loss function is defined as:

$$L_{AT} = \sum_j \left\| L_2(m_T^j) - L_2(m_S^j) \right\|_p \tag{4-12}$$

where p refers to norm type and is set to 2 in this research. The final loss combining basic cross-entropy loss, knowledge distillation loss ST and AT is:

$$L_{KD} = \alpha L_{CE} + \beta L_{ST} + \gamma L_{AT} \tag{4-13}$$

where L_{CE} is the cross-entropy loss of the student model, α, β and γ are the weights for three losses.

4.4.1.3. Hierarchical Knowledge Transfer

The goal of the proposed hierarchical knowledge transfer (HKT) training strategy is trying to exploit fully the parameter potential of the target model so that we could design a crack classification model as small and fast as possible while maintaining the performance. The scheme of the proposed training strategy is illustrated in Figure. 4.4. This training strategy includes three datasets and two models. The first dataset is a very large and rich image collection, such as ImageNet which contains more than 14 million images and 1000 classes. The second dataset is a relatively small but still rich image set, such as CIFAR-100 which contains 60k 32x32 images and 100 classes. The third dataset is the target crack dataset, which is the smallest one with only 4.8k images of two classes, i.e., with and without crack.

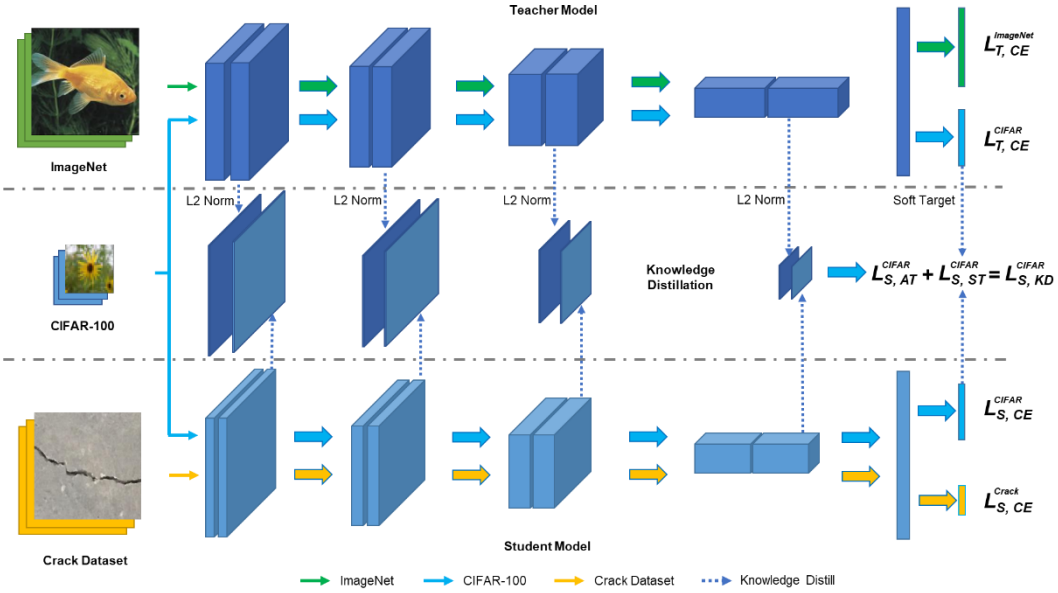


Figure. 4.4. Scheme of hierarchical knowledge transfer training strategy.

The teacher model, which has higher compacity than the target student model, is first pretrained on ImageNet and then finetuned on CIFAR-100. The pretraining process on ImageNet makes the teacher model perform better on CIFAR-100 and become a better teacher. Then the knowledge is distilled by ST and AT methods to the student model combined with its own cross-entropy loss. Finally, the student model is finetuned on the target crack dataset. Instead of finetuning the student model by frozen part of its parameters, it is finetuned on resized crack dataset with the same image size of CIFAR-100 (32x32) and then increased to 224x224, during which the whole set of parameters are trainable. Training with multi-scale image is also proved effective in [8, 26]. The knowledge transfer path is shown in Figure. 4.5. The training loss for each stage is described as equation (6) for transfer learning and (13) for knowledge distillation.

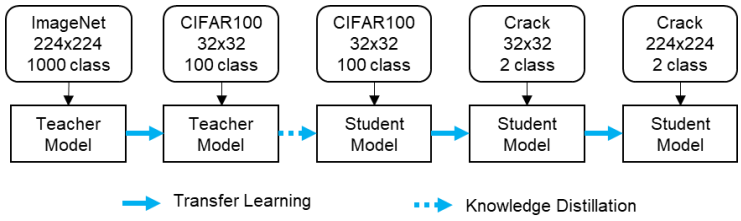


Figure. 4.5. Knowledge transfer path of the proposed training strategy.

4.4.2. DE-Net

4.4.2.1. Dilated Expansion Block

Dilated convolution, or atrous convolution, was initially published by [267] for semantic segmentation. It can enlarge receptive field of the convolution operator without increasing the

number of parameters. This is realised by skipping some points during convolution, depending on the dilation factor l. Formally, let $F: \mathbb{Z}^2 \longrightarrow \mathbb{R}$ be a discrete function, and $k: \Omega_r \longrightarrow \mathbb{R}$ be a discrete filter with $\Omega_r = [-r, r]^2 \cap \mathbb{Z}^2$ be its receptive field. The following equation is from [267]. The dilated convolution operator $*_l$ with dilation factor l at location i is:

$$(F *_l k)(i) = \sum_{j \in \Omega_r} F(i + l \cdot j)k(j) \tag{4-14}$$

when $l = 1$, it is equivalent to standard convolution. Concatenation of parallel dilated convolution layers of different dilation rates is called atrous spatial pyramid pooling (ASPP), proposed in [282] for semantic segmentation. It has high capacity to capture the spatial context information with high efficiency through the large receptive field of dilated convolution. While the dilated convolution emphasises on the spatial-wise information aggregation, the Squeeze-and-Excitation (SE) block focuses on recalibration of channel-wise responses, first proposed in [283]. The feature map is pooled to on dimensional vector, followed by two fully connected layers (FC) to compute the channel attention vector. The output is obtained by multiplying the attention vector with the original feature map. The MobileNetV3 architecture introduced the SE block into the inverted residual residuals and linear bottleneck in MobileNetV2. The bottleneck block first expands the input channels to 3 to 6 times by Conv 1x1 (convolution with 1x1 kernel size), followed by depth separable convolution consisting of one depth wise convolution (Dwise) and one Conv 1x1. Inside the depth separable convolution, the SE block is added. In this research, the dilated expansion block is proposed to replace the original expansion part of Conv 1x1 by an ASPP module, as shown in Figure. 4.6.

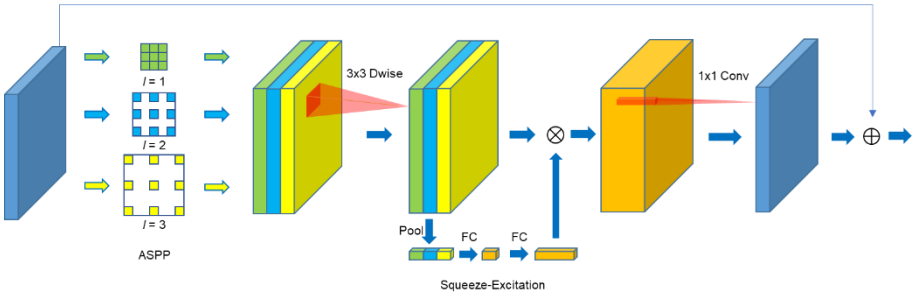


Figure. 4.6. Architecture of dilated expansion block.

The DE block can benefit from the ASPP in two aspects. Firstly, the receptive field is greatly enlarged with the little computation load increase due to the parallel paths of different dilated convolution. Secondly, since the spatial information is encoded into the channel-wise context after concatenation of dilated convolutions with different dilated ratio, the Squeeze-and-Excitation operation could carry out the channel-wise and spatial-wise recalibration at the

same time. The comparison of different block architecture used in four models is shown in Figure. 4.7.

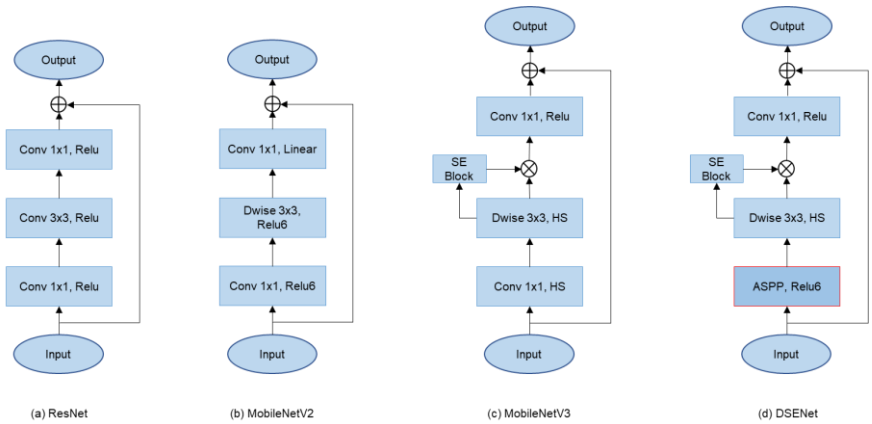


Figure. 4.7. Block architecture of (a) ResNet, (b) MobileNetV2, (c) MobileNetV3 and (d) DE-Net

4.4.2.2. Model Architectures of DE-Net

For crack classification, based on the MobileNetV3 architecture and DE block, a new model architecture was proposed named DE-Net. Its specification is listed in Table 4.1. Since the ASPP has provided a large receptive field, all the DE-blocks use 3x3 kernel size. The expiation rate (column Exp_rate) is set to 3 except the first DE-block. The output channel is kept as small as possible until the last several layers to reduce the computation expense. The non-linear activation layer (column NL) follows the MobileNetV3 configuration. The SE block and dilation expansion (column SE and Dilated) are always used together except the last DE-block. Similar as the MobileNets, the width multiplier is used to adjust the size of the model. As only one class is to be classified, the final channel number was set to 128.

Table 4.1. Specification of DE-Net Architecture

Layer	Input	Operator	ER	CO	SE	DL	NL	Stride
1	$224^2 \times 3$	Conv2d	-	16	-	-	HS	2
2	$112^2 \times 8$	DE-block	1	16	-	-	HS	2
3	$56^2 \times 8$	DE-block	3	16	-	-	RE	2
4	$28^2 \times 8$	DE-block	3	24	✓	✓	RE	1
5	$28^2 \times 8$	DE-block	3	24	✓	✓	HS	2
6	$14^2 \times 8$	DE-block	3	40	✓	✓	HS	1
7	$14^2 \times 8$	DE-block	3	40	✓	✓	HS	1
8	$14^2 \times 8$	DE-block	3	40	✓	✓	HS	1
9	$14^2 \times 8$	DE-block	3	48	✓	✓	HS	2
10	$7^2 \times 16$	DE-block	3	96	✓	-	HS	1
11	$7^2 \times 16$	Conv2d 1x1	-	96	-	-	HS	1
12	$7^2 \times 96$	AvgPool	-	-	-	-	-	1
13	$1^2 \times 96$	FC	-	128	-	-	HS	1
14	$1^2 \times 128$	FC	-	k	-	-	-	1

ER = expansion rate, CO = channel of output, SE = squeeze and excitation, DL = dilated, NL = nonlinear activation.

4.4.3. Experiments and Results

The proposed hierarchy knowledge transfer training strategy is used for tiny model development of crack classification problem. A crack dataset consisting of several public crack datasets and images from internet is built to validate the proposed training strategy and the dilated expansion block. The teacher networks chosen in the experiments are MobileNetV2 and ResNet-18. The student networks are MobileNetV2 with width multiplier 0.1, MobileNetV3 0.1 and DE-Net.

4.4.3.1. Crack Classification Data

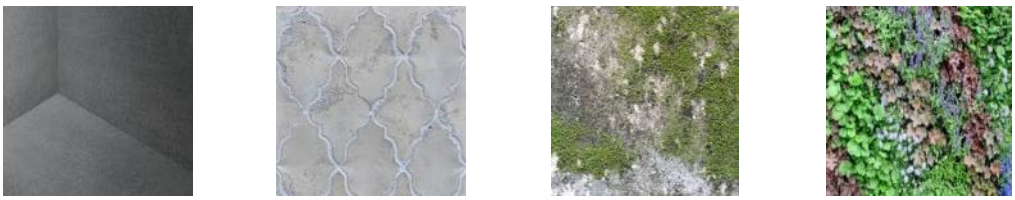
The images for crack classification are from two public crack classification datasets, i.e. Mendeley [241] and SDNET [242] for both positive and negative data, and two crack segmentation datasets for positive only data, i.e. CrackForest [284] and DeepCrack [243]. To train a crack classifier with higher generalization ability, we downloaded some non-crack images from Internet including concrete corner, concrete tile, moss, and plant on wall, with 275 images each category and 1100 in total. The data used for training are listed in Table 4.2. Some sample images are shown in Figure. 4.8 and Figure. 4.9. There are 2400 images for crack and non-crack data respectively, 400 of which are randomly chosen as testing data and not used in training, and another 400 images are chosen as validation data. Before fed into the network, some typical augmentations were applied such as rotation, crop, and flip.

Table 4.2. Data composition for crack classification training

Data Source	Crack Image	Non-Crack Image	Resolution
Mendeley	1360	1000	227×227
SDNET	385	300	256×256
CrackForest	118	0	448×448
DeepCrack	537	0	544×384
Internet	0	1100	-
Total	2400	2400	-

(a) Mendeley (b) SDNET (c) CrackForest (d) DeepCrack

Figure. 4.8. Sample crack images for crack classification



(a) Corner (b) Tile (c) Moss (d) Plant

Figure. 4.9. Sample non-crack images for crack classification

4.4.3.2. Model Configuration

Four model architectures were used during the experiment, including Resnet, MobileNetV2, MobileNetV3 small and DE-Net. MobileNetV2 and V3 are the second and third version of MobileNet, famous for its excellent trade-off between model size and performance. The width multiplier introduced by MobileNet is a hyperparameter for adjusting the model size, which was set to 1.0 and 0.1 as teacher and student network in the experiment. The models with a 1.0 width multiplier represent the standard benchmark models commonly used in literature for comparison. As the number of channels in each convolution layer must be quantized by a factor of 8, applying a width multiplier smaller than 0.1 has minimal impact on the overall model size. The architecture of these four networks is compared in Figure. 4.7. The implementation of MobileNet and ResNet was taken from torchvision [285] directly. The model size (number of parameters) of the four networks are compared in Table 4.3. Different combinations of teacher and student will be experimented in the next part.

In the following parts, testing of different teacher-student combinations was implemented to validate the effectiveness of the proposed training strategy. At the same time, the DE-Net was also evaluated regarding to both the training time and crack classification performance. The experiments include three parts, the performance analysis of DE-Net, the knowledge distillation on CIFAR-100 and classification on crack dataset.

Table 4.3. Comparison of teacher and student model size

Model	**RN-18**	**MN2-1.0**	**MN2-0.1**	**MN3-0.1**	**DEN-0.1**
Role	Teacher	Teacher	Student	Student	Student
Parameters (k)	11177.02	2225.15	76.98	31.78	32.77
Memory size (kB)	44786	6068	473	207	194

RN-18 = ResNet-18, MN2-1.0 = MobileNetV2 with width multiplier 1.0, MN2-0.1 = MobileNetV2 with width multiplier 0.1, MN3-0.1 = MobileNetV3 with width multiplier 0.1, DEN-0.1 = dilated excitation network with width multiplier 0.1.

4.4.3.3. Performance Analysis of DE-Net

Since CIFAR-100 is a well-explored dataset, the training strategy followed the typical setting [286], that is the initial learning rate was set to 0.1 and rescheduled by multistep learning rate decay with gamma of 0.2 and milestones of the 60^{th}, 120^{th} and 160^{th} epoch with 200 epochs in total with batch size of 128. The Stochastic gradient descent (SGD) optimiser was used with momentum of 0.9 and weight decay of 5e-4. This setting is not the optimal settings for all the models but was kept the same to maintain consistency.

To compare the parameter efficiency of the three model architectures, different width multipliers were used to change the number of parameters of models. These models were then trained on CIFAR-100 and the top-1 accuracy is compared in Figure. 4.10. Noted that the parameters are obtained from models with 100 output classes. With the same number of parameters, the DE-Net achieves the highest accuracy among the three models. It can also be observed that even with width multiplier as 1.0, the model size of DE-Net is still less than one third of MobileNetV3 but achieves 2% higher accuracy. This can lead to conclusion that DE-Net has the highest parameter efficiency and is most suitable for small model applications, such as crack classification.

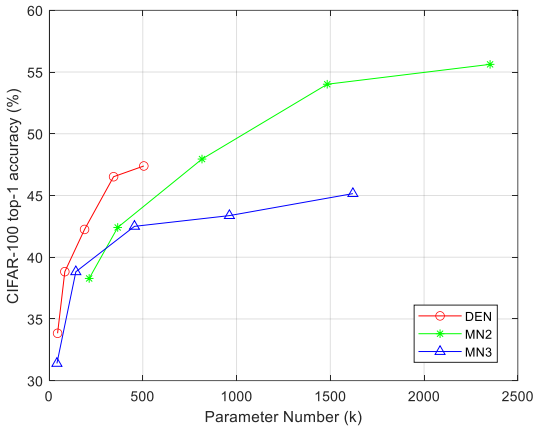


Figure. 4.10. Accuracy comparison of three models against parameter number trained on CIFAR-100.

In practice, the inference time latency of the model is a more important indicator than model size. To compare the latency, the inference time on CPU was recorded and averaged for CIFAR-100 images. The latency of the three architectures on CIFAR-100 is shown in Figure. 4.11. This shows similar trends as the parameter number. The DE-Net achieves the best latency-accuracy trade-off which only takes about 7 micro-second(ms) for one image.

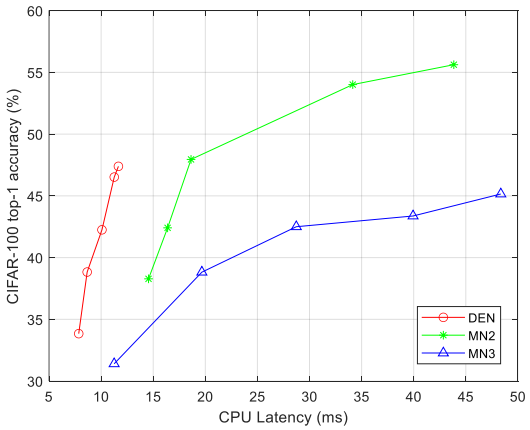
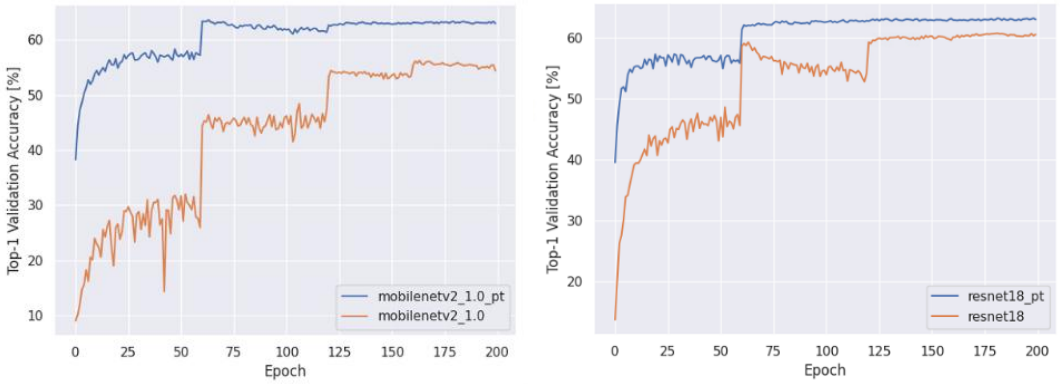


Figure. 4.11. Accuracy comparison of three models against inference CPU latency on CIFAR-100.

For ImageNet, due to the limit of GPU memory and computational power, the training of each architecture takes more than one week, only the models with width multiplier 0.1 were trained and compared. The top-1 accuracies on ImageNet test set of MobileNetV2 0.1, MobileNetV3 0.1, DE-Net 0.1, are 34.24%, 24.77, 25.87%, respectively. Since the model size of MobileNetV2 0.1 is more than twice of the other two, its accuracy is much higher. The DE-Net 0.1 is still better than MobileNetV3 0.1 with 1.1% higher accuracy.

4.4.3.4. Knowledge Distillation on CIFAR-100

The teacher model needs to be trained with the student model on the same dataset for knowledge distillation. Here the pretrained weights on ImageNet of MobileNetV2 with wm-1.0 and ResNet-18 from torchvision are used to boost the teacher model accuracy by transfer learning. The initial learning rate for pretrained weights was set to 0.01. The comparison of with/without pretrained weights of the two teacher models is shown in Figure. 4.12.



(a) MobileNetV2 1.0 (b) ResNet-18

Figure. 4.12. Validation accuracy with/without pretrained weights on ImageNet.

It can be observed that model with pretrained weights converges faster and has higher final performance. The final top-1 accuracies of MobileNetV2 and ResNet-18 are improved from 56.25 and 60.77 to 63.62% and 63.24%, respectively. It is proved that the student model will benefit from teacher model with higher accuracy, as in Table 4.4.

Table 4.4. Knowledge distillation results of different teacher-student pairs on CIFAR-100

Teacher	Student	Strategy	Accuracy (%)	Train Time (min)
-	MN2-0.1	Scratch	38.28	35.5
MN2-1.0	MN2-0.1	KD	41.06 (+2.78)	67.2
MN2-1.0-pt	MN2-0.1	KD	41.37 (+3.09)	66.7
RN-18	MN2-0.1	KD	40.87 (+2.59)	57.5
RN-18-pt	MN2-0.1	KD	41.12 (+2.84)	59.5
-	MN3-0.1	Scratch	31.40	24.6
MN2-1.0	MN3-0.1	KD	32.45 (+1.05)	50.0
MN2-1.0-pt	MN3-0.1	KD	33.01 (+1.61)	50.5
RN-18	MN3-0.1	KD	32.64 (+1.24)	39.9
RN-18-pt	MN3-0.1	KD	33.18 (+1.78)	40.0

-	DEN-0.1	Scratch	33.84	24.2
MN2-1.0	DEN-0.1	KD	36.54 (+2.70)	49.1
MN2-1.0-pt	DEN-0.1	KD	36.89 (+3.05)	48.8
RN-18	DEN-0.1	KD	36.49 (+2.65)	38.8
RN-18-pt	DEN-0.1	KD	37.76 (+3.92)	39.5

-pt = pretrained weights from ImageNet, KD = knowledge distillation.

As for the knowledge distillation configuration, the α, β and γ was set to 0.1, 0.9 and 10 empirically for all teacher-student pairs, following [287]. The temperature of ST method was set to 2. The mid-layers for AT method were chosen as the three layers that down-sample the feature maps to 16x16, 8x8, and 4x4. The comparison results of different teacher-student pairs are shown in in Table 4.4. The 'pt' in teacher model means this model is trained on CIFAR-100 with pretrained weights on ImageNet.

It can be observed that the knowledge distillation method can improve the top-1 accuracy by 3.09%, 1.22%, 3.88% for MobileNetV2 0.1, MobileNetV3 0.1 and DE-Net 0.1, respectively. For MobileNetV2 0.1, the MobileNetV2 1.0 teaches better because they have the same model architecture. While for MobileNetV3 0.1 and DE-Net 0.1, ResNet-18 is a better teacher. Since the MobileNetV2 0.1 has the largest number of parameters, it performs best among the three students. With similar number of parameters, DE-Net 0.1 achieved higher accuracy and better knowledge distillation improvement than MobileNetV3 0.1. In addition, the teacher models with pretrained weights on ImageNet improve the student model performance by 0.25%~1.27% top-1 accuracy. The comparison in chart form is shown in Figure. 4.13.

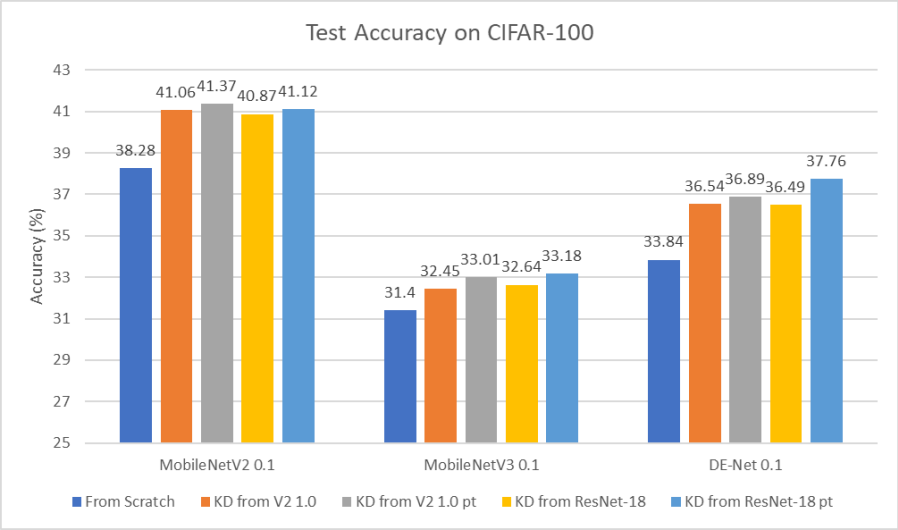


Figure. 4.13. Knowledge distillation results comparison for different teacher-student pairs on CIFAR-100.

4.4.3.5. Knowledge Transfer on Crack Dataset

Similar to the set up on CIFAR-100, the optimiser used was SGD with initial learning rate of 1e-1 for model training from scratch, and 1e-2 for finetuning. The weight decay was set to 5e-

4 with momentum of 0.9. The batch size was set to 32, and total number of epochs was 50. The scheduler of multistep learning rate decay was used with gamma of 0.2 and milestones of the 10^{th}, 20^{th} and 40^{th} epoch. The loss function is simply the binary-cross-entropy loss. When finetuning the pretrained weights following the typical scheme, the first 2/3 parameters of the whole network were frozen for the first 10 epochs. Alternatively, since the pretrained dataset CIFAR-100 has image size of 32x32, the pretrained weights can be finetuned all together on crack dataset with image size of 32x32. Then finetune the whole network with image size of 224x224 for another 40 epochs. During the training, only weights with the best validation accuracy were saved as the final weights. The results comparison of different training strategy on crack dataset are shown in Table 4.5 and Figure. 4.14.

Considering the accuracy only, the results show that pretraining on ImageNet is still the best choice disregard the training time. However, each model takes more than one week to train on ImageNet which is sometimes unacceptable. Instead, pretraining on CIFAR-100 takes only around one hour and can obtain comparable accuracy.

Table 4.5. Results of crack dataset with different training strategy

Model	Strategy	Accuracy (%)	Train Time (min)	Total Train Time (min)
MN2-0.1	Scratch	96.50	22.2	22.2
MN2-0.1	PT on IN	99.75 (+3.25)	22.3	8506
MN2-0.1	PT on CI	99.28 (+2.78)	22.4	89.1
MN2-0.1	KD on CI	99.40 (+2.90)	22.5	89.2
MN2-0.1	KD and FT with 32x32	99.76 (+3.26)	21.4	88.1
MN3-0.1	Scratch	96.15	21.5	21.5
MN3-0.1	PT on IN	99.16 (+3.01)	21.6	7954
MN3-0.1	PT on CI	97.12 (+0.97)	21.5	61.5
MN3-0.1	KD on CI	97.36 (+1.21)	21.6	61.6
MN3-0.1	KD and FT with 32x32	97.36 (+1.21)	20.1	60.1
DEN-0.1	Scratch	96.27	21.5	21.5
DEN-0.1	PT on IN	99.40 (+3.13)	21.8	7885
DEN-0.1	PT on CI	98.21 (+1.94)	21.7	61.2
DEN-0.1	KD on CI	98.56 (+2.29)	22.0	61.5
DEN-0.1	KD and FT with 32x32	98.68 (+2.41)	20.0	59.5

PT = pretrain, KD = knowledge distillation, FT = finetune, IN = ImageNet, CI = CIFAR-100, the total train time includes the pretrained time plus the crack dataset train time.

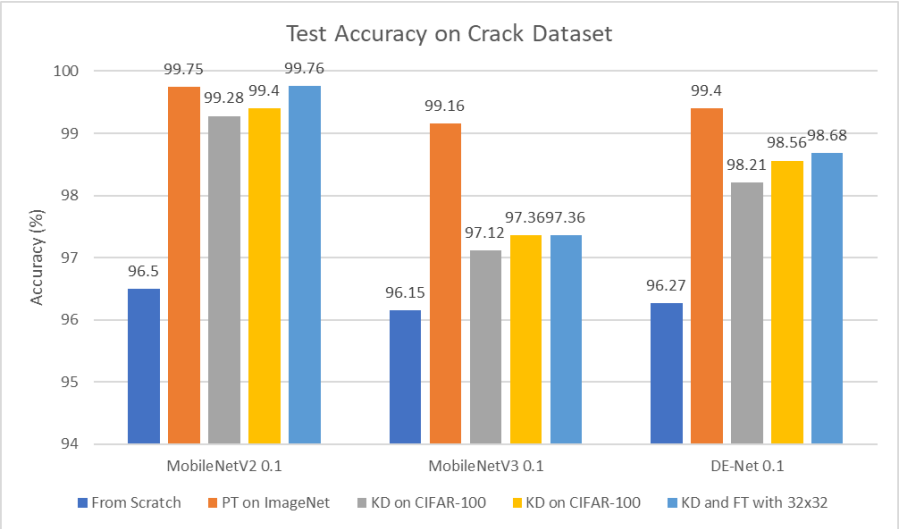


Figure. 4.14. Results comparison of crack dataset in chart with different training strategy.

For pretraining on CIFAR-100 dataset, the crack dataset accuracy is positively related to the pretraining accuracy. MobileNetV2 0.1 has the highest accuracy among the three models and achieved 99.76% accuracy and just surpasses using pretrained weights on ImageNet. All three models reach highest accuracy when using the complete hierarchical training strategy. By simply pretraining on CIFAR-100, an accuracy improvement of 0.97%~2.78% can be obtained. Then, the knowledge distillation gives additional 0.12%~0.35% accuracy uplift. The finetuning with 32x32 images makes no difference with standard method for MobileNetv3 0.1 but gives 0.36% and 0.12% increase to MobileNetV2 0.1 and DE-Net 0.1, respectively. As shown in Table 4.5, the total training time including pretraining on CIFAR-100 and finetuning on crack dataset is less than 1.5 hour for all the models, which is a significant reduce compared with one week on ImageNet. With similar number of parameters of MobileNetV3 0.1 and half of MobileNetV2 0.1, DE-Net 0.1 achieved 98.68 accuracy with hierarchical training strategy. Trade-off between model size and performance is well achieved. Therefore, the proposed hierarchical knowledge transfer training strategy is validated to be effective for different model architectures, and the proposed DE-Net is proved to have a good balance between model size and accuracy.

4.5. Crack Segmentation

4.5.1. Segmentation Model Candidates

For crack segmentation, the proposed DE-Net was used as the encoder for feature extraction. Four different segmentation head architectures were used for experiments, namely U-Net [288], Dilated Expansion U-Net (DEU-Net), DeepLabv3+ [282], and Lite Reduced Atrous

Spatial Pyramid Pooling (LR-ASPP) [264].

U-Net is a U-shaped encoder-decoder network architecture, with a contracting path to capture context and a symmetrical expanding path for precise localisation. Here, the variant of U-Net is used with the proposed DE-Net as backbone. The feature maps from different levels are used and aggregated through merge blocks, as shown in Figure 4.15 (a). Inside the merge blocks, the feature map from higher level is upsampled and concatenated with lower level feature map, followed by two 3x3 convolution blocks including batch normalization and ReLU activation. The segmentation result is obtained by aggregation from the highest to lowest level feature map. The DEU-Net has the same architecture as U-Net except that all the two 3x3 convolution blocks in each merge blocks are replaced by one proposed DE block with expansion rate as 3.

The DeepLabv3+ is based on the ASPP module which uses multiple atrous convolutions. Two feature maps from the encoder were used, whose sizes are 1/4 and 1/16 of the original image size containing low- and high-level features of the image. The ASPP was used to extract spatial features from the 1/16 feature map. This is then aggregated with the 1/4 feature map to form the segmentation results, as shown in Figure 4.15 (b).

The LR-ASPP was proposed together with MobileNetV3 for segmentation tasks. It simplified the ASPP block with fewer parameters but similar performance. The 1/16 feature map was separated into two branches and merged using attention mechanism. Similar to DeepLabv3+, the fusion of feature maps from two different resolutions was the predicted segmentation results, as shown in Figure 4.15 (c). Unlike the original architecture which uses the feature map of 1/8 resolution, here instead the feature map of 1/4 resolution was used to enable very thin crack detection. For crack segmentation, all the outputs from the three segmentation heads are followed by a 3x3 convolution block with 1 class output channel, and upsampling to original image size.

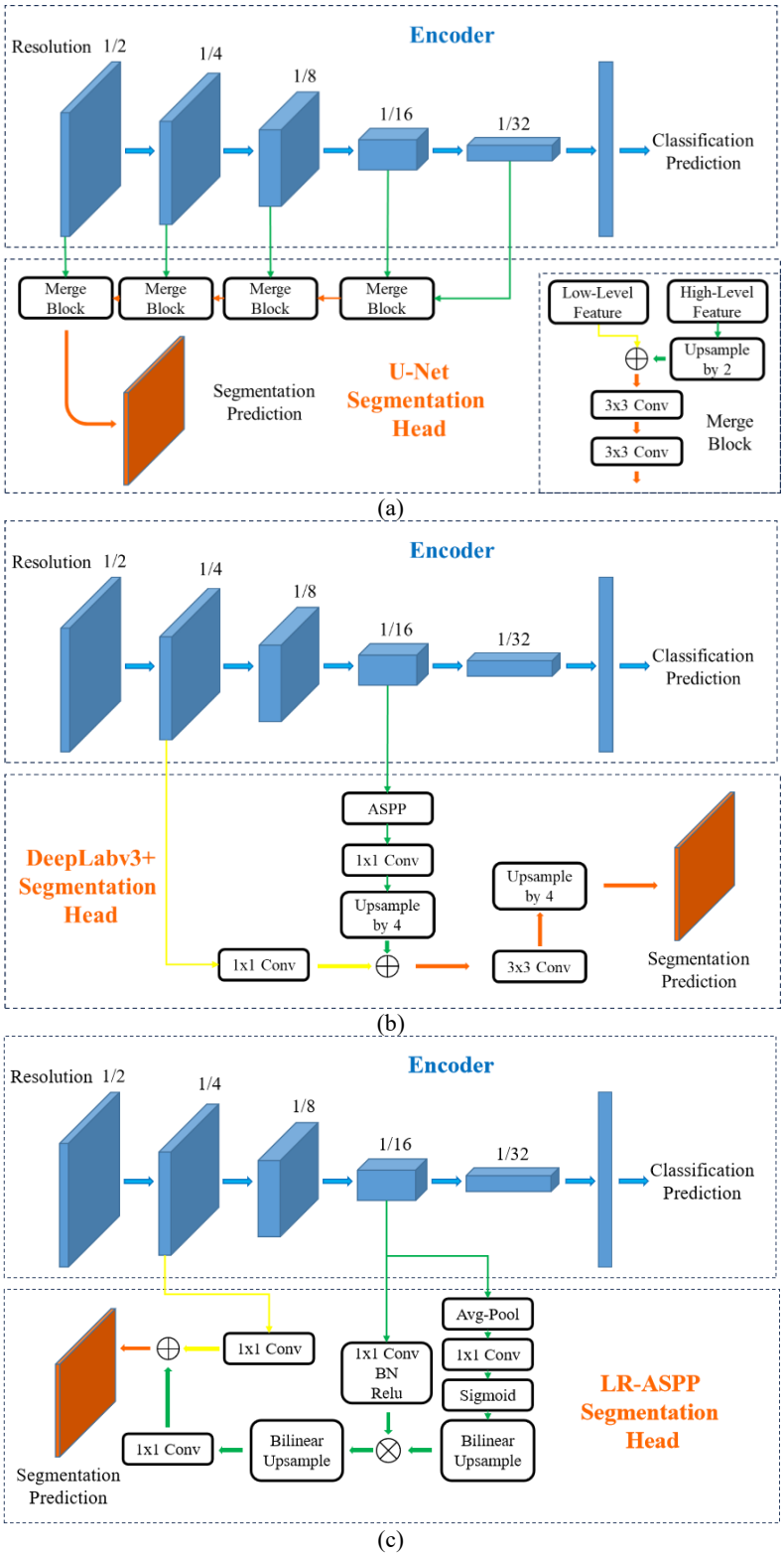


Figure 4.15. Crack segmentation models using different heads (a) U-Net, (b) DeepLabv3+, (c) LR-ASPP.

4.5.2. Experiments and Results

The feature extractor of the crack classification network is used as encoder of the segmentation

network and share the same weights pretrained on classification datasets. Four different segmentation architectures, namely U-Net, DEU-Net, DeepLabv3+, and LR-ASPP, are compared on three different segmentation datasets.

4.5.2.1. Crack Segmentation Data

The crack segmentation dataset includes three public datasets including Mendeley [289], CrackForest [284] and DeepCrack [243]. It should be noticed that the Mendeley crack segmentation dataset is different from the classification dataset. The other two datasets are the same as the one in the classification part but with segmentation labels. Instead of using the images from the three datasets all together for training, they were fed into the model separately. The model trained on one dataset will be tested on another two datasets for generalization ability evaluation. The data used for training are listed in Table 4.6.

Table 4.6. Data composition for crack segmentation training

Data Source	Numbers	Resolution
Mendeley	445	224×224
CrackForest	118	448×448
DeepCrack	537	544×384
Total	1100	-

Some sample images are shown in Figure 4.17. The Mendeley dataset is purely crack images of concrete wall. The CrackForest dataset is purely crack images of road pavement. While the DeepCrack dataset is a mix of road pavement and concrete wall. Similar as the classification setup, before fed into the network, data augmentations were applied to images and masks such as rotation, crop, and flip.

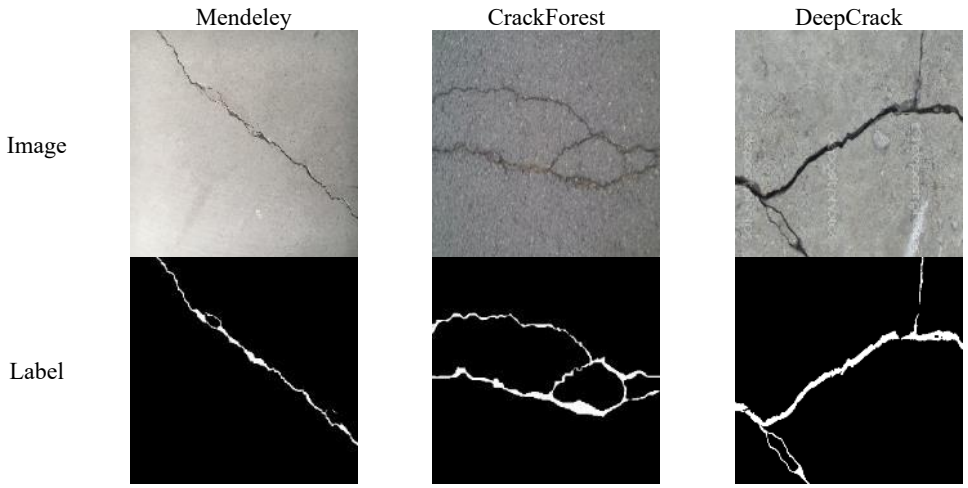


Figure 4.16. Sample crack images for crack segmentation.

4.5.2.2. Model Configuration

For segmentation model training, the DE-Net 0.1 will be used as the backbone followed by

four different segmentation heads, i.e., U-Net, DEU-Net, DeepLabv3+, and RL-ASPP. For U-Net, the feature maps of 1st, 2nd, 3rd, 5th, 9th layers are aggregated by merge blocks. The output channels of the merge blocks are 8, 16, 32, 64, respectively. The DEU-Net is the same as the U-Net except that the two convolution layers in each merge blocks are replaced by DE blocks with expansion rate of 3. The feature maps of 2nd and 5th are used for DeepLabv3+, corresponding to 1/4 and 1/16 resolution. The output channel of lower-level feature map is 48 and higher-level feature map is 128. Similarly in RL-ASPP, the 2nd and 5th feature maps are used. The output channels of the convolutional layers are all 128, as described in the original paper. The complete models with different segmentation heads are compared in Table 4.7. The parameters include the whole DE-Net with classification head. The DeepLabv3+ has the most parameters and largest memory size due to the ASPP module. Even though the U-Net used the layers from the backbone, its number of parameters are less than the DeepLabv3+. The parameters of DEU-Net are less than U-Net due to usage of DE block. The RL-ASPP has the least number of parameters due to its simplified architecture.

Table 4.7. Comparison of segmentation models with different segmentation heads

Model	U-Net	DEU-Net	DeepLabv3+	RL-ASPP
Layers from backbone	1,2,3,5,9	1,2,3,5,9	2,5	2,5
Parameters (k)	137.149	129.477	177.213	65.749
Memory size (kB)	644	621	814	343

In the following experiments, the same backbone DE-Net will be used but with different pretraining weights. The first set of weights is from crack classification training using pretrained from ImageNet. The other one is from the model using the proposed hierarchical knowledge transfer strategy. To make sure the backbone is unchanged and the segmentation model can be integrated with the classification model, during training, these weights will be frozen and do not participate the updating process.

4.5.2.3. Segmentation Results and Discussion

For all the segmentation heads and datasets, the model training configuration is the same. The training data was split into train and validation sets with split ratio of 0.8. The Adam optimiser was used with the learning rate set as 0.001 and weight decay as 0.0001. The number of epochs was 1000 and batch size was 16. The image size was the same as the classification model, that is 224×224. The model weights with best accuracy on validation set was saved as the final weights. Assume that a test is a self-dataset test if the training dataset and the test dataset are the same, otherwise it is a cross-dataset test.

A. Quantitative Analysis

The average self-dataset test F1 score of three datasets using two different backbones is shown in Figure 4.17. The first backbone is DE-Net pretrained on ImageNet and then crack classification dataset. The other one is trained by the proposed HKT strategy. It can be seen that ImageNet pretrained backbone provides a bit better performance for the first three models instead of LR-ASPP. The difference between the two backbones is within 1%. The average F1 score of U-Net is the highest when using the HKT strategy. While the DEU-Net has highest F1 score when pretrained on ImageNet. The performance of U-Net and DEU-Net is very close with less than 0.5% difference. This verified the parameter efficiency of DE block. The DeepLabv3+ is about 10% less than U-Net models with 62.33% average F1 score. While, the LR-ASPP has only 46.61% F1 score with backbone pretrained by HKT strategy. This can be explained by the fact that the output resolution of DeepLabv3+ and LR-ASPP before final upsampling is 1/4 of the original image, which is a drawback when dealing with small objects such as thin cracks.

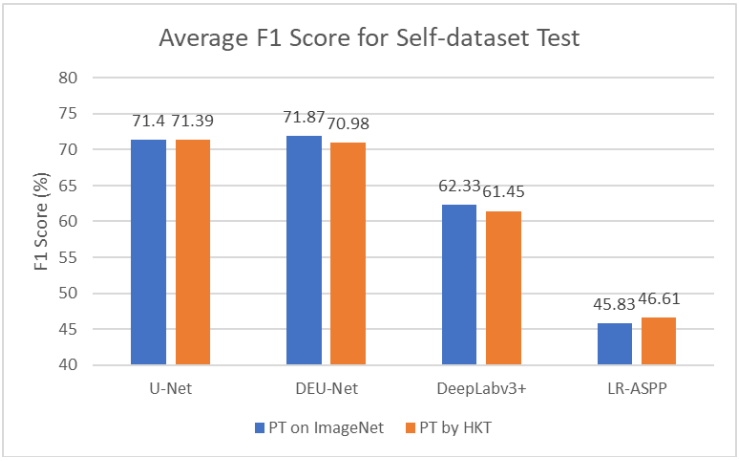


Figure 4.17. Average F1 score for self-dataset test of different segmentation heads and backbones.

The average cross-dataset test F1 score is shown in Figure 4.18. Compared with the self-dataset test, the average F1 score dropped significantly by about 14%. This is reasonable due to the domain difference among datasets. The DEU-Net has higher F1 score than U-Net by more than 1%, showing higher generalization ability of model architecture. The DeepLabv3+ is also 10% less than the U-Net and DEU-Net. The LR-ASPP still has the lowest F1 score of only 36.95%. The difference between the two backbones becomes more obvious. The ImageNet pretraining provides about 3% F1 score improve than the proposed HKT. This validated again that the model representation ability of ImageNet pretraining is higher than HKT. However, compared with the training time, this performance drop is still acceptable.

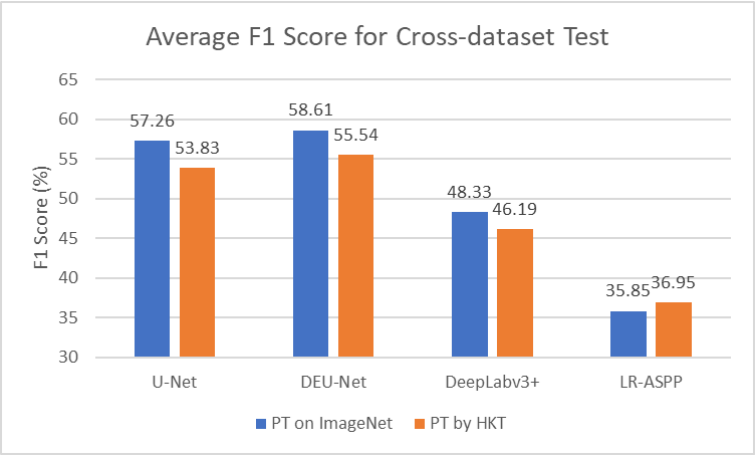


Figure 4.18. Average F1 score for cross-dataset test of different segmentation heads and backbones.

Table 4.8. Segmentation results of different segmentation heads and datasets

F1 score (%)	Train / Test	Mendeley	CrackForest	DeepCrack
U-Net	Mendeley	76.02	63.49	71.37
	CrackForest	27.27	61.99	41.86
	DeepCrack	65.40	53.56	76.16
DEU-Net	Mendeley	75.88	63.89	72.15
	CrackForest	28.87	63.28	47.91
	DeepCrack	64.61	55.81	73.77
DeepLabv3+	Mendeley	60.12	49.41	57.40
	CrackForest	28.71	50.58	32.66
	DeepCrack	58.18	50.78	73.63
LR-ASPP	Mendeley	41.95	35.40	39.74
	CrackForest	13.45	31.86	19.15
	DeepCrack	56.93	56.99	66.03

The model performance evaluated on the test set from different datasets using backbone trained by the proposed HKT strategy is listed in Table 4.8. It is interesting that the model trained on one dataset not always has the best performance for self-dataset tests, except DeepCrack dataset. This could be due to the small size of the datasets. For example, both U-Net and DEU-Net obtained best F1 score on Mendeley dataset for training datasets of Mendeley and CrackForest. The DeepLabv3+ obtained best performance on DeepCrack for training datasets of CrackForest and DeepCrack. The performance differences between self-dataset test and best cross-dataset test are within 1.5%. However, the LR-ASPP model performed best on DeepCrack dataset no matter what dataset it was trained on. The performance difference is from 15% to 26%. This is because the LR-ASPP architecture is more suitable for cracks in DeepCrack dataset that all very wide. This can be verified in Figure 4.20.

It is also important to investigate the influence of dataset on the generalization ability during cross-dataset test. Since the U-Net and DEU-Net have similar performance and better than the other two, the self- and cross-dataset test results of U-Net are compared in Figure 4.19. The models trained on Mendeley has the worst performance when segment the cracks in

CrackForest dataset with only 27.27% F1 score. While the test result on DeepCrack is relatively high with 65.5% F1 score. The F1 scores tested on Mendeley dataset are all very high from 63.49% to 76.02%, revealing that this dataset is an easy to be predicted dataset. The segmentation performance of CrackForest of model trained on DeepCrack is also very poor with only 41.86%. The F1 score of CrackForest is also the lowest among the self-dataset tests, indicating the CrackForest as a tough dataset. However, the cross-dataset test results of model trained on CrackForest has the highest average F1 score above 60%. Therefore, the CrackForest is the best dataset that provides the most generalization ability to model.

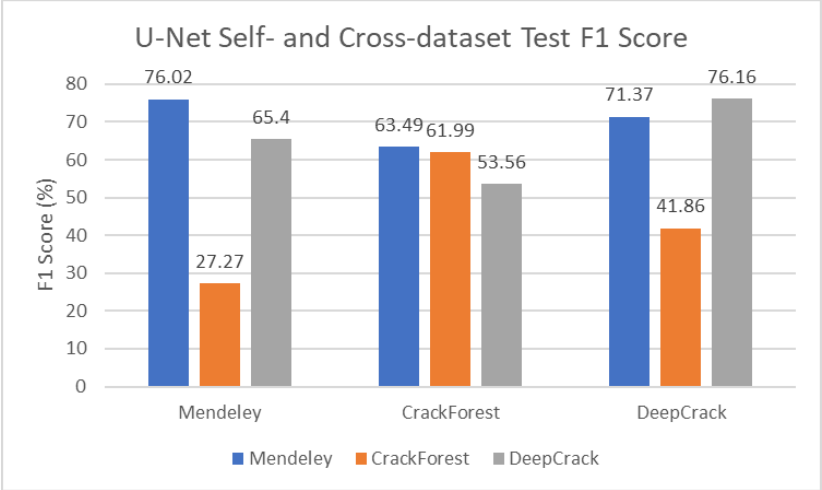


Figure 4.19. The U-Net self- and cross-dataset test F1 score comparison.

B. Qualitative Analysis

Some example predictions of the four architectures tested on three datasets are shown in Figure 4.20. The cracks in Mendeley and CrackForest are thin and long, while the ones in DeepCrack are much wider. For the thin cracks, both U-Net and DEU-Net has superior performance than the other two. The predicted crack masks are clear and accurate. The performance of DeepLabv3+ is ok but with less details due to that its prediction mask is 1/4 resolution of the original images before the final upsampling. The LR-ASPP performs poorly for thin cracks. It also outputs 1/4 resolution mask but with more noises and breaks in the mask. For test image from DeepCrack, all the four models perform well with higher than 93% F1 score. Because the Mendeley and CrackForest datasets have similar crack shape, it is reasonable that the results from the self-dataset test and the cross-dataset test are close. To objectively test the generalization ability influenced by the backbone and train dataset, a crack image was downloaded from internet, as shown in Figure 4.21. The U-Net architecture was chosen to segment this image with different backbone weights and train dataset. It is obvious that model trained on CrackForest has the best segmentation results with clear and complete crack mask.

The crack recognised by the other two are not consistent and has many false positive noises. The difference between the ImageNet pre-trained backbone and the HKT trained backbone is trivial. For models trained on CrackForest, the one with HKT trained backbone is less sensitive to non-crack pixels therefore has less false positive, such as the dents and falling off .

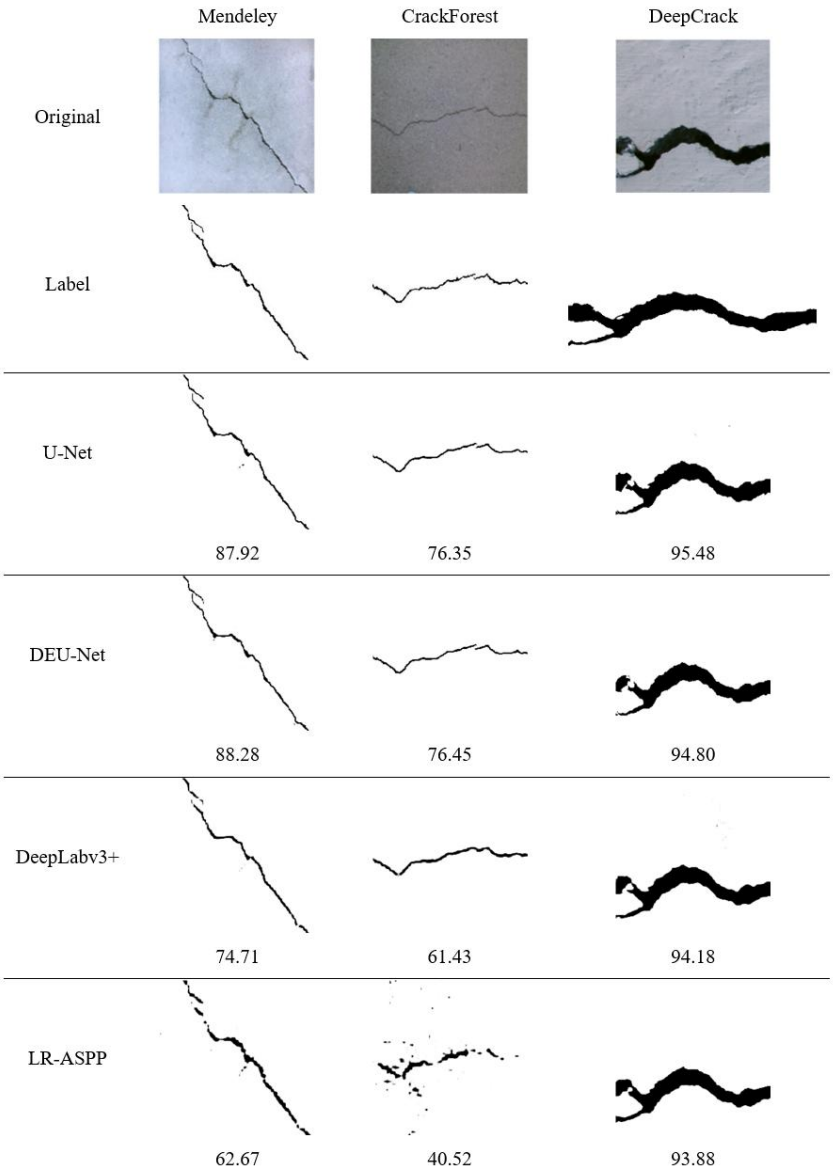


Figure 4.20. Example prediction results of four segmentation heads on different datasets, the dice score of segmentation is shown below the predicted masks.

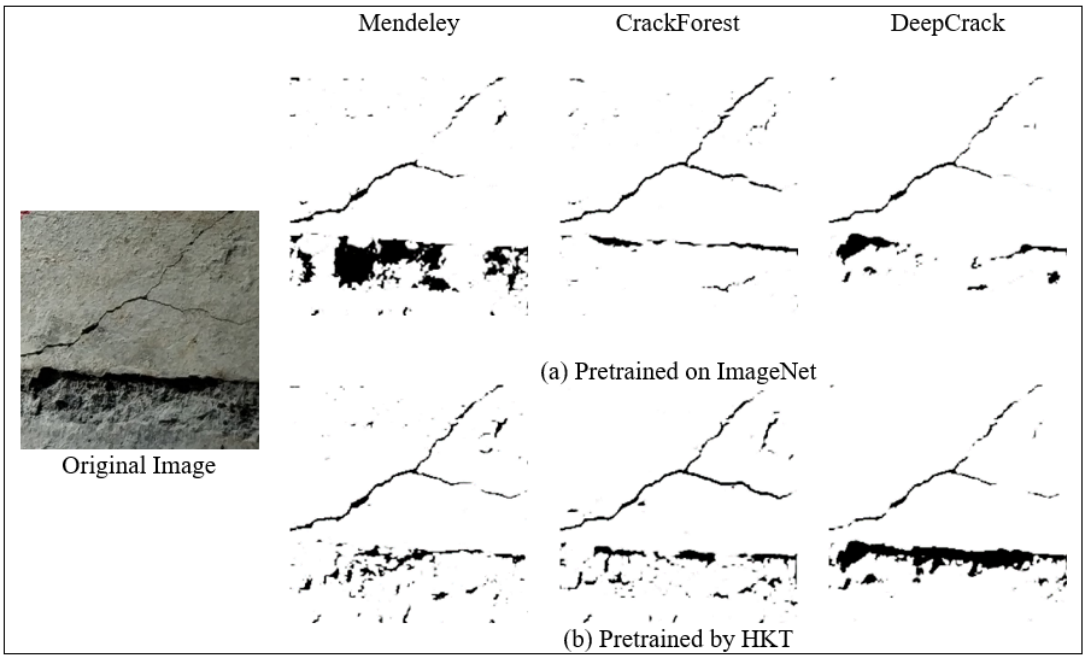


Figure 4.21. Example prediction results of U-Net trained on different datasets.

C. Efficiency Analysis

The efficiency of the four architectures is compared in Table 4.9. Even though the DEU-Net has less parameters than U-Net and DeepLabv3+, it has the longest inference time on all the devices. It is more obvious when tested on CPU and HoloLens 2 that the inference time is almost twice as the one of DeepLabv3+. This is due to the feature maps after concatenation and expansion has too many channels to process, making it not suitable for HoloLens 2 applications. Only U-Net and LR-ASPP can achieve more than 5 fps frame rate on HoloLens 2, with 24.3 and 21.1 inference time, respectively.

Table 4.9. Efficiency comparison of different segmentation heads. The accuracy of each model is the maximum average F1 score for self-dataset test

Model	Accuracy (%)	Parameters (k)	CPU (ms)	GPU (ms)	HoloLens 2 (ms)
U-Net	71.4	137.149	11.2	8.7	24.3
DEU-Net	71.87	129.477	54.6	9.4	101.4
DeepLabv3+	62.33	177.213	28.1	8.6	63.7
LR-ASPP	46.61	65.749	8.8	7.9	21.1

Overall, the U-Net performed best among the four segmentation architectures with good balance between inference speed and accuracy. Even though the DEU-Net has better parameter efficiency, its inference speed is much slower on CPU and HoloLens 2. Therefore, for AR application development, the U-Net architecture with DE-Net as backbone and trained on CrackForest dataset will be used for crack segmentation model.

4.6. System Validation and Verification

For integration of AI models into Unity, they were converted into ONNX format after training, including the DE-Net classification model and segmentation model with U-Net head architecture. The opset version 9 was used for compatibility. The Barracuda package was used to load and execute the converted model. Once the GUI design is finished in Unity, it can be compiled as VS Studio solution and deployed on to HoloLens 2 as a Universal Windows Program (UWP).

After training of the crack classification and segmentation models, the AR application can be developed and deployed on HoloLens 2 with these AI models integrated. The GUI of the AR application will be created in Unity using C#. The AI models will be incorporated via the Barracuda package. For the experiments, initial testing will be conducted with the RGB and depth cameras on HoloLens 2, followed by the evaluation of the AI models.

In lab environment, a crack image from Internet was projected on the wall for function test. The classification and segmentation results after pressing the 'Classification' and 'Segmentation' button are shown in Figure 4.22. Compared withFigure 4.21, since the classification results of the bottom three grids were negative, these image patches were not processed by the segmentation model, providing a better illustration of crack mask.

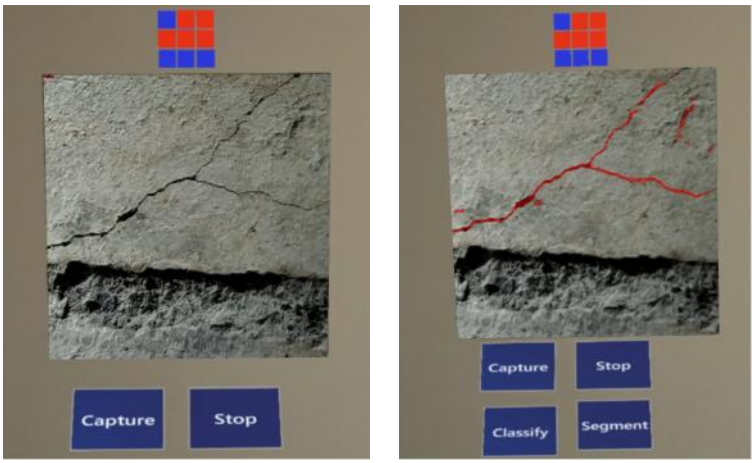


Figure 4.22. The classification (left) and segmentation (right) results for GUI test.

Based on the crack mask and point cloud, the crack can be measured regarding to its width and length. The crack mask is firstly connected using morphological close operation, followed by the skeletonization, as shown in Figure 4.23. In the skeletonized image, there are 6 connected components, with the longest two of size 498 and 61. The average crack width can be calculated by dividing the mask area by the total skeleton length. Therefore, the length and width of the longest crack is 498 and 5.58 pixels, respectively.

Figure 4.23. Postprocess of crack mask (left) morphological close and (right) skeletonization.

To measure the crack in real-world unit, the wall in point cloud was fitted as a 3D plane, represented by equation:

$$0.2330x - 0.0021y + 0.9725c + 2.4575 = 0 \quad (4\text{-}15)$$

The parameter of y-axis is almost 0, which means in the camera coordination, this plane is perpendicular to the XZ plane. The pixels in crack are firstly transformed from RGB image to RGB-D image, and then projected onto this plane. The crack length and width in RGB-D image are 154.8 pixels and 1.73 pixels. After mapping to point cloud, the crack length and width in real-world unit are 1.449 m and 16.3 mm, respectively.

The inference time is calculated from 10 repetitions of this crack measurement. The crack classification for the 9 image patches is 43.2 ms. The feature embeddings from the classification model are saved and used for crack segmentation. The segmentation time for the 5 image patches with detected crack is 71.5 ms. The final measurement time including the skeletonization and plane fitting is 65.4 ms. The total time is 180.1 ms (5.5 fps). Overall, all the requirements from the task clarification have been fulfilled.

4.7. Summary

In this chapter, a head mounted vision-based crack inspection system was developed by combining AR with AI technologies. A novel training strategy that combines the knowledge distillation and transfer learning was proposed to facilitate the model training. A novel model architecture was designed for real-time model inference on AR device. The whole program was implemented in Unity and deployed on Microsoft HoloLens 2. Following the proposed design methodology, the final developed system achieved a high accuracy real-time on-device crack inspection. The proposed design methodology was validated in this case study with an emphasis on the algorithm selection and optimisation.

The methodology ensured that the system design was tailored to the specific requirements of crack detection tasks in concrete walls. The early stages of context knowledge acquisition and

system specification played a key role in defining the hardware platform (Microsoft HoloLens 2), the image acquisition method (RGB and depth camera combination), and the software framework (Unity). These selections aligned with the need for real-time, portable, and interactive inspection applications.

During the concept and embodiment design stages, the methodology guided the modular design of the software system, including separate models for crack classification and segmentation. The proposed Hierarchical Knowledge Transfer (HKT) strategy and the development of DE-Net were direct outcomes of the methodology's emphasis on selecting and adapting machine learning models based on the task requirements and dataset availability.

5. The Second Case Study: Object-Level Wheat Head Disease Detection

This chapter reports the second case study of precision agriculture application targeting wheat head disease detection. This case study focuses on the algorithm selection for wheat head disease detection. The classification-based and anomaly detection methods are combined and applied for object-level wheat head disease detection. Addressing the challenge of limited data availability for wheat head disease images, an algorithm pipeline is proposed that leverages zero-shot learning and domain adaptation techniques. This case study is conducted on two open-source datasets to verify the effectiveness of the proposed pipeline as well as the feasibility of anomaly detection for wheat head disease detection. This case study aligns with the proposed design methodology by demonstrating its applicability in scenarios where data scarcity poses a significant challenge.

5.1. Introduction

In the pursuit of sustainable agriculture, precision agriculture has emerged as a significant paradigm shift, revolutionising traditional farming practices. A key aspect of this transformation is integrating state-of-the-art technologies into the agricultural sector, particularly focusing on precision spray robotic systems [290]. These advanced systems show promise in addressing the complex challenges faced by modern agriculture, including the requirement for higher yields, decreased resource waste, environmentally sustainable farming practices, and the critical issue of crop disease management. Wheat is one of the most important staple foods that provides a significant portion of the world's dietary calories. Fusarium Head Blight (FHB), (also known as wheat scab or ear blight, is one of the most devastating fungal diseases that can reduce yield by as much as 80% [291, 292].

To detect the diseased wheat head, some researchers have focused on using colour image for FHB detection and its feasibility has been widely validated [293-295]. Typically, the disease detection process was divided into two stages. As wheat heads are usually clustered in the field images, the first stage is to extract wheat heads from the images by segmentation [296] or bounding box detection [297]. The segmentation method usually requires more labelling effort and inference time than the bounding box method. After the extraction of wheat head, the disease can be classified or segmented based on its colour and texture information. For segmentation method, usually the binarization method is used based on the colour difference

of healthy part and diseased part [298]. The disease severity can be evaluated based on the extracted disease part. However, this approach requires either the labour-intensive task of hyperparameter tuning for different lighting conditions or the availability of carefully labelled disease segmentation data, both of which can be time consuming and prone to overfitting.

To address the lack of diseased wheat head images, this study developed an anomaly detection-based wheat head disease detection system for precise spray system. The requirements and specifications for the detection are derived from the task clarification. In the concept design stage, the concept of the system architecture is generated. As for the embodiment design, the development of the disease detection method is described. To address the issue of insufficient wheat head disease data, a novel detection algorithm pipeline is proposed that utilises only healthy wheat head data for model training. This algorithm pipeline combines classification-based detection and anomaly detection method, targeting wheat head bounding box detection and anomaly score prediction, respectively. To validate the proposed method, a comprehensive study of various anomaly detection methods is conducted on two wheat head disease datasets, demonstrating high disease classification accuracy, as well as the potential for applying anomaly detection methods to severity estimation. The entire detection pipeline is tested in real-time on a Jetson Nano, achieving a processing time of 327.9 ms per image, including both wheat head detection and disease classification.

5.2. Task Clarification

5.2.1. System Requirements and Specifications

- The detection method should be able to locate each diseased wheat head in the image as bounding boxes.
- Only the healthy wheat head images can be used for model training.
- The camera view angle for each dataset is fixed.
- The accuracy of the disease detection should be larger than 80%.
- Given the rover speed of 1.5 m/s and the distance between the camera and the spray unit of 1.8 m, the inference time for the whole processing pipeline should be within 1.2 second.

5.2.2. Related Works

For modern learning-based methods, the number of images in a dataset is a critical factor in determining model prediction accuracy [299]. This underscores the importance of cutting-edge technologies, like transfer learning [300] and domain adaptation [301], which enable the system to learn and adapt from diverse datasets, leading to a robust and reliable disease

detection mechanism. Currently, wheat datasets used in published research are often collected using dedicated instruments and are not publicly available. The Global Wheat Head Detection 2021 (GWHD) [302], which contains 193,634 labelled wheat heads from 7 countries/institutions such as UK, France, etc. This dataset has large diversity of geographic locations and wheat stages like filling, ripening, and post-flowering. However, it only has labels for wheat head and no information of any diseases. There are only a few public datasets focused on wheat head disease detection, including the Wheat Head Blight Dataset (WHBD) [303] and FHB with Severity (FHBS) dataset [304]. However, these datasets only have image-level annotations.

To address the lack of diseased wheat head images, anomaly detection methods can be employed. Anomaly detection is a recent trend in computer vision that uses only normal images for model training to identify anomalies. There are many existing anomaly detection methods, categorised as classification, probabilistic, reconstruction, and distance based method for one-class classification tasks [305]. Typical methods for these four categories include the Multi-variate Gaussian (MVG) used in PaDiM [222], K-Nearest Neighbor (KNN) used in PatchCore [223], Generative Adversarial Network (GAN) in AnoGAN [306], Support Vector Data Description (SVDD) in Patch SVDD [213], etc. This technique has been applied in some plant disease detection scenarios. For example, Jin et al. applied the generative adversarial network (GAN) to detect varies disease on grape leaves, such as black measles, black rot, and leaf blight [307]. Similarly, Bhugra used the GAN paired with the edge map to segment the disease area on the leaf images [308]. Other studies also focused on the leaf disease detection with reconstruction-based model [309, 310]. Detecting wheat head disease is more challenging than detecting diseases on leaves due to the more complex textures and clustered background of wheat heads. This complexity makes it difficult to apply methods such as GANs to wheat head images effectively. Therefore, a thorough study of different anomaly detection methods is necessary to address these challenges.

5.3. Concept Design

In order to develop a disease detection algorithm with high generalization ability and validate the robustness to various wheat head images, the GWHD dataset was used as training input, while the WHBD and FHBS datasets were used as validation data. Several technologies have been used during model training, including transfer learning, domain adaptation, anomaly detection, and data augmentation. This section aims to verify the feasibility of disease detection using deep learning with extremely unbalanced dataset that has no disease data at all.

The procedure of training detection model and using this model for disease detection is shown in Figure 5.1. Firstly, the base model YOLOv8n is fine-tuned on GWHD dataset using COCO-pretrained weights for wheat head bounding box detection task. This enables the base model to have perception of wheat head. Then the three feature layers P3, P4, and P5, are used to generate feature embeddings for wheat head. These feature embeddings are fitted to Multi-variate Gaussian Model (MGM). For disease prediction, the wheat head bounding box is firstly generated and used for feature embedding extraction. Then the distance between the feature embedding and the fitted GMM is used as anomaly score. The implementation details will be described in two sections since the first part which trains a bounding box detection model follows the typical detection task paradigm, while the second part that uses the feature embedding for disease detection falls into anomaly detection realm.

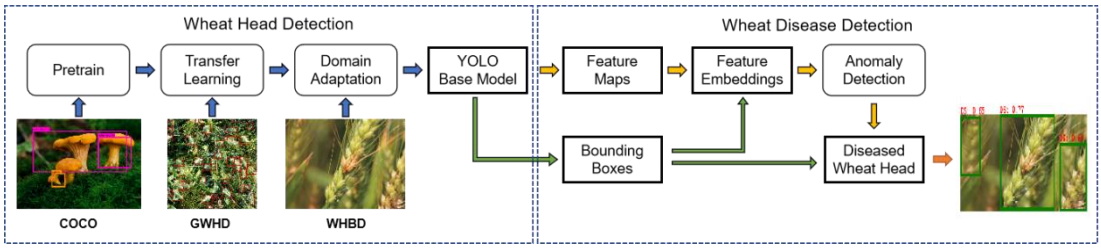


Figure 5.1. Disease detection model training and prediction

5.4. Wheat Head Detection

The wheat head detection is based on the state-of-the-art YOLOv8 model, which is a single-stage bounding box detection deep learning model that famous for excellent balance between speed and accuracy. The transfer learning and domain adaptation are used to enable the model to detect the wheat head bounding box in WBHD and FHBS using GWHD dataset as intermediate domain. The data augmentation such as random flip and mosaic is used during transfer learning on GWHD dataset.

5.4.1. YOLOv8 Model

This latest version of YOLO utilises both Feature Pyramid Network and Path Aggregation Network, which introduces numerous improvements compared to its predecessors [42]. It has three hyper-parameters to adjust the size and performance of the model, i.e., depth_multiple, width_multiple, and ratio, as listed in Table 5.1. The C2f block uses dense connections inside with varied number of Bottleneck blocks controlled by depth_multiple. The U block is upsampling block. The C block is concatenation of feature maps from different layers. The number of output channel for each convolution block (Conv) is adjusted by width_multiple and ratio. The smallest model architecture named YOLOv8n is used for wheat head detection

which has only 3.2 M parameters but achieved 37.3 map on COCO val2017 dataset.

Table 5.1. Comparison of different YOLOv8 models

Model	depth_multiple	width_multiple	ratio	params (M)	mAP50-90
YOLOv8n	0.33	0.25	2.0	3.2	0.373
YOLOv8s	0.33	0.50	2.0	11.2	0.449
YOLOv8m	0.67	0.75	1.5	25.9	0.502
YOLOv8l	1.00	1.00	1.0	43.7	0.529
YOLOv8x	1.00	1.25	1.0	68.2	0.539

For training on GWHD dataset, the loss function and weight update procedure are the same as the original YOLO model, defined as follows, from [42]:

$$L(\theta) = \frac{\lambda_{box}}{N_{pos}} L_{box}(\theta) + \frac{\lambda_{cls}}{N_{pos}} L_{cls}(\theta) + \frac{\lambda_{dfl}}{N_{pos}} L_{dfl}(\theta) + \phi\|\theta\|_2^2 \tag{5-1}$$

where θ is the parameters of the model, N_{pos} is the total number of cells containing and object, L_{box} is the bounding box loss, L_{cls} is the classification loss, L_{dfl} is the distribution focal loss. The λ_{box}, λ_{cls}, λ_{dfl} are the weighting factors of the three loss components, set as 7.5, 0.5, 1.5, respectively. The specific loss function can be defined as:

$$L_{box}(\theta) = \sum_{x,y} I_{c^*_{x,y}} \left[1 - q_{x,y} + \frac{\left\| b_{x,y} - \hat{b}_{x,y} \right\|_2^2}{\rho^2} + \alpha_{x,y} v_{x,y} \right] \tag{5-2}$$

$$L_{cls}(\theta) = \sum_{x,y} \sum_{c \in classes} y_c \log(\hat{y}_c) + (1 - y_c) \log(1 - \hat{y}_c) \tag{5-3}$$

$$L_{dfl}(\theta) = \sum_{x,y} I_{c^*_{x,y}} \left[-\left(q_{(x,y)+1} - q_{x,y}\right) \log\left(\hat{q}_{x,y}\right) + \left(q_{x,y} - q_{(x,y)+1}\right) \log\left(\hat{q}_{(x,y_{+1})}\right) \right] \tag{5-4}$$

where

$$q_{x,y} = IoU_{x,y} = \frac{\hat{\beta}_{x,y} \cap \beta_{x,y}}{\hat{\beta}_{x,y} \cup \beta_{x,y}}$$

$$v_{x,y} = \frac{4}{\pi^2} \left(\arctan\left(\frac{\omega_{x,y}}{h_{x,y}}\right) - \arctan\left(\frac{\hat{\omega}_{x,y}}{\hat{h}_{x,y}}\right) \right)^2 \tag{5-5}$$

$$\alpha_{x,y} = \frac{v}{1 - q_{x,y}}$$

where N_{pos} is the total number of cells containing and object, $I_{c^*_{x,y}}$ is indicator function for the cells containing an object, $\beta_{x,y}$ is the ground truth bounding box in format of (x_center, y_center, width, height), $\hat{\beta}_{x,y}$ is the predicted box, $b_{x,y}$ is the central point of the ground truth

bounding box, y_c is the ground truth class label, $q_{(x,y)+/-1}$ is the nearest predicted box IoU of left and right, ρ is the diagonal length of the smallest enclosing box covering the predicted and ground truth boxes.

5.4.2. Domain Adaptation and Transfer Learning

Domain adaptation and transfer learning are pivotal concepts in machine learning that enable models to leverage knowledge gained from one domain (source domain) to improve performance in a different but related domain (target domain) [43]. In domain adaptation, the objective is to minimise the distributional gap between the source and target domains, with the aim of making the model robust to changes in data distribution. This is typically achieved through methods such as adversarial training, where a domain discriminator is trained alongside the main model to encourage domain-invariant representations. Transfer learning, on the other hand, involves using pre-trained models on a source task to initialise a model for a target task. This process finetunes the target model with features learnt from the source task, thereby accelerating convergence and often leading to better generalisation on the target task, especially when data are scarce.

To train a model that is able to detect the wheat head disease in images from disease dataset, for example, the WHBD (target domain) without using its data at all, the domain adaptation method can be used via finetuning the model, that is pre-trained with COCO dataset (source domain), on GWHD dataset (intermediate domain). This process can be illustrated in Figure 5.2. The domains of the three datasets are simplified as Gaussian distributions, where the GWJD and WHBD are close to each other, while COCO dataset is far from these two. The aim of the training process is to pass knowledge from the COCO dataset domain to the WHBD domain through the GWHD domain. The purpose of using COCO as the the source domain and GWHD as intermediate domain, instead of using GWHD directly as that source domain, is the the COCO dataset has a lot more data, which usually means more model representative ability. This will be verified in the experiments section.

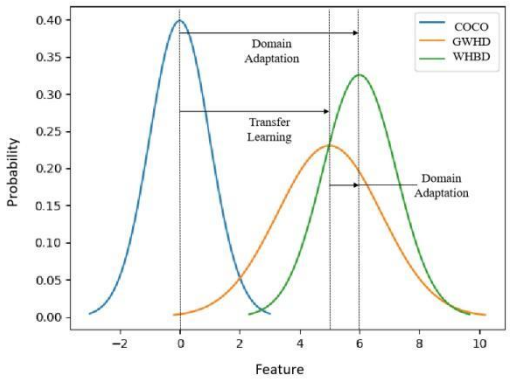


Figure 5.2. Illustration of domain adaptation and transfer learning on three datasets.

Mathematically, assume that the three datasets COCO, GWHD, WHBD are noted as D_C, D_G, and D_W, and defined as:

$$D_C = \{(x_{Ci}, y_{Ci}) \mid i\epsilon[1, N_C]\}$$
$$D_G = \{(x_{Gi}, y_{Gi}) \mid i\epsilon[1, N_G]\} \quad (5\text{-}6)$$
$$D_W = \{(x_{Wi}, y_{Wi}) \mid i\epsilon[1, N_W]\}$$

where (x_i, y_i) is image-label pair and N is the total number of images in the datasets. The bounding box detection task is defined as an optimisation problem of model parameters θ:

$$\hat{\theta} = \arg\min_{\theta} L(\theta|D) = \arg\min_{\theta} L(f_\theta|D) \quad (5\text{-}7)$$

where f_θ is the neural network model that maps the input data to bounding box. The domain adaptation by transfer learning can then be formalized as:

$$P(y_{Wi}|x_{Wi}) = f_{\theta_W}(x_{Wi}) = f_{\theta_G}(x_{Wi})$$
$$\theta_G = \arg\min_{\theta_G^0=\theta_C} L(\theta|D_G) \quad (5\text{-}8)$$
$$\theta_C = \arg\min_{\theta} L(\theta|D_C)$$

The prediction of wheat head bounding box for WHBD is achieved by using the model weights finetuned on dataset GWHD.

5.4.3. Experiments and Results

To verify the effectiveness of the proposed disease detection algorithm, the YOLOv8n model was trained on a desktop PC and its performance was evaluated in terms of wheat head detection and head disease detection.

5.4.3.1. GWHD

Regarding the training in the GWHD dataset, both YOLOv8n and YOLOv8s were trained with or without using pre-trained weights in the COCO dataset. The image size was 640x640. The hyperparameters follow exactly the original YOLO paper. The learning rate was set to 1e-3 and 1e-2 for training with and without pre-trained weights. The batch size was set to 16. The SGD optimiser was used with a momentum of 0.937 and a weight decay of 0.0005. In total, 100 epochs were used, and the weights with best validation performance were saved. Three warm-up epochs were used with warm up momentum of 0.8 and initial learning rate bias of 0.1. The mosaic augmentation was turned on and stopped 10 epochs before the final epoch. Other image augmentation includes scaling, translation, flip, contrast, etc.

The performance comparison of the four models trained on the GWHD dataset is listed in Table 5.2. The performance of YOLOv8n is good enough even when trained from scratch. All criteria significantly increased if they used pretrained weights from COCO for both YOLOv8n and YOLOv8s. With more than third times of the parameters, the YOLOv8s performed better than YOLOv8n for both training from scratch or pre-trained weights. However, the improvement is not significant, for example, only about 0.01 for precision and recall.

Table 5.2. Performance comparison of models trained on the GWHD dataset.

Model	Precision	Recall	mAP50	mAP50-90
YOLOv8n-scratch	0.905	0.811	0.892	0.492
YOLOv8n-pretrain	0.915	0.852	0.92	0.515
YOLOv8s-scratch	0.916	0.851	0.918	0.508
YOLOv8s-pretrain	0.926	0.86	0.931	0.531

Figure 5.3. Example wheat head detection results with GWHD and YOLOv8n-pretrain

Two examples of wheat head detection using YOLOv8n using pre-trained weights are shown in Figure 5.3. It is shown that almost all the wheat heads have been detected with the probability threshold set as 0.3. The smaller and more obscure the wheat head is, the less

probability is predicted by the model. It performed well even when multiple wheat heads are clustered together in the bottom right of this image.

5.4.3.2. WHBD

The trained models were then evaluated on the WHBD regarding the accuracy of the detection of the wheat head bounding box, as listed in Table 5.3. Unlike the GWHD dataset, where each image contains many wheat heads, in WHBD, basically only one wheat head appears in one image. The model was trained to recognise multiple small wheat heads in one image, making it almost impossible to detect large wheat heads in WHBD. Therefore, a small trick was used here that multiple images in WHBD were resized and assembled as a single image before being fed to the model. The resize ratio was determined based on the bounding box width/height ratio regarding the image size in GWHD, which is 0.075 and 0.071. For WHBD, it is 0.16 and 0.42, about 2 and 6 times of GWHD. The final assembled image from WHBD contains 6 rows and 2 columns of original images and resized to 640x640. The ground truth bounding boxes were also resized and translated according to the new image size. The performance comparison of the wheat head detection in WHBD dataset is listed in Table 5.3.

Table 5.3. Performance comparison of WHBD models for wheat head detection

Model	Precision	Recall	mAP50	mAP50-90
YOLOv8n-scratch	0.640	0.545	0.597	0.298
YOLOv8n-pretrain	0.748	0.728	0.781	0.375
YOLOv8s-scratch	0.672	0.570	0.600	0.324
YOLOv8s-pretrain	0.753	0.750	0.775	0.383

Even though the model has never seen any images from WHBD, it can achieve over 0.72 precision and recall through transfer learning. It is obvious that the pre-trained weights on COCO significantly improved the performance of YOLO models on dataset WHBD, for example, about 0.108 precision, 0.183 recall, 0.1 184 mAP50, and 0.077 mAP50-90 increase for YOLOv8n. This verified the method that uses transfer learning from COCO to GWHD and WHBD. An example of wheat head detection in WHBD using YOLOv8n is shown in Figure 5.4. As stated above, the image was resized before fed into the model. It can be seen that even the blurred wheat head on the left side can be recognised.

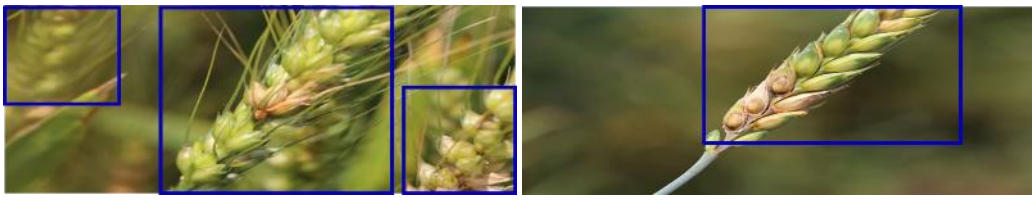


Figure 5.4. Examples of wheat head detection in WHBD using YOLOV8n-pretrain.

The WHBD also has images with clustered wheat head, as shown in Figure 5.5. It can be seen

that the detection results using the original image are very poor, with only one wheat head detected. When resizing the original image to half height, the results improved significantly with more than half the wheat heads correctly detected. This proved the importance of the object size in domain transfer.

Figure 5.5. Detection result of a sample image with clustered wheat heads in WHBD using YOLOV8n-pretrain with (left) original image size and (right) resized to half height.

5.4.3.3. FHBS

Since the wheat heads distribution in this dataset is also clustered, similar to the GWHD, no image preprocessing is applied except for the resizing to 640x640 resolution. Wheat head detection results on FHBS are listed in Table 5.4. The model configuration is the same as previously. The YOLO8n model has achieved over 0.86 and 0.82 precision and recall, which is close to the results on the GWHD dataset. The improvement through transfer learning is verified again, with a 0.071 precision, 0.065 recall, 0.128 mAP50, and 0.056 mAP50-90 increase.

Table 5.4. Performance comparison of YOLOv8 models on FHBS for wheat head detection.

Model	Precision	Recall	mAP50	mAP50-90
YOLOv8n-scratch	0.791	0.762	0.724	0.347
YOLOv8n-pretrain	0.862	0.827	0.852	0.403
YOLOv8s-scratch	0.875	0.828	0.811	0.412
YOLOv8s-pretrain	0.898	0.849	0.829	0.428

An example prediction result of this dataset is shown in Figure 5.6. It can be observed that most of the wheat heads have been detected. The prediction threshold is reduced to 0.2 due to the change of the dataset. Even with a lower threshold, there are few false positives but some false negatives, including heads overlapped with others or hidden behind leaves. The overall prediction quality is good enough for disease detection.

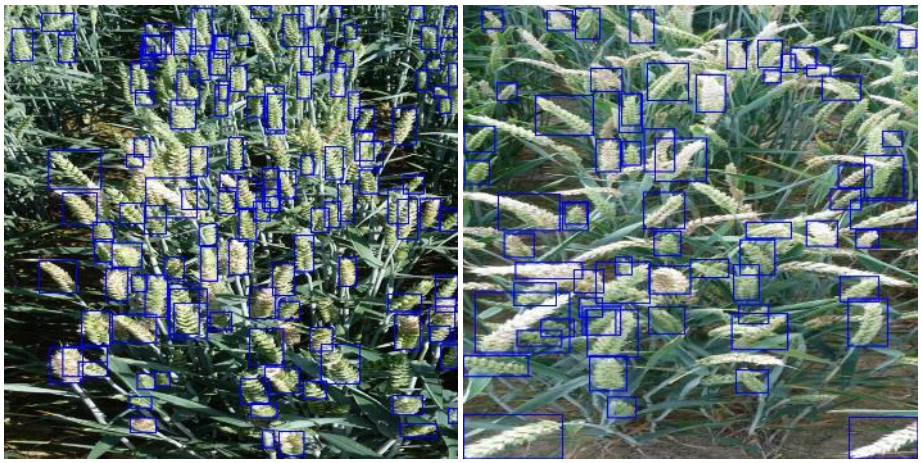


Figure 5.6.Example wheat head detection results with FHBS and YOLOv8n pretrain.

Regarding the model size, the YOLOv8s does have some improvement compared with YOLOv8n on WHBD, yet negligible (less than 0.03 on the four criterion). For Jetson Nano, whose computational resource is limited, the performance improvement of YOLOv8s does not justify its overhead. Therefore, for the development of the disease detection model, only YOLOv8n will be used and evaluated in the following section.

5.5. Wheat Head Disease Detection

In practice, it is usually difficult to obtain an adequate amount of high-quality and well-labelled disease images, which impedes the development of accurate and robust disease detection models. To address this critical limitation, an anomaly detection-based approach is proposed that uses the large number of images without disease. Before anomaly detection, the model trained from the last step is used directly by extracting its intermediate feature maps to generate feature embeddings for wheat head. The feature embeddings of healthy wheat heads are used to fit an anomaly detection model. Four different anomaly detection methods are used and compared, including the Multi-Variate Gaussian (MVG), K-Nearest Neighbor (KNN), Support Vector Data Description (SVDD), and Principal Component Analysis (PCA) based classification.

5.5.1. Algorithm Candidates

5.5.1.1. MVG

The anomaly detection algorithm follows the idea of Padim [44] with some modifications, as illustrated in Figure 5.7. The three layers of intermediate feature maps are used to extract feature embeddings. During training process, the images from GWHD dataset are used to generate the feature maps. Then the corresponding ground truth bounding boxes are resized

according to the size of each feature map. The tensors within each bounding boxes of each feature map are extracted, vectorised, and concatenated as the feature embeddings, $\boldsymbol{X} = \{\boldsymbol{x}^k, k \in [1, N]\}$ where N is the number of bounding boxes. Before the vectorisation and concatenation, the adaptive average pooling layer is used to adjust the length of the embeddings from each feature map. Finally, all the feature embeddings of wheat heads in GWHD are fitted to a multivariate Gaussian distribution $\mathcal{N}(\sum, \boldsymbol{\mu})$ where μ is the sample mean and the sample covariance $\sum$ is estimated as follows:

$$\Sigma = \frac{1}{N-1}\sum_{k=1}^{N}(\boldsymbol{x}^k - \boldsymbol{\mu})(\boldsymbol{x}^k - \boldsymbol{\mu})^{\mathrm{T}} + \epsilon \boldsymbol{I} \tag{5-9}$$

where the final regularisation term $\epsilon \boldsymbol{I}$ makes the sample covariance matrix full rank and invertible. By this way, each embedding contains information from different semantic levels and $\sum$ contains the inter-level correlations.

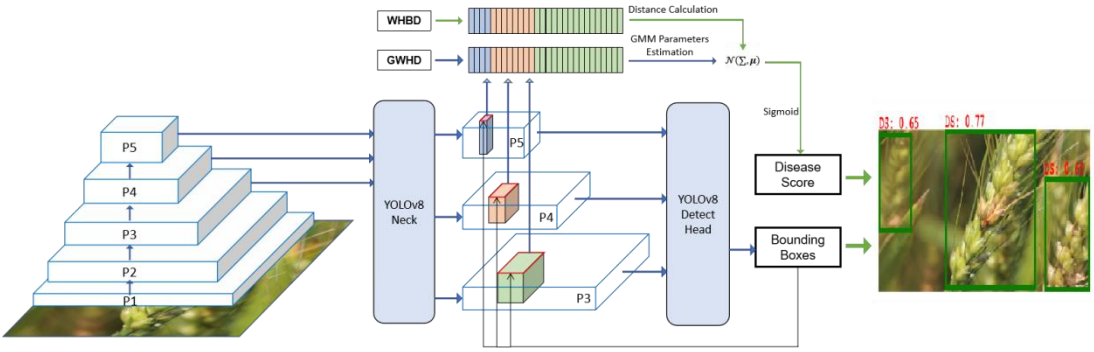


Figure 5.7. Illustration of disease detection by anomaly detection.

The Mahalanobis distance $M(\boldsymbol{x}^k)$ is used to calculate the anomaly score of the detected wheat head in WHBD, defined as:

$$M(\boldsymbol{x}^k) = \sqrt{(\boldsymbol{x}^k - \boldsymbol{\mu})^{\mathrm{T}} \sum^{-1} (\boldsymbol{x}^k - \boldsymbol{\mu})} \tag{5-10}$$

This can be interpreted as the distance between the test embedding and the learned distribution $\mathcal{N}(\sum, \boldsymbol{\mu})$. A higher distance means a high probability of anomaly. The sigmoid function is used to normalise the distance to probability from 0 to 1.

5.5.1.2. SVDD

The working flow of SVDD is similar to the MVG method, using the intermediate feature embeddings from the YOLO model. To fit an SVDD model, a neural network f_θ is trained to project the feature embeddings to another manifold, so that the feature embeddings are close

to each other with a sphere boundary. Formally, the objective of this method is:

$$\min_{\boldsymbol{\theta},\boldsymbol{c}} \frac{1}{N}\sum_{i=1}^{N}\left\|f_{\boldsymbol{\theta}}(\boldsymbol{x}^i)-\boldsymbol{c}\right\|^2 \tag{5-11}$$

where c is the centre of all the projected embeddings. The anomaly score is calculated as:

$$score = \|f_{\boldsymbol{\theta}}(\boldsymbol{x})-\boldsymbol{c}\|^2 \tag{5-12}$$

In this research, the neural network is a multi-layer perceptron with two fully connected layers. The output dimension is 32.

5.5.1.3. KNN

The KNN method does not have a model fitting process for one-class classification task. During inference, the anomaly score is calculated by the average distance between the tested embedding and its k-nearest neighbors of the normal samples:

$$score = \frac{1}{k}\sum_{i=1}^{k}\left\|\boldsymbol{x}-\boldsymbol{x}^i\right\|^2 \tag{5-13}$$

5.5.1.4. PCA

The PCA method seeks an orthogonal basis W that maximises the variance of the data $\mathcal{X} \in \mathbb{R}^D$ that D is the dimension of the feature embedding:

$$\max_{W} \textstyle\sum_{i=1}^{N}\left\|W\boldsymbol{x}^i\right\|^2 \quad \text{s.t. } WW^T = I \tag{5-14}$$

The $d \leq D$ components of $W \in \mathbb{R}^{d\times D}$ are the principal components that explain most of the data variance. When applying the PCA for classification, it is formulated as a reconstruction process using the orthogonal projection W^TW to a d-dimensional space. The anomaly score is therefore the reconstruction error:

$$score = \left\|\boldsymbol{x}^i - W^TW\boldsymbol{x}^i\right\|^2 \tag{5-15}$$

5.5.2. Experiments and Results

To evaluate the anomaly detection-based wheat head disease detection methods, the image patches of all the wheat heads are extracted from each dataset using the ground truth bounding box and then resized to 64x64 images. For the severity evaluation of FHBS, the anomaly score of each image is the average anomaly score of all the wheat heads in that image. Apart from the four feature embedding-based methods, the HSV colour thresholding is also evaluated.

5.5.2.1. WHBD

1) HSV Colour Thresholding

The images are firstly converted to HSV colour space. Then the average Hue value for each image as the colour feature. The normal and abnormal samples are then classified simply by thresholding the Hue values. This method is simple yet feasible. As can be seen in Figure 5.8. Negative samples, that is, wheat heads without disease are green in colour, with hue values mostly in the range 35 to 45. In contrast, the positive samples, i.e., wheat heads with disease are in yellow colour, with hue values mostly in the range 25 to 35. Therefore, the wheat heads can be classified by a threshold of hue value. The precision, recall, optimal F1, and AUROC using this algorithm are listed in Figure 5.10. Distribution of anomaly scores on WHBD using embedding-based methods.

Table 5.5. Precision and recall are calculated using the optimal F1. It achieved a 0.736 optimal F1 score and 0.765 AUROC.

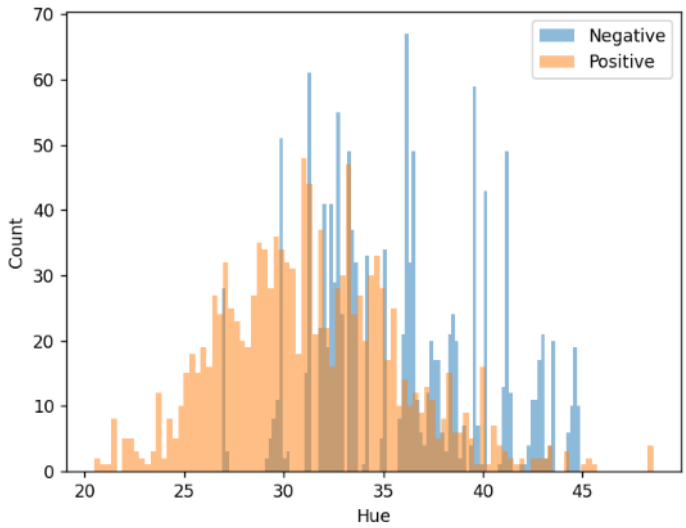


Figure 5.8. Histogram of Hue values for positive and negative samples in WHBD.

2) Feature Embedding-based Methods

For the following feature embedding based methods, the YOLOv8n model trained for wheat head detection is used directly to generate the intermediate feature maps. Since the disease data emphasis more on the colour instead of texture, the shallow layers provide more distinguishable features than deeper layers. Therefore, the first 7 layers of the model are used for feature extraction, with a total feature dimension of 464 for each wheat head image. The extracted feature embeddings are visualised in Figure 5.9, using the t-SNE method from scikit-learn library. It can be observed that the feature embeddings of positive samples are clustered

together. The negative samples are scattered in the feature space, similar to the HSV colour space.

For model fitting, 20% of the feature embeddings from negative samples are used for training. During the evaluation, other 80% of the negative feature embeddings and the same number of positive feature embeddings are used to calculate the anomaly scores using the fitted model. The anomaly score distributions of the four feature embedding-based methods are shown in Figure 5.10. The scores of negative and positive samples have different distributions using the four methods. It is obvious that the KNN method can classify the diseased feature embeddings better than all the others. It can be explained by the clustered distribution of the samples in the feature space. The MVG and SVDD have similar performance with partially overlapping between the positive and negative samples. The PCA method cannot classify the samples well that the scores of negative and positive samples have fewer difference.

Quantitatively, the optimal F1 and AUROC are calculated for these four methods, as listed in Table 5.5. The KNN method achieved the highest prediction performance with 0.963 optimal F1 and 0.992 AUROC. The MVG and SVDD have similar performance. The performance of the PCA method is even worse than that of the HSV thresholding method.

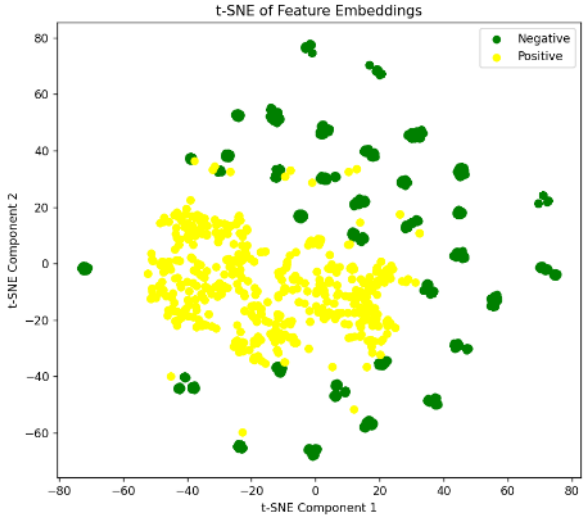
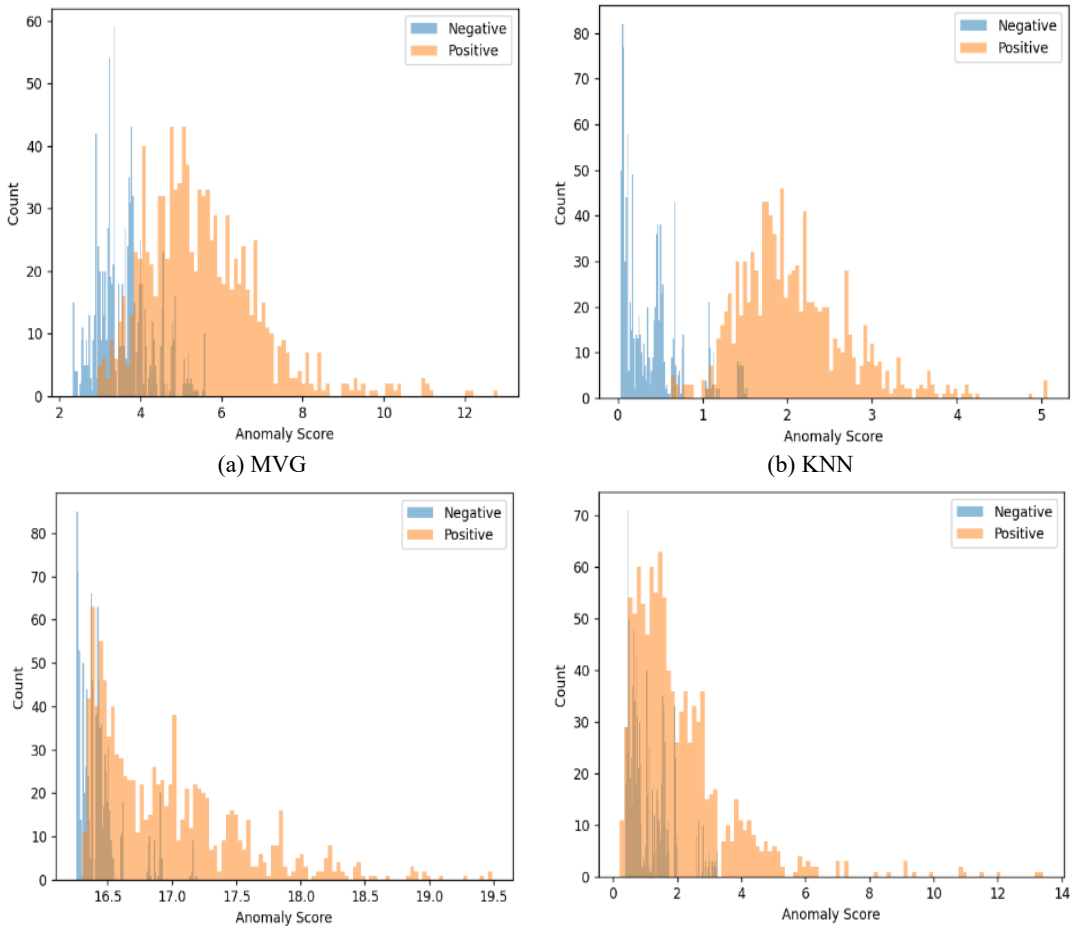


Figure 5.9. Feature embedding visualisation of WHBD. The feature embeddings are extracted from the first 7 layers of the YOLOv8n model.



(a) MVG (b) KNN

(c) SVDD (d) PCA

Figure 5.10. Distribution of anomaly scores on WHBD using embedding-based methods.

Table 5.5. Performance comparison of different disease detection algorithms on WHBD

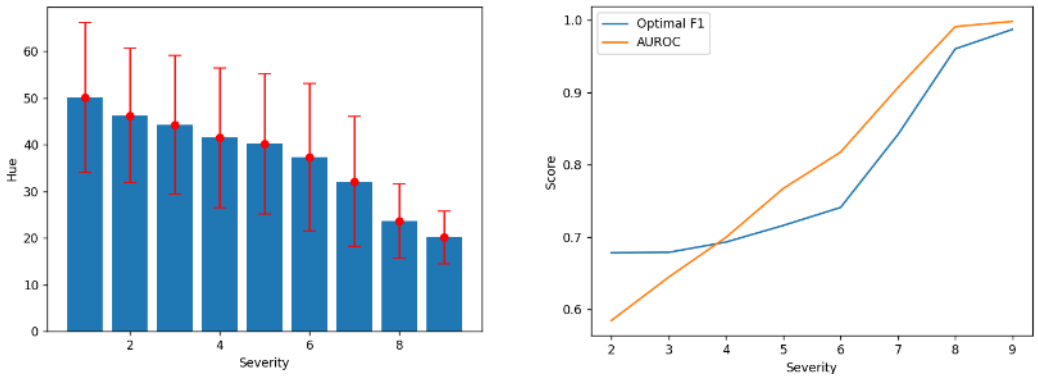
Model	Precision	Recall	Optimal F1	AUROC
HSV	0.643	0.858	0.736	0.765
MVG	0.806	0.892	0.847	0.914
KNN	0.956	0.970	0.963	0.992
SVDD	0.841	0.731	0.783	0.859
PCA	0.610	0.836	0.706	0.722

5.5.2.2. FHBS

1) HSV Colour Thresholding

Similarly, each wheat head image is converted to the HSV colour space, and the Hue value is averaged and compared in Figure 5.11 (a). This dataset has labelled the images regarding to the disease severity from 1 to 9. The images of severity 1 are negative samples and all the others are positive samples. For the one-class classification task in this research, the classification is done for each of the severity sets. The optimal F1 score and the AUROC for each severity class are plotted in Figure 5.11 (b). It is clear that with increasing severity, the better the prediction precision, especially from severity 7 to 9. The average scores for the criterion for

all the severities are listed in Figure 5.13.



(a) Hue value distribution (b) Classification accuracy

Figure 5.11. The classification of different severity of the disease in WHBD.

2) Feature Embedding-based Methods

For better visualisation, only the feature embeddings of severity 1, 4, 7, and 9 are plotted in Figure 5.12. The higher the more severity, the projected points in the feature space are distributed to the left. This verifies the feasibility of feature embedding-based methods for disease detection and severity estimation.

Similarly, 20% of the feature embeddings from the negative samples are used for model fitting and 80% for evaluation. The curve of optimal F1 scores and AUROCs using the four algorithms is shown in Figure 5.13. For this dataset, none of the feature embedding-based methods exceeds the simple HSV thresholding methods. However, for severity greater than 7, optimal F1 scores and AUROCs are all larger than 0.7.

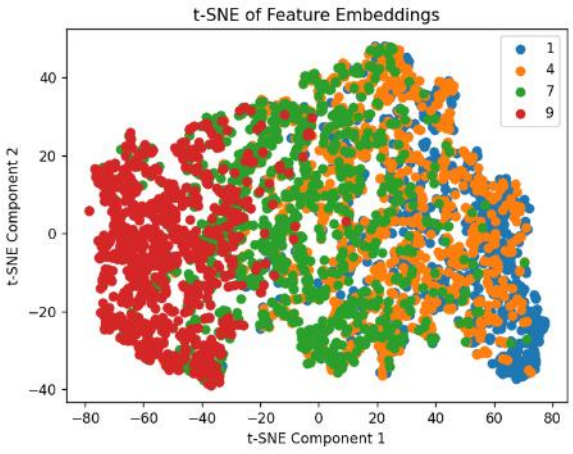
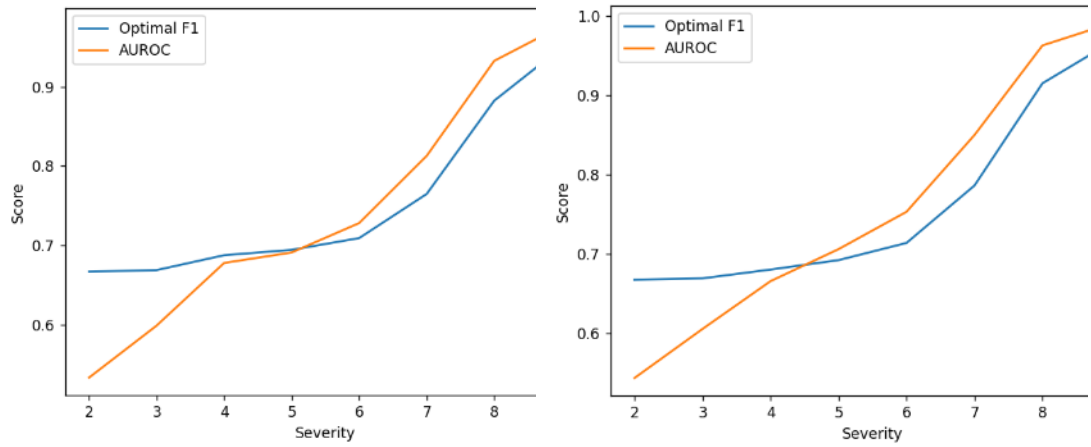
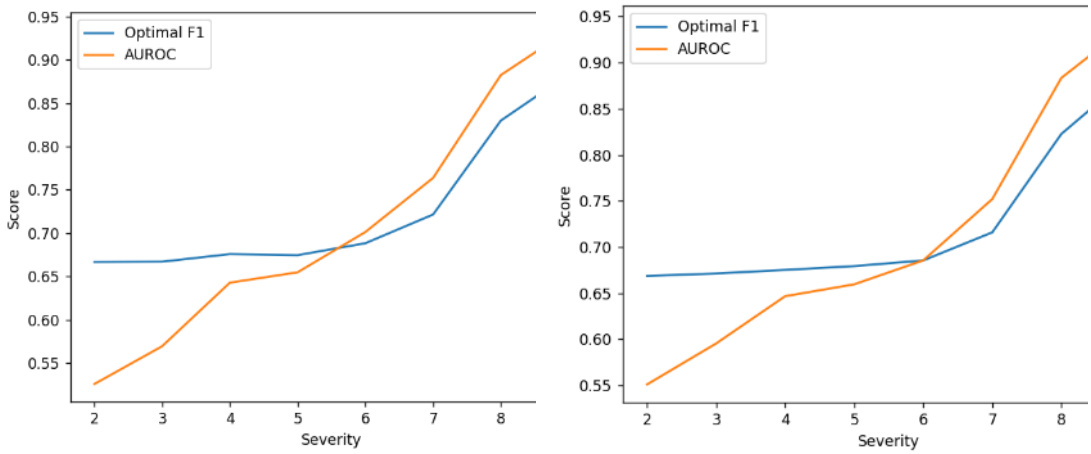


Figure 5.12. Feature embedding visualisation of FHBS.



(a) MVG (b) KNN



(c) SVDD (d) PCA

Figure 5.13. Distribution of anomaly scores on WHBD using embedding-based methods.

Table 5.6. Performance comparison of different disease detection algorithms on FHBS

Model	Precision	Recall	Optimal F1	AUROC
HSV	0.718	0.902	0.787	0.801
MVG	0.650	0.927	0.753	0.744
KNN	0.672	0.919	0.761	0.760
SVDD	0.616	0.907	0.726	0.709
PCA	0.604	0.935	0.726	0.714

5.6. System Validation and Verification

Finally, the whole model including wheat head bounding box detection and disease score calculation was run on Jetson Nano to evaluate the speed performance, as listed in Table 5.7. The Jetson Nano has Quad-core ARM Cortex-A57 CPU, 4 GB 64-bit LPDDR4 RAM, and NVIDIA Tegra X1 4GB GPU. The image size is the same as that for the training as 640x640 with a batch size of 16.

Table 5.7. Processing time (ms) of the head disease detection model on Jetson Nano

Model	Pre-process	Inference	Post-process	Distance	Total time
YOLOv8n	7.0	279.8	19.8	21.3	327.9

Even though with the smallest YOLOv8n model and run on GPU, the inference time for the bounding box still needs about 280 ms for one image. The distance calculation of feature embedding to MGM is relatively fast, similar to the post-processing part that calculates the NMS of bounding boxes. The total processing time is about 327.9 ms for each image. This means that the algorithm is able to run nearly real-time in practice. Assume that the rover runs in 1.5 m/s and the grid size is 0.5 m. This means that going through this grid takes about 1/3 second, which is just enough to process one image.

5.7. Summary

In this chapter, the proposed design methodology is adopted for the development of wheat head disease detection methods using the basic algorithm selection and optimisation process. To address the lack of wheat head disease data, a zero-shot learning based wheat head disease detection algorithm was proposed targeting real-time in-field wheat inspection. The proposed method was verified to have the ability to predict wheat head disease without using any diseased data during model development, and real-time inference when deployed on embedded systems such as Jetson Nano. This case study helps to validate the research direction of using the domain transfer and anomaly detection method for the detection of wheat head disease when the disease data is unavailable.

The wheat head disease detection case study aligns with the proposed design methodology by demonstrating its applicability in scenarios where data scarcity poses a significant challenge. During the context knowledge acquisition and system specification stages, the scarcity of annotated wheat head disease images was identified as a critical constraint. This insight informed the subsequent algorithm selection process, where a combination of classification-based methods and anomaly detection techniques was proposed to leverage the larger availability of healthy wheat head images.

The concept design stage guided the decision to adopt a pipeline integrating zero-shot learning and domain adaptation techniques. The anomaly detection-based approach verified the methodology's adaptiveness by shifting from conventional supervised learning to a more data-efficient paradigm, which exploits knowledge transfer from related domains.

Moreover, the embodiment design and verification stages supported the model development process by incorporating systematic evaluation across two open-source datasets. This demonstrated the methodology’s role in guiding algorithm selection, performance assessment, and generalization evaluation.

6. The Third Case Study: Pixel-Level Car Engine Surface Anomaly Detection

In this chapter, the proposed design methodology is adopted for the design of a car engine manufacturing anomaly detection system. The third case study focuses on the dataset establishment and the refined algorithm selection and optimisation process of the design methodology. It aims to provide a comparative study of pixel-level anomaly detection methods on car engine images. Following the proposed algorithm development process, two car engine datasets with synthetic and real anomalies are used as a validation and test dataset, respectively. Five typical deep learning anomaly detection methods are compared on these two datasets regarding image collection configurations and anomaly characteristics. Based on the findings from the comparative study, the best model was identified and further optimised for the real anomaly dataset, which achieved zero false negative and minimal false positives. This case study addresses the challenge of developing a robust AVI system in a more controlled industrial environment with a focus on comparative evaluation and model optimisation.

6.1. Introduction

With increased competition, globalisation, and uncertainty in today's markets, the manufacturing companies face more challenges in producing the desired products more efficiently. State-of-the-art techniques, such as machine vision and AI techniques, allow manufacturing systems to learn from big data in order to realise a connected and intelligent industrial practice, known as smart manufacturing [311]. By 2030, the size of the global smart manufacturing market is expected to grow from US$ 249.46 billion in 2021 to US$ 576.21 billion in 2028, with an annual growth rate of 12% [312]. Manufacturing automotive parts is a complex and precision-driven process, where even minor surface defects can lead to significant consequences, ranging from appearance problems to critical functional failures. Conventional methods, which rely on manual inspection or simple automated systems, suffer from problems of accuracy, consistency, and efficiency. Skilled human inspectors may inadvertently miss minor defects or make mistakes due to fatigue, while AVI systems, despite providing greater consistency, can struggle to spot subtle anomalies, particularly in intricate geometries such as those encountered in combustion engines [132].

Deep learning technology presents a transformative opportunity to overcome these limitations. Using advanced algorithms and data processing techniques, deep learning-based systems

provide a promising solution for the reliable detection of various surface anomalies. Research has shown that algorithms trained on large image dataset can identify defects in various materials, such as metals, textiles and composites, with high accuracy [313]. Convolutional neural networks (CNNs), in particular, have been highlighted for their effectiveness in learning defect characteristics [314]. Li et al. [315] and Yang et al. [316] reviewed the state-of-the-art deep learning methods in manufacturing defect detection. The main challenges identified by the authors include small object, complex background, data scarcity, and achieving high precision and fast detection, etc. This chapter focuses on addressing the issue of data scarcity.

Defect images from the production line are usually rare and difficult to collect. The scarcity of labelled data and the unpredictability of defect types pose significant challenges. Solutions to this problem primarily stem from two directions: data enhancement and anomaly detection. Data enhancement strategies focus on expanding the dataset though data augmentation [317] and synthesis [318], while anomaly detection method aims to detect the anomalies that are unseen in training data. However, implementing these advanced technologies for the car engine surface defect detection presents its own challenges, including illumination configuration, model selection, and model optimisation, due to its complex geometric structure. To our knowledge and based on the literature review, no study has explored the usage of anomaly detection in engine surface defect detection. Therefore, this chapter provides a comparative study of vision-based deep learning anomaly methods for car engine surface defect detection. The anomaly detection methods avoid the need for a database of defective images, which can be difficult to collect in a manufacturing environment where defects are usually rare.

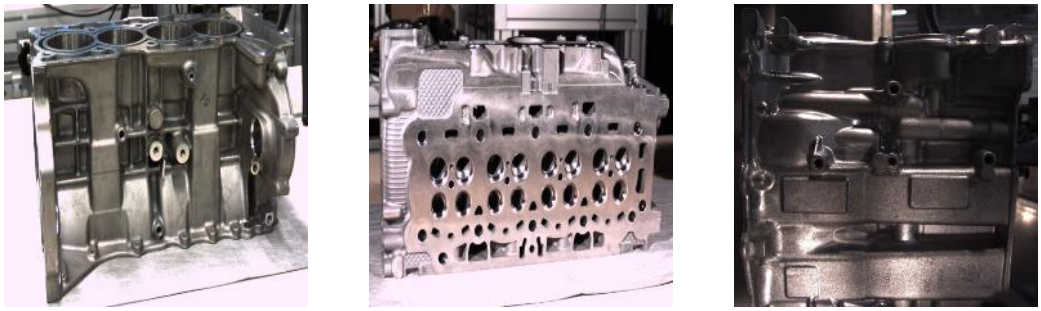
Following the algorithm selection and optimisation process in the proposed design methodology, this chapter developed the anomaly detection method using two datasets: the synthetic anomaly dataset as supplementary dataset, and real anomaly dataset as target dataset for algorithm optimisation. This study assesses anomaly detection performance in terms of image collection configuration and anomaly characteristics, targeting different parts of the car engine like engine head, block and underside.

6.2. Task Clarification

6.2.1. System Requirements and Specifications

Through the analysis of the need, the requirements of the car engine inspection system have been generated and summarised as follows:

- A vision system that can detect various manufacturing defects in multiple car parts.
- The detection targets include three critical areas of the engine, including the engine block, engine head, and engine under, as shown in Figure 6.1.
- The defect types are not constrained, including but not limited to choc, porosity, etc.
- The relative position of the camera and the parts are fixed.
- The controlled lighting condition is applied and optical filters are employed during the image capture process.
- The datasets fewer than 150 defect-free images for each category are available for AI model training.
- The system must not only detect the presence of defects, but also accurately determine their number and the specific location on the part.
- The area of the defects should be quantitatively detected as accurately as possible.
- The AI model should be compatible with the TensorFlow platform.
- The processing and inference time for analysing one image should be within the range of several seconds.
- The system should be adaptable to new types of defects and scalable to different types or inspection stations without significant reconfiguration.



(a) Engine Block (b) Engine Head (c) Engine Under

Figure 6.1. The three different viewpoints of the combustion engine for defect detection

6.2.2. Related Works

6.2.2.1. Engine Surface Defect Detection

Traditional methods rely on specific illumination and image processing to identify defects. For example, Liao et al. designed an inspection system under multiple directional illuminations so that defects on machined surface can be emphasised [319]. Wang et al. compared the deep learning method, VGG16 and InceptionV3, with the traditional method, HOG and SVM, in an enhanced vehicle parts dataset [320]. The result shows a significant improvement in deep learning methods for defect classification. Abagiu et al. proposed an inspection system for the

classification in engine pistons using CNN [20]. These image-level classification methods can predict if a defect is presented but cannot locate it. Zhu et al. proposed an Mask-RCNN based network to detect small defects in engine parts with bounding box [321]. A microscope is used to obtain detailed images of the parts, which is not practical on the production line. All these deep learning-based methods require hundreds of defect images for training. Instead, Abagiu et al. proposed using a structural similarity index measure for pixel-level defect detection [322] by comparing the test image with a reference image. However, this method requires the reference image to be well aligned with the test image under the same illumination.

6.2.2.2. Anomaly Synthesis

Data synthesis is useful for dataset enhancement, especially for industrial applications where data collection can be costly. The general data synthesis method includes the variational auto-encoder (VAE), generative adversarial network (GAN), diffusion model, etc. These methods have been adopted in defect detection. Li et al. [323] proposed to use synthetic 3D data for training the detection model to reduce the cost of time, energy and material during data collection. The trained model was used to detect geometric defects in additive manufacturing. Through GAN, Zhang et al. proposed a Defect-GAN network to generate realistic defect images. The network is capable of generating defects in normal images and remove the defects to generate normal images. Instead of image-level synthesis, some studies focus on anomaly synthesis and imposition on the original images. For example, the DTD method utilises the DTD texture dataset to create anomalies in different patterns, including line, colour, size, bend, and shape [324]. The CutPaste method randomly cropped and pasted images from the same dataset to simulate anomalies [218]. The natural synthetic anomalies (NSA) method employs Poisson image editing to seamlessly paste the copped image patch onto the target image [325]. The DRÆM method used the Perlin noise generator to create a variety of anomaly shapes [326]. To study the influence of anomaly characteristics such as size, brightness, and contrast, this research also proposed a new anomaly synthesis method that generates random polygon-shaped anomalies with controlled anomaly characteristics, including size, brightness, and contrast.

6.3. Concept Design

From the task clarification, only defect-free images are available for model training due to the lack of defect images. It is also required to detect the exact numbers and positions of the defects in each image. Therefore, the pixel-level anomaly detection method is adopted for this system. According to the task clarification, for surface defect detection, a monocular RGB camera with

LEDs is sufficient. An optical filter can be applied to reduce interference from the surface reflection of the metal parts.

Due to the limited number of collected data, the refined algorithm selection and optimisation process is adopted. The image synthetic and processing techniques are used to establish a validation dataset. The preliminary analysis is then carried out on this validation dataset to comprehensively compare the image collection parameters, synthesis methods, and detection models. Based on the preliminary analysis, the best image collection method and detection model can be selected for this system. Then the dataset with real anomalies are used for the algorithm optimisation based on the findings from the preliminary study.

6.4. Dataset Establishment

6.4.1. Image Collection

Due to reflection of the engine surface, some parts of the engine can be overexposed, leading to false positive detections. Reflective light on the smooth surface of the engine block is reduced by adding a polariser filter with a specific rotation angle. For comparison, five different filter conditions are applied, including without filter, a filter rotated at 0, 45, 90, and 135 degrees, as shown in Figure 6.2. When the filter is applied, the exposure time is increased to maintain the same image brightness. The reflected light is significantly reduced when the filter is rotated to 90 degrees.

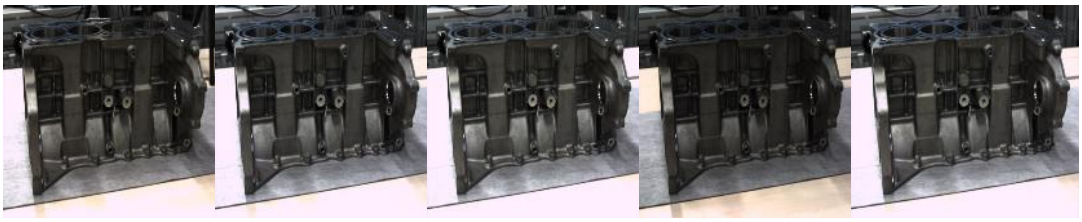


Figure 6.2. Sample images of different filter conditions: no filter, filter rotated at 0, 45, 90, 135 degrees, from left to right.

Exposure time is another important factor that affects areas of overexposure and underexposure. During data collection, three exposure times were used for each filter and lighting configuration to cover different exposure settings. The intermediate exposure time t was determined by the illuminance E in lux, from [327]:

$$t = \frac{N^2 C}{ES} \tag{6-1}$$

where N is the f-number, C is the incident-light meter calibration constant, with common value 250 used, S is the ISO arithmetic speed. The exposure time was halved and doubled to provide

a range of settings, as shown in Figure 6.3.

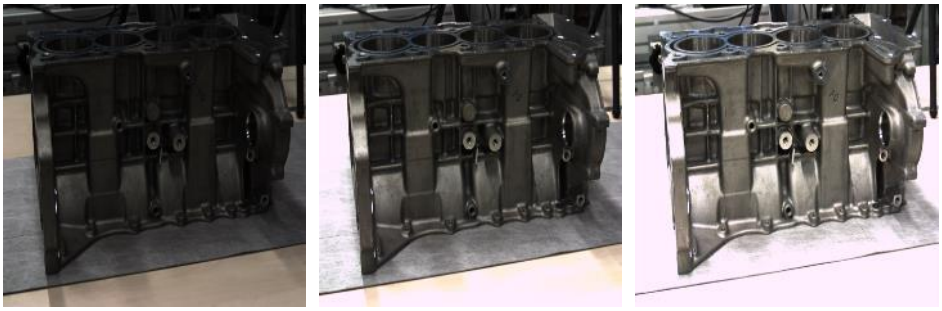


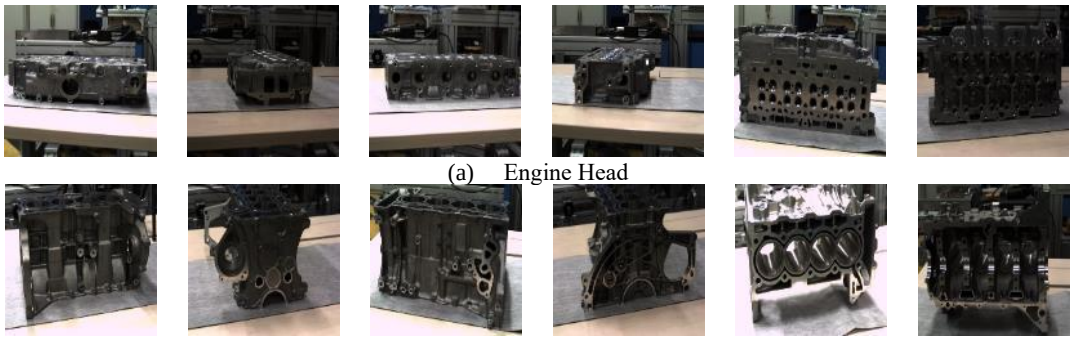
Figure 6.3. Sample images of engine block with different exposure times of 100 ms, 200 ms, and 400 ms under ceiling lights.

Different light directions and light intensities can affect the brightness distribution of the target object. In this project, three different lighting configurations were used. The first configuration utilised lights from the ceiling. The second used one LED light from the left-hand side of the object. The third configuration included an additional LED light from the right-hand side of the object. This approach allowed for a variety of lighting directions and intensities to be included, as shown in Figure 6.4.

Figure 6.4. Sample images of an engine block with three lighting conditions: lights from the ceiling, one LED from left-hand side, and two LEDs from the left and right-hand side.

Images of two engine parts, i.e., one engine block and one engine head, are captured to build the dataset. With each set of image capture parameters, one of the six faces of the engine part are placed to face toward the cameras. In total, 540 images were collected for each part of the engine. Images of the six faces of engine head and engine block are shown in Figure 6.5.

The image collection setup is shown in Figure 6.6. The two cameras used are HikVision MV-CE100-30GM with an image resolution of 3840x2748. Two LEDs are located on the two sides of the target engine block/head. Exposure time is controlled by the collection software. The other camera parameters are kept the same during the whole collection, including the white balance, gain, focus, etc.



(a) Engine Head

(b) Engine Block

Figure 6.5. Example images taken for different faces of engine head and engine block.

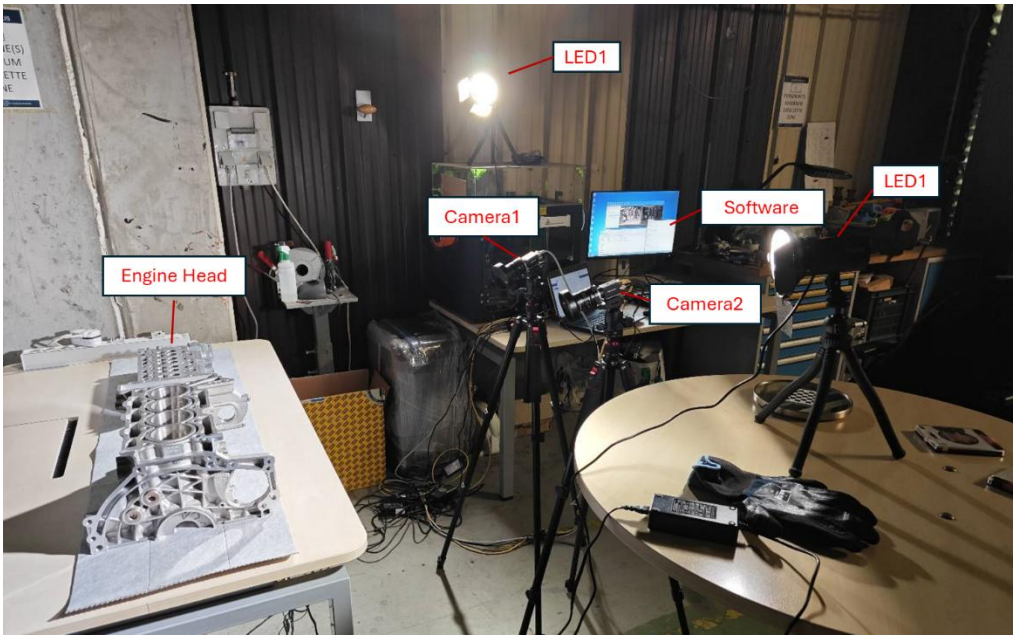


Figure 6.6. Image collection setup for the engine head with two cameras and two LEDs.

Another set of images with real anomalies was captured using the system developed by the Stellantis Centre Technique de Carrières (formerly PSA Group) in France. Several engines after the machining process, with and without anomalies, during production are placed at a fixed position in front of the camera. The camera resolution is 3088x2064. Since the object is in the same position for all the images, the background in the image can be cropped out. The sample images with and without anomalies are shown in Figure 6.7. This object is named as engine under. In total, 55 images are normal images, among which 45 images are randomly selected for training and 10 images with anomalies at different positions.

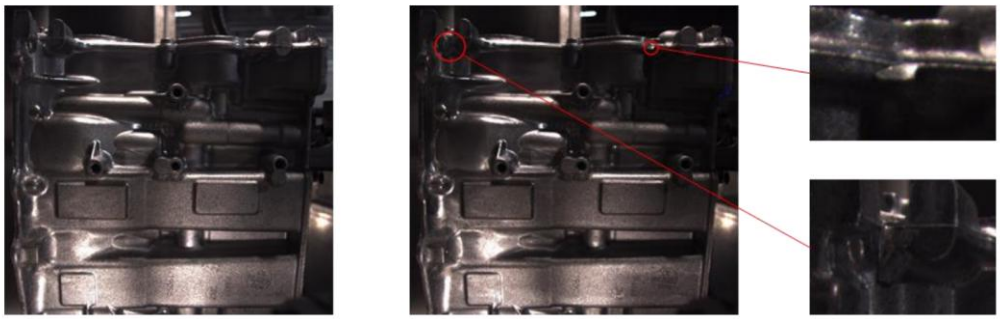


Figure 6.7. Sample images of engine under without (left) and with (right) anomalies.

6.4.2. Image Synthesis

Anomalies are rare and building an anomaly dataset is difficult. Therefore, some researchers have used image synthetic techniques to mimic the anomalies and enrich the dataset. In this project, three image synthesis techniques have been adopted to build engine anomaly datasets, including Natural Synthetic Anomalies (NSA) [325], Perline noise generator [326], and polygons with controlled area and brightness distribution. The aims of using synthetic datasets are: 1) study the effects of the image collection parameters, 2) compare the effectiveness of the image synthesis methods, and 3) compare the effectiveness of the anomaly detection methods.

6.4.2.1. NSA

The NSA method tries to clone image patches from one image to another by Poisson image editing [328]. To synthesise one anomaly image by NSA method, the source and target images are randomly selected. Then an image patch from the source image is randomly resized and pasted onto the target image at a random position with Poisson image editing. The number of image patches for each anomaly image varies from 1 to 3. Finally, the pixel-wise labels for each anomaly image are generated. One sample anomaly image for engine block generated by NSA method is shown in Figure 6.8.

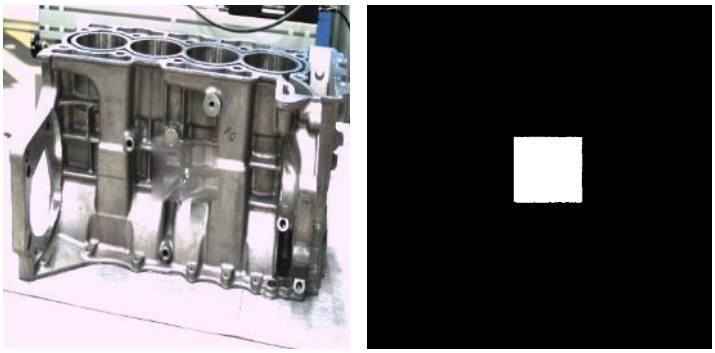


Figure 6.8. Sample synthetic anomaly image (left) and label (right) for engine block generated by the NSA method

6.4.2.2. Perlin

Following the anomaly generation in the DRAEM [326] method, the Perlin noise image is generated and binarized to capture a variety of anomaly shapes. This noise image is then used as an anomaly mask for the source image to generate the anomalies, as an example in Figure 6.9. The augmentation functions are used to augment the source image, including posterise, sharpness, solarize, equalize, brightness change, colour change, and auto-contrast.

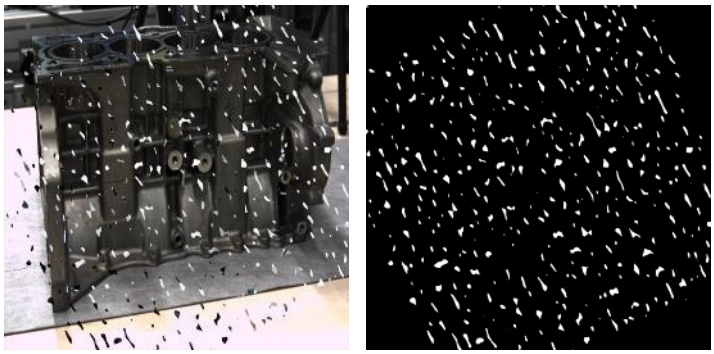


Figure 6.9. Sample synthetic anomaly image (left) and label (right) for the engine block generated by the Perlin noise image.

6.4.2.3. Polygon

The NSA method can only generate rectangular shaped anomalies and the Perlin noise image has less control to the anomaly shapes and positions. To address the above problems, a simple polygon generator is used to generate more controlled anomalies, as shown in Figure 6.10. The number of polygon sides varies in the range of 3 to 6. The area of the polygon ranges from 100 to 5000 pixels. The contrast of the anomaly with respect to the target image is within 0.5 to 1.5.

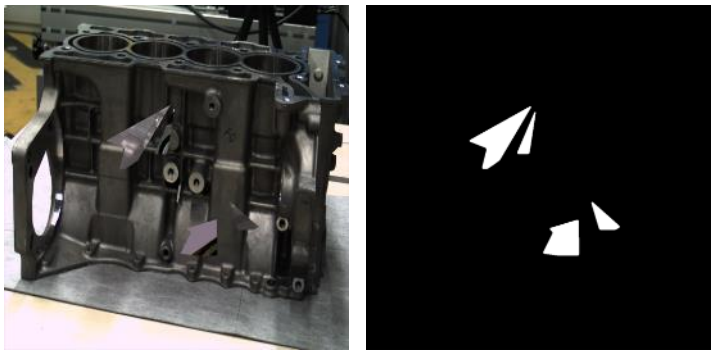


Figure 6.10. Sample synthetic anomaly image (left) and label (right) for engine block generated by random polygons.

6.4.3. Datasets with Synthetic and Real Anomalies

Two datasets are used for system verification and validation. The first dataset consists of images with synthetic anomalies. For each engine part, the 540 original images are randomly

split into 270 images for training and 270 test images. Additional 270 training images are generated with synthetic anomalies using different methods. The 270 test images are randomly divided into 135 good images and 135 defective images with synthetic anomalies. In total, the whole dataset contains 4860 images. The second dataset contains images with real anomalies. It contains 45 training images and 20 test images. Among the test images, 10 of them are normal images and another 10 contain 18 anomalies in total.

6.5. Synthetic Anomaly Analysis

6.5.1. Anomaly Detection Algorithm Candidates

For the preliminary analysis, five state-of-the-art anomaly detection algorithms are chosen, including the GANomaly[185], NSA [328], STFPM [224], FastFlow [221], PaDiM [222].

6.5.1.1. GANomaly

Generative Adversarial Networks (GANs), which consist of a generator and a discriminator, can be particularly effective for anomaly detection due to their ability to model complex data distributions. The GANomaly model trains a GAN to determine if an image contains anomaly. An encoder-decoder network is first used to reconstruct the input image as a generator. A discriminator is used to determine whether the reconstructed image is fake or real. Therefore, this model can only be used for image-level classification.

6.5.1.2. NSA

The NSA method takes the synthetic anomaly images to train an encoder-decoder CNN with ResNet as the backbone. The model is trained to directly predict the anomaly mask, similar to the traditional segmentation task. For better prediction results, logistic labels are used for model training:

$$y_{logistic} = \frac{y_{binary}}{1 + \exp\left(-k(y_{continuous} - y_0)\right)} \tag{6-2}$$

where y_{binary} is the binary label generated during image synthesis, $y_{continuous}$ is the colour difference between the synthetic image and original image, k and y_0 are the intensity logistic parameters, set as 1/6 and 20 as default. ResNet-18 is used as the encoder without the final classification layers. The decoder uses the same ResNet blocks and up-samples the feature embedding to original image size. The binary cross-entropy loss is used for binary labels and mean square error loss for logistic labels.

6.5.1.3. STFPM

The Student-Teacher Feature Pyramid Matching (STFPM) algorithm is based on a student-teacher network architecture. The teacher network is typically pre-trained on a large, diverse dataset to capture a wide array of features. The student network is trained on a more specific dataset, typically the normal data for the anomaly detection task. The algorithm leverages Feature Pyramid Networks (FPNs), which are a popular architecture in computer vision used to detect objects at various scales within an image. Anomaly detection is performed by comparing the feature pyramids of the student and teacher networks. For input image I, the difference on the l-th feature map from the teacher $F_t^l(I)$ and student network $F_s^l(I)$ is calculated as the L_2 norm distance:

$$L^l(I) = \frac{1}{2}\left\|F_t^l(I) - F_s^l(I)\right\|_2^2 \tag{6-3}$$

6.5.1.4. FastFlow

The FastFlow is an anomaly detection algorithm that utilises Normalizing Flows (NFs). The NFs are designed to transform data from a complex, multi-dimensional distribution into a simpler distribution. They are "normalising" because they transform data into a standard form, and they are "flows" because they allow the transformation to be reversed. In FastFlow, the feature map from the feature extractor is transformed to a standard normal distribution. The probability density of each location on the feature map is estimated for anomaly detection. The flow model $f: X \to Z$ is a bijective invertible mapping that projects the image features $x \in p_X(x)$ to hidden variable $z \in p_Z(z)$. It consists of a stack of the invertible transformation flow blocks f_i. The model distribution on X can be defined by the change of the hidden variable as:

$$p_X(x) = p_Z(z)\left|\mathbf{det}\left(\frac{\partial z}{\partial x}\right)\right| \tag{6-4}$$

The log likelihoods for image features can be estimated from $p_Z(z)$ by:

$$\log p_X(x) = \log p_Z(z) + \log\left|\mathbf{det}\left(\frac{\partial z}{\partial x}\right)\right| = \log p_Z\big(f_\theta(x)\big) + \log\left|\mathbf{det}\left(\frac{\partial f_\theta(x)}{\partial x}\right)\right| \tag{6-5}$$

where $z \sim \mathcal{N}(o, I)$ and the $\frac{\partial f_\theta(x)}{\partial x}$ is the Jacobian of the flow model, θ is parameters of the flow model. The invertible flow model satisfies that $z = f_\theta(x)$ and $x = f_\theta^{-1}(z)$.

6.5.1.5. PaDiM

The Patch Distribution Modelling (PaDiM) algorithm leverages patch-based fully convolutional network to detect the anomalies based on the feature embeddings modelling.

For each patch, PaDiM models the distribution of features extracted by a pre-trained CNN. It uses multivariate Gaussian distributions (MGD) to capture the statistical properties of the features of normal patches. The anomaly is detected by comparing the feature embedding of the test image patch against the learnt distributions. The feature embeddings $X_{i,j} = \{x_{i,j}^n, n \in [1, N]\}$ of N image patches at location (i, j) extracted from the pretrained CNN are modelled as a multivariate Gaussian distribution $\mathcal{N}(\mu_{i,j}, \sum_{i,j})$, where $\mu_{i,j}$ and $\sum_{i,j}$ are the mean and covariance of $X_{i,j}$. The covariance $\sum_{i,j}$ is estimated as:

$$\sum_{i,j} = \frac{1}{N-1}\sum_{n=1}^{N}\left(x_{i,j}^n - \mu_{i,j}\right)\left(x_{i,j}^n - \mu_{i,j}\right)^{\mathrm{T}} + \epsilon \boldsymbol{I} \tag{6-6}$$

where the $\epsilon \boldsymbol{I}$ is regulation term to make the covariance matrix full rank and invertible. Each image patch is associated with a multivariate Gaussian distribution that carries information of different semantic levels. During the test, the feature embeddings of the image patches from the test image are extracted at each location. The anomaly score is obtained by Mahalanobis distance calculation between the feature embeddings and the multivariate Gaussian distribution:

$$M\left(x_{i,j}\right) = \sqrt{\left(x_{i,j} - \mu_{i,j}\right)^{\mathrm{T}} \sum_{i,j}^{-1} \left(x_{i,j} - \mu_{i,j}\right)} \tag{6-7}$$

The Mahalanobis distances at different image locations form the anomaly map. The image-level anomaly score is the highest score in the anomaly map.

6.5.2. Experiment Setup

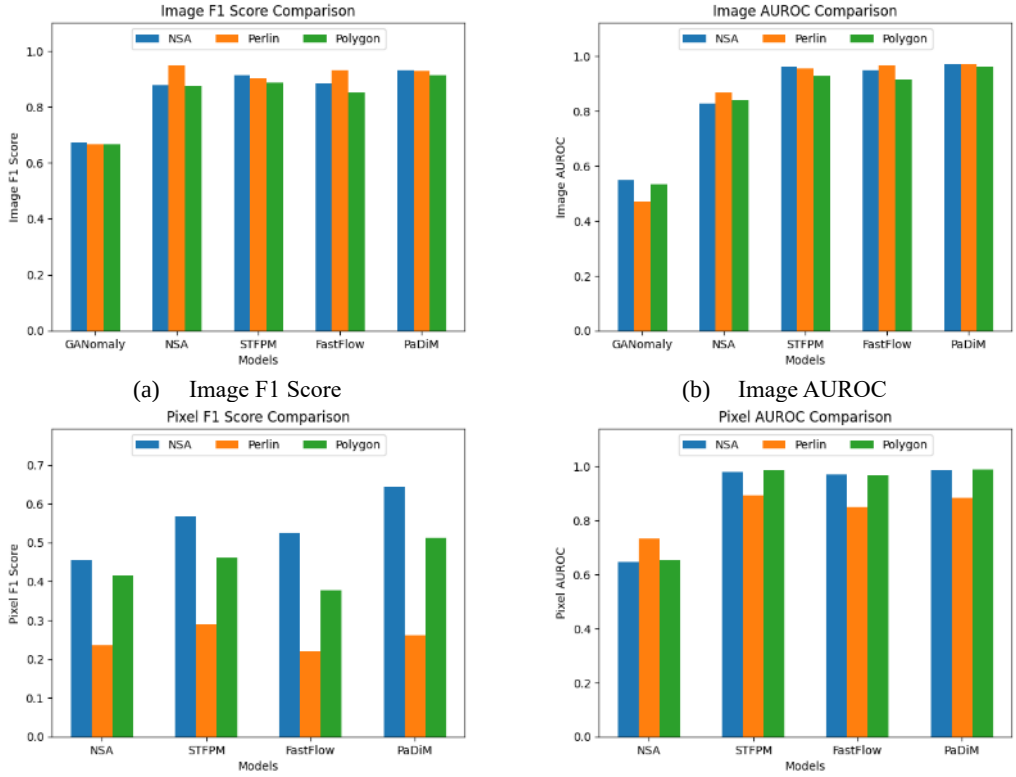
Anomalib [329] is an open source deep learning libraries that has collected a number of state-of-the-art anomaly detection algorithms. A good API design has made it a good choice for algorithms selection and development for this system. The default parameters are used for each anomaly detection model. ResNet18 is used as the backbone network for models who need one. The input image size is set to 256x256. ImageNet normalisation is used for all the images.

To evaluate the performance of anomaly prediction, two sets of metrics are used. The first set is for image and pixel level analysis, including Optimal F1 and AUROC, which have been widely used in anomaly detection. Another one is anomaly level analysis focusing on the anomaly score of each anomaly, regarding to anomaly size, brightness, and contrast.

6.5.3. Image and Pixel Level Analysis

6.5.3.1. Training with the full dataset

To compare the performance of different models for different target objects, the models are trained and tested with the full dataset, as shown in Figure 6.11. For image-level performance, the image F1 score and AUROC are all above 0.8 except for the GANomaly model with only less than 0.7 image F1 score and less than 0.6 AUROC. The other four models have similar performance but react differently to the synthesis methods. The reconstruction-based models, NSA and FastFlow, perform best on the Perlin dataset, while the feature embedding-based methods, STFPM and PaDiM perform best on the NSA dataset. As for the pixel level performance, there is a large deviation between the F1 score and AUROC. The F1 score is much lower than the AUROC, typically below 0.65 for the NSA dataset, and below 0.5 for the other two datasets. This deviation is caused by the pixel-level imbalance between normal and abnormal pixels. The Perlin dataset has many small anomalies that only occupy a small part of the image. Although the anomalies from Polygon and NSA datasets are typically large patch, therefore showing a higher F1 score. The GANomaly model is not included due to its incapability of pixel-level detection.



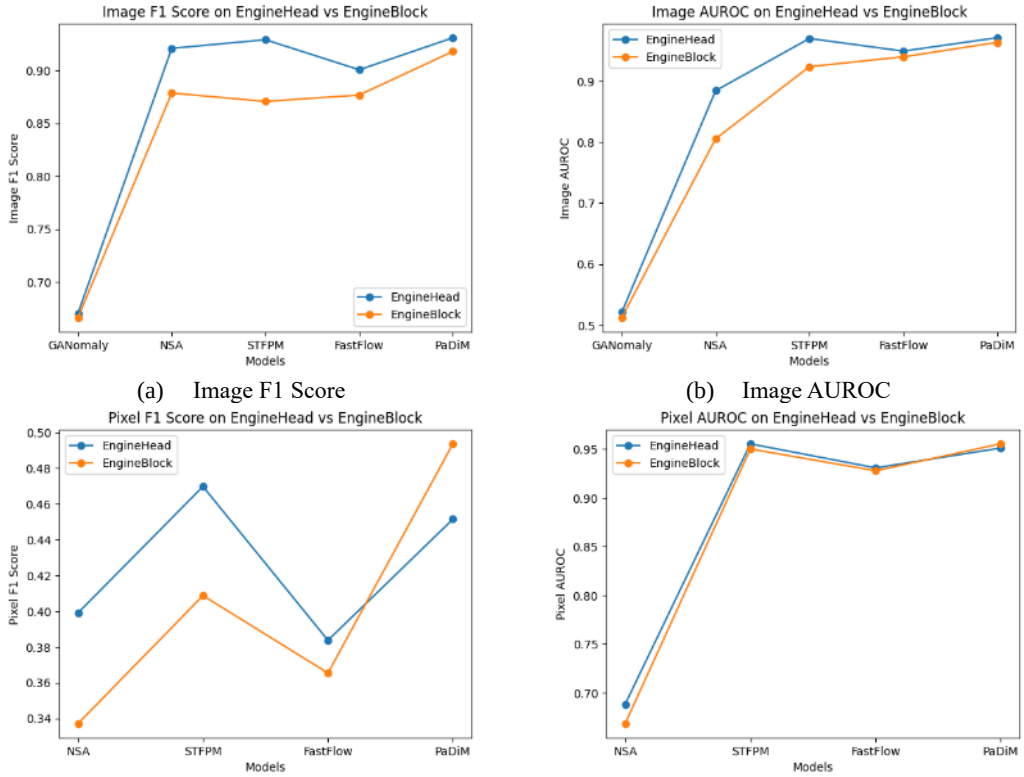
(a) Image F1 Score (b) Image AUROC

(c) Pixel F1 Score (d) Pixel AUROC

Figure 6.11. Model comparison with respect to image and pixel level metrics.

In terms of the target object, the performance of the models has the same trends for engine head and engine block. The engine block dataset is more difficult than the engine head, with

difference within 0.1 on all the evaluation metrics, as can be seen in Figure 6.12.



(a) Image F1 Score (b) Image AUROC

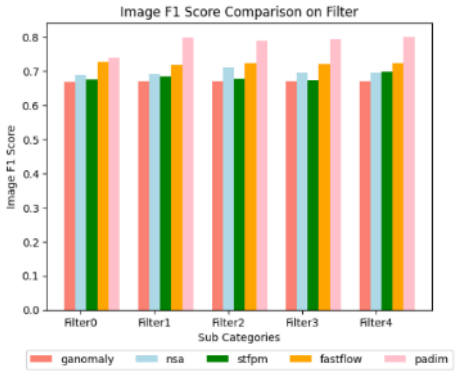
(c) Pixel F1 Score (d) Pixel AUROC

Figure 6.12. Model performance comparison on engine head and engine block.

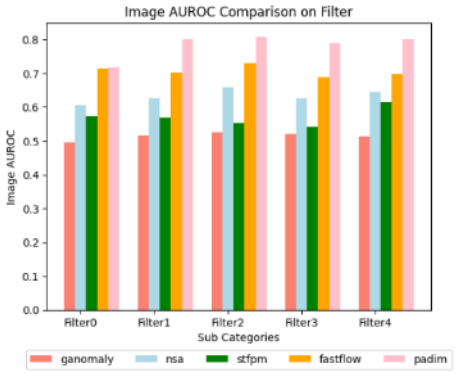
6.5.3.2. Filter

To compare the effects of the polarised filter, only data with specified filter condition are used for training but tested on the entire test dataset with all the filter conditions, as shown in Figure 6.13. The Filter0-4 corresponds to no filter, and filter rotated at 0, 45, 90, and 135 degrees, respectively. The overall performance of Filter0, which does not use filter, has the lowest performance for all the anomaly detection models. The other four filter conditions exhibit similar performance at both the image and pixel levels, with no single condition significantly outperforming the others.

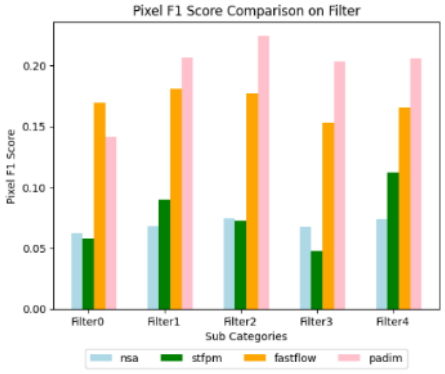
As for the models, the PaDiM model achieves the best performance over all the filter conditions except the pixel F1 on Filter0. This could be explained by the lack of training data when the dataset is divided into five subsets with different filter conditions. All models except PaDiM need iterative training on the dataset. Consequently, the PaDiM is less sensitive to image scarcity and filter variations.



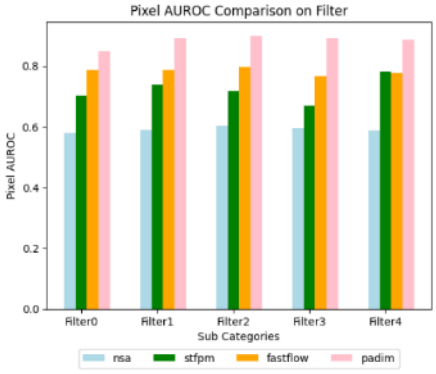
(a) Image F1 Score



(b) Image AUROC

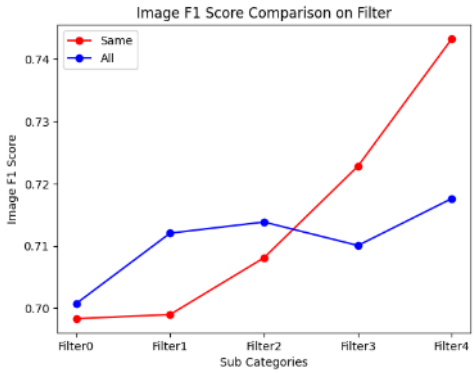


(c) Pixel F1 Score

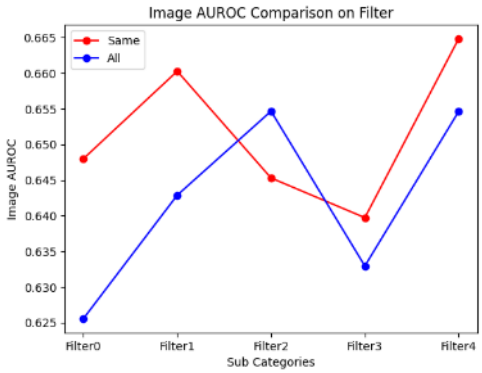


(d) Pixel AUROC

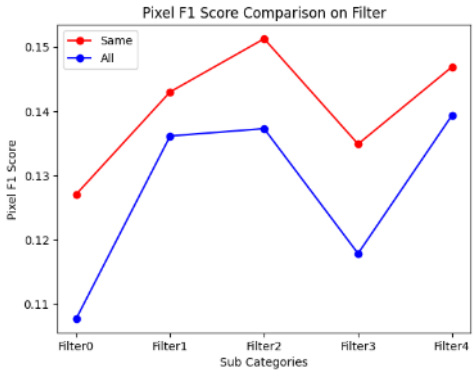
Figure 6.13. Model comparison under different filter conditions: F0-no filter, F1-filter rotated at 0°, F2-45°, F3-90°, F4-135°.



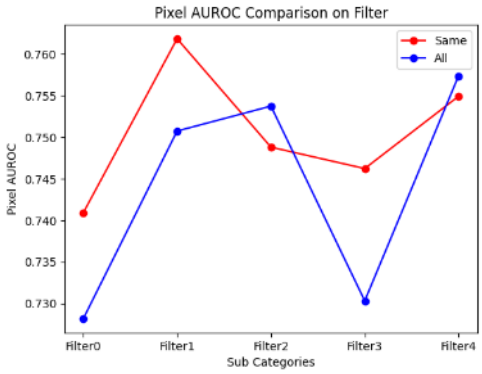
(a) Image F1 Score



(b) Image AUROC



(c) Pixel F1 Score



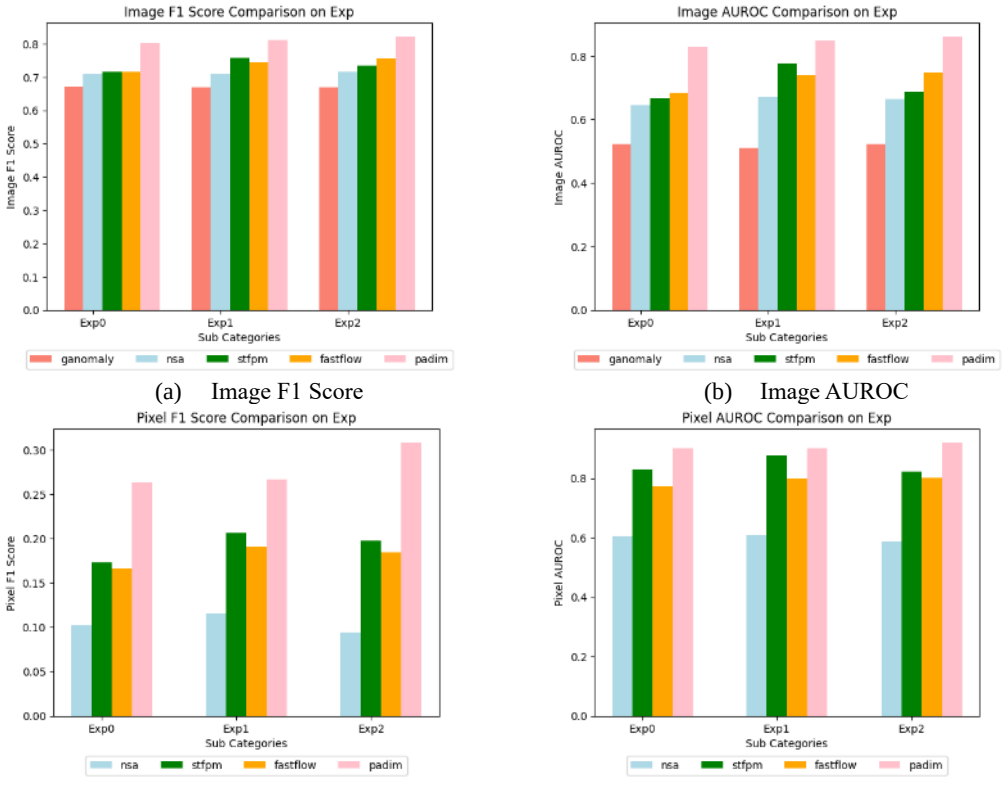
(d) Pixel AUROC

Figure 6.14. Comparison of models under different filter conditions and tested on dataset with the same filter condition or dataset with all filter conditions.

In real applications, the filter condition is mostly kept unchanged during development and deployment. Therefore, the models are also tested with dataset that has the same filter condition with the training dataset, as shown in Figure 6.14. Interestingly, the model does not always perform better on the dataset with the same filter condition. The accuracy across different filter conditions using various metrics shows a degree of randomness. This suggests that for synthetic anomalies, the filter condition has a limited impact on detection accuracy.

6.5.3.3. Exposure

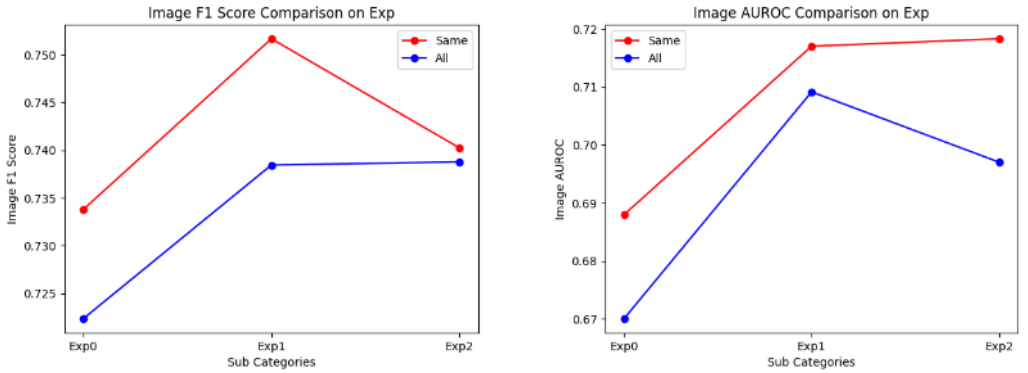
Three training datasets with three exposure levels, namely Exp0, Exp1, and Exp2, are established for model training. The exposure time of Exp1 is calculated by Eq.(1). The time of Exp0 and Exp2 is half and double of Exp1. The performance of anomaly detection on the three datasets is shown in Figure 6.15. The PaDiM model consistently performs the best across all datasets and metrics. A comparison with the average performance across models is shown in Figure 6.16. The models perform best with the normal exposure dataset, but the difference in performance between the different exposure levels is less than 0.02 for all models. It is obvious that models trained and tested on the same exposure levels perform better than tested on the whole dataset with all the exposure levels.



(a) Image F1 Score (b) Image AUROC

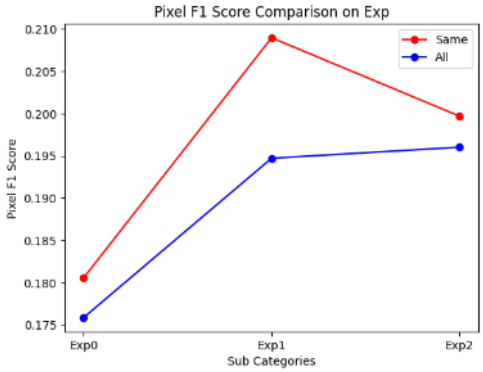
(c) Pixel F1 Score (d) Pixel AUROC

Figure 6.15. Model comparison on different exposure levels: Exp0-under exposure, Exp1-normal exposure, Exp2-over exposure.

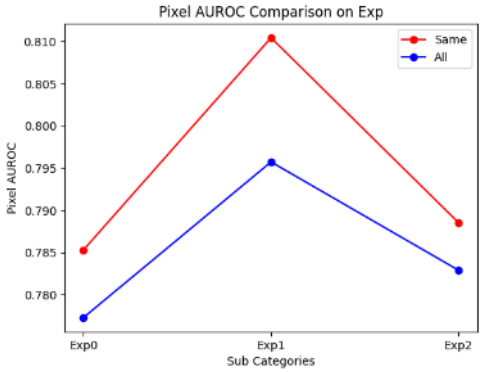


(a) Image F1 Score

(b) Image AUROC



(c) Pixel F1 Score

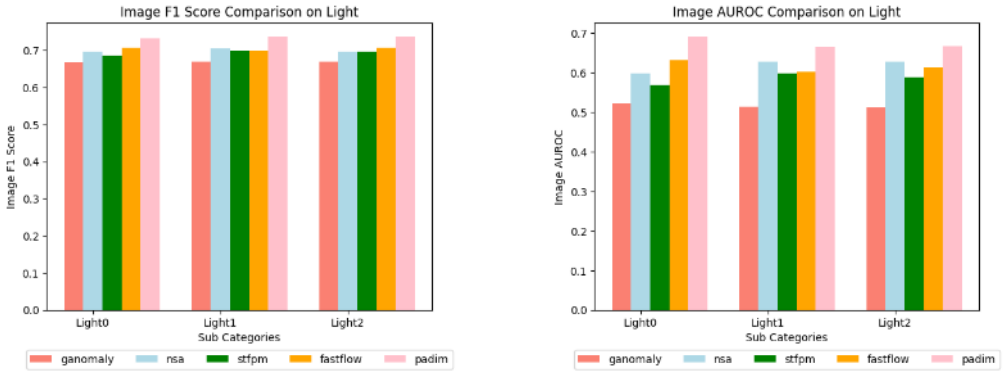


(d) Pixel AUROC

Figure 6.16. Model comparison on different exposure level and tested on dataset with the same exposure level or dataset with all the exposure levels.

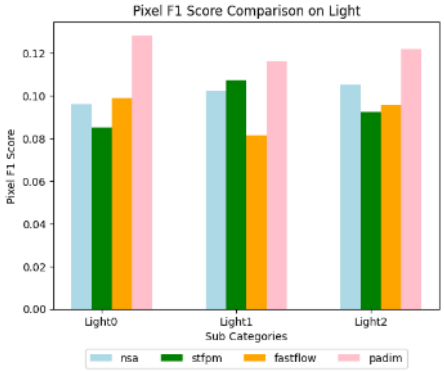
6.5.3.4. Lighting

Three training datasets with three lighting conditions, namely Light0, Light1, and Light2, are established for model training. The Light0 to Light2 correspond to lights from the the ceiling, one LED from the left-hand-hand side, and two LEDs from left and right-hand side. The performance of anomaly detection on three datasets are shown in Figure 6.17. The performance differences among the models are much smaller compared to the changes caused by different filters or exposure levels. Furthermore, the performance is similar across the three lighting conditions, indicating that the anomaly detection models have consistent but low sensitivity to lighting variations. This can also be observed from Figure 6.18 that the lighting condition itself has a limited impact on performance. However, the difference between testing on dataset with specific lighting conditions and dataset with all lighting conditions are much larger, with 0.05-0.1 gap on image and pixel accuracy. This indicates that the lighting conditions should remain unchanged during training and testing. Otherwise, the results will be variant unless the diversity of lighting conditions in the training dataset should be ensured to be close to the actual ones.

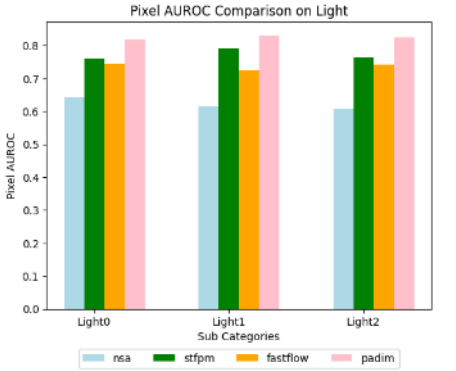


(a) Image F1 Score

(b) Image AUROC

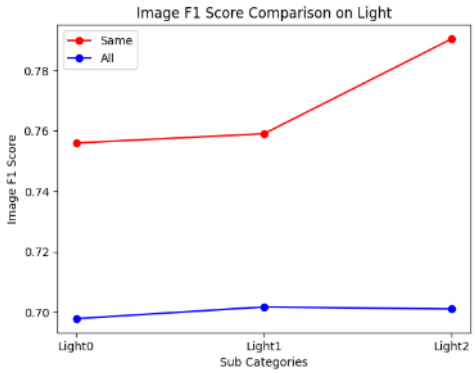


(c) Pixel F1 Score

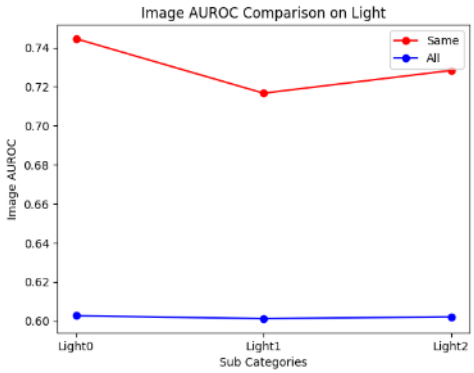


(d) Pixel AUROC

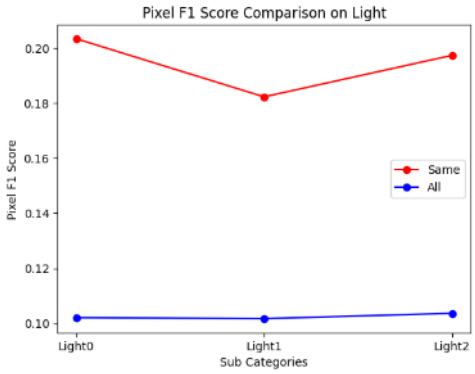
Figure 6.17. Model comparison under different lighting conditions: Light0-ceil light, Light1-one LED from left, Light2-two LEDs from left and right.



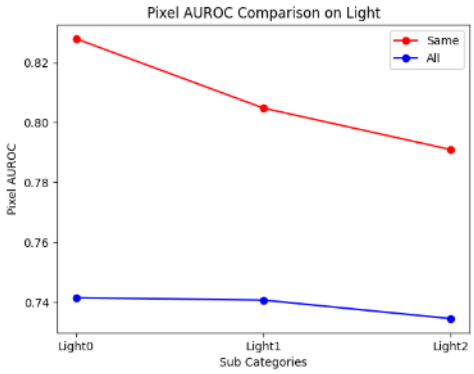
(a) Image F1 Score



(b) Image AUROC



(c) Pixel F1 Score



(d) Pixel AUROC

Figure 6.18. Model comparison on different lighting conditions and tested on dataset with the same lighting condition or dataset with all the lighting conditions.

6.5.3.5. Summary

Overall, in terms of the anomaly detection methods, the GANomaly model performed the worst in image-level detection. The other four methods performed similarly when trained with the whole dataset. However, the PaDiM model performed the best when trained with subcategories dataset, showing highest robustness across different image collection parameters, including the filter, exposure, and lighting.

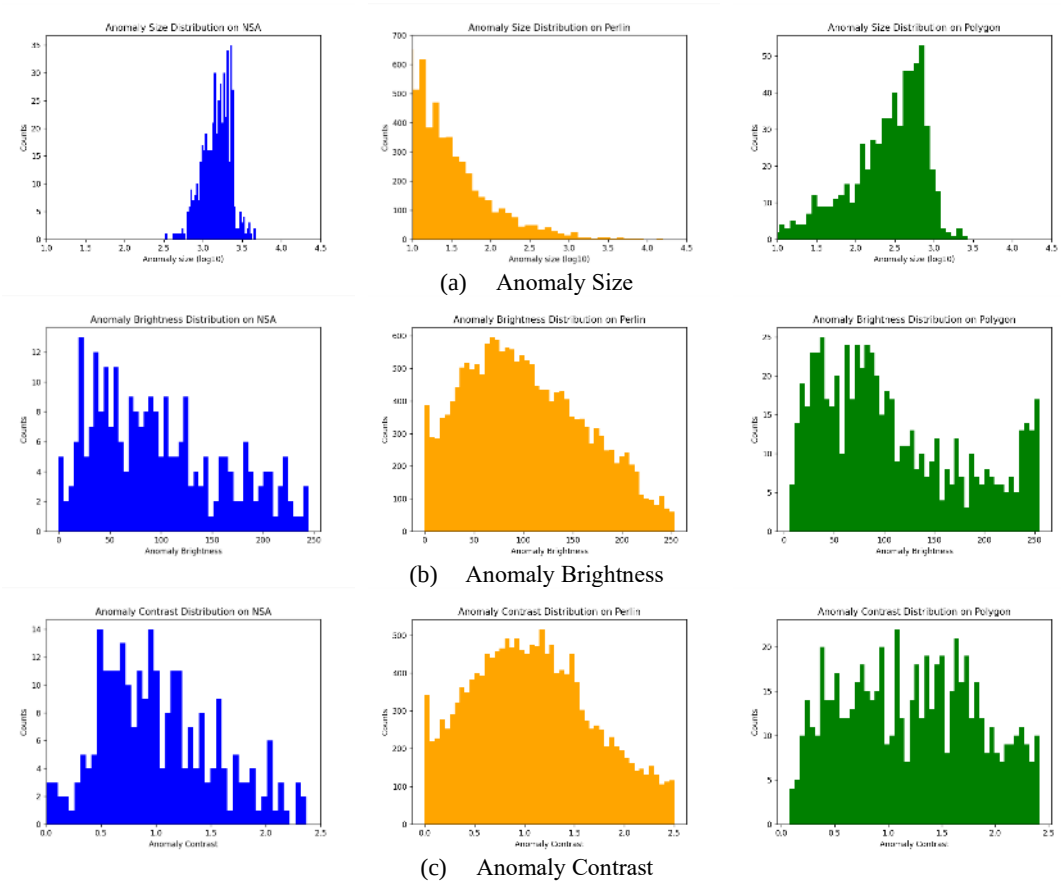
Among the three datasets, the NSA and Perlin datasets are easier than the Polygon dataset for image-level anomaly detection. For pixel-level detection, the NSA dataset is significantly easier than the other two. This can be explained by the fact that the NSA dataset contains anomalies in large patch shapes, while the Perlin and Polygon datasets contain anomalies in random shapes and smaller sizes. More detailed datasets comparison can be found in the next section about anomaly level analysis.

For datasets with controlled image collection parameters, the filter has less impact on anomaly detection. Although reducing reflective light can decrease overexposed areas of the target part, this effect can also be achieved by adjusting the exposure time. The polariser filter may be particularly useful when defects are bright and easily confused with other overexposed areas, which is not the case for this dataset. With respect to exposure time, on average, the best performance for anomaly detection is achieved at normal exposure levels. Maintaining consistency between exposure levels during training and testing is beneficial for optimal performance. This consistency is essential for lighting conditions and is more important than the specific lighting condition itself.

6.5.4. Anomaly Level Analysis

6.5.4.1. Anomaly Distribution

The anomaly level analysis focuses on the effects of anomaly characteristics on detection, including anomaly size, anomaly brightness, and anomaly contrast. Firstly, the distribution of these three anomaly characteristics is compared in Figure 6.19. For a better view, the anomaly sizes are presented on a log10 scale on the x-axis. It can be seen that the NSA dataset in Figure 6.19 (a) contains anomalies concentrated between 2.7 and 3.5 (approximately 500 to 3000 pixels) in a 256x256 resolution image. The Perlin dataset has anomalies less than 100 pixels. The Polygon dataset has anomalies distributed around 1.5 to 3 (approximately 30 to 1000 pixels). The brightness and contrast distribution for the three datasets is similar and normally distributed.



(a) Anomaly Size

(b) Anomaly Brightness

(c) Anomaly Contrast

Figure 6.19. Comparison of the anomaly characteristics of NSA, Perlin, and Polygon datasets, regarding anomaly size, brightness, and contrast.

6.5.4.2. Anomaly Size

The anomaly score relative to the scaled anomaly size for different datasets and anomaly detection models is compared in Figure 6.20. The anomaly score for each anomaly is the maximum prediction score within the anomaly area. For each bin of the anomaly size in the plot, the anomaly score is averaged for anomalies within the size range. For all three datasets, the anomaly score increases linearly with the scaled anomaly size. Only a small number of anomalies have size larger than 3.0 from the Perlin dataset, the drop of anomaly score when the size is larger than 3.5 is not crucial.

Regarding to the anomaly detection models, the STFPM model remains close to 0.5 for all the anomaly size. The PaDiM model performs better than FastFlow when the scaled anomaly size is smaller than 3.0. The FastFlow model performs the best when the scaled anomaly size is larger than 3.0 except one point that the NSA model surpassed FastFlow at 3.7. The anomaly score of the NSA model exceeds 0.5 only when the scaled anomaly size is larger than 3.5.

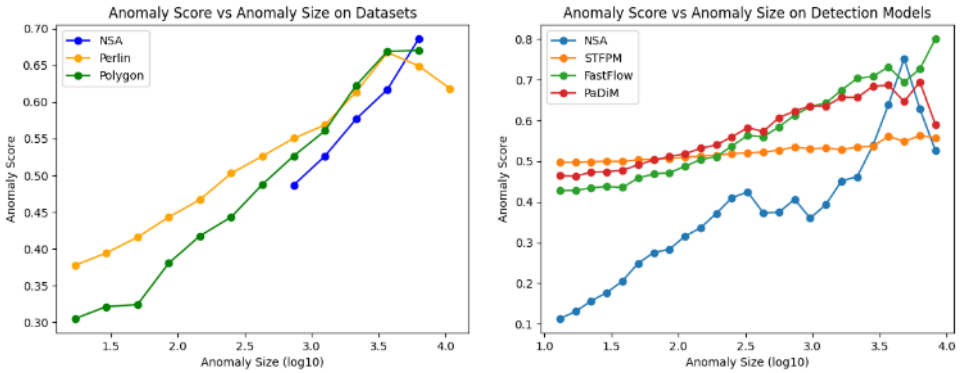


Figure 6.20. The anomaly score in relation to the anomaly size for different datasets (left) and anomaly detection models (right).

6.5.4.3. Anomaly Brightness

The anomaly score relative to the anomaly brightness for different datasets and anomaly detection models is compared in Figure 6.21. It can be observed that the NSA and Polygon datasets are less sensitive to brightness variations. In contrast, for the Perlin dataset, the anomaly score is significantly higher for anomaly brightness between 50 and 200. Regarding the models, the anomaly score increases steadily when the brightness is less than 170 and then drops slightly. Due to the image normalisation process of the models, the absolute brightness of the anomalies has less impact on the anomaly score than the relative brightness, the anomaly contrast, as discussed below.

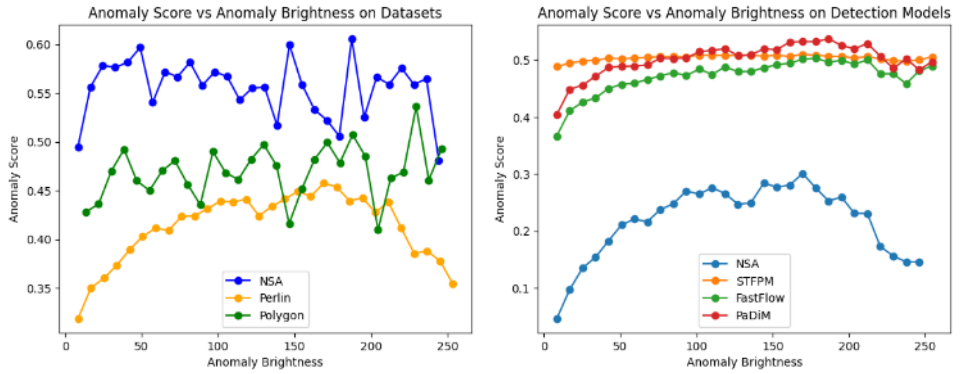


Figure 6.21. The anomaly score with respect to the anomaly brightness for different datasets (left) and anomaly detection models (right).

6.5.4.4. Anomaly Contrast

The anomaly score with respect to the anomaly contrast for different datasets and anomaly detection models is compared in Figure 6.22. All three datasets show increase trend of anomaly score with the increase of the contrast, especially for the Perlin dataset, which has much greater number of anomalies. The anomaly score of the STFPM model still remains around 0.5 with a slight increase with the contrast. The PaDiM model performs better than FastFlow for all the

contrast ranges. The NSA model struggles to achieve an anomaly score greater than 0.2.

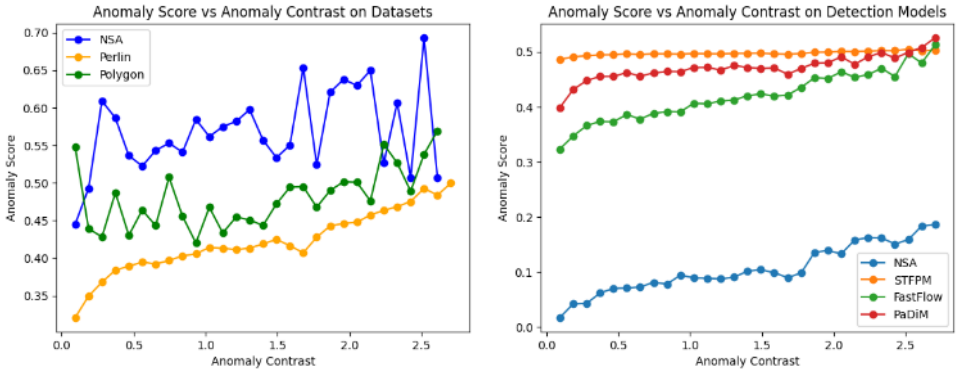


Figure 6.22. The anomaly score with respect to the anomaly contrast for different datasets (left) and anomaly detection models (right).

6.5.4.5. Summary

The three datasets cover different distributions of the anomaly size but have similar distributions of the brightness and contrast. For anomaly size, the anomaly score increases linearly with the scaled anomaly size. The anomaly brightness has less influence on the anomaly score than the anomaly contrast. A higher anomaly score is achieved for brightness between 50 and 200. A higher anomaly contrast usually leads to higher anomaly scores.

6.6. Real Anomaly Analysis

In this section, the model performance is evaluated on the real anomaly dataset. Since the NSA method still requires synthetic anomalies during the training stage, the three synthesis methods are used to generate synthetic anomalies. To evaluate the influence of anomaly size, different input image resolutions for the models are used to change the anomaly size in pixel, including 256, 512, and 1024. Image augmentation is then used to enrich the training dataset, and its influence is evaluated. Similarly to the synthetic anomaly analysis, image and pixel level analysis is performed first, followed by the anomaly level analysis.

6.6.1. Image and Pixel Level Analysis

First, the NSA method is evaluated on the real anomaly dataset with three different synthesis methods to train image generation. The results regarding the four image and pixel metrics are listed in Table 6.1. It is obvious that the model trained on the Perlin dataset outperforms the other two significantly. However, the overall performance is poor with the best pixel F1 score only 0.009. For the following evaluation, only the Perlin synthetic dataset will be used for the NSA method.

Table 6.1. Evaluation of NSA method on real anomaly dataset using different synthesis method.

Synthesis method	Image F1 Score	Image AUROC	Pixel F1 Score	Pixel AUROC
NSA	0.667	0.150	0.001	0.360
Perlin	0.869	0.790	0.009	0.539
Polygon	0.701	0.634	0.003	0.521

The results of the five anomaly detection models with different input image resolutions are listed in Table 6.2. The GANomaly method cannot distinguish the anomalous image from the normal images due to the small size of the anomalies. The NSA method is found to be more difficult to locate anomalies when image size increases to 512. All the other three models achieve a pixel AUROC of more than 0.9. However, the image level detection and the pixel F1 score of STFPM and FastFlow are not satisfying with most pixel F1 scores less than 0.1. An increase in image size to 512 is beneficial for GANomaly, STFPM, and PaDiM, but detrimental to NSA and FastFlow.

The PaDiM model demonstrates clear superiority over the other models, exhibiting complete accuracy in image-level classification and the highest pixel-level performance. The optimal performance is observed when the image size is 512. Compared to the synthetic dataset, this notable discrepancy between the PaDiM and the other models can be attributed to the reduction of the training images from 270 to 45. Additionally, given that the PaDiM model extracts and fits patch-level features at fixed positions for all images, it benefits from a dataset in which all objects are positioned consistently within the image.

PaDiM is the only model that does not require iterative model training, resulting in the fastest training time of only 32 seconds, even for an image size of 1024. It also has the fastest inference speed with 3.35 fps for an image size of 256. The downside is that its model size explodes with image size. While the models of NSA, STFPM, and FastFlow are much smaller.

Table 6.2. Evaluation of anomaly detection models on a real anomaly dataset.

Model	Image Size	Image F1 Score	Image AUROC	Pixel F1 Score	Pixel AUROC	Train Time (s)	Inference FPS	Model Size (MB)
GANomaly	256	0.667	0.240	-	-	265	0.86	2109.0
	512	0.690	0.631	-	-	1775	0.47	8219.7
	1024	-	-	-	-	-	-	-
NSA	256	0.790	0.790	0.009	0.539	776	1.46	43.9
	512	0.857	0.790	0.006	0.611	4557	1.18	43.9
	1024	-	-	-	-	-	-	-
STFPM	256	0.741	0.640	0.051	0.952	178	3.32	35.0
	512	0.800	0.630	0.112	0.986	340	3.06	37.6
	1024	0.833	0.680	0.059	0.968	397	2.11	54.5
FastFlow	256	0.833	0.780	0.033	0.906	76	3.29	86.8
	512	0.714	0.530	0.009	0.789	28	2.85	125.7
	1024	-	-	-	-	-	-	-
PaDiM	256	0.909	0.940	0.151	0.990	6	3.35	177.2
	512	1.000	1.000	0.262	0.994	11	3.00	674.2
	1024	0.952	0.950	0.258	0.997	32	2.11	2664.1

To enrich the training dataset, the image augmentation techniques, such as rotation, translation, brightness, and contrast adjustment, are used to increase the training images from 45 to 180. The results of all the models trained on the augmentation dataset are listed in Table 6.3. Despite increasing the number of training images four times, there is little improvement in most models. In particular, STFPM and FastFlow benefit more from image augmentation, whereas the other models show little or negative improvement. This is due to the fact that simple augmentation methods do not sufficiently enrich the image features.

The PaDiM model remains the best performer. Although the image-level accuracy drops slightly and is no longer 100% accurate, the best pixel F1 score improves significantly from 0.262 to 0.322 when the image size is 1024. With the augmented dataset, the increase in image size improves recognition accuracy, particularly the pixel F1 score, which increases from 0.190 to 0.322. However, the pixel AUROC drops by 0.004 and 0.001 for image size of 256 and 1024.

Table 6.3. Evaluation of anomaly detection models on real anomaly dataset with image augmentation. The red digits indicate decrease and green digits indicate increase.

Model	Image Size	Image F1 Score	Image AUROC	Pixel F1 Score	Pixel AUROC
GANomaly	256	0.667 (0)	0.310 (+0.070)	-	-
	512	0.667 (-0.023)	0.300 (-0.331)	-	-
	1024	-	-	-	-
NSA	256	0.833 (+0.043)	0.770 (-0.020)	0.008 (-0.001)	0.585 (+0.046)
	512	0.869 (+0.012)	0.780 (-0.010)	0.005 (-0.001)	0.460 (-0.151)
	1024	-	-	-	-
STFPM	256	0.769 (+0.028)	0.720 (+0.080)	0.063 (+0.012)	0.971 (+0.019)
	512	0.800 (0)	0.690 (+0.060)	0.118 (+0.006)	0.987 (+0.001)
	1024	0.800 (-0.033)	0.680 (0)	0.066 (+0.007)	0.973 (+0.005)
FastFlow	256	0.869 (+0.036)	0.850 (+0.070)	0.113 (+0.080)	0.908 (+0.002)
	512	0.818 (+0.104)	0.790 (+0.260)	0.058 (+0.049)	0.930 (+0.141)
	1024	-	-	-	-
PaDiM	256	0.909 (0)	0.970 (+0.03)	0.190 (+0.039)	0.986 (-0.004)
	512	0.952 (-0.048)	0.990 (-0.01)	0.242 (-0.020)	0.994 (0)
	1024	0.952 (0)	0.980 (+0.03)	0.322 (+0.064)	0.996 (-0.001)

Overall, similar to the synthetic dataset, all the models struggle to accurately locate the anomalies, leading to low pixel F1 score. Among these models, the PaDiM is the most robust and accurate model. The change in image size changes the anomaly size and the details of the image. The best image size for the PaDiM model is 512. Image augmentation is helpful for the STFPM and FastFlow model but not the others. All the performance of STFPM and FastFlow increased after augmentation, except the image F1 score of STFPM at image size 1024 drops by 0.033. The anomaly level analysis regarding the image size and anomaly characteristics is carried out in the next section.

6.6.2. Anomaly Level Analysis

Due to its significant superiority over the other models, the PaDiM model is selected as the core model for this anomaly detection application. The anomaly score of each anomaly is calculated for the PaDiM model with respect to the anomaly size, brightness, and contrast. As can be seen in Figure 6.23, the anomaly score changes to the upper right when the image size increases. This verifies the preliminary analysis that the anomaly score increases with the anomaly size. The anomaly scores for all anomalies are higher than 0.3 when the image size is 1024. This indicates that all anomalies can be identified if the threshold is 0.3, although this results in an increase in the number of false positives from 35 to 282. Given that the lighting conditions of the engine datasets are identical, the anomaly contrast is directly proportional to the anomaly brightness, resulting in a similar distribution of anomaly-wise performance. The fitted lines from real anomalies and synthetic anomalies in Figure 6.23 show similar trends, indicating the effectiveness of the synthetic anomaly analysis.

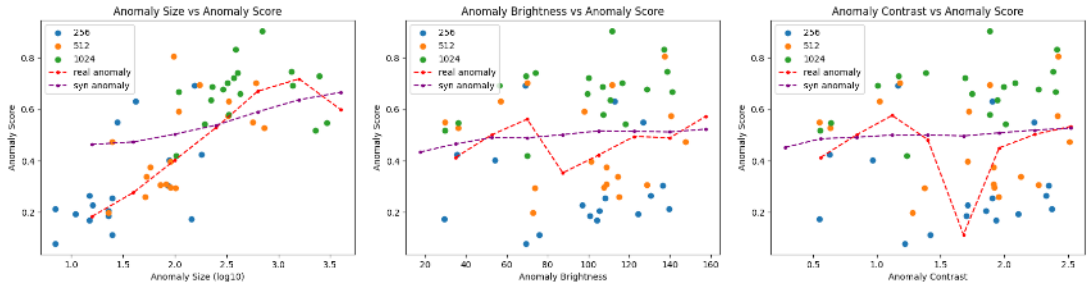


Figure 6.23. The anomaly score with respect to the anomaly size (left), anomaly brightness (middle) and anomaly contrast (right). The fitted lines of anomaly score from synthetic anomalies and real anomalies are compared.

The prediction results for a sample image with different image sizes are shown in Figure 6.24. The original image contains four anomalies with different size and brightness. More anomalies are detected when the image size increases. The only anomaly detected when the image size is 256, in the upper middle of the image, has the highest brightness. However, the largest anomaly shows a relatively high anomaly score in the anomaly map and cannot be fully detected due to its low brightness and low contrast with the surrounding area. The smallest anomaly near the largest anomaly cannot be detected for any image size. It is observed that the larger the image size, the more false positives detected by the model, especially at the background.

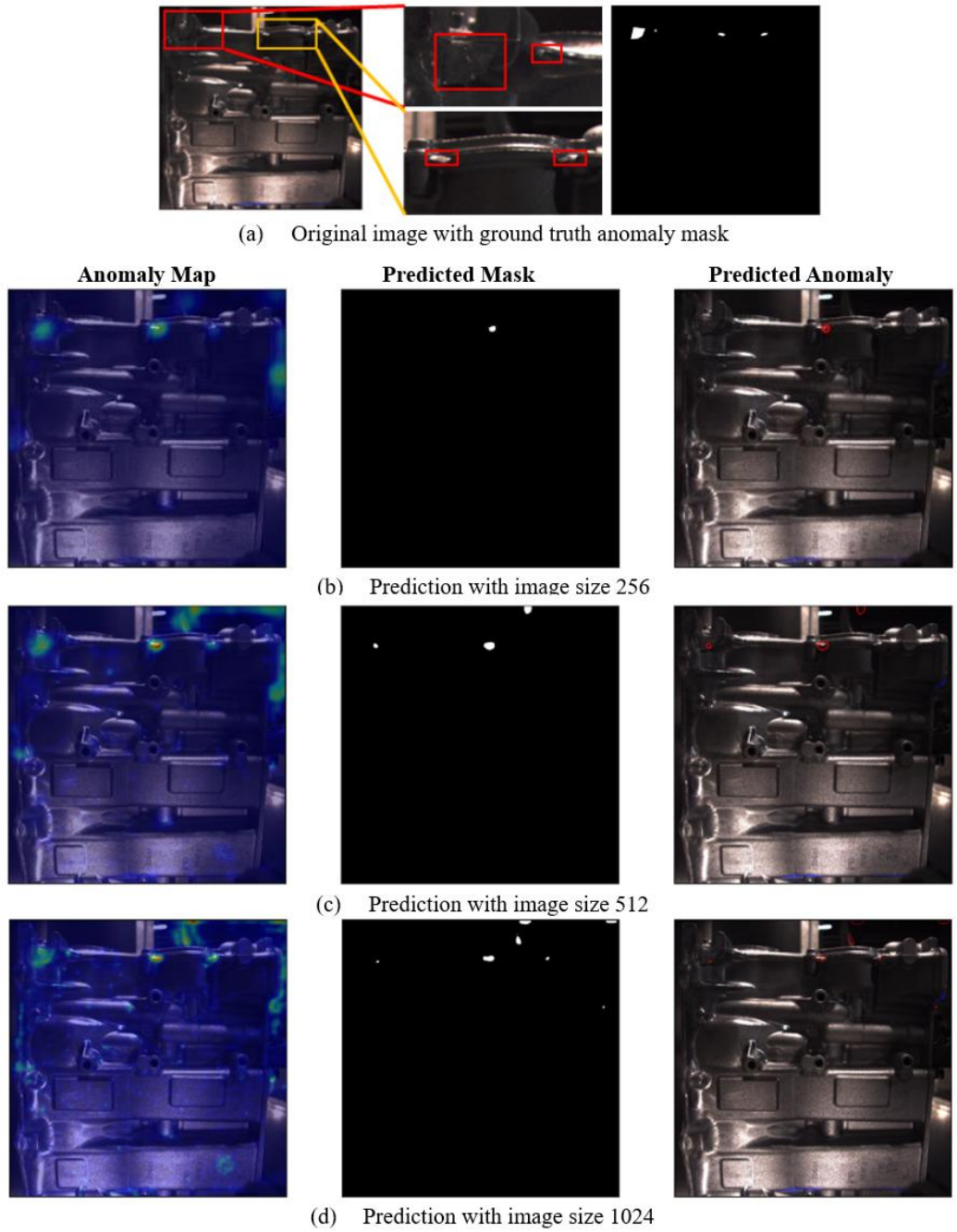


Figure 6.24. PaDiM prediction results on a sample image with different image sizes.

6.7. Model Optimisation

The accuracies of general-task models are limited when applied directly to this real-world application. For this specific application, certain observations can help enhance the performance during the anomaly detection process:

1) The feature extractor used for all models is ResNet-18 for concept proofing. This can be replaced by more powerful feature extractors.

2) Many false positives are detected in the background area which could be eliminated by background removal.
3) For this complicated part, the level of variations is different in different image part. A global anomaly score threshold is not the optimised solution.

Based on the above observations, some improvements can be made to the anomaly detection process, including the ViT feature extractor, background removal, and image tiling.

6.7.1. Feature Extractor

The feature extractor used in previous experiments is ResNet-18, which can be replaced by more powerful feature extractor, the Vision Transformer (ViT) [330]. The comparison of using different feature extractors is listed in Table 6.4. The image size is set to 1024. For ViT model sets, the final feature layer with 256 dimension is used. The ViT model sets have much more model parameters and better representation ability than the ResNet. The PaDiM achieves 100% image level accuracy when using ViT-large model. The pixel F1 score increases significantly from 0.258 to 0.386 when using the ViT-huge model.

Table 6.4. Comparison of using different feature extractors for the PaDiM model.

Feature Extractor	Parameters (M)	Image F1	Image AUROC	Pixel F1	Pixel AUROC
ResNet-18	11	0.952	0.950	0.258	0.997
ViT-base	86	0.909	0.930	0.294	0.995
ViT-large	307	1.000	1.000	0.282	0.994
ViT-huge	632	0.952	0.990	0.386	0.995

6.7.2. Background Removal

Since the engine is placed in the same place in the image, the background can be removed using the Segment Anything Model (SAM) [331] with a rough bounding box prompt. The extraction result and anomaly map prediction are shown in Figure 6.25. The original image and the ground truth are the same as in Figure 6.24. The false positives in the background have been eliminated. The image F1 score, AUROC, pixel F1 score, AUROC are 0.909, 0.940, 0.431, and 0.997, respectively.

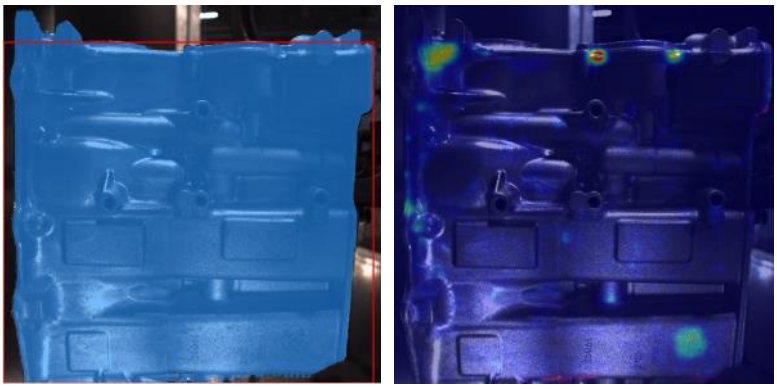


Figure 6.25. Background removal result using SAM and bounding box prompt for image with anomaly (left) and corresponding anomaly map (right).

6.7.3. Image Tiling

By image tiling, each image is divided into 3x3 image patches to form 9 sub-datasets. The image size for each patch is 1024. Each dataset is trained separately to have its own threshold. Also, the anomaly size is also enlarged at each image patch. The prediction results for image patches that contain anomalies are shown in Figure 6.26. The object mask from the SAM is used to filter the background. The prediction quality has been greatly improved, with all four anomalies identified with only one false positive. The average image F1 score, AUROC, and pixel F1 score, AUROC are 1.000, 1.000, 0.662, 0.996, respectively. Both the image and the pixel level accuracy have been dramatically improved.

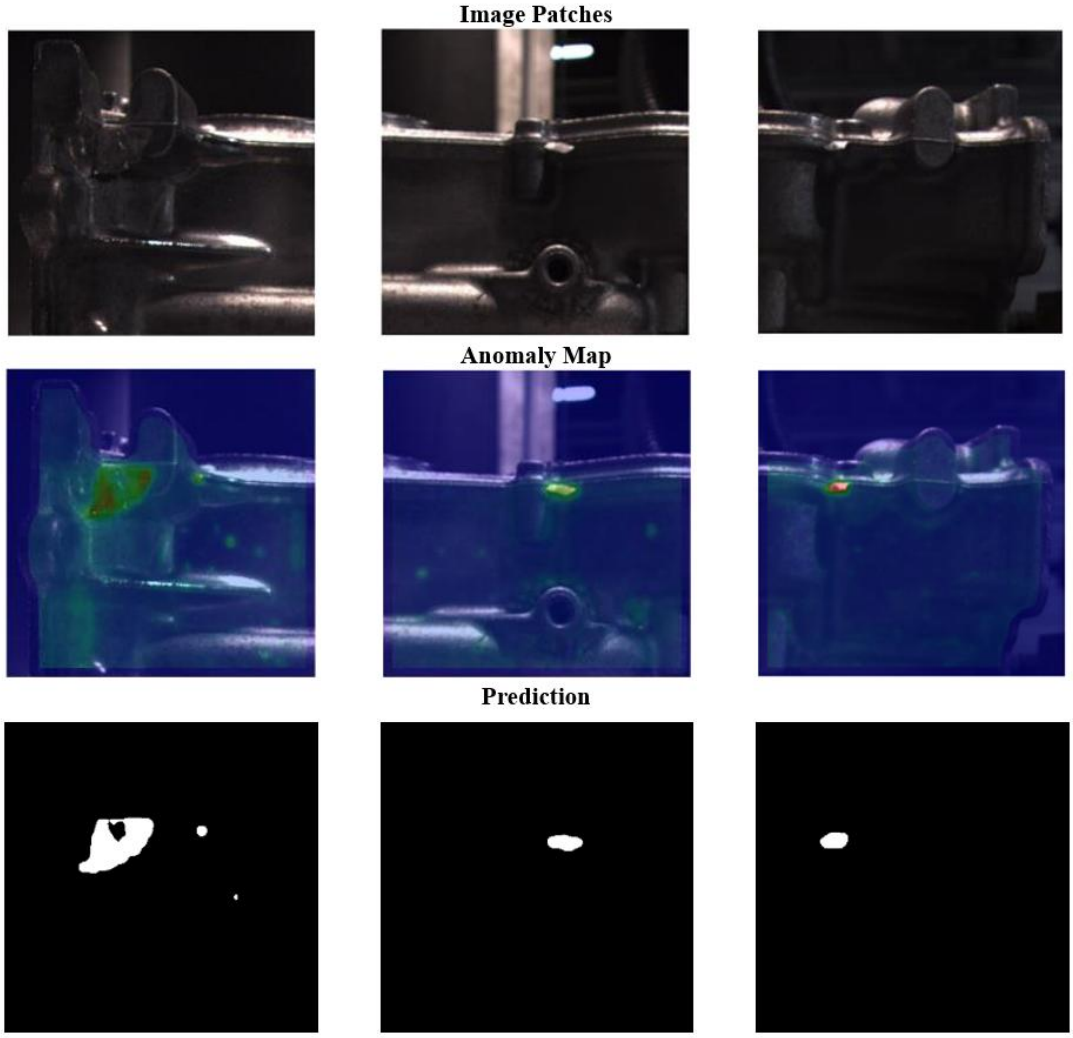


Figure 6.26. Prediction results of image patches containing anomalies by image tiling.

Although anomaly detection has achieved image level, there are still some false positives in normal and abnormal images when the threshold is set to achieve the best F1 score, as shown in Figure 6.27. The anomaly score is high across the entire image patch due to the change in lighting condition. Most of the detected false anomalies are at the edges of the engine with high brightness. Some very small shining points on the engine are also identified as anomalies.

Overall, by a set of improvements, including better feature extractor, background removal, and image tiling, the anomaly detection accuracy using the PaDiM model has been significantly improved at both the image level and the pixel level. All anomalies can be successfully identified with the cost of false positives in 7 images.

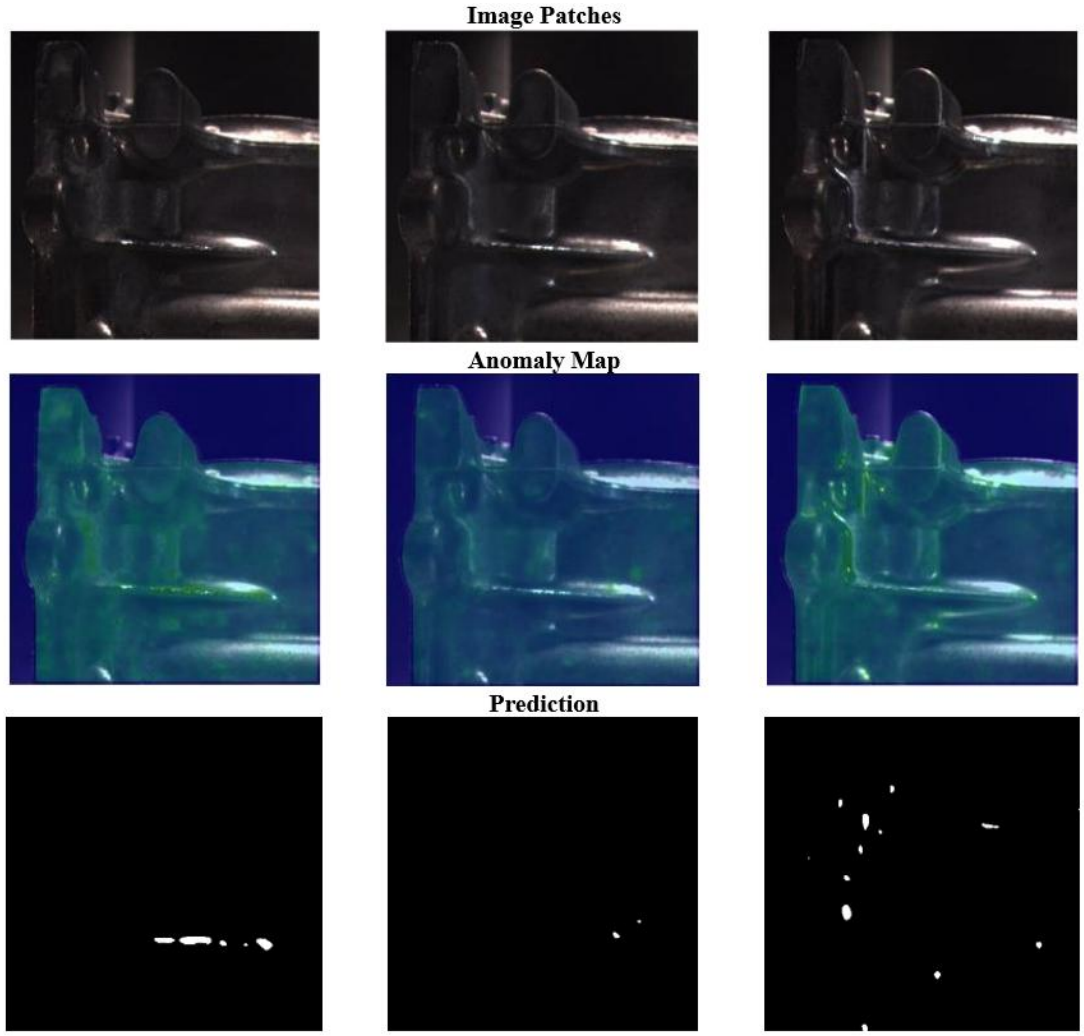


Figure 6.27. Sample false positive predictions for good images using the image tilling method.

Overall, by a set of improvements, including better feature extractor, background removal, and image tiling, the anomaly detection accuracy using PaDiM model has been significantly improved at both image level and pixel level, as listed in Table 6.5. All the anomalies can be successfully identified with the cost of false positives in 7 images.

Table 6.5. Model improvements for real anomaly dataset. The red digits indicate decrease and green digits indicate increase.

Improvement	Image F1 Score	Image AUROC	Pixel F1 Score	Pixel AUROC
Original	0.952	0.950	0.258	0.997
ViT-huge	0.952 (0)	0.990 (+0.040)	0.386 (+0.128)	0.995 (-0.002)
Background Removal	0.909 (-0.043)	0.940 (-0.010)	0.431 (+0.173)	0.997 (0)
Image Tiling	1.000 (+0.048)	1.000 (+0.050)	0.662 (+0.404)	0.996 (-0.001)

6.8. Summary

To design an anomaly detection system for quality monitoring of engine production, this case study follows the proposed algorithm selection and optimisation process. It first presents a

comparative study involving different anomaly synthesis methods, anomaly characteristics, and anomaly detection techniques. Based on the findings from the synthetic dataset, specific improvements were made to the real anomaly dataset to achieve zero false negative prediction. Specifically, this study successfully established two datasets for engine parts, containing both synthetic and real anomalies. The synthetic dataset was generated using different image collection configurations and three different anomaly synthesis methods. A comprehensive comparative analysis was conducted using five anomaly detection methods, evaluating image, pixel, and anomaly-level detection accuracy across the three synthesis methods, three anomaly characteristics, and five detection models. Ultimately, the best-performing model from this analysis was selected and optimised for the real anomaly dataset, achieving zero false negative.

The study highlights the effectiveness of the methodology in guiding the systematic development and optimisation of AVI systems. The case study demonstrates the importance of structured dataset creation as an essential step of the design methodology. By establishing two datasets with both synthetic and real anomalies, the study highlights how the dataset's characteristics significantly impact model performance and how synthetic data can support model development in the absence of sufficient real anomaly samples.

For algorithm selection and comparative study, the study adopts a comparative approach to evaluate five state-of-the-art anomaly detection algorithms, reflecting the design methodology's emphasis on method benchmarking and selection. This process provides comprehensive insights into the strengths and limitations of different methods under varying anomaly characteristics and image acquisition conditions, which is a critical aspect of AVI system design.

The study further validates the importance of optimisation as an iterative process in the proposed design methodology. By fine-tuning hyperparameters and applying post-processing techniques, the selected model achieves zero false negatives and minimal false positives, demonstrating how optimisation significantly enhances detection performance.

7. The Fourth Case Study: 3D-Level Spacecraft Anomaly Detection

This chapter reports the fourth as well as the last case study of a vision-based anomaly detection system for modular and reconfigurable spacecraft. Central to this case study is the development of a 3D image capture system for dataset establishment and development of a reconfigurable algorithm pipeline. The algorithm selection and optimisation process is also applied, presenting a distinct extension of the proposed comprehensive design methodology by addressing the unique challenges associated with 3D-level anomaly detection. Specifically, three distinct image capture systems were developed, consisting of two sets of stereo cameras and one structured light camera. The software system was developed to be reconfigurable. The backbone of anomaly detection is formed by 3D reconstruction and registration algorithms. Two datasets are used for algorithm selection, including a small satellite model and an engineering feature block. The selected algorithm is optimised on the modular spacecraft dataset. The final system is evaluated through a series of anomaly detection tasks, including surface, interface, and reconfiguration anomaly detection.

7.1. Introduction

In space exploration and satellite technology, the concept of modular and reconfigurable spacecraft is a paradigm shift. Traditionally, missions entail launching singular, fixed-purpose satellites specifically designed for predetermined tasks. However, the limitations inherent in this approach, such as high costs, long development times, and inflexibility to adapt to changing mission requirements, have spurred the development of modular and reconfigurable spacecraft [332]. Modular and reconfigurable spacecraft are a novel class of space vehicles that offer a dynamic and versatile approach to space missions. They depart from the conventional "one-size-fits-all" approach by allowing for the assembly of spacecraft using standardised, interchangeable components, or modules, akin to building blocks. These modules can be configured and reconfigured in orbit, allowing a single spacecraft to undertake various missions, adapt to unexpected challenges, or extend its operational lifespan [333].

Vision systems are a crucial aspect of the spacecraft system, potentially allowing for the automatic monitoring and control of spacecraft and robots in the expansive realm of outer space. These advanced systems possess the ability to obtain and transmit valuable spatial and geometric data from camera images, which can then be used by a robotic planner. This planner

directs the robot in its regular duties and, importantly, allows instantaneous reactions to unexpected occurrences, enabling fast adaptation of the robotic operational conduct to prevent probable collisions [334].

This chapter outlines the development of the vision system for the modular and reconfigurable spacecraft. The system objectives and requirements are identified during the task clarification stage. Then the concept of vision system architecture and software platform is generated. The embodiment design focuses on the dataset establishment and algorithm selection and optimisation. Three datasets are collected using the stereo camera, two of which are used as validation dataset for algorithm selection, and the other one are used as test dataset for algorithm improvement. A software platform is developed to enable reconfigurable algorithm nodes and chains. The final system is verified and validated through a series of anomaly detection tasks.

7.2. Task Clarification

7.2.1. System Requirements and Specifications

The specifications derived from the task clarification are summarised as follows:

• A vision inspection system that can detect anomaly of the modular and reconfigurable spacecraft, including surface, interface, and reconfiguration anomaly.
• A 3D reconstruction of the satellite could be produced from the observation by a camera, with a maximum surface average error of 1cm.
• The 3D pose estimation of the spacecraft modules could be computed based on the image stream obtained by the camera mounted on the walking manipulator, with a mean error not larger than 1 cm translation and 5° rotation.
• Initial detection of a module does not require real-time capabilities but needs to be achieved to meet the above accuracy.

7.2.2. Related Works

7.2.2.1. Vision-based Spacecraft Inspection System

The use of computer vision techniques for spacecraft inspection can be traced back to the implementation of the Space Vision System by the Canadian Space Agency [335, 336]. This vision system was utilised on the space shuttle Columbia flight in November 1996 for element berthing and space station assembly operations. Another vision system used for space missions is the Advanced Video Guidance Sensor. This system demonstrated its efficacy through its use

on the Defence Advanced Research Projects Agency Orbital Express mission in 2007 for spacecraft guidance [337]. A vision-based navigation system was developed by Alonso et al. [338] for formation flying of spacecraft. The position sensing diode was used together with an optical sensor to estimate spacecraft state using the Lyapunov and contraction mapping approach. However, there is a limited availability of vision systems tailored specifically for the inspection of modular and reconfigurable spacecraft.

Typically, there are three types of vision system for in space applications regarding the image capture system, including monocular camera, stereo camera, and laser camera. The monocular-based vision system uses a single camera to capture RGB image of the spacecraft for inspection. To retrieve 3D information of the targets, the monocular camera is always supported by other additional components, such as markers [339] or structured light [340]. Through stereo matching, the 3D point cloud can be obtained for the targets together with the RGB image using stereo cameras [341]. Laser-based methods include using laser range finder [342] for distance measurement or LIDAR target 3D reconstruction from LIDAR [343]. With regard to the utilisation of additional light sources, vision systems can be classified into passive and active systems. In space applications, LEDs and structured lighting represent two common light sources. LEDs are employed not only to enhance the target's brightness but also, at times, their arrangement can serve as patterns or markers for monocular vision systems. Structured light, on the other hand, finds utility in both monocular and stereo vision applications. For laser-based vison systems, it usually does require an extra light source.

7.2.2.2. 3D Reconstruction and Localisation

During the past decade, several works have addressed the problem of 3D reconstruction from RGB data. Usually, the problem is reduced to a camera localisation step, followed by a fusion process of the 3D data in one unique environmental representation. The developed 3D reconstruction solution evolved in quality from variant of the iterative closest point algorithm with the capacity of handling sparse data or dense data in a small area to hierarchical optimisation of the reconstruction over large areas.

The iterative closest point (ICP) algorithm is an iterative process that refines the estimation of pose between two point clouds. Pamerleau proposed an ICP comparison framework for multiple ICP variations [344]. It is reported that the ICP using point-to-plane distance was 77% faster than ICP using point-to-point distance. In addition, the point-to-plane pose error for the ICP was 60-90% lower than the point-to-point pose error. In [345], Henry et al. proposed an algorithm that uses the Surfel representation, which refers to 3D points taken from a depth

image and enriched with additional descriptive information, such as surface normal. The reconstruction algorithm initiates by identifying SIFT features in the stereo images. These features are then matched with a RANSAC estimation followed by a point-to-plane ICP refinement. This idea was extended in [346] with a new energy function defined as the weighted sum of a geometric term based on the consolidated point-to-plane distance and a photo consistency term based on the colour difference.

Instead of reinventing the wheel, some existing software can be used with maturely developed implementation of 3D processing pipeline. InFuse is a software equipped with a collection of data fusion algorithms. It contains a 3D reconstruction pipeline composed by a point cloud reconstruction step from a pair of stereo images, an ICP matching step between consecutive point clouds and point cloud fusion step by weighted average [347]. Based on the InFuse library, this study will focus on the development of a vision inspection system for modular spacecraft.

7.3. Concept Design

The vision processing subsystem is composed of a vision processing platform and a fixed camera. The processing platform is a standard computer with dedicated memory and processors and input/output peripherals such as a monitor, a keyboard, and a mouse. The processing platform will host an Ubuntu Operating System and will execute the vision processing algorithms. The camera will be installed on a stand at a pre-determined location beside the satellite mock. The camera field of view will cover the upper surface of the satellite, granting clear visibility to the non-occluded spacecraft modules. The space illumination conditions will be simulated by setting the demonstrator in a dark room with a black background. The lighting conditions will approximate those that could be experienced in space. A dark background will be placed behind the demonstrator to absorb light and minimise features detected by the vision system. Three adjustable servicer LED lamps will be located on the side of the camera and will illuminate the visible side of the demonstrator.

To detect anomalies in spacecraft, a point cloud is generated through 3D reconstruction using either a stereo camera or a structured light camera. This allows for image capture of the the evaluation of a the determination of the poses of all observed spacecraft through feature matching, which could then be used for CAD model registration. By calculating the distance between the point cloud and the registered CAD model, anomalies can be identified according to a predefined threshold.

7.4. Dataset Establishment

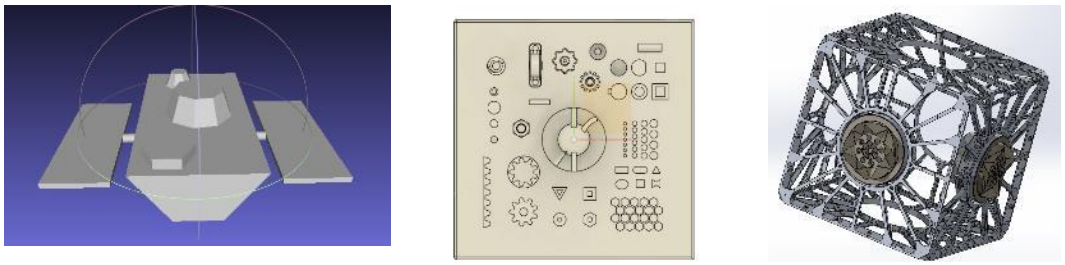
To validate the vision inspection system, three datasets are collected using different 3D printed models, including the small satellite, the engineering feature block, and modular satellite. The small satellite dataset serves as a feasibility study of 3D reconstruction and localisation from the stereo camera pair. The engineering feature block dataset focuses on comparison of different 3D localisation methods. The modular satellite dataset contains the target object of the MOSAR system and is used for final system validation by anomalies detection tasks. The small satellite dataset was collected by my colleagues Dr. Mutian and Dr. Alessandro Bianco, while the other two datasets were collected by myself.

7.4.1. Hardware Configuration

The hardware for the vision subsystem was selected for the lab environment demonstrator. The control unit was selected as small size workstation with an a good computing power. Two completely same set of cameras and lens are selected for stereo vision with high resolution imaging ability. Additional two PTZ cameras with adjustable focal length were used for selective zooming function. Furthermore, an RGB-D camera using binocular structured light technology was bought for better point cloud reconstruction. These components are listed in **Error! Reference source not found.** with detailed specifications.

7.4.2. Data Collection

To validate the vision inspection system, three datasets are collected using different models. The first model is a small satellite with typical satellite shape that has one main body and two solar panels, captured by HikVision stereo cameras. The second model is a block designed to contain typical engineering features, captured by both HikVision stereo cameras and Revopoint RGB-D camera. The third model is the finalised modular spacecraft, captured by all the three camera sets of, including another DataVideo PTZ stereo cameras. The CADs of the three modules are shown in Figure 7.1.



(a) (b) (c)

Figure 7.1. CAD of the three modules for vision system, including (a) small satellite model, (b) engineering feature block, and (c) modular spacecraft.

7.4.2.1. Small Satellite

For the image capture of this model, the vision system consists of a SCARA robot, a pair of Hikvision stereo cameras, a set of 4 controllable lightings in a dark room, and a conveyor belt which could move the model, as shown in Figure 7.2. Stereo cameras were attached to the SCARA robot and instructed to rotate in a circular motion with the model in the centre. In total, 21 pairs of stereo images were captured with robot arms rotated every 4.5^{o} in the range of 90^{o}. A sample image pair is shown in Figure 7.3. The colour different was caused by the white balance discrepancy from product inconsistence of HikVision camera. This dataset aims to test the stereo reconstruction and pose estimation function of the vision system.

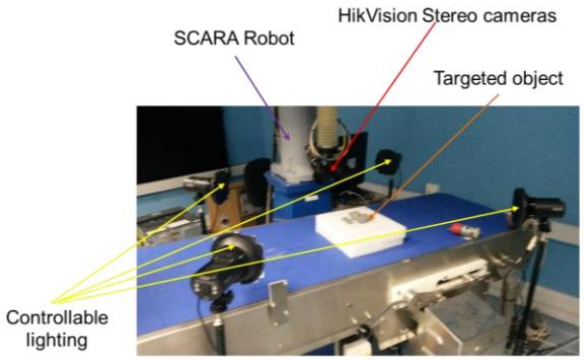


Figure 7.2. Vision system setup for small satellite model.

Figure 7.3. Sample image of small satellite captured by HikVision stereo cameras.

7.4.2.2. Engineering Feature Block

An engineering feature block was designed containing most of the typical engineering features, such as circle, triangle, polygon, holes of different depth,, thread, etc. It was 3D printed to test the 3D reconstruction and registration performance of the vision system. For 3D printing precision evaluation, measurements were carried out with calliper, height gage, and dial indicator, giving a maximum error of 0.271 mm. Therefore, the 3D printed model is accurate enough and the CAD model is used as the point cloud ground truth.

Both the stereo camera and RGB-D camera were used for image capture from different angle,

distance, exposure time, and illumination conditions. The system set up is shown in Figure 7.4. The EFB was placed on a rotating plate so that it can be rotated with a minimum angle interval of 5°. The stereo cameras and RGB-D camera were placed at two sides of the model. Three LED lamps were placed around the model to create different illumination directions and intensity.

Figure 7.4. Vision system setup for EFB model image capture.

The two cameras were placed at two difference distances from the EFB. According to the field of view, the HikVision camera was placed about 50 cm and 70 cm away, while the Revopoint camera was placed about 30 cm and 50 cm away. Three view angles of the camera with respect to the EFB were used, about 30, 45, and 60 degrees. Four sets illumination were configured with different numbers of LEDs turned on. For each illumination configuration, three exposure times of HikVision stereo cameras were used based on the measured exposure value (EV), as listed in Table 7.1. The exposure time for RGB camera of Revopoint RGB-D was set as auto-exposure. Due to the usage of structured light, the depth camera is not sensitive to the illumination, therefore three different exposure values were used but the same for all the illumination conditions. The EFB was rotated for 360 ° with angle interval of 10°. A sample image pair from HikVision camera with distance 50 cm, view angle 60, and illumination level 0, exposure time 300 ms is shown in Figure 7.5. Another sample image from Revopoint RGB-D camera with distance 30 cm, view angle 45, illumination level 2, depth camera exposure time 10 ms, is shown in Figure 7.6. This dataset aims to test the multi-view stereo reconstruction, RGB-D image quality, line segment based monocular image pose estimation, and point cloud based pose estimation.

Table 7.1. Illumination configuration for EFB image capture

Level	Description	Lux	Exposure value	HikVision Exposure (ms)	Revopoint Exposure (ms)
0	Room Lights	211	6.4	200, 400, 800	10, 20, 30

1	One back LED	320	7.0	125, 250, 500	10, 20, 30
2	Two lateral LEDs	342	7.1	125, 250, 500	10, 20, 30
3	Three LEDs	685	8.1	62, 125, 250	10, 20, 30

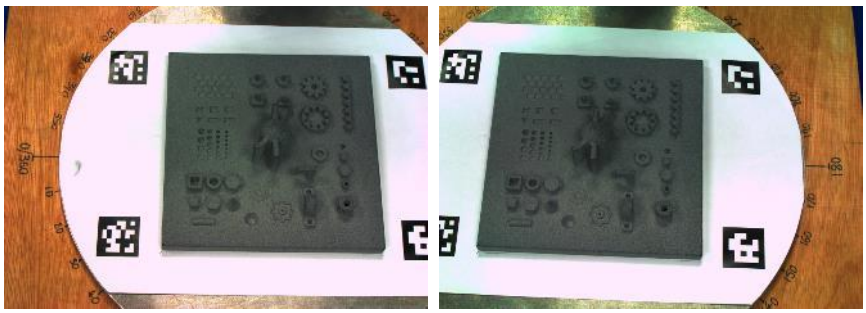


Figure 7.5. Sample EFB image of stereo image pair from HikVision Camera.

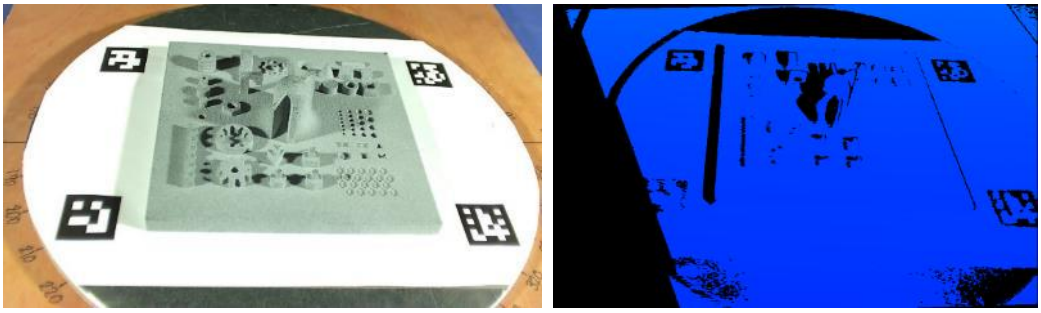


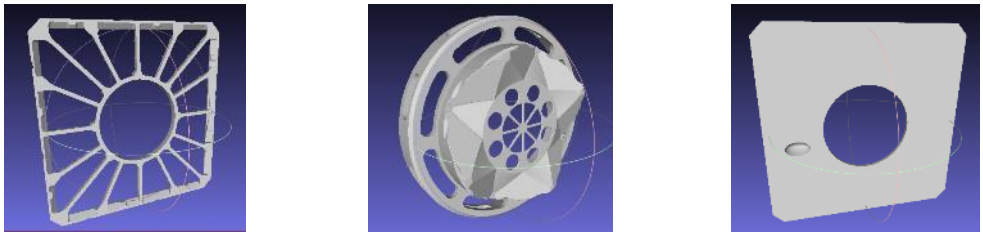
Figure 7.6. Sample EFB image of RGB and depth from Revopoint camera.

7.4.2.3. Modular Satellite

After finalising the modular satellite design, it is 3D printed for testing in one tenth of the original size. The components of one modular satellite including frame, interface and cover, which are separately printed, as shown in Figure 7.7. The system setup is shown in Figure 7.8. The three camera sets were used for data collection, including HikVision stereo cameras, Revopoint RGB-D camera, and DataVideo PTZ stereo cameras. The robot arm is Universal Robot UR10 to simulate the walking manipulator. The module body and interface were printed separately so that the interface anomaly can be easily created. There are two focal lengths configured for the PTC stereo cameras targeting close and far view of the satellite, with the camera intrinsic and extrinsic parameters calibrated separately.

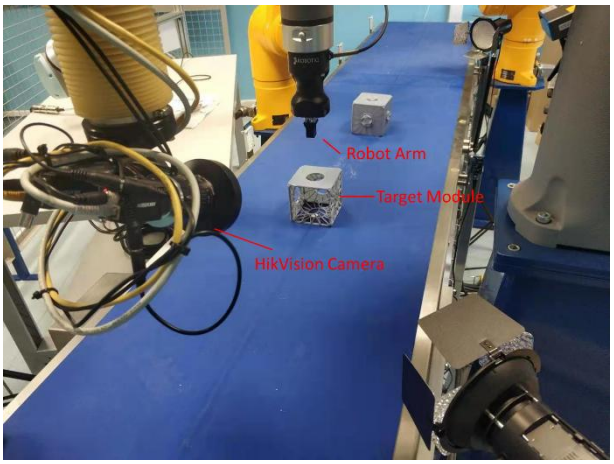
To simulate the surface anomaly of the module body, two different kinds of anomaly were designed in CAD and 3D printed, including one extrude and one recess with 5 mm depth both in ellipse shape with major axis 8 cm and minor axis 4 cm. The interface anomaly is designed as protrusion or recession with respect to the cover plate. For reconfiguration anomaly test, the experiments were simply designed with one to three modules presented in the view and test if the modules can be detected with correct number and location. The sample images with surface

anomaly taken by the three sets of cameras are shown in Figure 7.9, Figure 7.10, Figure 7.11, respectively. This dataset aims to test whether the designed vision inspection system can successfully detect the surface, interface, and reconfiguration anomalies.

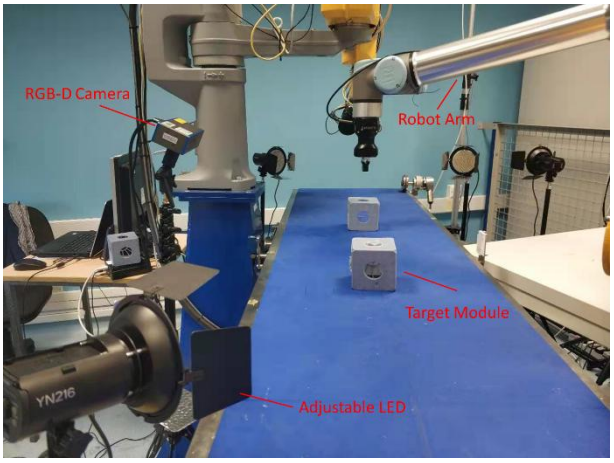


(a) (b) (c)

Figure 7.7. CAD model of modular satellite components including (a) satellite frame, (b) interface, and (c) cover with anomaly.



(a) HikVision Stereo Cameras



(b) Revopoint RGB-D Camera

(c) DataVideo PTZ Stereo Cameras

Figure 7.8. Vision system setup for modular satellite model image capture using three sets of cameras.

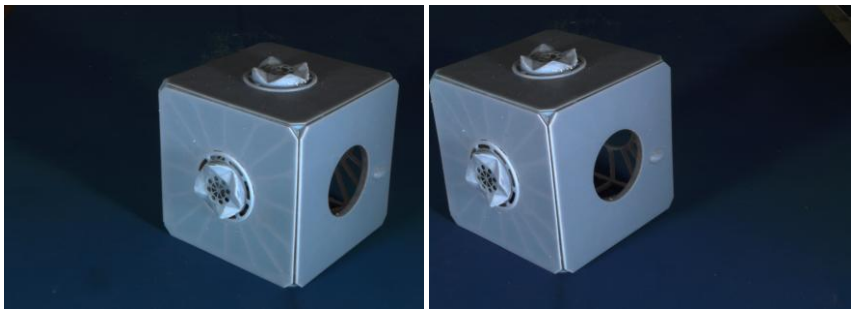


Figure 7.9. Sample modular satellite image with surface anomaly from HikVision stereo cameras.

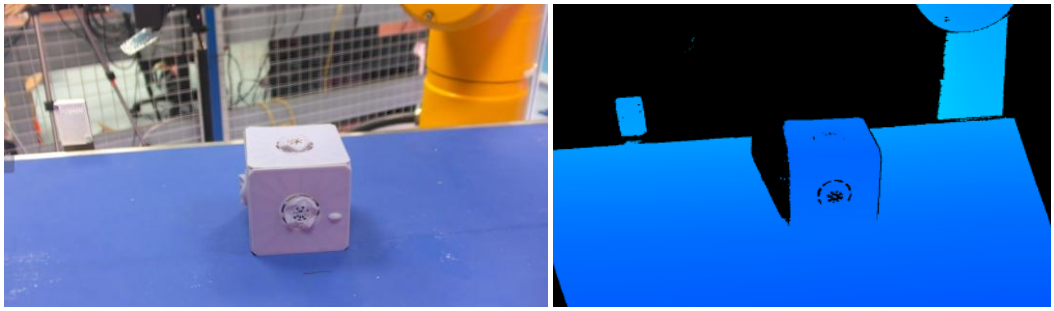


Figure 7.10. Sample modular satellite image with surface anomaly from Revopoint RGB-D camera.

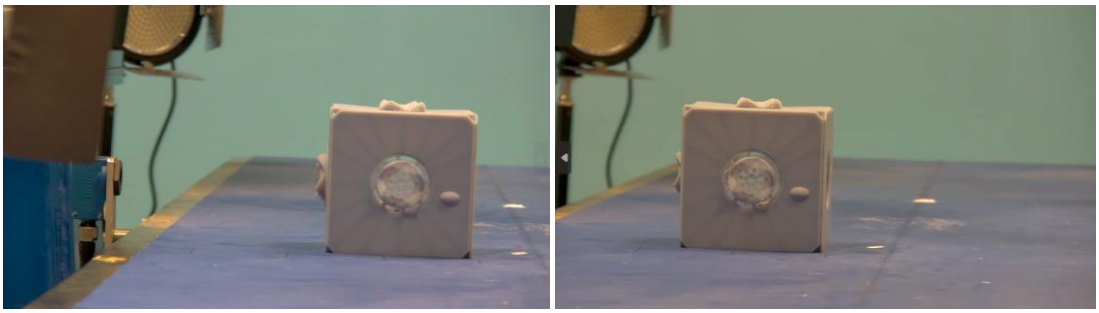


Figure 7.11. Sample modular satellite image with surface anomaly from DataVideo PTC stereo camera.

7.5. Object Pose Estimation

7.5.1. Algorithms for Object Pose Estimation

Pose estimation is the last step before the CAD model compared to the reconstructed point cloud. Therefore, it is the most important part of the vision subsystem, as it directly influences the accuracy of the three inspection requirements including surface, interface, and reconfiguration anomaly detection. For stereo and RGB-D based pose estimation, the image feature extraction is critical for model recognition and localisation. In this study, the image features are used to obtain an initial model pose estimation. The point cloud based method is then used to refine the pose.

7.5.1.1. Image Feature based Pose Estimation

The image features of a model in RGB image can be used for initial pose estimation by matching the image features with 3D model features. Typically used image features for the 2D-3D correspondence include a key point, line segment, and curve.

A. Key Point Detection

The key point is an important image feature that is widely used for object recognition and localisation. Generally, there are two kinds of key point in computer vision, that is, traditional feature points which are category-agnostic and generated from a small image patch, such as SIFT, and semantic feature points of specific objects and generated from AI model. The AI-generated key points enable direct mapping to 3D models due to their inclusion of semantic information, unlike traditional feature points which only provide low-level feature. In practice, AI-based key points are more robust than traditional feature points if the model is well trained with sufficient data. Among the AI-based key points detector, including regression, heatmap, segmentation, and bounding box, the heatmap-based key points detection method is used in this study because it has better interpretability. The output of heatmap-based detector is a set of heatmaps with each showing the probability density of the key point at each pixel location. The position with the highest probability is chosen as the position of that key point.

For this study of satellite model pose estimation, an AI model is trained to detect the predefined key-points of the satellite model in one RGB image. The key-points are then used for initial pose estimation by solving the PnP problem. The AI model uses a one-layer hourglass CNN architecture [348], with output of 19 heatmaps representing the positions of 19 key-points. The input image size is 256x256 and the output heatmap size is 96x96. The architecture of the network is shown in Figure 7.12.

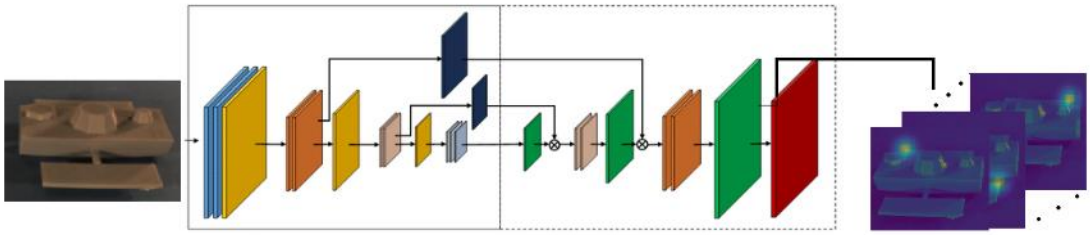


Figure 7.12. Hourglass model for satellite key points detection.

For supervision, the ground truth of each point is also a heatmap as a 2D Gaussian with standard deviation of 1 pixel and cantered on the point location. The Mean-Squared Error (MSE) loss is applied for back propagation.

B. Edge based Image Feature

Edge detection in computer vision is a fundamental process that aims to identify and extract linear patterns or edges within an image, providing valuable information for tasks like object recognition and scene analysis. This is particularly significant for industrial components that include many fabricated features, such as straight line, curve, and circle. Unlike other algorithms that use edges from the edge detector as an image feature, the line segment and curve are more meaningful and contain less noises. Therefore, the line segment and the curve together comprise the edge feature of industrial parts for pose estimation.

In this study, the Line Segment Detector (LSD) algorithm [349] is used. The LSD algorithm is a widely recognised method for detecting line segments in images. It is known for its efficiency and robustness in identifying line segments of varying lengths and orientations within images, making it a valuable tool in computer image processing applications. Compared to the Hough transform, the LSD has several advantages. It takes into grey scale image as input instead of binary image. The algorithm operates in linear time and does not require parameter tuning, making it highly advantageous for space applications where lighting conditions may vary significantly.

The curve is another significant edge feature of 2D images that can be used for pose estimation through edge point matching. Usually, detection of curves is the first step of circle and ellipse detection. Here, the curve detection is the same as the one in circle and ellipse detection but used directly for pose estimation. In this study, the edge drawing method was used for edge extraction [135], which introduced the Helmholtz principle to eliminate invalid detections. Then the curve detection follows the same procedure presented in [350] for the circle detection. The procedure is summarised as follows:

1. Detect edge segments by the edge drawing method.
2. Extend and merge edges to fill the gap of several pixels.
3. Convert the remaining edge segments into line segments using Edlines in [351].
4. Detect curves by combining line segments with turning angle as a criterion.
5. Validate the curves by the Helmholtz principle.

The Helmholtz principle states that whenever some large deviation from randomness occurs, a structure is perceived. Based on this, the ε -meaningful event is used to capture the meaningful edge segments. We say that an event that is ε - meaningful if the expectation of the number of occurrences of this event is less than ε under the *a contrario* random assumption. When $\varepsilon \leq 1$, we simply say that the event is meaningful. The number of false alarms (NFA) is defined as:

$$NFA = N_{conf} \cdot B(n,k,p) \leq \varepsilon \tag{7-1}$$

$$B(n,k,p) = \sum_{i=k}^{n} \binom{n}{i} p^i (1-p)^{n-i} \tag{7-2}$$

where N_{conf} is the number of different possible configurations one could have for the searched object, p is the precision of the detection, usually probability of a Bernoulli distribution. For the curve, N_{conf} is defined as:

$$N_{conf} = \sum_{i} \frac{l_i(l_i - 1)}{2} \tag{7-3}$$

where l_i is the length of the segment. For the edge of engineering features, the angle change should be consistent. According to the Helmholtz principle, an edge segment is meaningful if the probability of this segment happening is rather low, with the assumption that each angle change is independent and randomly distributed. The segment validation problem can be reformulated with p being 0.125. Instead of focusing on the gradient distribution along the segment, this method focuses on the sub-segment angle distribution. This assumes that the extracted segments are valid and looks for segments aligned with engineering features.

The edge features, comprising of both line segments and curves, can be used for pose estimation in two distinct ways. The first method is for estimating the pose of monocular images. The 3D model edges are back projected to the image, and the extracted edge features are matched with the 3D model edges. Once the point-to-point correspondence is established between image edges and 3D edges, the PnP problem can be solved to estimate the pose of the 3D model in camera coordinates. The other method is for RGB-D image pose estimation. The

2D edge features can be mapped directly to 3D points in the RGB-D image. Then the 3D points can be used either for 3D feature fitting and matching, or directly for point cloud matching with the 3D model. In this study, both methods will be used.

J. Perspective-n-Points

The Perspective-n-Point (PnP) problem is a fundamental challenge in computer vision and robotics, particularly in the context of pose estimation. It involves determining the position and orientation (pose) of a camera relative to a set of 3D points in a scene, given their corresponding 2D projections in the camera's image. Mathematically, given a set of 3D points in world coordinate system, represented as $\{X_i \mid i = 1, 2, ..., n.\}$, and their corresponding 2D points in image coordinate system, represented as $\{ x_i \mid i = 1, 2, ..., n.\}$, the camera's intrinsic parameters including the focal length (f_x, f_y), principal point (c_x, c_y), and lens distortion parameters, The goal is to find the camera's extrinsic parameters, which specify its pose in the world coordinate system, including rotation matrix $\boldsymbol{R}$, which describes the camera's orientation, and the translation vector $\boldsymbol{t}$, which represents the camera's position. Generally, the PnP problem can be defined as follows.

$$f(X_i, x_i, \boldsymbol{R}, \boldsymbol{t}) = 0 \tag{7-4}$$

where f represents the mapping between 3D world points and their corresponding 2D image projections, taking into account the camera's intrinsic parameters, as well as the rotation matrix $\boldsymbol{R}$ and the translation vector $\boldsymbol{t}$. Solving the PnP problem typically involves finding the values of $\boldsymbol{R}$ and $\boldsymbol{t}$ that minimise the error in the mapping f, often expressed as the sum of squared reprojection errors:

$$error = \sum_i \|x_i - \boldsymbol{K}[\boldsymbol{R}|\boldsymbol{t}]X_i\| \tag{7-5}$$

where $\boldsymbol{K}$ is the camera intrinsic matrix and

$$\boldsymbol{K}[\boldsymbol{R}|\boldsymbol{t}] = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \tag{7-6}$$

Various algorithms, such as the Direct Linear Transform (DLT) or iterative methods like Levenberg-Marquardt, can be employed to find the optimal values for $\boldsymbol{R}$ and $\boldsymbol{t}$ that best align the 3D points with their corresponding 2D projections, thereby solving the PnP problem and estimating the camera's pose.

In practice, the problem of PnP can be sensitive to outliers, which refer to incorrect or noisy

correspondences between 3D points and their 2D projections. The Random Sample Consensus (RANSAC) algorithm performs excellently in addressing these outliers by choosing subsets of correspondences, fitting pose models, and identifying the consensus set of inliers that concurs best agree with the model. By doing so, the algorithm considerably enhances the precision and accuracy of pose estimation, even in the existence of noisy data. RANSAC is an iterative method that is used to calculate the parameters of the mathematical model from a data set containing outliers. The method deliberately excludes outliers, ensuring that they do not affect the estimated values. The number of iterations N can be determined from:

$$N = \frac{\log(1-p)}{\log(1-w^m)} \tag{7-7}$$

where m is the lowest number of points that uniquely define the model, w is the percentage of inliers, and p is the desired probability that the RANSAC algorithm will produce at least one useful result in this run.

Apart from using in the PnP problem, the RANSAC can also be used in any matching algorithms where outliers may appear, such as line segment 3D matching, model registration, etc.

7.5.1.2. Point Cloud based Pose Estimation

The point cloud based pose estimation matches the reconstructed point cloud and the 3D model in camera coordinate system. The object point cloud together with the RGB image can be obtained by stereo reconstruction or directly from the RGB-D camera, both of which provide the pixels to 3D points correspondence. The iterative closest point (ICP) method is the most popular way for point cloud matching because of its easy implementation and effectiveness.

A. Stereo Reconstruction

Stereo reconstruction is a computer vision technique used to reconstruct the 3D structure of a scene or objects by analysing the disparities between corresponding points in stereo images captured by two cameras. This process is based on the principles of triangulation and stereo geometry. The epipolar geometry describes the geometric relationship between two cameras and their corresponding points in stereo images. The fundamental matrix $\boldsymbol{F}$ represents this relationship and is used to restrict the search for the corresponding points. The epipolar constraint can be expressed as follows:

$$(x')^T \boldsymbol{F} x = 0 \tag{7-8}$$

where x and x' are the homogeneous image coordinates of a point in the left and right images, respectively, and $\boldsymbol{F}$ is the fundamental matrix. The disparity map represents the pixel-wise horizontal shift (disparity) between corresponding points in the left and right images. According to the epipolar constraint, the disparities can be calculated simply by the difference of the horizontal coordinates of the corresponding points in the left and right images. Once the disparity map is obtained, the depth values for each pixel can be calculated by:

$$Z = f\frac{B}{disparity} \tag{7-9}$$

where f is the focal length of the camera lens and B is the baseline distance between the two cameras (the distance between their optical centres).

By calculating the disparities and applying the depth estimation equation, the 3D representation of the scene or objects can be created from the stereo images. The calculation of disparities through stereo matching is the most important part of stereo reconstruction since it directly influences the accuracy of the depth calculation. Stereo matching quality depends on the matching algorithm and image quality. With smooth and reflective surfaces that contain large area of similar pixel values, the stereo matching always failed to estimate the disparity. The structure light with predefined pattern can be used to add features to the images and significantly improve the stereo matching quality.

B. ICP Registration

Once the point cloud of the object is obtained, the pose of the object can be estimated by the point cloud matching between the reconstructed point cloud C_r and point cloud sampled from the mesh model C_m. The goal is to find the transformation $\boldsymbol{T}$ composed of a 3D rotation matrix $\boldsymbol{R}$ and a 3D translation vector $\boldsymbol{t}$ that minimises the sum of squared distances between corresponding points in C_r and the transformed C_m, expressed by:

$$distance = \|p_r - (\boldsymbol{R}p_m + \boldsymbol{T})\|^2 \tag{7-10}$$

where p_r and p_m are points in C_r and C_m.

The Iterative Closest Point (ICP) algorithm is a widely used technique in computer vision and robotics to align two sets of 3D points through an iterative optimisation process. It is commonly employed for tasks such as point cloud registration, 3D object alignment, and mapping. The primary goal of ICP is to find the transformation (translation and rotation) that best aligns one set of points to another. The procedure of ICP following the step:

1) Initialisation: Start with an initial guess for the transformation $\boldsymbol{T}$ (often an identity matrix).

2) Point Correspondence: Find the nearest neighbours in the the set C_m for each point in set C_r. This establishes point correspondences.

3) Alignment Estimation: Estimate the transformation (R, t) that minimises the sum of squared distances between the corresponding points. Rotation Estimation: Compute the optimal rotation matrix $\boldsymbol{R}$ using Singular Value Decomposition (SVD) of the cross-covariance matrix between the corresponding points. Translation Estimation: Calculate the translation vector $\boldsymbol{t}$ as the mean of the differences between the corresponding points.

4) Update Transformation: Update the transformation $\boldsymbol{T}$ with the estimated ($\boldsymbol{R}$, $\boldsymbol{t}$).

5) Termination Criterion: Check if a termination criterion is met (e.g., small change in transformation or a maximum number of iterations). If not, repeat steps 2-4.

6) Final Transformation: The final transformation $\boldsymbol{T}$ aligns the set C_m with the set C_r.

ICP iteratively refines the transformation until convergence. Variants of ICP exist to address these challenges, such as Robust ICP (which handles outliers) and Point-to-Plane ICP (which considers the surface normal for more accurate alignment). It is important to note that ICP can be sensitive to the choice of initial alignment, data noise, and the presence of outliers. In this study, initial pose estimation was used through image feature based point estimation methods.

7.5.2. Small Satellite Model Pose Estimation

The dataset of small satellite model was used to test the stereo reconstruction and pose estimation performance of the vision system. The stereo matching algorithm parameters tuning strategy was firstly investigated. Then the reconstructed point cloud from different view angles was merged to obtain more point clouds. Finally, the key point based pose estimation was used to obtain the initial pose for final ICP registration.

7.5.2.1. Stereo Reconstruction

The stereo matching algorithm used is from the OpenCV library, which is a semi-global block matching (SGBM) algorithm modified from [352]. There are 11 parameters in the tuning of the algorithm for the generation of disparity maps. It would be time-consuming and inefficient to use a trial-and-error approach to find the optimal parameter set for the MOSAR vision image capture work. The Taguchi method was used to reduce the number of experiments [353]. Taguchi uses orthogonal arrays to stipulate the way to conduct a minimum number of experiments, which give sufficient information of the effect of a parameter upon a performance

parameter. The alternative is the factorial method, which involves altering all parameters, resulting in unnecessary testing of all possible combinations of parameters. Using Taguchi for the 11 parameters (3 levels for 10 parameters and 1 parameter at 2 levels) results in a L54 (2^1 plus 3^10) orthogonal array, that is 54 tests compared to 118098 tests for the full factorial method. Software tools such as Minitab can be used to produce the Taguchi test. The values used for the 11 parameters in the test are listed in Table 7.2.

Table 7.2. Values for parameter tuning of stereo matching algorithm

MinDisp			**NumDisp**			**BlockSize**		
Level 1	Level 2	Level 3	Level 1	Level 2	Level 3	Level 1	Level 2	Level 3
450	500	550	22	24	26	3	7	9
SpecRange			**SpecSize**			**MaxDiff**		
Level 1	Level 2	Level 3	Level 1	Level 2	Level 3	Level 1	Level 2	Level 3
1	3	5	75	125	160	4000	5000	6000
UniRatio			**Smooth1**			**Smooth2**		
Level 1	Level 2	Level 3	Level 1	Level 2	Level 3	Level 1	Level 2	Level 3
8	10	12	100	200	300	400	800	1200
Cap			**TwoPass**					
Level 1	Level 2	Level 3	Level 1	Level 2				
300	500	700	0	1				

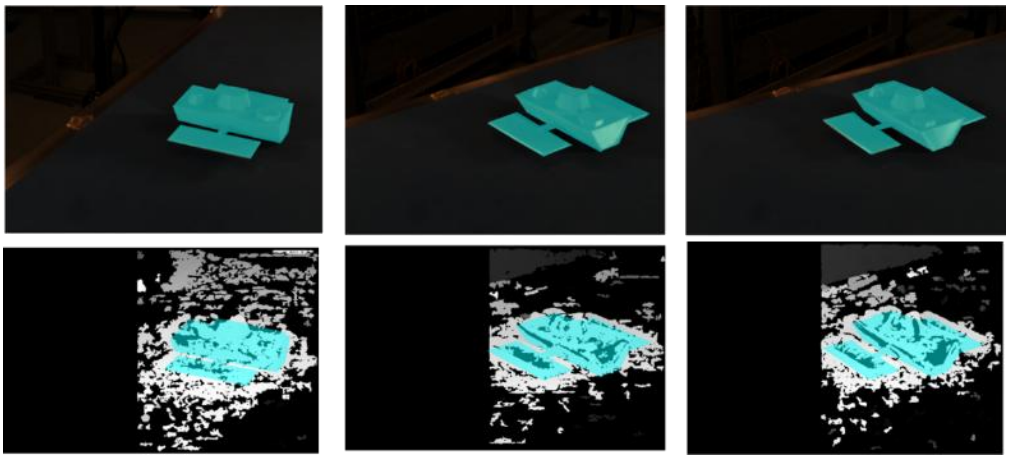


Figure 7.13. Comparison of reconstruction results of three parameter combinations.

The criterion to evaluate the performance of a combination of the parameters is the number of points reconstructed for the small satellite. The density in region of interest (DROI) was used which calculates the ratio of matched pixels and total pixels for the small satellite model. The pixels of the small satellite model were extracted by colour segmentation. The comparison of reconstruction results of three parameter combinations is shown in Figure 7.13. The blue mask shows the segmentation results of the small satellite. The DROIs from left to right are 0.80, 0.77, and 0.74, respectively. According to DROI, the best parameter combination of

parameters can be determined.

7.5.2.2. Point Cloud Merge

The point cloud from one point of view contains many noises and holes. To improve the quality of the point cloud, the point cloud from different points of view can be merged as a new point cloud. In both the lab environment and the space application, the rough relative position of the target with respect to the camera can be obtained from the robot control module. Here, the point cloud from another viewpoint is first transformed to the first viewpoint using the rough relative position, and then refined by the ICP algorithm.

To estimate the initial transformation, the transformation matrix is estimated through the geometrical layout of the vision system and the camera rotation during image capture. The coordinate system of the vision system for the small satellite model is shown in Figure 7.14. The transformation from the camera and object can be estimated. For different viewpoints, a new transformation matrix can be generated with the knowledge that the camera was rotated 4.5° centred at the object. After the initial transform estimation, the point clouds can be merged using the ICP algorithm with the first as the target. An example of point cloud merge result is shown Figure 7.15 using the point cloud from the 4th, 5th, and 6th viewpoints. It can be seen that the number of points has increased and many holes have been filled.

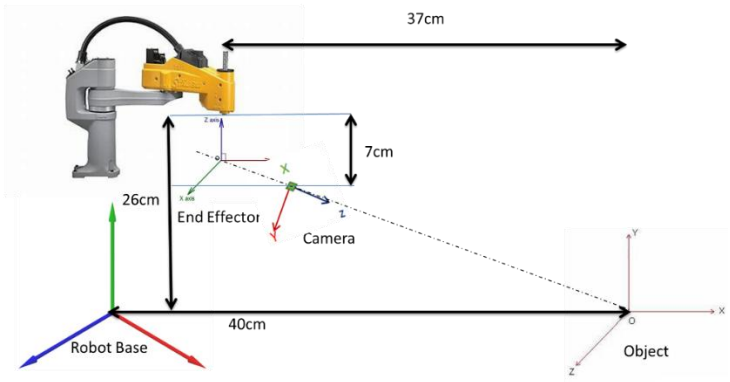


Figure 7.14. The coordinate system of the vision system for the small satellite model.

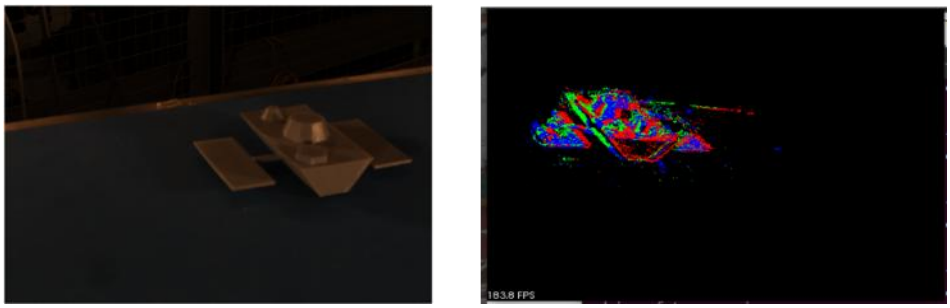


Figure 7.15. Point cloud merge result using the 4th, 5th, and 6th viewpoints.

7.5.2.3. Key Point Detection and Pose Estimation

AI-based key point detection requires a labelled dataset for model training. Therefore, the dataset of satellite images is labelled with the ground truth key points positions. The open-source software LabelMe is used for labelling the points. For the small satellite model, 19 points are selected, mainly the vertices of the satellite model, as shown in Figure 7.16. The labelled images are fed to the hourglass model for training of key point heatmap generation. The input is the undistorted and rectified single RGB image with size 256x256. Then 19 heatmaps with size 96x96 are generated from the AI model. The mean square error (MSE) loss function is used to calculate the gradients. Adam optimiser is used for gradient update with initial learning rate of 0.001 and weight decay 1e-4. The batch size is set to 8 and 1000 epochs are used in total. The final loss after training is 1e-4.

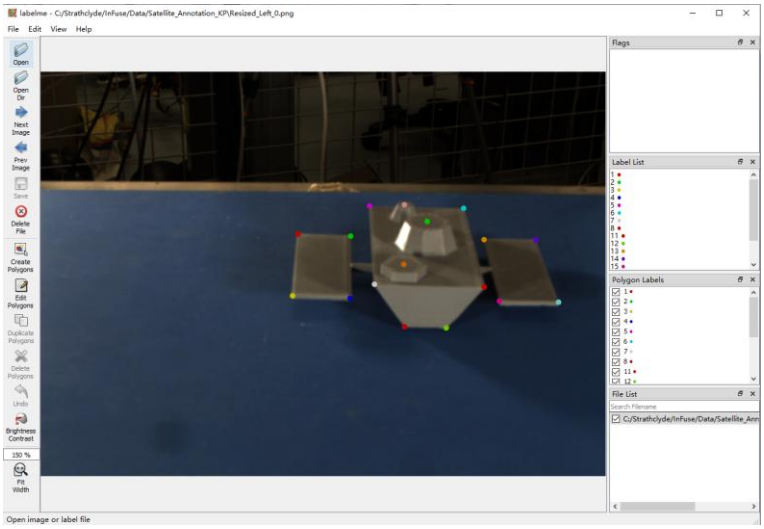


Figure 7.16. Key points selected for the small satellite model.

Once the model training is completed, the key points of the satellite model can be predicted for each image. The corresponding 3D coordinates of these points on the CAD model are also known. With the camera intrinsic matrix calibrated, by solving the projective-n-points (PnP) problem, rotation and translation vectors can be obtained for initial pose estimation. The 2D projective points of transformed 3D points are compared with the predicted points, revealing the precision of PnP, as shown in Figure 7.17. The green points are the predicted key points using AI model. The red points are the projected 3D vertices points after transformation using the rotation and translation vectors obtained from solving the PnP problem. It can be observed that the red points align well with the predefined key points.

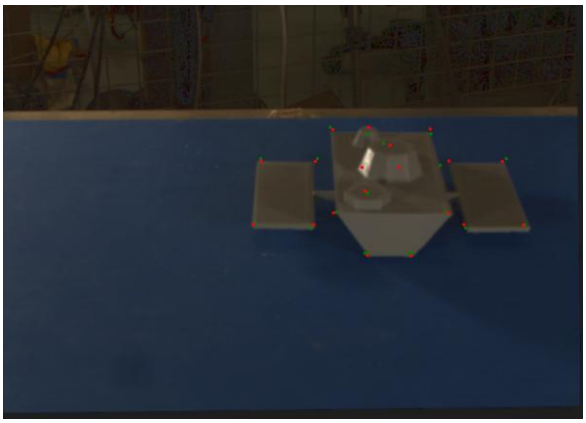


Figure 7.17. Comparison of key point prediction (green dots) with the ground truth (red dots).

Once the initial pose is estimated from PnP, the ICP algorithm can be sued to refine the pose. The satellite model is sampled as a point cloud from the CAD mesh file. This point cloud is first transformed using the vectors from initial pose estimation as shown in Figure 7.18. Then the ICP algorithm is used for point cloud registration, from which a more accurate pose estimation is obtained. The transformed 3D mesh model is compared with the reconstructed point cloud to evaluate the accuracy of pose estimation, as shown in Figure 7.19. The 3D mesh of the satellite model is transformed and displayed together with the reconstructed point cloud from stereo images. Compared to the initial pose estimation, the CAD model using the refined pose aligned much better with the point cloud. The ICP algorithm was also applied without using the initial pose estimation from PnP, as shown in Figure 7.20, showing a rotation error. Overall, the feasibility of stereo-based pose estimation has been verified using the small satellite model.

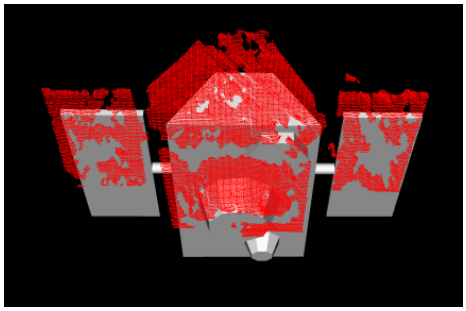


Figure 7.18. The CAD model (grey) is transformed using matrix from PnP and compared with the point cloud (red points).

However, as an initial pose estimation, the accuracy is enough for postprocessing, such as aligning algorithm like iterative closest point (ICP). The below image shows the alignment using the ICP algorithm.

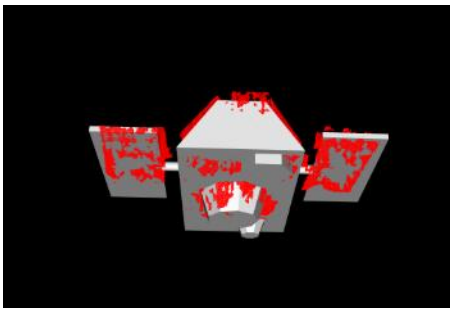


Figure 7.19. The CAD model (grey) is transformed using matrix from ICP with initial pose from PnP and compared with the point cloud (red points).

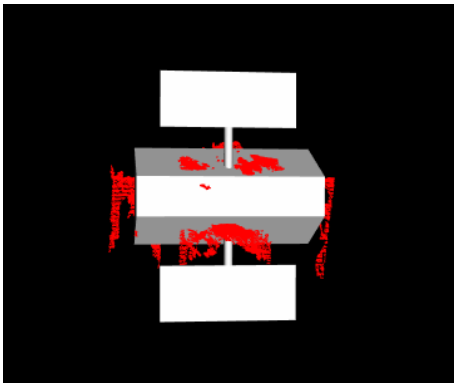


Figure 7.20. The CAD model (grey) is transformed using matrix from ICP without initial pose and compared with the point cloud (red points) showing 90 degree rotation error.

7.5.3. Engineering Feature Block Pose Estimation

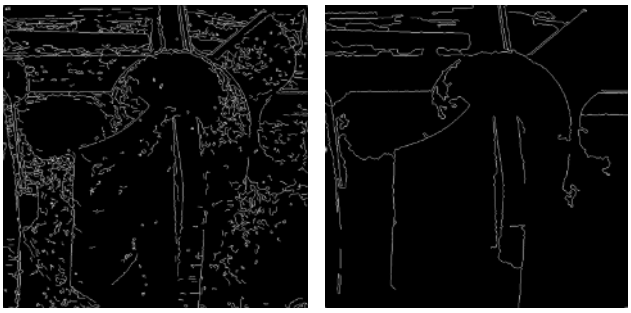
The EFB dataset was collected for general purpose engineering feature recognition, localisation, and defect detection. In this study, the EFB dataset was used for the evaluation of edge feature extraction, monocular image edge-based pose estimation, and defect detection.

7.5.3.1. Edge-based Pose Estimation

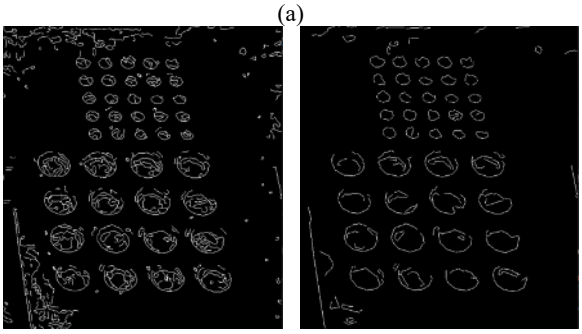
The extraction of edge features, such as lines, circles, and arcs, is critical for recognising parts in engineering. For the edge-based pose estimation, which will be described shortly, the edge pixels are the key part regardless of whether it belongs to a line or a circle. Traditional edge detectors such as Canny can be used directly, but the detection results always contain useless edge segments and noises. The line or circle detector can only find a small part of useful edges. Therefore, in this study, the edges from edge detector are filtered to extract useful edges through the NFA-based false detection control, including line segments and arcs. The result of the edge filter is shown in Figure 7.21 using the EFB dataset. It is obvious that most of the noise from the edge pixels has been filtered out.

For edge-based pose estimation, the ground truth edge is first generated using the CAD model and back projection, as shown in Figure 7.22. This requires an initial pose estimation for the

object similar to the ICP registration, which can be obtained through key-point detection or as prior knowledge from the system setup. Then the edge pixels from the image are matched to the ground truth edges through a search in the direction of the edge normal. After edge matching, the image pixels from the extracted edge can be mapped to the 3D points from the ground truth edge. Finally, the pose can be estimated by solving the PnP problem. The registration result is shown in Figure 7.23 and Figure 7.24. It can be seen that the ground truth edge has been well aligned with the EFB in the image through only a monocular image. The distance between the matched edge points is shown in Figure 7.23 with red colour showing a higher matching error.



(a)



(b)

Figure 7.21. Comparison of edges extracted from EFB before (left) and after (right) NFA-based filter.

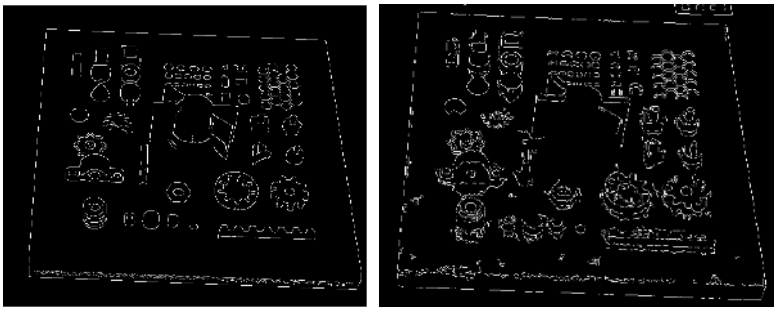


Figure 7.22. The ground truth edge generated from CAD and the extracted and filtered edge from image.

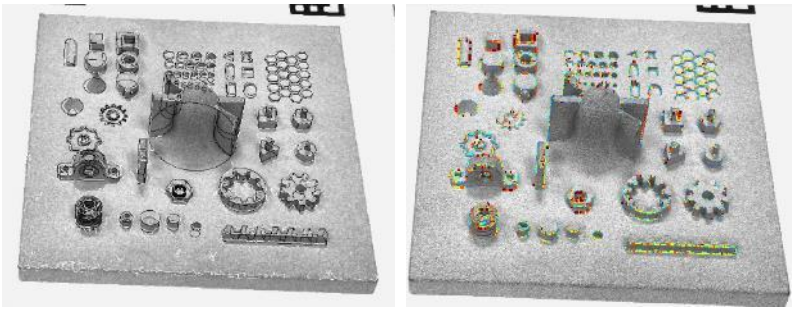


Figure 7.23. Ground truth edge (right) and matching error (right) overlayed on the left camera image after alignment by solving PnP. The red colour means higher matching error.

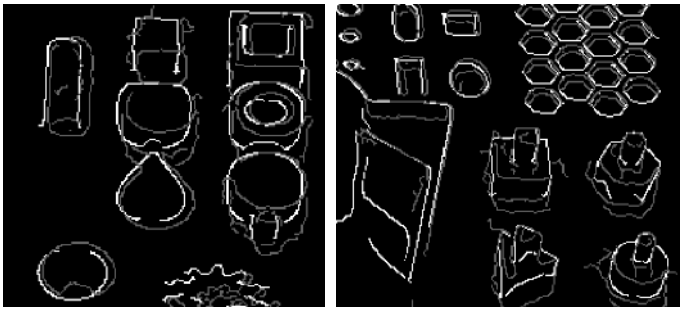


Figure 7.24. Results of edge-based registration for the upper left (left) and lower right (right) parts of EFB.

To quantitatively assess the quality of different alignments, the average model edge to image edge distance is used as a criterion, as listed in Table 7.3. Other two methods are also implemented using 3D edge information. The edge pixels are mapped to the 3D points through stereo reconstruction and registered with the 3D model edges. The SVD method uses the point correspondence from the edge matching and calculates the transformation from 3D image edge points to model edge points. The ICP method simply registers the two edge point cloud. It is shown that using only the monocular image and the PnP method, the model registration accuracy is comparable to the ones with 3D information. The ICP method achieved the best performance. It should be noted that the ICP method used only the 3D edge points of both the reconstruction and the CAD model. The ICP using the complete reconstructed points will have much better accuracy, which will be described in the next section.

Table 7.3. The comparison of edge-based model registration

Method	Edge Distance (px)	Point Cloud Distance (mm)
PnP	1.80	0.7553
SVD	1.78	0.7321
ICP	1.69	0.6132

7.5.3.2. Point Cloud Registration

Stereo images from HikVision stereo cameras were used for point cloud reconstruction. The reconstructed point cloud is shown in Figure 7.25. Unlike the small satellite model, the point cloud quality is higher with fewer holes and noises due to the feature-rich surface of the 3D

printed model through multi-jet fusion. The reconstructed point cloud is registered to the sampled point cloud from CAD model for pose estimation with the pose from edge-based method as initial pose, as shown in Figure 7.26. It can be seen that the reconstructed point cloud is well aligned with the sampled point cloud. The position of each feature part of the EFB is well estimated. The mean distance from the reconstructed point cloud to the CAD model is 0.4209. The point cloud from the RGB-D camera is also shown in Figure 7.25. The mean distance after ICP registration is 0.3861, which gives better point cloud quality.

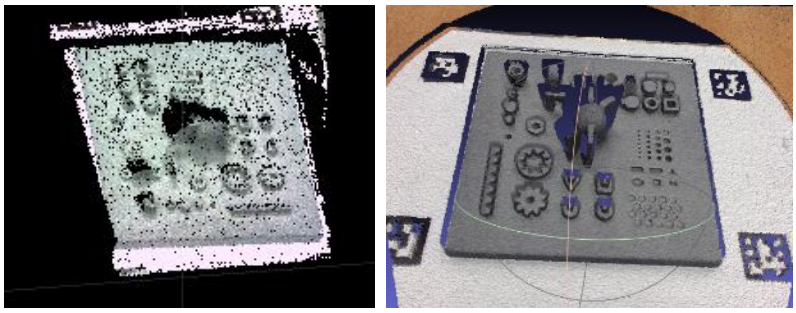


Figure 7.25. The EFB point cloud from the stereo reconstruction (left) and the RGB-D camera (right).

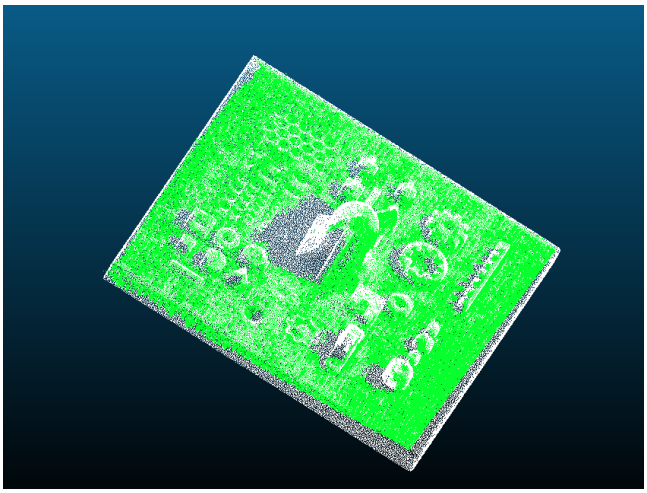


Figure 7.26. Pose estimation of EFB through ICP registration.

7.6. Modular Spacecraft Inspection

7.6.1. Algorithm Selection and Optimisation

From the pose estimation experiments for small satellite model and engineering feature block, it shows the necessity of initial pose estimation and high accuracy of the ICP algorithm. For initial pose estimation, the key-point detection AI model is accurate, but relies on labelled training data and lacks generalisation ability. Edge-based pose estimation is effective and easy to implement. Therefore, for modular spacecraft inspection, the edge-based pose estimation is used for the initial estimation. Then the ICP algorithm is used to refine the pose. For anomaly

detection, the distance between the reconstructed point cloud after registration with the CAD model is calculated. The 3D points with distance higher than the preset anomaly tolerance are regarded as anomaly.

Due to the simple structure of the spacecraft, the edge-based pose estimation can be optimised by simple orthogonal 3D line matching. It also needs to be modified to fit this application given that a multiple number of spacecrafts could be present in one image. The 3D lines already matched will be removed from the matching pool. An additional procedure is required to filter out false detection by visibility check.

The pipeline of spacecraft anomaly detection is defined as: image acquisition, point cloud reconstruction, segment 3D extraction, segment 3D matching, model position detection, model inlier extraction, and module anomaly detection. The workflow is shown in Figure 7.27.

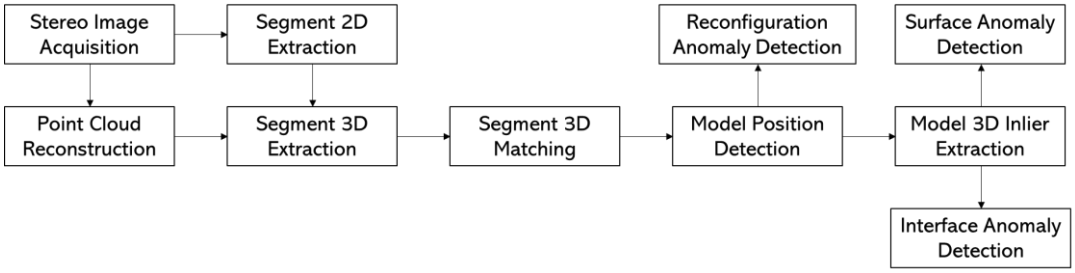


Figure 7.27. Workflow of satellite anomaly detection.

First, the stereo images are acquired from a stereo camera with intrinsic and extrinsic parameters calibrated. Then the stereo images are undistorted and rectified based on the calibration parameters. The stereo matching algorithm is used to calculate the disparities of the stereo image pair and reconstruct the point cloud of the scene. From the RGB image of the left camera, the 2D image features such as the line segment are extracted and mapped to 3D points. The 3D points of the line segments are fitted as 3D line segments. By matching the extracted 3D line segments and model line segments, the model can be detected with a 6 DoF pose. The model inlier point cloud is then extracted from the reconstructed point cloud from the estimated position. The above steps will continue until no more models can be detected. The positions of all the models will be compared with the predefined configuration for reconfiguration anomaly detection. The extracted model inlier will be compared with the model CAD for surface and interface anomaly detection.

The algorithm development focuses on three nodes, i.e., 3D segment extraction, 3D segment matching, and 3D model inlier extraction.

1) 3D Segment Extraction

Edge is one of the most simple but important features for object detection. The algorithms for 2D segment detection have been very mature, but the detection of 3D segments that rely solely on the point cloud is still difficult. In this application, through the RGB-D image, this problem can be solved accurately and efficiently, as shown in Figure 7.28. First, the 2D segment is extracted on the RGB image with the line segment detector (LSD). Then the pixels belonging to the segments are mapped to 3D points through the transformation matrix between the RGB image and the depth image. Finally, the 3D points of the same segment will be fitted into the 3D segment through the RANSAC algorithm.

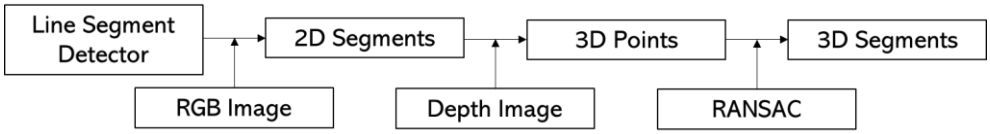


Figure 7.28. Workflow of the segment 3D extraction node.

2) 3D Segment Matching

Once the 3D segments are extracted, the segment matching algorithm can be performed to search for cubes among the segments. It is impossible to search with brute force for all possible transformations, as shown in Figure 7.29. Instead, the repeated RANSAC algorithm was used with some constraints. Initially, all orthogonal segment pairs are found from the 3D segments of the last step and the cube model, forming two sets of segment pairs. Then, two pairs from each set are randomly selected and the transformation matrix from one to the other is calculated. The segments of the cube model are then transformed using the matrix and compared with the extracted segments to calculate the overlapping length. Once the overlapping length of more than two segments is larger than a threshold and the overall length is larger than another threshold, this match will be treated as a success. This process will be iterated multiple times until no success match is found. This step always tries to find all the possible models even though some of which are invalid and will be filtered out in the next step.

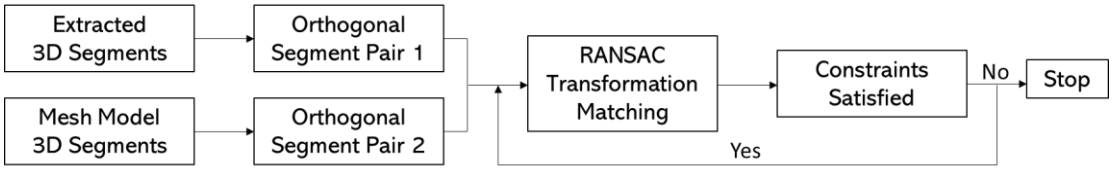


Figure 7.29. Workflow of 3D segment matching node.

3) Model Inlier Extraction

After the segment matching process, the inlier points from the point cloud will be extracted. Simple use of the distance from point to model mesh will introduce many outliers. Therefore,

loop of reprojection and ICP alignment combination was used to determine the inliers, as shown in Figure 7.30. The transformed mesh model after the matching process is reprojected to the depth map, and the points inside the reprojected meshes are treated as inlier candidates. Then the ICP algorithm is used to align the inlier candidates with the sample point cloud of the mesh model to get a new transformation of the model. Repetition of the reprojection and ICP alignment twice can obtain a good result of inliers. To filter out the invalid model from the last step, two thresholds are introduced to imply constraint to the detected model, one for the total number of the inliers and one for the number of inliers of each visible faces of the model. The last step is to calculate all the distances of each inlier point to the model, which will be used for anomaly detection by the region growing method.

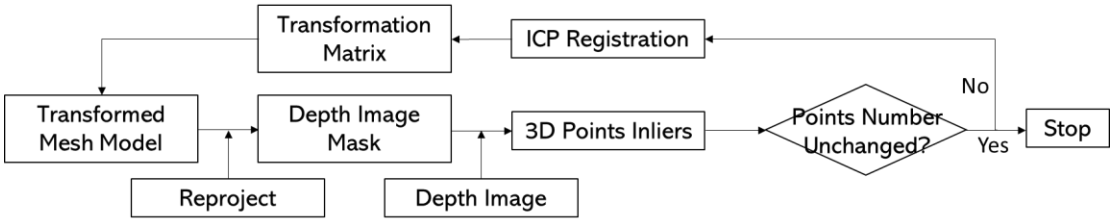


Figure 7.30. Workflow of model inlier extraction node.

7.6.2. Test Procedures

For surface anomaly detection, three scenarios were designed, including i) one undamaged module, ii) one module in the view with 5 mm surface protrusion, iii) one module in the view with 5 mm surface recess, and iv) one module undamaged and another module with 5 mm surface protrusion.

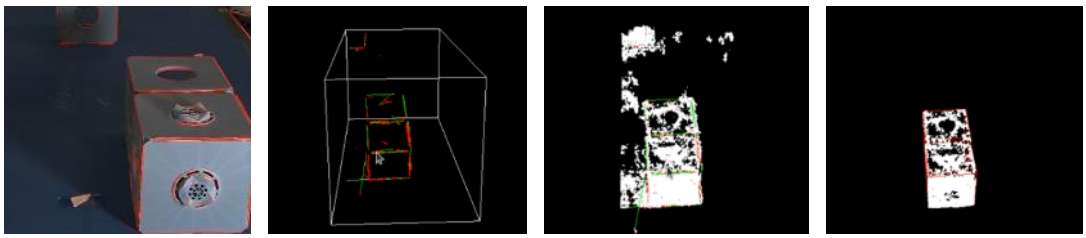
For interface anomaly detection, three scenarios were designed including i) one undamaged module, ii) one module in the view with 5 mm interface protrusion, iii) one module in the view with 5 mm interface recess, and iv) one module undamaged and another module with 5 mm surface protrusion and 5 mm interface recess.

For reconfiguration anomaly detection, two scenarios were designed including i) two modules in the view and ii) three modules in the view.

7.6.3. Stereo Camera based Pose Estimation

For the modular spacecraft pose estimation, the line segments were used as image features for the initial pose estimation. Compared with key points, the line segment is easy to extract without the need for deep learning model training. Since the modular spacecraft is in cubic shape, the 12 3D line segments of a cube of the same size are used for segment matching. The

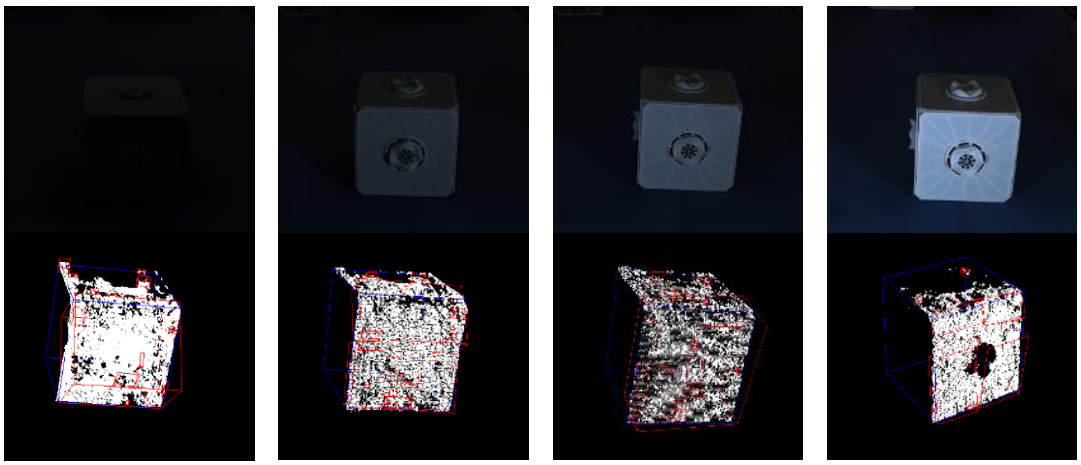
pixels from extracted 2D line segments are mapped to 3D points and fitted as line segments by RANSAC. These extracted 3D line segments are then compared with the model line segments for spacecraft detection. The output results through the module detection pipeline are shown in Figure 7.31. It can be seen that there are many noises of line segments leading to multiple false positive detection of modules. Therefore, several constraints are applied to filter out false positive detection, such as orthogonal line segment detection, module collision detection, inlier points threshold, etc. The final detection results will show the position of each modular spacecraft, which can be used for reconfiguration anomaly detection.



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

Figure 7.31. Module detection pipeline for stereo images including (a) image line segment detection, (b) 3D line segment fitting, (c) 3D line segment matching, and (d) model inlier extraction.

First, the influence of illumination is compared in Figure 7.32. The images were pre-processed before point cloud reconstruction, such as contrast adjustment and histogram balance. It can be seen that the quality of the point cloud with medium light is the best among the results. In all the cases, the module pose has been correctly estimated.



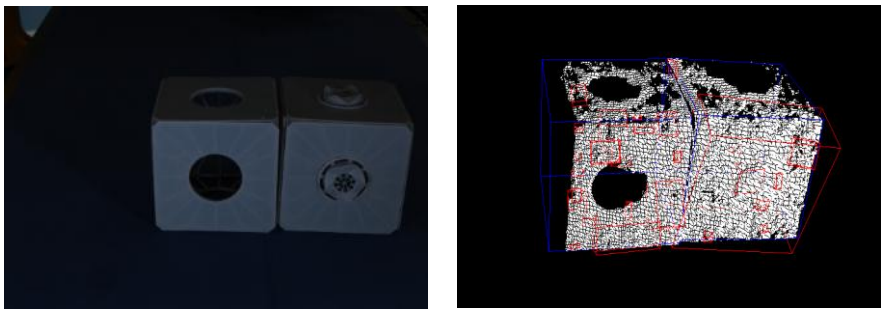
no light low light medium light intense light

Figure 7.32. Comparison of point cloud reconstruction and pose estimation for a HikVision camera using different illumination.

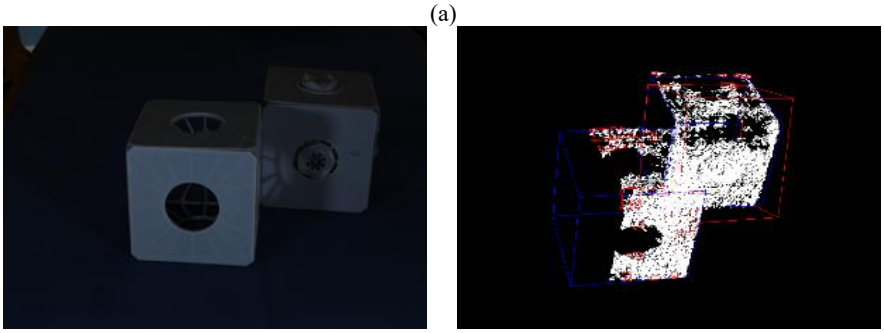
The result of the pose estimation using one module with recession surface anomaly is shown in Figure 7.33, which contains the image from the left camera and the reconstructed point cloud. The model inlier is extracted through the back projection method. It can be seen that the module position has been successfully detected and aligned well with the cube line

segments. However, the point cloud quality is so poor that many false positive anomalies were detected, shown in the red bounding box. The two modules pose estimation is shown in Figure 7.34 with three different configurations. Due to the limitation of field of view of the stereo camera, the module away from the right side of the left image cannot be completely reconstructed. Nevertheless, all modules have been detected correctly. Similarly, the poses of three modules are correctly detected in Figure 7.35.

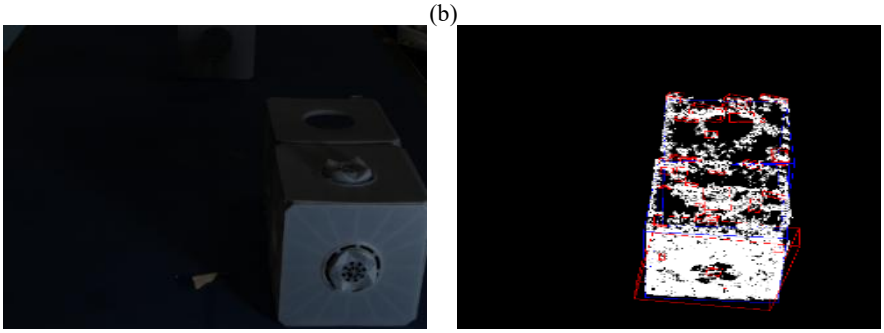
Figure 7.33. Single modular spacecraft pose estimation using the HikVision stereo camera.



(a)



(b)



(c)

Figure 7.34. Two modules pose an estimation of different configuration using a HikVision stereo camera.

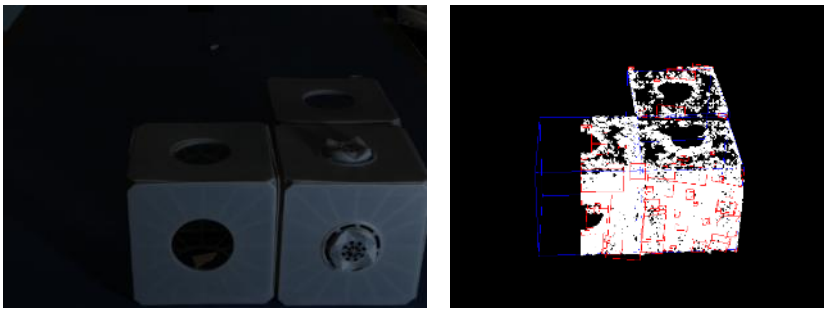


Figure 7.35. Three modules pose estimation of different configuration using HikVision stereo camera.

The selective zooming is enabled by the use of a pair of PTZ cameras whose direction of view and focal length can be digitally controlled. To make them a stereo camera pair, each orientation and zoom position need to be calibrated for point cloud reconstruction. Two positions are calibrated here for demonstration, i.e., far view and close view, as shown in Figure 7.36 and Figure 7.37, respectively. The close view mode has better point cloud reconstruction quality than far view when the module is far from the cameras.

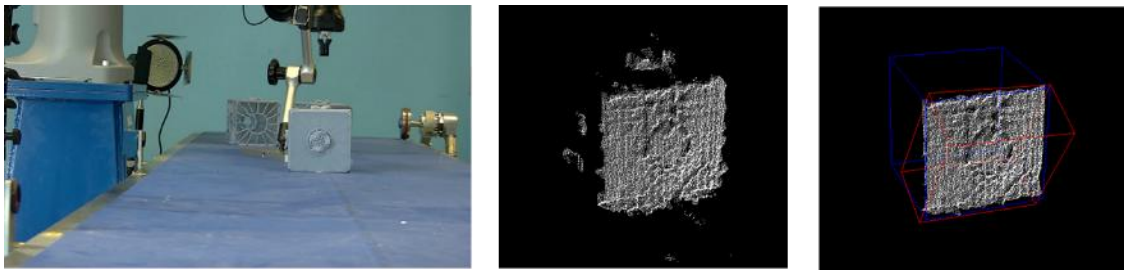


Figure 7.36. Far view mode pose estimation of single module using DataVideo stereo cameras.

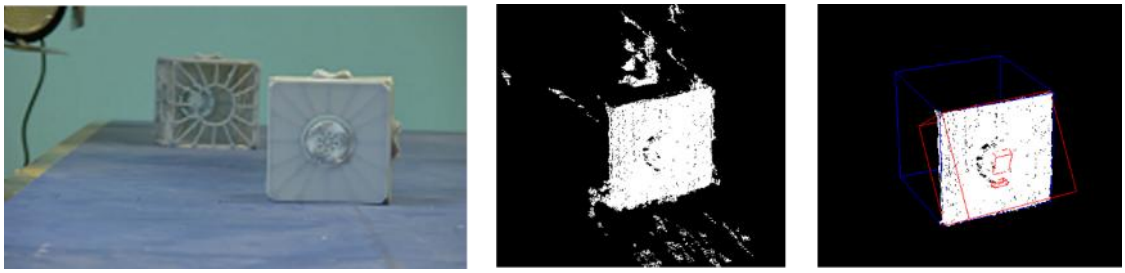


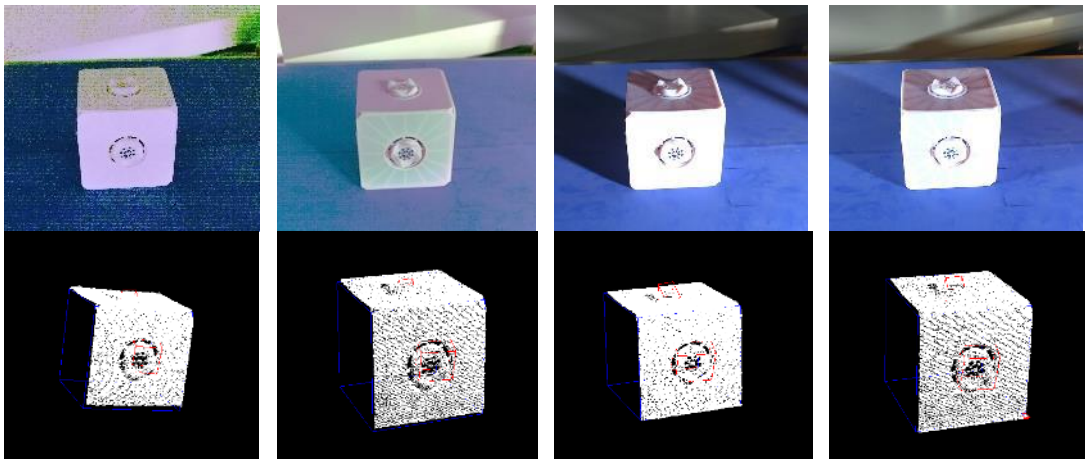
Figure 7.37. Close view mode pose estimation of single module using DataVideo stereo cameras.

Due to the texture-less surface of the the module, the error of reconstructed point cloud from both HikVision and DataVideo stereo cameras is larger than 5mm. Therefore, it is impossible to detect the designed 5mm surface and interface anomaly using the current system. However, they are still capable of reconfiguration anomaly detection.

7.6.4. RGB-D based Anomaly Detection

Following the same procedure of the stereo-based anomaly detection, the modules are detected with line segment matching. The point cloud from the RGB-D camera is compared with the

spacecraft mesh model for anomaly detection. Similarly, different illumination conditions are compared in Figure 7.38. Since the point cloud is generated using an active light source, that is, the structured light, the quality of the point cloud is not influenced by the environment light condition. It can be seen that the point cloud and detection results are the same for all the illumination conditions.



(a) no light (b) low light (c) medium light (d) intense light

Figure 7.38. Comparison of point cloud reconstruction and pose estimation for the Revopoint camera using different illumination.

The comparison of the single module point cloud quality from three camera sets is listed in Table 7.4. It can be observed that the DataVideo PTZ stereo camera has the worst performance with far view configuration. In the close view configuration, both the number of points and quality have increased. The distance error was reduced to 2.8 mm. However, for all the stereo cameras, the distance error is too large to detect the designed 5 mm surface and interface anomaly. But the precision is good enough for reconfiguration anomaly detection. The RevoPoint structured light RGB-D camera has the best point cloud quality with only 0.5 mm distance error, which makes it a good camera for all the anomaly detection.

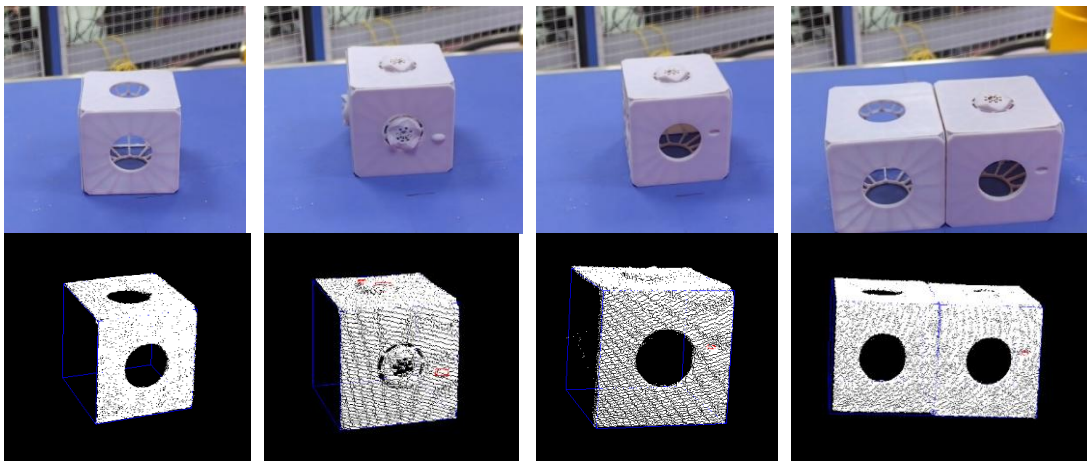
Table 7.4. Comparison of point cloud quality from three camera sets

Camera Type	HikVision Stereo Camera	RevoPoint RGB-D Camera	DataVideo PTZ Far	DataVideo PTZ Close
Number of Points	37979	32818	14924	56778
Mean Distance Erro (mm)	3.1	0.48	4.7	2.8
Distance Error Standard Deviation (mm)	2.2	0.51	3.7	1.5

1) Surface Anomaly Detection

Following the procedure defined in Section 0, the vision system was tested in four scenarios, as shown in Figure 7.39. The detection result of the undamaged module shows no anomaly

detected. The protrusion and recess of the other three scenarios were detected successfully. For the second scenario, the top interface was detected with an anomaly, which is due to an installation error. When two modules were placed together, surface error can still be detected. Therefore, the system performed well in surface anomaly detection. Quantitatively, the detected surface protrusion is 4.3 mm in depth, compared to the ground truth 5 mm. The detected depth of the surface recess is 2.3 mm, which is much less than the ground truth. This is due to the fact that points within the thickness of the cover, which is about 1.5 mm, are not counted as anomaly.

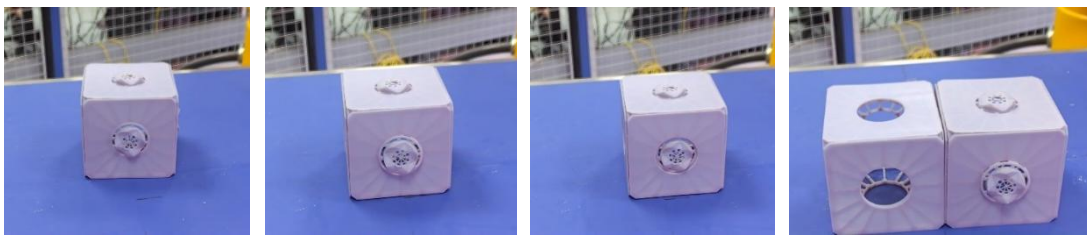


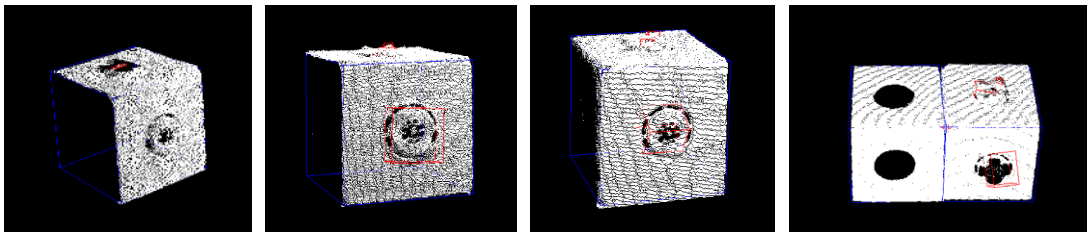
(a) (b) (c) (d)

Figure 7.39. Surface anomaly detection results using Revopoint camera including (a) no damage, (b) surface protrusion, (c) surface recess, and (d) one undamaged and one with surface recess.

2) Interface Anomaly Detection

Similarly to surface anomaly detection, there are four scenarios for interface anomaly detection, as shown in Figure 7.40. The experiment was designed to change the position of the front face interface. However, due to the installation error, the top face interface was detected as anomaly in all the four scenarios. This validated the repeatability of the system. For all front face interface, the designed anomalies have been correctly detected. The detected interface protrusion is 3.6 mm and the recess is 3.2 mm.





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

Figure 7.40. Results of the detection results using RevoPoint camera, including (a) undamaged, (b) protrusion of the interface, (c) recess of the interface, and (d) one undamaged and one with recess of the interface.

3) Reconfiguration Anomaly Detection

The reconfiguration anomaly detection is simply a comparison of the detected module configuration with the predefined configuration. The most critical part is the correct detection of the positions of all the modules in the view. Therefore, the evaluation of reconfiguration anomaly detection is simplified as a multi-module pose estimation. As shown in Figure 7.41 and Figure 7.42 with two and three modules presented, the positions of all the modules have been detected. Comparison of the detected module positions with the defined configuration depends on the implementation of the MOSAR mission.

Overall, the designed vision system has passed all MOSAR tests by using RGB-D camera. The standard stereo cameras, however, only managed to realise the reconfiguration anomaly detection, but not the surface and interface anomaly detection.

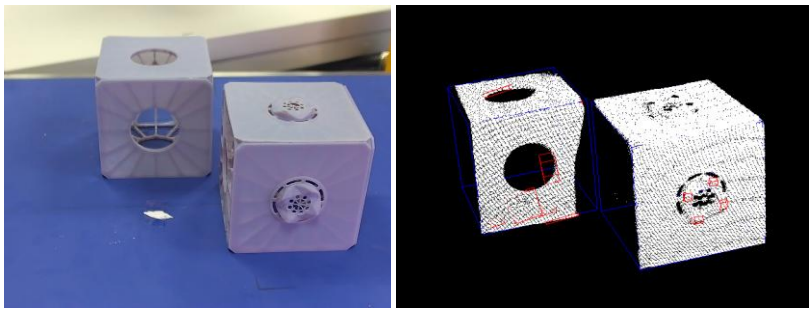


Figure 7.41. Two modules pose estimation using a Revopoint camera.

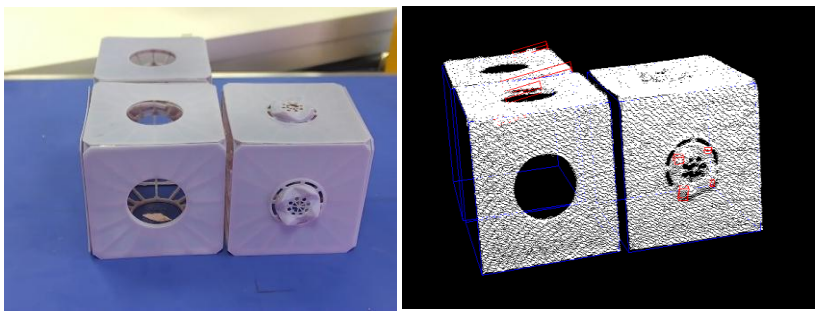
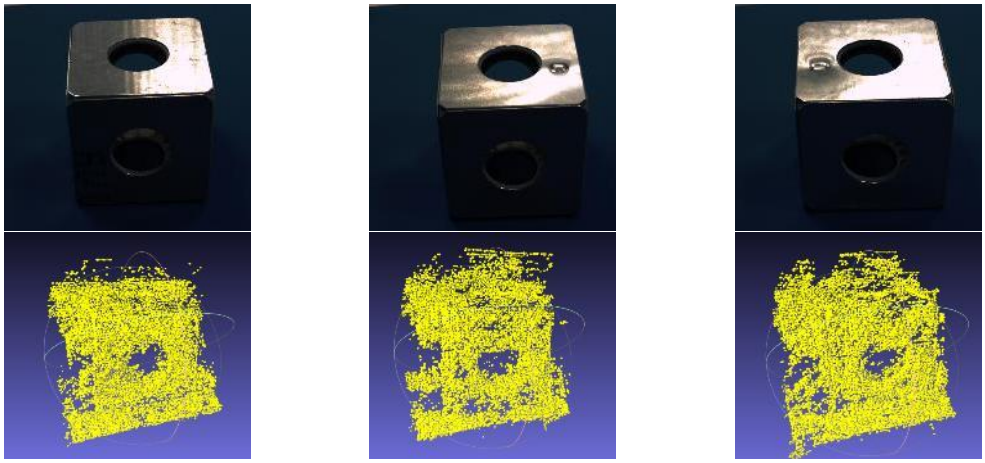


Figure 7.42. Three modules pose estimation using Revopoint camera.

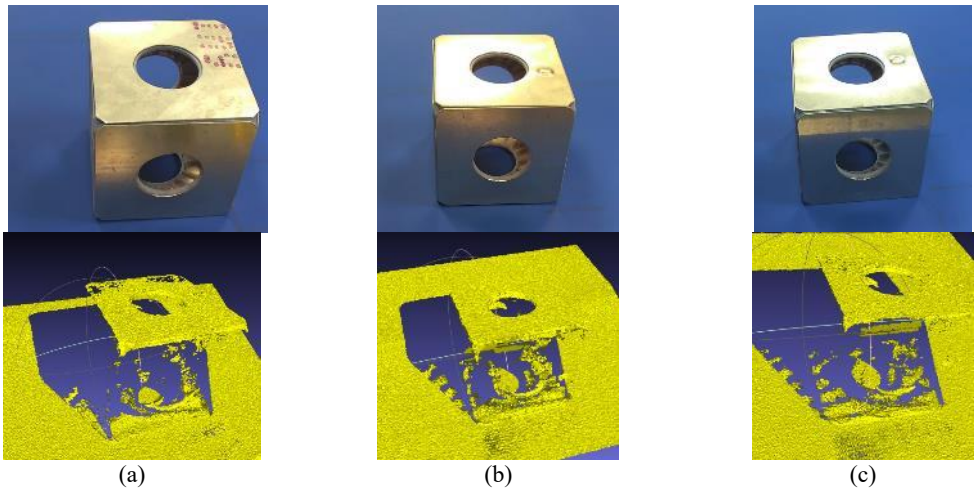
7.6.5. Metal Cover

The 3D printed material is plastic, which is easier for both point cloud reconstruction from stereo camera and structured light camera. However, in practice, the spacecraft is usually covered by metal plates. Therefore, a dataset of modular spacecraft with metal plate cover were also collected, as shown in Figure 7.43 and Figure 7.44. The point cloud from stereo cameras is sparser and contains more noise for metal cover than the one for the plastic cover. As for the structured light camera, since the metal surface is reflective, the structured light cannot be captured and decoded correctly, leading to failure of point cloud reconstruction for the front face of the module. Even though the point cloud for the top face is still good, surface and interface anomaly detection still cannot be realised. Therefore, anomaly detection for spacecraft with metal surface spacecraft was failed.



(a) (b) (c)

Figure 7.43. Left camera images (top) and reconstructed point cloud (bottom) of modular spacecrafts with metal cover from HikVision stereo cameras.



(a) (b) (c)

Figure 7.44. RGB images (top) and point cloud (bottom) of modular spacecrafts with metal cover from RevoPoint camera.

7.7. Summary

In this chapter, an AVI system is designed for modular and reconfigurable spacecraft. It includes three image capture systems for 3D reconstruction. Three datasets are established to validate the AVI system, including the small satellite model, engineering feature block, and modular spacecraft. The first two datasets were used for the algorithm selection and optimisation process, followed by a pose estimation algorithm pipeline to detect the anomaly of the target spacecraft. The proposed pipeline combines the information from both the RGB and point cloud for accurate pose estimation and model inlier extraction The three key algorithms include segment 3D extraction, segment 3D matching, and model inlier extraction. With identified module position and its inlier point cloud, the anomaly can be detected by comparing the point cloud with the CAD model. Three sets of cameras are used and tested for the vision system. The final system is validated through a series of spacecraft anomaly detection tasks, including the surface, interface, and reconfiguration anomaly detection.

The fourth case study on modular spacecraft anomaly detection significantly contributes to the proposed comprehensive design methodology by extending its applicability to 3D vision-based inspection systems. This case study demonstrates that the proposed design methodology is not limited to 2D image-based systems but can be effectively applied to 3D anomaly detection tasks. The incorporation of 3D reconstruction and registration algorithms into the algorithm selection and optimisation process broadens the methodology's scope to address more complex and multi-dimensional inspection applications.

The study highlights the importance of customising image acquisition systems to the target application. By developing and evaluating three different 3D image capture systems (stereo cameras and structured light camera), the case study validates the methodology's principle of sensor selection and system customisation as a critical step in the AVI system design process.

It has also been verified that the algorithm selection and optimisation process can be applied to 3D-level data. The process facilitated the identification of the most suitable 3D reconstruction and anomaly detection algorithms, as well as their optimisation for different anomaly types, showcasing the generalisability of the proposed methodology across different data modalities.

8. Conclusion and Future Work

8.1. Introduction

This thesis provides an investigation of AVI system design for defect detection across various localisation precision levels and diverse industrial applications. Based on the literature review, the knowledge gaps have been identified including the absence of a comprehensive design methodology for AVI systems, challenges in real-time detection on devices with limited computational resources, and under-exploration of the anomaly detection methods in AVI systems when negative sample data is scarce.

To address these challenges, this thesis proposed a comprehensive design methodology for AVI systems. The methodology emphasises defect detection method development through systematic dataset establishment and algorithm selection. Its effectiveness is demonstrated through four case studies: concrete wall crack inspection, wheat head disease detection, car engine anomaly detection, and modular spacecraft anomaly detection. These case studies encompass a range of defect detection tasks, from image-level analysis to 3D-level inspections, highlighting the versatility of the proposed approach.

This chapter first outlines the key findings from the design methodology and case studies, followed by the contributions to knowledge. Finally, it discusses the limitations and potential directions for future work.

8.2. Key Findings

8.2.1. AVI System Design Methodology

First, a design methodology is proposed to support the design of AVI systems with a focus on defect detection method development. Through the process model, the design evolves from an abstract and qualitative concept to a specific and quantitative embodiment, and finally a complete AVI system. During the development of the defect detection method, the algorithm selection and optimisation process is adapted based on the availability of target data. Following this design methodology, four AVI systems are designed and developed to address various detection tasks across the construction, agricultural, manufacturing, and space industries. An improved version of Table 3.1, comparing the four AVI systems is presented in Table 8.1. The four case studies have covered and validated different aspects of the design methodology. Specifically, the algorithm selection and optimisation process has been adapted by each case

study for the defect detection method development, targeting different tasks.

Table 8.1. Comparison of case studies for methodology validation

	The First Case Study	The Second Case Study	The Third Case Study	The Fourth Case Study
Target Object	Concrete Wall	Wheat Head	Car Engine	Modular Spacecraft
Detection Task	Image and pixel-level	Object-level	Pixel-level	3D-level
Defect	Crack	Head blight	Surplus	Extrusion, protrusion
Requirements	Real-time High accuracy	Real-time No disease data for training	High accuracy No defect data for training	High accuracy No defect data for training
Detection Method Category	Classification-based detection	Classification-based detection and anomaly detection	Anomaly detection	Anomaly detection
Camera	Monocular RGB, depth camera	Monocular RGB	Monocular RGB	Stereo camera, structured light camera
Target Dataset	Open-source crack datasets for classification and segmentation	Open-source wheat head datasets and wheat head disease datasets	Collected car engine datasets with real anomalies	Collected stereo and RGBD images for small satellite and engineering feature block
Supplementary Dataset	-	-	Collected car engine datasets with synthetic anomalies	Collected stereo image pairs and RGBD images for modular spacecraft
System Platform	AR headset	Agri-Rover	Production line	Spacecraft
Algorithm Pipeline	Crack classification, segmentation, measurement	Wheat head bounding box detection, anomaly detection	Background removal, image tiling, anomaly detection	3D reconstruction, pose estimation, anomaly detection
Detection Method	CNN	CNN	Pretrained ViT	Registration
Optimisation	Novel training strategy, novel model architecture	Domain transfer	SAM-based background removal, image tiling, ViT-based feature extractor	3D segment extraction and matching
User Interface	AR application	Image window with bounding box and confidence	Image window with heatmap mask	Point cloud window with bounding box
Accuracy (%)	98.68 classification 71.4 segmentation	74.8 WHBD head detection 86.2 FHBS head detection 95.6 WHBD disease detection 71.8 FHBS disease detection	66.2 pixel F1 score 99.6 pixel AUROC	5 cm surface and interface anomalies successfully detected
Inference speed	5.5 fps on HoloLens 2 for crack measurement	3 fps on Jetson Nano	-	-

8.2.2. The First Case Study: Concrete Crack Inspection

Specifically, the first case study for image- and pixel-level detection presents the development of a head-mounted vision-based inspection system, employing a synergistic blend of AR and AI technologies. The system, implemented in Unity and deployed on Microsoft HoloLens 2,

divides crack images into patches for AI-driven classification and segmentation. A novel hierarchical training strategy, combining transfer learning and knowledge distillation, accelerates the model generation for real-time classification. A new convolution block architecture, DE-Net, was developed, showing a remarkable balance between accuracy, model size, and training time. For crack segmentation, various segmentation heads like U-Net and DEU-Net were implemented, tested on different datasets to evaluate their performance in self-dataset and cross-dataset scenarios. The results favoured the U-Net and DEU-Net for their superior performance and parameter efficiency, with U-Net demonstrating particular effectiveness when trained on the CrackForest dataset. The final integration combined the classification and segmentation models with a user-friendly GUI in Unity, successfully deploying the system on HoloLens 2. In lab tests, the system effectively classified and segmented crack images, using point cloud data from HoloLens 2 to measure crack dimensions in real-world units. This comprehensive system was validated in a lab environment, confirming its potential for practical application in engineering and construction scenarios.

8.2.3. The Second Case Study: Wheat Head Disease Detection

The second case study for object-level detection details the design and development of a wheat head disease detection method for precision spray system. This system highlights the incorporation of cutting-edge technological approaches in the model training process to address the data scarcity. Techniques such as transfer learning, domain adaptation, anomaly detection, and data augmentation have been applied to enhance the system capabilities. Particularly, the zero-shot learning was employed to tackle the challenges posed by a significantly imbalanced dataset. The deep learning model pre-trained on a large wheat head detection dataset can be used for the detection task on another dataset via domain adaption method. Without any labelled data of wheat head disease, the anomaly detection methods were compared on two datasets for disease detection. The HSV color thresholding method significantly outperformed the feature embedding-based methods on the FHBS dataset. While all methods could distinguish between healthy heads and those with severity level 9, the accuracy for detecting lower severity disease heads requires improvement.

This is the first time the anomaly detection technique is used for wheat head disease detection. This method can also be easily extended to other disease detection tasks through the proposed pipeline, i.e., domain transfer from one larger dataset to the target dataset, feature analysis of negative and positive samples, and apply anomaly detection methods for the assigned disease detection task..

8.2.4. The Third Case Study: Car Engine Surface Anomaly Detection

In the third case study for pixel-level detection, the proposed design methodology was applied to develop a car engine manufacturing defect detection system. This case study firstly presents a comparative study involving different anomaly synthesis methods, anomaly characteristics, and anomaly detection techniques. Based on the findings from this study, specific improvements were made to the real anomaly dataset to achieve zero false negatives prediction. Specifically, two engine parts datasets with synthetic and real anomalies are established. The synthetic dataset contains varied image collection configurations and synthetic anomalies generated from three synthesis methods. In this synthetic dataset, a range of anomaly characteristics has been covered, including anomaly size, brightness, and contrast. A comparative study was conducted using five anomaly detection methods tested on datasets with both synthetic and real anomalies. It is revealed that the polarised filter can reduce the reflective light on the engine surface, however, it has limited impact on the anomaly detection on the synthetic dataset. Exposure time affects the size of overexposed and underexposed area. The study shows that the medium level exposure is the best for anomaly detection. As for lighting conditions, there is no perfect lighting condition, but it is critical to maintain the same lighting for image collection for both training and testing. Finally, the best performed model, PaDiM, is selected as the base model and improvements were made for the real anomaly dataset, including the implementation of a better feature extractor, background removal, and image tiling. This case study offers critical insights into developing and optimising anomaly detection method for vision-based car engine manufacturing defect inspection systems.

8.2.5. The Fourth Case Study: Modular Spacecraft Anomaly Detection

In the fourth case study for 3D level detection, a specialised vision-based inspection system was developed for the anomaly detection in modular spacecraft. Based on the pose estimation experiments on two supplementary datasets, an algorithm pipeline was introduced and optimised for the modular spacecraft. The proposed pipeline incorporated three key algorithms: segment 3D extraction, segment 3D matching, and model inlier extraction, each playing a critical role in the anomaly detection process. These algorithms enabled the system to accurately identify the position of spacecraft modules and extract their inlier point clouds, facilitating the detection of anomalies on the spacecraft's surface, interfaces, and during reconfiguration processes. To support and test the vision system, three different sets of cameras were utilised. This visual inspection system was validated with 3D printed spacecraft in lab environments.

Overall, following the proposed design methodology, the four designed and developed AVI systems have been verified and validated to meet the specifications and requirements of their respective applications.

8.3. Contribution to Knowledge

8.3.1. Overall Contributions to Knowledge

This research addresses the AVI system design for defect detection in dealing with a range of application requirements. Through a thorough literature review, the limitations and knowledge gaps in this field have been provided, including the absence of a comprehensive design methodology for AVI systems, challenges in real-time detection on devices with limited computational resources, and under-exploration of the anomaly detection methods in AVI systems. By addressing these knowledge gaps, this research makes significant contributions to the field, including a comprehensive design methodology, and the AVI systems developed through these case studies.

To address the first knowledge gap regarding the absence of design methodology, a comprehensive design methodology was proposed for the design of vision-based defect detection systems with a focus on defect detection. The design process model in the methodology can be used to guide the integration of different components for the development of the defect detection method. In addition, an algorithm selection and optimisation process is proposed and can be adapted to different applications depending on the data availability. This design methodology has been validated through four case studies.

Addressing the second knowledge gap related to the accuracy-efficiency trade-off on resource constrained device, a novel training strategy and a novel CNN model architecture are proposed. The training strategy combines the knowledge distillation and transfer learning to facilitate the model training process on both efficiency and accuracy. The proposed CNN model architecture integrated the dilated convolution to increase the parameter efficiency, leading to a smaller and more accurate model. Both the proposed training strategy and model architecture were validated in a crack inspection system on the AR device HoloLens 2.

Targeting the third knowledge gap concerning data scarcity in most industrial applications, the anomaly detection method was explored, deployed, and validated on three case studies, including wheat head disease detection, car engine defect detection, and modular spacecraft anomaly detection. All three applications require no defect data during model training. To apply the anomaly detection method, following the design methodology, the algorithm

candidates are compared, and the best algorithm is optimised for the target application.

8.3.2. Case Study Specific Contributions

Each case study also provides their specific contribution to knowledge. In the case study of AR based wall crack inspection system, the proposed model architecture is used as the backbone of four different segmentation architectures and evaluated on three different datasets. An AR application that integrates the classification model and segmentation model are developed and deployed on HoloLens 2.

In terms of the wheat head disease detection, a zero-shot wheat head disease detection algorithm is proposed based on domain transfer and anomaly detection. The domain transfer enables the model to detect wheat head in the test dataset which has completely different data distribution. The anomaly detection enables the model to detect wheat disease without using any disease data during training. Then a comprehensive study of various anomaly detection methods is conducted on two wheat head disease datasets, demonstrating high disease classification accuracy, as well as the potential for applying anomaly detection methods to severity estimation.

The contributions of the third case study, the car engine defect detection, include two datasets for engine parts with synthetic anomalies and real anomalies, the findings from the image-, pixel- and anomaly-level analysis on the two datasets regarding to the data collection configurations, anomaly characteristics, and anomaly detection methods, and finally an optimised algorithm pipeline to achieve zero false negative for car engine defect detection.

For the case study of modular spacecraft inspection system, a reconfigurable defect detection software was developed so that each algorithm node can be configured for specific tasks. An image capture system with three different cameras was developed. Additionally, three datasets were established using different 3D printed models and cameras.

8.4. Limitations and Future Work

8.4.1. AVI System Design Methodology

The proposed design methodology was used to design four different AVI systems to address two key challenges in defect detection method development: the accuracy-efficiency trade-off and data scarcity. In addressing the accuracy-efficiency trade-off, only the crack inspection system was developed to validate the proposed training strategy and model architecture. Future research will aim to conduct further experiments to validate these strategies in other

applications such as surface inspection of metal pipe, electronic components, food production, etc. For the anomaly detection methods used in wheat head disease and car engine defect detection, only existing anomaly detection methods were deployed and compared without further modification. Future work will focus on proposing novel anomaly detection methods for general anomaly detection tasks and validating them across these applications.

Other limitations of the design methodology include the multi-perspective models and the defect detection development process. The current multi-perspective models cover only the most general models within AVI systems, while more specific models in certain applications remain unexplored. Additionally, the defect detection development process outlines only basic steps for method development and requires more detailed guidance. Future work will aim to develop adaptable, application-specific multi-perspective models and provide a more detailed defect detection development process.

8.4.2. The First Case Study: Concrete Crack Inspection

The current system relies on two stages of crack detection including crack classification and crack segmentation. This could sometimes lead to high latency during on-site usage and cause uncomfortable for users. It also requires further refinement to enhance its accuracy and efficiency, particularly under varied environmental conditions. Additionally, the ability to perform consistently in diverse real-world environments remains to be fully tested and validated. The future work to refine the system includes:

- Implement model quantisation techniques to reduce the model size without significant loss of accuracy, enhancing the system's suitability for devices with limited processing capabilities.

- Utilise Neural Architecture Search (NAS) to optimise the network structure, potentially improving the system's performance and efficiency.

- Refine the AR application based on feedback from real-world tests. This could involve improving the user interface, improving the design of the interaction, and ensuring stable performance in various environmental conditions.

- Explore the use of more advanced AR features, such as real-time data overlay and interactive 3D models, to provide a more comprehensive and user-friendly experience.

- Conduct extensive field tests in different settings, such as in various weather conditions and on different types of concrete structures, to rigorously assess the system's versatility and reliability.

- Gather and analyse data from real-world usage to identify any shortcomings and areas for improvement.

8.4.3. The Second Case Study: Wheat Head Disease Detection

The limitations of the precision spray system can be mainly attributed to two aspects: the detection algorithm and the physical spray mechanism. First, the accuracy of the defection algorithm is constrained by a lack of comprehensive data. Second, the robustness of the spray system needs to be improved. There have been occasions when the pressure and flow rate of the system have decreased significantly, especially when navigating obstacles. Therefore, the directions of future work include:

- Develop and fine-tune the disease detection algorithm to improve its sensitivity and accuracy, particularly to reduce false negatives. This could be done by collecting more data and optimisation of the detection algorithm.

- Improve the durability and resistance to environmental factors such as moisture, dust, and temperature variations, ensuring its longevity and reliability in agricultural settings.

- Deploy the system in actual field conditions and conduct extensive tests to evaluate its performance across various agricultural settings and crop types. Collect and analyse data from these deployments to identify areas for improvement.

- Improve system integration with smart farming technologies, such as IoT sensors and farm management software, to provide a more holistic approach to crop health and resource management.

8.4.4. The Third Case Study: Car Engine Surface Anomaly Detection

The car part defect detection system, particularly for complex structures like cylinder head or engine underside, can yield many false positives and false negatives, indicating a need for further algorithmic refinement. Achieving a balance between image-level and pixel-level accuracy remains challenging, as optimising one often leads to a compromise on the other.

There are several areas where further work could possibly enhance the defect detection accuracy of the system, especially for challenging components like cylinder heads and engine undersides:

- Investigate more sophisticated image processing and heatmap generation techniques. This can include adaptive filtering, advanced edge detection methods, and machine learning-based image enhancement based on the characteristics of cylinder heads and engine undersides.

- Explore hybrid models that combine the strengths of different anomaly detection algorithms. For example, integrating models that are effective in identifying large defects with those proficient in detecting finer anomalies could produce a more robust system.

- Implement more advanced feature extraction methods that can accurately capture the nuances of complex structures in cylinder heads and engine undersides.

- Develop dynamic anomaly scoring systems that adjust thresholds based on the specific characteristics of each car part and defect type. This could help reduce false positives and negatives by providing a more detailed assessment of anomalies.

- Collect and incorporate a more diverse range of data, including rare defect types and variations in car part manufacturing, to improve the model's ability to generalise and handle a wider array of defects.

8.4.5. The Fourth Case Study: Modular Spacecraft Anomaly Detection

The current inspection system has limitations in accurately reconstructing 3D models, especially when dealing with reflective surfaces such as metal. This could impact the precision of pose estimation and anomaly detection. System performance under varying conditions of space, such as extreme lighting and temperature variations, might not be fully optimised. To address these limitations, the future work will focus on:

- Develop advanced algorithms or incorporate new sensor technologies to improve 3D reconstruction, particularly for challenging surfaces like metal.

- Explore the use of polarised light or specialised filters to mitigate the issues caused by reflective surfaces.

- Tailor and optimise the system to handle a wider range of materials and surface textures found in spacecraft, enhancing its applicability.

- Implement machine learning techniques to improve the system’s ability to adapt and accurately interpret data from different materials and environmental conditions.

- Use AI to enhance the system’s capability for real-time processing and decision making in space environments.

- Conduct rigorous tests in environments that closely simulate space conditions to validate and refine the performance of the system.

References

[1] T. S. Newman and A. K. Jain, "A survey of automated visual inspection," *Computer vision and image understanding,* vol. 61, no. 2, pp. 231-262, 1995.

[2] H.-F. Ng, "Automatic thresholding for defect detection," *Pattern recognition letters,* vol. 27, no. 14, pp. 1644-1649, 2006.

[3] A.-A. Tulbure, A.-A. Tulbure, and E.-H. Dulf, "A review on modern defect detection models using DCNNs–Deep convolutional neural networks," *Journal of Advanced Research,* vol. 35, pp. 33-48, 2022.

[4] S. K. Dwivedi, M. Vishwakarma, and A. Soni, "Advances and researches on non destructive testing: A review," *Materials Today: Proceedings,* vol. 5, no. 2, pp. 3690-3698, 2018.

[5] M. Gupta, M. A. Khan, R. Butola, and R. M. Singari, "Advances in applications of Non-Destructive Testing (NDT): A review," *Advances in Materials and Processing Technologies,* vol. 8, no. 2, pp. 2286-2307, 2022.

[6] A. Fahr, *Aeronautical applications of non-destructive testing*. DEStech Publications, Inc, 2013.

[7] S. Iliopoulos *et al.*, "Detection and evaluation of cracks in the concrete buffer of the Belgian Nuclear Waste container using combined NDT techniques," *Construction and Building Materials,* vol. 78, pp. 369-378, 2015.

[8] F. Guibert, M. Rafrafi, D. Rodat, E. Prothon, N. Dominguez, and S. Rolet, "Smart NDT tools: Connection and automation for efficient and reliable NDT operations," in *19th World Conf. Non-Destructive Test*, 2016, pp. 1-10.

[9] A. Chabot, N. Laroche, E. Carcreff, M. Rauch, and J.-Y. Hascoët, "Towards defect monitoring for metallic additive manufacturing components using phased array ultrasonic testing," *Journal of Intelligent Manufacturing,* vol. 31, no. 5, pp. 1191-1201, 2020.

[10] A. Lopez, R. Bacelar, I. Pires, T. G. Santos, J. P. Sousa, and L. Quintino, "Non-destructive testing application of radiography and ultrasound for wire and arc additive manufacturing," *Additive Manufacturing,* vol. 21, pp. 298-306, 2018.

[11] H. A. Gabbar, A. Chahid, M. J. A. Khan, O. G. Adegboro, and M. I. Samson, "CTIMS: Automated defect detection framework using computed tomography," *Applied Sciences,* vol. 12, no. 4, p. 2175, 2022.

[12] Z.-C. Yuan, Z.-T. Zhang, H. Su, L. Zhang, F. Shen, and F. Zhang, "Vision-based defect detection for mobile phone cover glass using deep neural networks," *International Journal of Precision Engineering and Manufacturing,* vol. 19, no. 6, pp. 801-810, 2018.

[13] B. W. Drinkwater and P. D. Wilcox, "Ultrasonic arrays for non-destructive evaluation: A review," *NDT & e International,* vol. 39, no. 7, pp. 525-541, 2006.

[14] G. Zenzinger, J. Bamberg, W. Satzger, and V. Carl, "Thermographic crack detection by eddy current excitation," *Nondestructive Testing and Evaluation,* vol. 22, no. 2-3, pp. 101-111, 2007.

[15] Y. Chung, R. Shrestha, S. Lee, and W. Kim, "Thermographic inspection of internal defects in steel structures: analysis of signal processing techniques in pulsed thermography," *Sensors,* vol. 20, no. 21, p. 6015, 2020.

[16] W. Sun, S. Brown, and R. Leach, "An overview of industrial X-ray computed

tomography," 2012.
[17] A. Du Plessis, S. G. le Roux, J. Els, G. Booysen, and D. C. Blaine, "Application of microCT to the non-destructive testing of an additive manufactured titanium component," *Case Studies in Nondestructive Testing and Evaluation,* vol. 4, pp. 1-7, 2015.
[18] M. Baygin, M. Karakose, A. Sarimaden, and A. Erhan, "Machine vision based defect detection approach using image processing," in *2017 international artificial intelligence and data processing symposium (IDAP)*, 2017: Ieee, pp. 1-5.
[19] B. Caiazzo, M. Di Nardo, T. Murino, A. Petrillo, G. Piccirillo, and S. Santini, "Towards Zero Defect Manufacturing paradigm: A review of the state-of-the-art methods and open challenges," *Computers in Industry,* vol. 134, p. 103548, 2022.
[20] M. M. Abagiu, D. Cojocaru, F. Manta, and A. Mariniuc, "Detecting Machining Defects inside Engine Piston Chamber with Computer Vision and Machine Learning," *Sensors,* vol. 23, no. 2, p. 785, 2023.
[21] X. Zheng, S. Zheng, Y. Kong, and J. Chen, "Recent advances in surface defect inspection of industrial products using deep learning techniques," *The International Journal of Advanced Manufacturing Technology,* pp. 1-24, 2021.
[22] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Computational intelligence and neuroscience,* vol. 2018, no. 1, p. 7068349, 2018.
[23] J. Chai, H. Zeng, A. Li, and E. W. Ngai, "Deep learning in computer vision: A critical review of emerging techniques and application scenarios," *Machine Learning with Applications,* vol. 6, p. 100134, 2021.
[24] A. Ioannidou, E. Chatzilari, S. Nikolopoulos, and I. Kompatsiaris, "Deep learning advances in computer vision with 3d data: A survey," *ACM computing surveys (CSUR),* vol. 50, no. 2, pp. 1-38, 2017.
[25] Z. Wang *et al.*, "A convolutional neural network-based classification and decision-making model for visible defect identification of high-speed train images," *Journal of Sensors,* vol. 2021, pp. 1-17, 2021.
[26] Q. Jiang, D. Tan, Y. Li, S. Ji, C. Cai, and Q. J. A. S. Zheng, "Object Detection and Classification of Metal Polishing Shaft Surface Defects Based on Convolutional Neural Network Deep Learning," vol. 10, no. 1, p. 87, 2020.
[27] D. Tabernik, S. Šela, J. Skvarč, and D. Skočaj, "Segmentation-based deep-learning approach for surface-defect detection," *Journal of Intelligent Manufacturing,* vol. 31, no. 3, pp. 759-776, 2020.
[28] W. Ying, J. Cuiyun, and Z. Yanhui, "Pipe defect detection and reconstruction based on 3D points acquired by the circular structured light vision," *Advances in Mechanical Engineering,* vol. 5, p. 670487, 2013.
[29] J. Guo, P. Liu, B. Xiao, L. Deng, and Q. Wang, "Surface defect detection of civil structures using images: Review from data perspective," *Automation in Construction,* vol. 158, p. 105186, 2024.
[30] S. R. Saufi, Z. A. B. Ahmad, M. S. Leong, and M. H. Lim, "Challenges and opportunities of deep learning models for machinery fault detection and diagnosis: A review," *Ieee Access,* vol. 7, pp. 122644-122662, 2019.
[31] B. Batchelor, F. Waltz, and M. Snyder, "A design methodology for industrial vision systems," in *Optomechanical and Electro-Optical Design of Industrial*

Systems, 1988, vol. 959: SPIE, pp. 126-145.
[32] H. Golnabi and A. Asadpour, "Design and application of industrial machine vision systems," *Robotics and Computer-Integrated Manufacturing,* vol. 23, no. 6, pp. 630-637, 2007.
[33] Y. Chen, Y. Ding, F. Zhao, E. Zhang, Z. Wu, and L. Shao, "Surface defect detection methods for industrial products: A review," *Applied Sciences,* vol. 11, no. 16, p. 7657, 2021.
[34] J. Huang *et al.*, "Speed/accuracy trade-offs for modern convolutional object detectors," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7310-7311.
[35] D. Hou, T. Liu, Y.-T. Pan, and J. Hou, "AI on edge device for laser chip defect detection," in *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, 2019: IEEE, pp. 0247-0251.
[36] J. Liu *et al.*, "Deep industrial image anomaly detection: A survey," *Machine Intelligence Research,* vol. 21, no. 1, pp. 104-135, 2024.
[37] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger, "MVTec AD--A comprehensive real-world dataset for unsupervised anomaly detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 9592-9600.
[38] Q. Luo, X. Fang, L. Liu, C. Yang, and Y. Sun, "Automated visual defect detection for flat steel surface: A survey," *IEEE Transactions on Instrumentation and Measurement,* vol. 69, no. 3, pp. 626-644, 2020.
[39] X. Fang, Q. Luo, B. Zhou, C. Li, and L. Tian, "Research progress of automated visual surface defect detection for industrial metal planar materials," *Sensors,* vol. 20, no. 18, p. 5136, 2020.
[40] J. J. Lin, A. Ibrahim, S. Sarwade, and M. Golparvar-Fard, "Bridge inspection with aerial robots: Automating the entire pipeline of visual data capture, 3D mapping, defect detection, analysis, and reporting," *Journal of Computing in Civil Engineering,* vol. 35, no. 2, p. 04020064, 2021.
[41] W. Ming *et al.*, "A comprehensive review of defect detection in 3C glass components," *Measurement,* vol. 158, p. 107722, 2020.
[42] S. Alahakoon, Y. Q. Sun, M. Spiryagin, and C. Cole, "Rail flaw detection technologies for safer, reliable transportation: a review," *Journal of Dynamic Systems, Measurement, and Control,* vol. 140, no. 2, 2018.
[43] W. Cao, Q. Liu, and Z. He, "Review of pavement defect detection methods," *IEEE Access,* vol. 8, pp. 14531-14544, 2020.
[44] R. Manish, A. Venkatesh, and S. D. Ashok, "Machine vision based image processing techniques for surface finish and defect inspection in a grinding process," *Materials Today: Proceedings,* vol. 5, no. 5, pp. 12792-12802, 2018.
[45] V. Bruni and D. Vitulano, "A generalized model for scratch detection," *IEEE transactions on image processing,* vol. 13, no. 1, pp. 44-50, 2004.
[46] L. Song, W. Lin, Y.-G. Yang, X. Zhu, Q. Guo, and J. Xi, "Weak micro-scratch detection based on deep convolutional neural network," *IEEE Access,* vol. 7, pp. 27547-27554, 2019.
[47] W. Li, L. Zhang, C. Wu, Z. Cui, and C. Niu, "A new lightweight deep neural network for surface scratch detection," *The International Journal of Advanced Manufacturing Technology,* vol. 123, no. 5-6, pp. 1999-2015, 2022.
[48] T. Lilienblum, P. Albrecht, R. Calow, and B. Michaelis, "Dent detection in car

bodies," in *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, 2000, vol. 4: IEEE, pp. 775-778.
[49] L. Wang, L. Luo, P. Zheng, T. Zheng, and S. He, "A fast dent detection method for curved glass using deep convolutional neural network," in *2019 IEEE 13th International Conference on Anti-counterfeiting, Security, and Identification (ASID)*, 2019: IEEE, pp. 117-121.
[50] S. Barua, F. Liou, J. Newkirk, and T. Sparks, "Vision-based defect detection in laser metal deposition process," *Rapid Prototyping Journal,* vol. 20, no. 1, pp. 77-85, 2014.
[51] V. Rebuffel, S. Sood, and B. Blakeley, "Defect detection method in digital radiography for porosity in magnesium castings," *Materials Evaluation, ECNDT,* 2006.
[52] Y. Yao, S. T. E. Tung, and B. Glisic, "Crack detection and characterization techniques—An overview," *Structural Control and Health Monitoring,* vol. 21, no. 12, pp. 1387-1413, 2014.
[53] H. S. Munawar, A. W. Hammad, A. Haddad, C. A. P. Soares, and S. T. Waller, "Image-based crack detection methods: A review," *Infrastructures,* vol. 6, no. 8, p. 115, 2021.
[54] W. Mukupa, G. W. Roberts, C. M. Hancock, and K. Al-Manasir, "A review of the use of terrestrial laser scanning application for change detection and deformation monitoring of structures," *Survey review,* vol. 49, no. 353, pp. 99-116, 2017.
[55] P. Arjun and T. Mirnalinee, "Machine parts recognition and defect detection in automated assembly systems using computer vision techniques," *Rev. Téc. Ing. Univ. Zulia,* vol. 39, no. 1, pp. 71-80, 2016.
[56] P. Kunakornvong and P. Sooraksa, "Machine vision for defect detection on the air bearing surface," in *2016 International Symposium on Computer, Consumer and Control (IS3C)*, 2016: IEEE, pp. 37-40.
[57] T. Czimmermann *et al.*, "Visual-based defect detection and classification approaches for industrial applications—a survey," *Sensors,* vol. 20, no. 5, p. 1459, 2020.
[58] Z. Ren, F. Fang, N. Yan, and Y. Wu, "State of the art in defect detection based on machine vision," *International Journal of Precision Engineering and Manufacturing-Green Technology,* pp. 1-31, 2021.
[59] W. Hou, D. Zhang, Y. Wei, J. Guo, and X. Zhang, "Review on computer aided weld defect detection from radiography images," *Applied Sciences,* vol. 10, no. 5, p. 1878, 2020.
[60] O. Duran, K. Althoefer, and L. D. Seneviratne, "Automated pipe defect detection and categorization using camera/laser-based profiler and artificial neural network," *IEEE Transactions on Automation Science and Engineering,* vol. 4, no. 1, pp. 118-126, 2007.
[61] M. Chang, B.-C. Chen, J. L. Gabayno, and M.-F. Chen, "Development of an optical inspection platform for surface defect detection in touch panel glass," *International Journal of Optomechatronics,* vol. 10, no. 2, pp. 63-72, 2016.
[62] C. Bakolias and A. K. Forrest, "Dark-field Scheimpflug imaging for surface inspection," in *Machine Vision Applications in Industrial Inspection V*, 1997, vol. 3029: SPIE, pp. 57-68.
[63] Z. Xue-Wu, D. Yan-Qiong, L. Yan-Yun, S. Ai-Ye, and L. Rui-Yu, "A vision

inspection system for the surface defects of strongly reflected metal based on multi-class SVM," *Expert Systems with Applications,* vol. 38, no. 5, pp. 5930-5939, 2011.
[64] R. Zhang, P.-S. Tsai, J. E. Cryer, and M. Shah, "Shape-from-shading: a survey," *IEEE transactions on pattern analysis and machine intelligence,* vol. 21, no. 8, pp. 690-706, 1999.
[65] J. Aloimonos, "Shape from texture," *Biological cybernetics,* vol. 58, no. 5, pp. 345-360, 1988.
[66] J. Wu *et al.*, "Single image 3d interpreter network," in *European Conference on Computer Vision*, 2016: Springer, pp. 365-382.
[67] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, "A comparison and evaluation of multi-view stereo reconstruction algorithms," in *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, 2006, vol. 1: IEEE, pp. 519-528.
[68] S. Ullman, "The interpretation of structure from motion," *Proceedings of the Royal Society of London. Series B. Biological Sciences,* vol. 203, no. 1153, pp. 405-426, 1979.
[69] H. Kawasaki, R. Furukawa, R. Sagawa, and Y. Yagi, "Dynamic scene shape reconstruction using a single structured light pattern," in *2008 IEEE conference on computer vision and pattern recognition*, 2008: Ieee, pp. 1-8.
[70] Z. Xiong, Q. Li, Q. Mao, and Q. Zou, "A 3D laser profiling system for rail surface defect detection," *Sensors,* vol. 17, no. 8, p. 1791, 2017.
[71] K. Song and Y. Yan, "A noise robust method based on completed local binary patterns for hot-rolled steel strip surface defects," *Applied Surface Science,* vol. 285, pp. 858-864, 2013.
[72] X. Lv, F. Duan, J.-j. Jiang, X. Fu, and L. Gan, "Deep metallic surface defect detection: The new benchmark and detection network," *Sensors,* vol. 20, no. 6, p. 1562, 2020.
[73] J. Božič, D. Tabernik, and D. Skočaj, "Mixed supervision for surface-defect detection: From weakly to fully supervised learning," *Computers in Industry,* vol. 129, p. 103459, 2021.
[74] G. Song, K. Song, and Y. Yan, "Saliency detection for strip steel surface defects using multiple constraints and improved texture features," *Optics and Lasers in Engineering,* vol. 128, p. 106000, 2020.
[75] Y. Huang, C. Qiu, and K. Yuan, "Surface defect saliency of magnetic tile," *The Visual Computer,* vol. 36, no. 1, pp. 85-96, 2020.
[76] Y. Liu, J. Yao, X. Lu, R. Xie, and L. J. N. Li, "DeepCrack: A deep hierarchical feature learning architecture for crack segmentation," vol. 338, pp. 139-153, 2019.
[77] P. Bergmann, X. Jin, D. Sattlegger, and C. Steger, "The mvtec 3d-ad dataset for unsupervised 3d anomaly detection and localization," *arXiv preprint arXiv:2112.09045,* 2021.
[78] S. M. Pizer *et al.*, "Adaptive histogram equalization and its variations," *Computer vision, graphics, and image processing,* vol. 39, no. 3, pp. 355-368, 1987.
[79] F. Russo, "An image enhancement technique combining sharpening and noise reduction," *IEEE Transactions on Instrumentation and Measurement,* vol. 51, no. 4, pp. 824-828, 2002.

[80] H. J. Nussbaumer and H. J. Nussbaumer, *The fast Fourier transform*. Springer, 1982.
[81] Y. Han and P. Shi, "An adaptive level-selecting wavelet transform for texture defect detection," *Image and Vision Computing,* vol. 25, no. 8, pp. 1239-1248, 2007.
[82] J. Chen, J. Benesty, Y. Huang, and S. Doclo, "New insights into the noise reduction Wiener filter," *IEEE Transactions on audio, speech, and language processing,* vol. 14, no. 4, pp. 1218-1234, 2006.
[83] S. Singla and R. Sharma, "Medical image stitching using hybrid of sift & surf techniques," *International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE),* vol. 3, no. 8, pp. 838-842, 2014.
[84] Q. Zhou, B. Zhong, Y. Zhang, J. Li, and Y. Fu, "Deep alignment network based multi-person tracking with occlusion and motion reasoning," *IEEE Transactions on Multimedia,* vol. 21, no. 5, pp. 1183-1194, 2018.
[85] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of big data,* vol. 6, no. 1, pp. 1-48, 2019.
[86] X. Zheng, T. Chalasani, K. Ghosal, S. Lutz, and A. Smolic, "Stada: Style transfer as data augmentation," *arXiv preprint arXiv:1909.01056,* 2019.
[87] D. Shumin, L. Zhoufeng, and L. Chunlei, "AdaBoost learning for fabric defect detection based on HOG and SVM," in *2011 International conference on multimedia technology*, 2011: IEEE, pp. 2903-2906.
[88] I. Cetiner, A. A. Var, and H. Cetiner, "Classification of knot defect types using wavelets and KNN," *Elektronika ir elektrotechnika,* vol. 22, no. 6, pp. 67-72, 2016.
[89] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size," *arXiv preprint arXiv:1602.07360,* 2016.
[90] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proceedings of the AAAI conference on artificial intelligence*, 2017, vol. 31, no. 1.
[91] L. Zhang, F. Yang, Y. D. Zhang, and Y. J. Zhu, "Road crack detection using deep convolutional neural network," in *2016 IEEE international conference on image processing (ICIP)*, 2016: IEEE, pp. 3708-3712.
[92] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440-1448.
[93] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, "A Review of Yolo algorithm developments," *Procedia Computer Science,* vol. 199, pp. 1066-1073, 2022.
[94] W. Liu *et al.*, "Ssd: Single shot multibox detector," in *European conference on computer vision*, 2016: Springer, pp. 21-37.
[95] G. B. Coleman and H. C. Andrews, "Image segmentation by clustering," *Proceedings of the IEEE,* vol. 67, no. 5, pp. 773-785, 1979.
[96] R. Pohle and K. D. Toennies, "Segmentation of medical images using adaptive region growing," in *Medical Imaging 2001: Image Processing*, 2001, vol. 4322: SPIE, pp. 1337-1346.
[97] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1520-1528.
[98] N. Siddique, S. Paheding, C. P. Elkin, and V. Devabhaktuni, "U-net and its

variants for medical image segmentation: A review of theory and applications," *Ieee Access,* vol. 9, pp. 82031-82057, 2021.
[99] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961-2969.
[100] A. Kirillov *et al.*, "Segment anything," *arXiv preprint arXiv:2304.02643,* 2023.
[101] S. He, R. Bao, J. Li, P. E. Grant, and Y. Ou, "Accuracy of segment-anything model (sam) in medical image segmentation tasks," *arXiv preprint arXiv:2304.09324,* 2023.
[102] M. Ahmadi, A. G. Lonbar, A. Sharifi, A. T. Beris, M. Nouri, and A. S. Javidi, "Application of segment anything model for civil infrastructure defect assessment," *arXiv preprint arXiv:2304.12600,* 2023.
[103] Z. Chen, W. K. Wong, Z. Zhong, J. Liao, and Y. Qu, "Effective Transfer of Pretrained Large Visual Model for Fabric Defect Segmentation via Specifc Knowledge Injection," *arXiv preprint arXiv:2306.16186,* 2023.
[104] G.-P. Ji, D.-P. Fan, P. Xu, M.-M. Cheng, B. Zhou, and L. Van Gool, "SAM Struggles in Concealed Scenes--Empirical Study on" Segment Anything"," *arXiv preprint arXiv:2304.06022,* 2023.
[105] W. Ji, J. Li, Q. Bi, W. Li, and L. Cheng, "Segment anything is not always perfect: An investigation of sam on different real-world applications," *arXiv preprint arXiv:2304.05750,* 2023.
[106] L. M. Song, M. P. Wang, L. Lu, and H. J. Huan, "High precision camera calibration in vision measurement," *Optics & Laser Technology,* vol. 39, no. 7, pp. 1413-1420, 2007.
[107] S. Dworkin and T. Nye, "Image processing for machine vision measurement of hot formed parts," *Journal of materials processing technology,* vol. 174, no. 1-3, pp. 1-6, 2006.
[108] R. G. Lins and S. N. Givigi, "Automatic crack detection and measurement based on image analysis," *IEEE Transactions on Instrumentation and Measurement,* vol. 65, no. 3, pp. 583-590, 2016.
[109] D. Huang, S. Du, G. Li, C. Zhao, and Y. Deng, "Detection and monitoring of defects on three-dimensional curved surfaces based on high-density point cloud data," *Precision Engineering,* vol. 53, pp. 79-95, 2018.
[110] Y. Tang, Q. Wang, H. Wang, J. Li, and Y. Ke, "A novel 3D laser scanning defect detection and measurement approach for automated fibre placement," *Measurement Science and Technology,* vol. 32, no. 7, p. 075201, 2021.
[111] A. G. Howard *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017.
[112] Y. Guo, H. Shi, A. Kumar, K. Grauman, T. Rosing, and R. Feris, "Spottune: transfer learning through adaptive fine-tuning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4805-4814.
[113] C. Pinto, Y. Gkoufas, A. Reale, S. Seelam, and S. Eliuk, "Hoard: A distributed data caching system to accelerate deep learning training on the cloud," *arXiv preprint arXiv:1812.00669,* 2018.
[114] A. Polino, R. Pascanu, and D. Alistarh, "Model compression via distillation and quantization," *arXiv preprint arXiv:1802.05668,* 2018.
[115] T. Hoefler, D. Alistarh, T. Ben-Nun, N. Dryden, and A. Peste, "Sparsity in deep learning: Pruning and growth for efficient inference and training in neural

networks," *The Journal of Machine Learning Research,* vol. 22, no. 1, pp. 10882-11005, 2021.
[116] T. Chen *et al.*, "{TVM}: An automated {End-to-End} optimizing compiler for deep learning," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, 2018, pp. 578-594.
[117] X. Hu, L. Chu, J. Pei, W. Liu, and J. Bian, "Model complexity of deep learning: A survey," *Knowledge and Information Systems,* vol. 63, pp. 2585-2619, 2021.
[118] D. Racki, D. Tomazevic, and D. Skocaj, "A compact convolutional neural network for textured surface anomaly detection," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018: IEEE, pp. 1331-1339.
[119] Y. Li, H. Huang, Q. Xie, L. Yao, and Q. Chen, "Research on a surface defect detection algorithm based on MobileNet-SSD," *Applied Sciences,* vol. 8, no. 9, p. 1678, 2018.
[120] C. Goutte and E. Gaussier, "A probabilistic interpretation of precision, recall and F-score, with implication for evaluation," in *European conference on information retrieval*, 2005: Springer, pp. 345-359.
[121] M. H. Zweig and G. Campbell, "Receiver-operating characteristic (ROC) plots: a fundamental evaluation tool in clinical medicine," *Clinical chemistry,* vol. 39, no. 4, pp. 561-577, 1993.
[122] M. A. Rahman and Y. Wang, "Optimizing intersection-over-union in deep neural networks for image segmentation," in *International symposium on visual computing*, 2016: Springer, pp. 234-244.
[123] R. R. Shamir, Y. Duchin, J. Kim, G. Sapiro, and N. Harel, "Continuous dice coefficient: a method for evaluating probabilistic segmentations," *arXiv preprint arXiv:1906.11031,* 2019.
[124] P. Henderson and V. Ferrari, "End-to-end training of object class detectors for mean average precision," in *Computer Vision–ACCV 2016: 13th Asian Conference on Computer Vision, Taipei, Taiwan, November 20-24, 2016, Revised Selected Papers, Part V 13*, 2017: Springer, pp. 198-213.
[125] M. French, "Conceptual design for engineers. Springer, Berlin Heidelberg New York," 1999.
[126] W. Beitz, G. Pahl, and K. Grote, "Engineering design: a systematic approach," *Mrs Bulletin,* vol. 71, 1996.
[127] X.-T. Yan and R. Zante, "A mechatronic design process and its application," in *Mechatronics in Action: Case Studies in Mechatronics–Applications and Education*: Springer, 2010, pp. 55-70.
[128] V. VDI, "2206. Design Handbook 2206, Design methodology for mechatronic systems," ed: VDI Publishing Group, Düsseldorf, 2003.
[129] S. Saha and S. S. Bhattacharyya, "Design methodology for embedded computer vision systems," in *Embedded computer vision*: Springer, 2009, pp. 27-47.
[130] T. Menzies and P. Haynes, "The methodology of methodologies, or, evaluating current methodologies: Why and how," in *Technology of Object-Oriented Languages and Systems: TOOLS*, 1994, vol. 15, pp. 83-92.
[131] M. Sharifzadeh, R. Amirfattahi, S. Sadri, S. Alirezaee, and M. Ahmadi, "Detection of steel defect using the image processing algorithms," in *The International Conference on Electrical Engineering*, 2008, vol. 6, no. 6th

International Conference on Electrical Engineering ICEENG 2008: Military Technical College, pp. 1-7.
[132] Z. Ren, F. Fang, N. Yan, and Y. Wu, "State of the art in defect detection based on machine vision," *International Journal of Precision Engineering and Manufacturing-Green Technology,* vol. 9, no. 2, pp. 661-691, 2022.
[133] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, "Deep learning for anomaly detection: A review," *ACM computing surveys (CSUR),* vol. 54, no. 2, pp. 1-38, 2021.
[134] Z. Hocenski, S. Vasilic, and V. Hocenski, "Improved canny edge detector in ceramic tiles defect detection," in *IECON 2006-32nd Annual Conference on IEEE Industrial Electronics*, 2006: IEEE, pp. 3328-3331.
[135] C. Topal and C. Akinlar, "Edge drawing: a combined real-time edge and segment detector," *Journal of Visual Communication and Image Representation,* vol. 23, no. 6, pp. 862-872, 2012.
[136] B. Kundu, K. White, and C. Mastrangelo, "Defect clustering and classification for semiconductor devices," in *The 2002 45th Midwest Symposium on Circuits and Systems, 2002. MWSCAS-2002.*, 2002, vol. 2: IEEE, pp. II-II.
[137] F. G. Bulnes, R. Usamentiaga, D. F. García, and J. Molleda, "Vision-based sensor for early detection of periodical defects in web materials," *Sensors,* vol. 12, no. 8, pp. 10788-10809, 2012.
[138] K. P. White, B. Kundu, and C. M. Mastrangelo, "Classification of defect clusters on semiconductor wafers via the Hough transformation," *IEEE Transactions on Semiconductor Manufacturing,* vol. 21, no. 2, pp. 272-278, 2008.
[139] F.-l. Wang and B. Zuo, "Detection of surface cutting defect on magnet using Fourier image reconstruction," *Journal of Central South University,* vol. 23, no. 5, pp. 1123-1131, 2016.
[140] C.-h. Chan and G. K. Pang, "Fabric defect detection by Fourier analysis," *IEEE transactions on Industry Applications,* vol. 36, no. 5, pp. 1267-1276, 2000.
[141] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International journal of computer vision,* vol. 1, no. 4, pp. 321-331, 1988.
[142] Q. Yang and J. A. Marchant, "Accurate blemish detection with active contour models," *Computers and electronics in agriculture,* vol. 14, no. 1, pp. 77-89, 1996.
[143] J. Yang *et al.*, "Development of an optical defect inspection algorithm based on an active contour model for large steel roller surfaces," *Applied optics,* vol. 57, no. 10, pp. 2490-2498, 2018.
[144] P. Roy, S. Dutta, N. Dey, G. Dey, S. Chakraborty, and R. Ray, "Adaptive thresholding: A comparative study," in *2014 International conference on control, Instrumentation, communication and Computational Technologies (ICCICCT)*, 2014: IEEE, pp. 1182-1186.
[145] R. M. Haralick, K. Shanmugam, and I. H. Dinstein, "Textural features for image classification," *IEEE Transactions on systems, man, and cybernetics,* no. 6, pp. 610-621, 1973.
[146] D.-M. Tsai, M.-C. Chen, W.-C. Li, and W.-Y. Chiu, "A fast regularity measure for surface defect detection," *Machine Vision and applications,* vol. 23, pp. 869-886, 2012.
[147] T. Ojala, M. Pietikäinen, and D. Harwood, "A comparative study of texture

measures with classification based on featured distributions," *Pattern recognition,* vol. 29, no. 1, pp. 51-59, 1996.
[148] Z. Guo, L. Zhang, and D. Zhang, "A completed modeling of local binary pattern operator for texture classification," *IEEE transactions on image processing,* vol. 19, no. 6, pp. 1657-1663, 2010.
[149] Y.-h. Ai and K. Xu, "Surface detection of continuous casting slabs based on curvelet transform and kernel locality preserving projections," *Journal of Iron and Steel Research International,* vol. 20, no. 5, pp. 80-86, 2013.
[150] B. Zorić, T. Matić, and Ž. Hocenski, "Classification of biscuit tiles for defect detection using Fourier transform features," *ISA transactions,* vol. 125, pp. 400-414, 2022.
[151] D. Gabor, "Theory of communication. Part 1: The analysis of information," *Journal of the Institution of Electrical Engineers-part III: radio and communication engineering,* vol. 93, no. 26, pp. 429-441, 1946.
[152] G. H. Granlund, "In search of a general picture processing operator," *Computer Graphics and Image Processing,* vol. 8, no. 2, pp. 155-173, 1978.
[153] A. Kumar and G. K. Pang, "Defect detection in textured materials using Gabor filters," *IEEE Transactions on industry applications,* vol. 38, no. 2, pp. 425-440, 2002.
[154] J. L. Raheja, S. Kumar, and A. Chaudhary, "Fabric defect detection based on GLCM and Gabor filter: A comparison," *Optik,* vol. 124, no. 23, pp. 6469-6474, 2013.
[155] L. Bissi, G. Baruffa, P. Placidi, E. Ricci, A. Scorzoni, and P. Valigi, "Automated defect detection in uniform and structured fabrics using Gabor filters and PCA," *Journal of Visual Communication and Image Representation,* vol. 24, no. 7, pp. 838-845, 2013.
[156] G. Lambert and F. Bock, "Wavelet methods for texture defect detection," in *proceedings of international conference on image processing*, 1997, vol. 3: IEEE, pp. 201-204.
[157] X. Zhou *et al.*, "A surface defect detection framework for glass bottle bottom using visual attention model and wavelet transform," *IEEE Transactions on Industrial Informatics,* vol. 16, no. 4, pp. 2189-2201, 2019.
[158] T. Wang, Y. Chen, M. Qiao, and H. Snoussi, "A fast and robust convolutional neural network-based defect detection model in product quality control," *The International Journal of Advanced Manufacturing Technology,* vol. 94, no. 9, pp. 3465-3471, 2018.
[159] Y. He, K. Song, Q. Meng, and Y. Yan, "An end-to-end steel surface defect detection approach via fusing multiple hierarchical features," *IEEE Transactions on Instrumentation and Measurement,* vol. 69, no. 4, pp. 1493-1504, 2019.
[160] X. Cheng and J. Yu, "RetinaNet with difference channel attention and adaptively spatial feature fusion for steel surface defect detection," *IEEE Transactions on Instrumentation and Measurement,* vol. 70, pp. 1-11, 2020.
[161] M. S. Dizaji and D. K. Harris, "3D InspectionNet: a deep 3D convolutional neural networks based approach for 3D defect detection on concrete columns," in *Nondestructive Characterization and Monitoring of Advanced Materials, Aerospace, Civil Infrastructure, and Transportation XIII*, 2019, vol. 10971: SPIE, pp. 67-77.

[162] K.-H. Lee, H. W. Lee, and G. J. Yun, "A defect detection framework using three-dimensional convolutional neural network (3D-CNN) with in-situ monitoring data in laser powder bed fusion process," *Optics & Laser Technology,* vol. 165, p. 109571, 2023.
[163] K. Han *et al.*, "A survey on vision transformer," *IEEE transactions on pattern analysis and machine intelligence,* vol. 45, no. 1, pp. 87-110, 2022.
[164] L. M. Dang, H. Wang, Y. Li, T. N. Nguyen, and H. Moon, "DefectTR: End-to-end defect detection for sewage networks using a transformer," *Construction and Building Materials,* vol. 325, p. 126584, 2022.
[165] K. An and Y. Zhang, "LPViT: a transformer based model for PCB image classification and defect detection," *IEEE Access,* vol. 10, pp. 42542-42553, 2022.
[166] J. Wang, G. Xu, F. Yan, J. Wang, and Z. Wang, "Defect transformer: An efficient hybrid transformer architecture for surface defect detection," *Measurement,* vol. 211, p. 112614, 2023.
[167] D.-M. Tsai, G.-N. Li, W.-C. Li, and W.-Y. Chiu, "Defect detection in multi-crystal solar cells using clustering with uniformity measures," *Advanced Engineering Informatics,* vol. 29, no. 3, pp. 419-430, 2015.
[168] K. Zheng, Y.-S. Chang, K.-H. Wang, and Y. Yao, "Thermographic clustering analysis for defect detection in CFRP structures," *Polymer Testing,* vol. 49, pp. 73-81, 2016.
[169] S.-S. Yang, Y.-H. He, Z.-L. Wang, and W.-S. Zhao, "A method of steel strip image segmentation based on local gray information," in *2008 IEEE International Conference on Industrial Technology*, 2008: IEEE, pp. 1-4.
[170] Y. Ma, Q. Li, Y. Zhou, F. He, and S. Xi, "A surface defects inspection method based on multidirectional gray-level fluctuation," *International Journal of Advanced Robotic Systems,* vol. 14, no. 3, p. 1729881417703114, 2017.
[171] I. Jovančević *et al.*, "3D point cloud analysis for detection and characterization of defects on airplane exterior surface," *Journal of Nondestructive Evaluation,* vol. 36, pp. 1-17, 2017.
[172] F. Tajeripour, E. Kabir, and A. Sheikhi, "Fabric defect detection using modified local binary patterns," *EURASIP Journal on Advances in Signal Processing,* vol. 2008, pp. 1-12, 2007.
[173] Q. Luo, Y. Sun, P. Li, O. Simpson, L. Tian, and Y. He, "Generalized completed local binary patterns for time-efficient steel surface defect classification," *IEEE Transactions on Instrumentation and Measurement,* vol. 68, no. 3, pp. 667-679, 2018.
[174] Y. Zhang, Z. Tang, and R. Yang, "Data anomaly detection for structural health monitoring by multi-view representation based on local binary patterns," *Measurement,* vol. 202, p. 111804, 2022.
[175] T. Kim and C. H. Park, "Anomaly pattern detection in streaming data based on the transformation to multiple binary-valued data streams," *Journal of Artificial Intelligence and Soft Computing Research,* vol. 12, no. 1, pp. 19-27, 2022.
[176] A. Çalışkan, "A new ensemble approach for congestive heart failure and arrhythmia classification using shifted one-dimensional local binary patterns with long short-term memory," *The Computer Journal,* vol. 65, no. 9, pp. 2535-2546, 2022.

[177] C. Ciamberlini, F. Francini, G. Longobardi, P. Poggi, P. Sansoni, and B. Tiribilli, "Weaving defect detection by Fourier imaging," in *Vision Systems: Applications*, 1996, vol. 2786: SPIE, pp. 9-18.
[178] D.-M. Tsai and C.-Y. Hsieh, "Automated surface inspection for directional textures," *Image and Vision computing,* vol. 18, no. 1, pp. 49-62, 1999.
[179] A. Noiboar and I. Cohen, "Anomaly detection in three dimensional data based on Gauss Markov random field modeling," in *2004 23rd IEEE Convention of Electrical and Electronics Engineers in Israel*, 2004: IEEE, pp. 448-451.
[180] A. Goldman and I. Cohen, "Anomaly subspace detection based on a multi-scale Markov random field model," *Signal Processing,* vol. 85, no. 3, pp. 463-479, 2005.
[181] G.-H. Hu, G.-H. Zhang, and Q.-H. Wang, "Automated defect detection in textured materials using wavelet-domain hidden Markov models," *Optical Engineering,* vol. 53, no. 9, pp. 093107-093107, 2014.
[182] H.-g. Bu, J. Wang, and X.-b. Huang, "Fabric defect detection based on multiple fractal features and support vector data description," *Engineering Applications of Artificial Intelligence,* vol. 22, no. 2, pp. 224-235, 2009.
[183] Y.-H. Liu, Y.-C. Liu, and Y.-Z. Chen, "High-speed inline defect detection for TFT-LCD array process using a novel support vector data description," *Expert Systems with Applications,* vol. 38, no. 5, pp. 6222-6231, 2011.
[184] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special lecture on IE,* vol. 2, no. 1, pp. 1-18, 2015.
[185] S. Akcay, A. Atapour-Abarghouei, and T. P. Breckon, "Ganomaly: Semi-supervised anomaly detection via adversarial training," in *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part III 14*, 2019: Springer, pp. 622-637.
[186] P. Perera, R. Nallapati, and B. Xiang, "Ocgan: One-class novelty detection using gans with constrained latent representations," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 2898-2906.
[187] N. Cohen and Y. Hoshen, "Sub-image anomaly detection with deep pyramid correspondences," *arXiv preprint arXiv:2005.02357,* 2020.
[188] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger, "Uninformed students: Student-teacher anomaly detection with discriminative latent embeddings," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 4183-4192.
[189] D. Lu, X. Liao, F. Xu, and J. Bai, "Anomaly detection method for substation equipment based on feature matching and multi-Semantic classification," in *2021 6th Asia Conference on Power and Electrical Engineering (ACPEE)*, 2021: IEEE, pp. 109-113.
[190] X. Zhou, G. Liu, X. Zhang, B. D. Prasad, X. Gu, and Y. Li, "Re2FAD: A differential image registration and robust image fusion method framework for power thermal anomaly detection," *Optik,* vol. 259, p. 168817, 2022.
[191] P. Hong-Seok and T. U. Mani, "Development of an inspection system for defect detection in pressed parts using laser scanned data," *Procedia Engineering,* vol. 69, pp. 931-936, 2014.
[192] M. Chen, T. Kanade, D. Pomerleau, and H. A. Rowley, "Anomaly detection

through registration," *Pattern Recognition,* vol. 32, no. 1, pp. 113-128, 1999.
[193] R. Ren, T. Hung, and K. C. Tan, "A generic deep-learning-based approach for automated surface inspection," *IEEE transactions on cybernetics,* vol. 48, no. 3, pp. 929-940, 2017.
[194] J. C. Cheng and M. Wang, "Automated detection of sewer pipe defects in closed-circuit television images using deep learning techniques," *Automation in Construction,* vol. 95, pp. 155-171, 2018.
[195] Y. Liu, K. Xu, and J. Xu, "Periodic surface defect detection in steel plates based on deep learning," *Applied Sciences,* vol. 9, no. 15, p. 3127, 2019.
[196] S. Y. Lee, B. A. Tama, S. J. Moon, and S. Lee, "Steel surface defect diagnostics using deep convolutional neural network and class activation map," *Applied Sciences,* vol. 9, no. 24, p. 5449, 2019.
[197] F. Chang, M. Liu, M. Dong, and Y. Duan, "A mobile vision inspection system for tiny defect detection on smooth car-body surfaces based on deep ensemble learning," *Measurement Science and Technology,* vol. 30, no. 12, p. 125905, 2019.
[198] Y. Huang, C. Qiu, X. Wang, S. Wang, and K. Yuan, "A compact convolutional neural network for surface defect inspection," *Sensors,* vol. 20, no. 7, p. 1974, 2020.
[199] Y. Liu, Y. Yuan, C. Balta, and J. Liu, "A light-weight deep-learning model with multi-scale features for steel surface defect classification," *Materials,* vol. 13, no. 20, p. 4629, 2020.
[200] L. Xiao, B. Wu, and Y. Hu, "Surface defect detection using image pyramid," *IEEE Sensors Journal,* vol. 20, no. 13, pp. 7181-7188, 2020.
[201] X. Tao, D. Zhang, W. Hou, W. Ma, and D. Xu, "Industrial weak scratches inspection based on multifeature fusion network," *IEEE Transactions on Instrumentation and Measurement,* vol. 70, pp. 1-14, 2020.
[202] R. Neven and T. Goedemé, "A multi-branch U-Net for steel surface defect type and severity segmentation," *Metals,* vol. 11, no. 6, p. 870, 2021.
[203] J. Luo, Z. Yang, S. Li, and Y. Wu, "FPCB surface defect detection: A decoupled two-stage object detection framework," *IEEE Transactions on Instrumentation and Measurement,* vol. 70, pp. 1-11, 2021.
[204] D.-M. Tsai, S.-K. S. Fan, and Y.-H. Chou, "Auto-annotated deep segmentation for surface defect detection," *IEEE Transactions on Instrumentation and Measurement,* vol. 70, pp. 1-10, 2021.
[205] L. Cui, X. Jiang, M. Xu, W. Li, P. Lv, and B. Zhou, "SDDNet: A fast and accurate network for surface defect detection," *IEEE Transactions on Instrumentation and Measurement,* vol. 70, pp. 1-13, 2021.
[206] Y. Wu, Y. Qin, Y. Qian, F. Guo, Z. Wang, and L. Jia, "Hybrid deep learning architecture for rail surface segmentation and surface defect detection," *Computer‐Aided Civil and Infrastructure Engineering,* vol. 37, no. 2, pp. 227-244, 2022.
[207] M. Chen *et al.*, "Improved faster R-CNN for fabric defect detection based on Gabor filter with Genetic Algorithm optimization," *Computers in Industry,* vol. 134, p. 103551, 2022.
[208] P. Lu, J. Jing, and Y. Huang, "MRD-net: An effective CNN-based segmentation network for surface defect detection," *IEEE Transactions on Instrumentation*

and Measurement, vol. 71, pp. 1-12, 2022.
[209] Y. Liu, H. Xiao, J. Xu, and J. Zhao, "A rail surface defect detection method based on pyramid feature and lightweight convolutional neural network," *IEEE Transactions on Instrumentation and Measurement,* vol. 71, pp. 1-10, 2022.
[210] L. Gao, J. Zhang, C. Yang, and Y. Zhou, "Cas-VSwin transformer: A variant swin transformer for surface-defect detection," *Computers in Industry,* vol. 140, p. 103689, 2022.
[211] W. Li *et al.*, "Deep learning based online metallic surface defect detection method for wire and arc additive manufacturing," *Robotics and Computer-Integrated Manufacturing,* vol. 80, p. 102470, 2023.
[212] Y. Zhang *et al.*, "Development of a cross-scale weighted feature fusion network for hot-rolled steel surface defect detection," *Engineering Applications of Artificial Intelligence,* vol. 117, p. 105628, 2023.
[213] J. Yi and S. Yoon, "Patch svdd: Patch-level svdd for anomaly detection and segmentation," in *Proceedings of the Asian conference on computer vision*, 2020.
[214] C. Hu, K. Chen, and H. Shao, "A semantic-enhanced method based on deep SVDD for pixel-wise anomaly detection," in *2021 IEEE International Conference on Multimedia and Expo (ICME)*, 2021: IEEE, pp. 1-6.
[215] T. Matsubara, K. Sato, K. Hama, R. Tachibana, and K. Uehara, "Deep generative model using unregularized score for anomaly detection with heterogeneous complexity," *IEEE Transactions on Cybernetics,* vol. 52, no. 6, pp. 5161-5173, 2020.
[216] W. Liu *et al.*, "Towards visually explaining variational autoencoders," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8642-8651.
[217] C.-C. Tsai, T.-H. Wu, and S.-H. Lai, "Multi-scale patch-based representation learning for image anomaly detection and segmentation," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022, pp. 3992-4000.
[218] C.-L. Li, K. Sohn, J. Yoon, and T. Pfister, "Cutpaste: Self-supervised learning for anomaly detection and localization," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 9664-9674.
[219] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery," in *International conference on information processing in medical imaging*, 2017: Springer, pp. 146-157.
[220] D. Gudovskiy, S. Ishizaka, and K. Kozuka, "Cflow-ad: Real-time unsupervised anomaly detection with localization via conditional normalizing flows," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022, pp. 98-107.
[221] J. Yu *et al.*, "Fastflow: Unsupervised anomaly detection and localization via 2d normalizing flows," *arXiv preprint arXiv:2111.07677,* 2021.
[222] T. Defard, A. Setkov, A. Loesch, and R. Audigier, "Padim: a patch distribution modeling framework for anomaly detection and localization," in *International Conference on Pattern Recognition*, 2021: Springer, pp. 475-489.
[223] K. Roth, L. Pemula, J. Zepeda, B. Schölkopf, T. Brox, and P. Gehler, "Towards total recall in industrial anomaly detection," in *Proceedings of the IEEE/CVF*

Conference on Computer Vision and Pattern Recognition, 2022, pp. 14318-14328.
[224] G. Wang, S. Han, E. Ding, and D. Huang, "Student-teacher feature pyramid matching for anomaly detection," *arXiv preprint arXiv:2103.04257,* 2021.
[225] H. Deng and X. Li, "Anomaly detection via reverse distillation from one-class embedding," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 9737-9746.
[226] K. Batzner, L. Heckler, and R. König, "Efficientad: Accurate visual anomaly detection at millisecond-level latencies," *arXiv preprint arXiv:2303.14535,* 2023.
[227] C. Huang, H. Guan, A. Jiang, Y. Zhang, M. Spratling, and Y.-F. Wang, "Registration based few-shot anomaly detection," in *European Conference on Computer Vision*, 2022: Springer, pp. 303-319.
[228] Y. Zheng, X. Wang, R. Deng, T. Bao, R. Zhao, and L. Wu, "Focus your distribution: Coarse-to-fine non-contrastive learning for anomaly detection and localization," in *2022 IEEE International Conference on Multimedia and Expo (ICME)*, 2022: IEEE, pp. 1-6.
[229] J. C. Borg, X.-T. Yan, and N. P. Juster, "Exploring decisions' influence on life-cycle performance to aid “design for Multi-X”," *AI EDAM,* vol. 14, no. 2, pp. 91-113, 2000.
[230] X.-T. Yan, "A multiple perspective product modeling and simulation approach to engineering design support," *Concurrent Engineering,* vol. 11, no. 3, pp. 221-234, 2003.
[231] P. Prasanna *et al.*, "Automated crack detection on concrete bridges," *IEEE Transactions on automation science and engineering,* vol. 13, no. 2, pp. 591-599, 2014.
[232] M. Lehman, "The american society of civil engineers’ report card on america’s infrastructure," in *Women in Infrastructure*: Springer, 2022, pp. 5-21.
[233] J. Act, "Infrastructure Investment and Jobs Act," 2021.
[234] X. Ji, Z. Miao, and R. Kromanis, "Vision-based measurements of deformations and cracks for RC structure tests," *Engineering Structures,* vol. 212, p. 110508, 2020.
[235] D. Ai, G. Jiang, S.-K. Lam, P. He, and C. Li, "Computer vision framework for crack detection of civil infrastructure—A review," *Engineering Applications of Artificial Intelligence,* vol. 117, p. 105478, 2023.
[236] B. F. Spencer Jr, V. Hoskere, and Y. Narazaki, "Advances in computer vision-based civil infrastructure inspection and monitoring," *Engineering,* vol. 5, no. 2, pp. 199-222, 2019.
[237] K. Xu and Y. Lu, "Numerical simulation study of spallation in reinforced concrete plates subjected to blast loading," *Computers & structures,* vol. 84, no. 5-6, pp. 431-438, 2006.
[238] N.-D. Hoang and Q.-L. Nguyen, "Metaheuristic optimized edge detection for recognition of concrete wall cracks: a comparative study on the performances of roberts, prewitt, canny, and sobel algorithms," *Advances in Civil Engineering,* vol. 2018, pp. 1-16, 2018.
[239] Y. Hamishebahar, H. Guan, S. So, and J. Jo, "A Comprehensive Review of Deep Learning-Based Crack Detection Approaches," *Applied Sciences,* vol. 12, no. 3, p. 1374, 2022.

[240] M. M. Islam, M. B. Hossain, M. N. Akhtar, M. A. Moni, and K. F. Hasan, "CNN based on transfer learning models using data augmentation and transformation for detection of concrete crack," *Algorithms,* vol. 15, no. 8, p. 287, 2022.
[241] Ç. F. Özgenel, "Concrete crack images for classification," *Mendeley Data, v1 http://dx. doi. org/10.17632/5y9wdsg2zt,* vol. 1, 2018.
[242] M. Maguire, S. Dorafshan, and R. J. Thomas, "SDNET2018: A concrete crack image dataset for machine learning applications," 2018.
[243] Y. Liu, J. Yao, X. Lu, R. Xie, and L. Li, "DeepCrack: A deep hierarchical feature learning architecture for crack segmentation," *Neurocomputing,* vol. 338, pp. 139-153, 2019.
[244] S. Kulkarni, S. Singh, D. Balakrishnan, S. Sharma, S. Devunuri, and S. C. R. Korlapati, "CrackSeg9k: a collection and benchmark for crack segmentation datasets and frameworks," in *European Conference on Computer Vision*, 2022: Springer, pp. 179-195.
[245] S. Zhao, F. Kang, and J. Li, "Non-Contact Crack Visual Measurement System Combining Improved U-Net Algorithm and Canny Edge Detection Method with Laser Rangefinder and Camera," *Applied Sciences,* vol. 12, no. 20, p. 10651, 2022.
[246] F. Kucuksubasi and A. Sorguc, "Transfer learning-based crack detection by autonomous UAVs," *arXiv preprint arXiv:1807.11785,* 2018.
[247] T. Yamaguchi, T. Shibuya, M. Kanda, and A. Yasojima, "Crack inspection support system for concrete structures using head mounted display in mixed reality space," in *2019 58th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, 2019: IEEE, pp. 791-796.
[248] Y. J. Cha, W. Choi, and O. Büyüköztürk, "Deep learning‐based crack damage detection using convolutional neural networks," *Computer‐Aided Civil and Infrastructure Engineering,* vol. 32, no. 5, pp. 361-378, 2017.
[249] J. Deng, Y. Lu, and V. C. S. Lee, "Concrete crack detection with handwriting script interferences using faster region‐based convolutional neural network," *Computer‐Aided Civil and Infrastructure Engineering,* vol. 35, no. 4, pp. 373-388, 2020.
[250] X. Yang, H. Li, Y. Yu, X. Luo, T. Huang, and X. Yang, "Automatic pixel‐level crack detection and measurement using fully convolutional network," *Computer‐Aided Civil and Infrastructure Engineering,* vol. 33, no. 12, pp. 1090-1109, 2018.
[251] K. Chen, G. Reichard, X. Xu, and A. Akanmu, "Automated crack segmentation in close-range building façade inspection images using deep learning techniques," *Journal of Building Engineering,* vol. 43, p. 102913, 2021.
[252] Y. Zheng, Y. Gao, S. Lu, and K. M. Mosalam, "Multistage semisupervised active learning framework for crack identification, segmentation, and measurement of bridges," *Computer‐Aided Civil and Infrastructure Engineering,* vol. 37, no. 9, pp. 1089-1108, 2022.
[253] Y. Jiang and C. Zhao, "Attention classification-and-segmentation network for micro-crack anomaly detection of photovoltaic module cells," *Solar Energy,* vol. 238, pp. 291-304, 2022.
[254] G. Yang *et al.*, "Datasets and processing methods for boosting visual inspection of civil infrastructure: A comprehensive review and algorithm comparison for

crack classification, segmentation, and detection," *Construction and Building Materials,* vol. 356, p. 129226, 2022.
[255] F. Fang, L. Li, M. Rice, and J.-H. Lim, "Towards real-time crack detection using a deep neural network with a Bayesian fusion algorithm," in *2019 IEEE international conference on image processing (ICIP)*, 2019: IEEE, pp. 2976-2980.
[256] F.-C. Chen and M. R. Jahanshahi, "NB-FCN: Real-time accurate crack detection in inspection videos using deep fully convolutional network and parametric data fusion," *IEEE Transactions on Instrumentation and Measurement,* vol. 69, no. 8, pp. 5325-5334, 2019.
[257] J. Pang, H. Zhang, H. Zhao, and L. Li, "DcsNet: a real-time deep network for crack segmentation," *Signal, Image and Video Processing,* pp. 1-9, 2022.
[258] S. Jiang and J. Zhang, "Real‐time crack assessment using deep neural networks with wall‐climbing unmanned aerial system," *Computer‐Aided Civil and Infrastructure Engineering,* vol. 35, no. 6, pp. 549-564, 2020.
[259] A. Doulamis, N. Doulamis, E. Protopapadakis, and A. Voulodimos, "Combined convolutional neural networks and fuzzy spectral clustering for real time crack detection in tunnels," in *2018 25th IEEE International Conference on Image Processing (ICIP)*, 2018: IEEE, pp. 4153-4157.
[260] T. Chisholm, R. Lins, and S. Givigi, "FPGA-based design for real-time crack detection based on particle filter," *IEEE Transactions on Industrial Informatics,* vol. 16, no. 9, pp. 5703-5711, 2019.
[261] D. Ma, H. Fang, N. Wang, B. Xue, J. Dong, and F. Wang, "A real-time crack detection algorithm for pavement based on CNN with multiple feature layers," *Road Materials and Pavement Design,* pp. 1-17, 2021.
[262] Z. Yu, Y. Shen, and C. Shen, "A real-time detection approach for bridge cracks based on YOLOv4-FPM," *Automation in Construction,* vol. 122, p. 103514, 2021.
[263] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 18-23 June 2018 2018, pp. 4510-4520, doi: 10.1109/CVPR.2018.00474.
[264] A. G. Howard *et al.*, "Searching for MobileNetV3," *2019 IEEE/CVF International Conference on Computer Vision (ICCV),* pp. 1314-1324, 2019.
[265] M. Tan and Q. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," presented at the Proceedings of the 36th International Conference on Machine Learning, Proceedings of Machine Learning Research, 2019. [Online]. Available: https://proceedings.mlr.press/v97/tan19a.html.
[266] M. Tan and Q. Le, "EfficientNetV2: Smaller Models and Faster Training," presented at the Proceedings of the 38th International Conference on Machine Learning, Proceedings of Machine Learning Research, 2021. [Online]. Available: https://proceedings.mlr.press/v139/tan21a.html.
[267] F. Yu and V. Koltun, "Multi-Scale Context Aggregation by Dilated Convolutions," 2016. [Online]. Available: http://arxiv.org/abs/1511.07122.
[268] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *International conference on artificial neural networks*, 2018: Springer, pp. 270-279.

[269] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering,* vol. 22, no. 10, pp. 1345-1359, 2009.
[270] J. Xie, B. Huang, and S. Dubljevic, "Transfer learning for dynamic feature extraction using variational bayesian inference," *IEEE Transactions on Knowledge and Data Engineering,* 2021.
[271] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, 2009: Ieee, pp. 248-255.
[272] M. Abadi *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467,* 2016.
[273] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems,* vol. 32, 2019.
[274] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.
[275] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531,* vol. 2, no. 7, 2015.
[276] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *International Journal of Computer Vision,* vol. 129, no. 6, pp. 1789-1819, 2021.
[277] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," *arXiv preprint arXiv:1612.03928,* 2016.
[278] W. Liu, D. Gong, M. Tan, J. Q. Shi, Y. Yang, and A. G. Hauptmann, "Learning distilled graph for large-scale social network data clustering," *IEEE Transactions on Knowledge and Data Engineering,* vol. 32, no. 7, pp. 1393-1404, 2019.
[279] F. M. Nardini, C. Rulli, S. Trani, and R. Venturini, "Distilled Neural Networks for Efficient Learning to Rank," *IEEE Transactions on Knowledge and Data Engineering,* 2022.
[280] Z. Liu, X. Gu, H. Yang, L. Wang, Y. Chen, and D. Wang, "Novel YOLOv3 Model With Structure and Hyperparameter Optimization for Detection of Pavement Concealed Cracks in GPR Images," *IEEE Transactions on Intelligent Transportation Systems,* pp. 1-11, 2022, doi: 10.1109/TITS.2022.3174626.
[281] D. Ungureanu *et al.*, "Hololens 2 research mode as a tool for computer vision research," *arXiv preprint arXiv:2008.11239,* 2020.
[282] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking Atrous Convolution for Semantic Image Segmentation," 06/17 2017.
[283] J. Hu, L. Shen, and G. Sun, "Squeeze-and-Excitation Networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 18-23 June 2018 2018, pp. 7132-7141, doi: 10.1109/CVPR.2018.00745.
[284] Y. Shi, L. Cui, Z. Qi, F. Meng, and Z. J. I. T. o. I. T. S. Chen, "Automatic road crack detection using random structured forests," vol. 17, no. 12, pp. 3434-3445, 2016.
[285] S. Marcel and Y. Rodriguez, "Torchvision the machine-vision package of torch," presented at the Proceedings of the 18th ACM international conference on Multimedia, Firenze, Italy, 2010. [Online]. Available: https://doi.org/10.1145/1873951.1874254.

[286] T. Devries and G. W. Taylor, "Improved Regularization of Convolutional Neural Networks with Cutout," *CoRR,* vol. abs/1708.04552, / 2017. [Online]. Available: http://arxiv.org/abs/1708.04552.
[287] G. Aguilar, Y. Ling, Y. Zhang, B. Yao, X. Fan, and C. Guo, "Knowledge distillation from internal representations," in *Proceedings of the AAAI conference on artificial intelligence*, 2020, vol. 34, no. 05, pp. 7350-7357.
[288] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, 2015: Springer, pp. 234-241.
[289] Ç. F. Özgenel. *Concrete Crack Segmentation Dataset*, doi: 10.17632/jwsn7tfbrp.1, doi: 10.17632/jwsn7tfbrp.1.
[290] U. Shafi, R. Mumtaz, J. García-Nieto, S. A. Hassan, S. A. R. Zaidi, and N. Iqbal, "Precision agriculture techniques and practices: From considerations to applications," *Sensors,* vol. 19, no. 17, p. 3796, 2019.
[291] C. Dweba *et al.*, "Fusarium head blight of wheat: Pathogenesis and control strategies," *Crop protection,* vol. 91, pp. 114-122, 2017.
[292] M. McMullen *et al.*, "A unified effort to fight an enemy of wheat and barley: Fusarium head blight," *Plant disease,* vol. 96, no. 12, pp. 1712-1728, 2012.
[293] E. Cambaza, S. Koseki, and S. Kawamura, "Why RGB imaging should be used to analyze Fusarium graminearum growth and estimate deoxynivalenol contamination," *Methods and protocols,* vol. 2, no. 1, p. 25, 2019.
[294] R. Ruan, S. Ning, A. Song, A. Ning, R. Jones, and P. Chen, "Estimation of Fusarium scab in wheat using machine vision and a neural network," *Cereal Chemistry,* vol. 75, no. 4, pp. 455-459, 1998.
[295] L. Qiongyan, J. Cai, B. Berger, M. Okamoto, and S. J. Miklavcic, "Detecting spikes of wheat plants using neural networks with Laws texture energy," *Plant Methods,* vol. 13, pp. 1-13, 2017.
[296] R. Qiu, C. Yang, A. Moghimi, M. Zhang, B. J. Steffenson, and C. D. Hirsch, "Detection of fusarium head blight in wheat using a deep neural network and color imaging," *Remote Sensing,* vol. 11, no. 22, p. 2658, 2019.
[297] S. Khaki, N. Safaei, H. Pham, and L. Wang, "WheatNet: A lightweight convolutional neural network for high-throughput image-based wheat head detection and counting," *Neurocomputing,* vol. 489, pp. 78-89, 2022.
[298] D. Zhang *et al.*, "Using neural network to identify the severity of wheat Fusarium head blight in the field environment," *Remote Sensing,* vol. 11, no. 20, p. 2375, 2019.
[299] J. G. A. Barbedo, "Impact of dataset size and variety on the effectiveness of deep learning and transfer learning for plant disease classification," *Computers and electronics in agriculture,* vol. 153, pp. 46-53, 2018.
[300] M. Hasan, B. Tanawala, and K. J. Patel, "Deep learning precision farming: Tomato leaf disease detection by transfer learning," in *Proceedings of 2nd international conference on advanced computing and software engineering (ICACSE)*, 2019.
[301] Z. K. Hartley and A. P. French, "Domain adaptation of synthetic images for wheat head detection," *Plants,* vol. 10, no. 12, p. 2633, 2021.
[302] E. David *et al.*, "Global wheat head detection 2021: An improved dataset for benchmarking wheat head detection methods," *Plant Phenomics,* 2021.

[303] *Wheat head blight dataset*, Science Data Bank, 2023-08-14, doi: 10.11922/sciencedb.p00001.00008.
[304] D. Rößle *et al.*, "Efficient Noninvasive FHB Estimation using RGB Images from a Novel Multiyear, Multirater Dataset," *Plant Phenomics,* vol. 5, p. 0068, 2023.
[305] L. Ruff *et al.*, "A unifying review of deep and shallow anomaly detection," *Proceedings of the IEEE,* vol. 109, no. 5, pp. 756-795, 2021.
[306] T. Schlegl, P. Seeböck, S. M. Waldstein, G. Langs, and U. Schmidt-Erfurth, "f-AnoGAN: Fast unsupervised anomaly detection with generative adversarial networks," *Medical image analysis,* vol. 54, pp. 30-44, 2019.
[307] H. Jin, Y. Li, J. Qi, J. Feng, D. Tian, and W. Mu, "GrapeGAN: Unsupervised image enhancement for improved grape leaf disease recognition," *Computers and Electronics in Agriculture,* vol. 198, p. 107055, 2022.
[308] S. Bhugra, V. Kaushik, A. Gupta, B. Lall, and S. Chaudhury, "AnoLeaf: Unsupervised Leaf Disease Segmentation via Structurally Robust Generative Inpainting," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023, pp. 6415-6424.
[309] A. Benfenati, P. Causin, R. Oberti, and G. Stefanello, "Unsupervised deep learning techniques for automatic detection of plant diseases: reducing the need of manual labelling of plant images," *Journal of Mathematics in Industry,* vol. 13, no. 1, p. 5, 2023.
[310] T. Kim, H. Kim, K. Baik, and Y. Choi, "Instance-aware plant disease detection by utilizing saliency map and self-supervised pre-training," *Agriculture,* vol. 12, no. 8, p. 1084, 2022.
[311] H. S. Kang *et al.*, "Smart manufacturing: Past research, present findings, and future directions," *International journal of precision engineering and manufacturing-green technology,* vol. 3, pp. 111-128, 2016.
[312] S. Sahoo and C.-Y. Lo, "Smart manufacturing powered by recent technological advancements: A review," *Journal of Manufacturing Systems,* vol. 64, pp. 236-250, 2022.
[313] A. Bailly *et al.*, "Effects of dataset size and interactions on the prediction performance of logistic regression and deep learning models," *Computer Methods and Programs in Biomedicine,* vol. 213, p. 106504, 2022.
[314] P. M. Bhatt *et al.*, "Image-based surface defect detection using deep learning: A review," *Journal of Computing and Information Science in Engineering,* vol. 21, no. 4, p. 040801, 2021.
[315] C. Li, J. Li, Y. Li, L. He, X. Fu, and J. Chen, "Fabric defect detection in textile manufacturing: a survey of the state of the art," *Security and Communication Networks,* vol. 2021, pp. 1-13, 2021.
[316] J. Yang, S. Li, Z. Wang, H. Dong, J. Wang, and S. Tang, "Using deep learning to detect defects in manufacturing: a comprehensive survey and current challenges," *Materials,* vol. 13, no. 24, p. 5755, 2020.
[317] X. Ren, W. Lin, X. Yang, X. Yu, and H. Gao, "Data augmentation in defect detection of sanitary ceramics in small and non-iid datasets," *IEEE Transactions on Neural Networks and Learning Systems,* vol. 34, no. 11, pp. 8669-8678, 2022.
[318] 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,* pp. 1-14, 2022.
[319] Y. Liao, X. Weng, C. Swonger, and J. Ni, "Defect detection and classification of machined surfaces under multiple illuminant directions," in *Applications of Digital Image Processing XXXIII*, 2010, vol. 7798: SPIE, pp. 541-556.
[320] W. Liqun, W. Jiansheng, and W. Dingjin, "Research on vehicle parts defect detection based on deep learning," in *Journal of Physics: Conference Series*, 2020, vol. 1437, no. 1: IOP Publishing, p. 012004.
[321] H. Zhu, Y. Wang, and J. Fan, "IA-Mask R-CNN: improved anchor design Mask R-CNN for surface defect detection of automotive engine parts," *Applied Sciences,* vol. 12, no. 13, p. 6633, 2022.
[322] M. M. Abagiu, D. Cojocaru, F. L. Manta, and A. Mariniuc, "Detection of a surface defect on an engine block using computer vision," in *2021 22nd International Carpathian Control Conference (ICCC)*, 2021: IEEE, pp. 1-5.
[323] R. Li, M. Jin, and V. C. Paquit, "Geometrical defect detection for additive manufacturing with machine learning models," *Materials & Design,* vol. 206, p. 109726, 2021.
[324] T. Aota, L. T. T. Tong, and T. Okatani, "Zero-shot versus many-shot: Unsupervised texture anomaly detection," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023, pp. 5564-5572.
[325] H. M. Schlüter, J. Tan, B. Hou, and B. Kainz, "Natural synthetic anomalies for self-supervised anomaly detection and localization," in *European Conference on Computer Vision*, 2022: Springer, pp. 474-489.
[326] V. Zavrtanik, M. Kristan, and D. Skočaj, "Draem-a discriminatively trained reconstruction embedding for surface anomaly detection," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 8330-8339.
[327] D. A. Kerr, "Apex-additive system of photographic exposure," *Issue,* vol. 7, no. 2007.08, p. 04, 2007.
[328] P. Pérez, M. Gangnet, and A. Blake, "Poisson image editing," in *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, 2023, pp. 577-582.
[329] S. Akcay, D. Ameln, A. Vaidya, B. Lakshmanan, N. Ahuja, and U. Genc, "Anomalib: A deep learning library for anomaly detection," in *2022 IEEE International Conference on Image Processing (ICIP)*, 2022: IEEE, pp. 1706-1710.
[330] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929,* 2020.
[331] A. Kirillov *et al.*, "Segment anything," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4015-4026.
[332] M. Goeller, J. Oberlaender, K. Uhl, A. Roennau, and R. Dillmann, "Modular robots for on-orbit satellite servicing," in *2012 IEEE international conference on robotics and biomimetics (ROBIO)*, 2012: IEEE, pp. 2018-2023.
[333] G. Hu *et al.*, "Modular self-reconfigurable spacecraft: Development status, key technologies, and application prospect," *Acta Astronautica,* 2023.
[334] J. Esper, "Modular, adaptive, reconfigurable systems: technology for sustainable, reliable, effective, and affordable space exploration," in *AIP Conference Proceedings*, 2005, vol. 746, no. 1: American Institute of Physics, pp. 1033-1043.
[335] D. P. Goodwin *et al.*, "Orbiter space vision system on space shuttle flight STS-

80," in *Visual Information Processing VI*, 1997, vol. 3074: SPIE, pp. 18-28.
[336] H. Pinkney and S. MacLean, "Machine Vision in Space," *Canadian Aeronautics and Space Journal,* vol. 39, no. 2, pp. 63-77, 1993.
[337] R. T. Howard, A. F. Heaton, R. M. Pinson, and C. K. Carrington, "Orbital express advanced video guidance sensor," in *2008 IEEE Aerospace Conference*, 2008: IEEE, pp. 1-10.
[338] R. Alonso, J. Crassidis, and J. Junkins, "Vision-based relative navigation for formation flying of spacecraft," in *AIAA guidance, navigation, and control conference and exhibit*, 2000, p. 4439.
[339] C.-C. J. Ho and N. H. McClamroch, "Automatic spacecraft docking using computer vision-based guidance and control techniques," *Journal of guidance, control, and dynamics,* vol. 16, no. 2, pp. 281-288, 1993.
[340] X.-H. Gao, B. Liang, L. Pan, Z.-H. Li, and Y.-C. Zhang, "A monocular structured light vision method for pose determination of large non-cooperative satellites," *International Journal of Control, Automation and Systems,* vol. 14, no. 6, pp. 1535-1549, 2016.
[341] W. Xu, Q. Xue, H. Liu, X. Du, and B. Liang, "A pose measurement method of a non-cooperative GEO spacecraft based on stereo vision," in *2012 12th International Conference on Control Automation Robotics & Vision (ICARCV)*, 2012: IEEE, pp. 966-971.
[342] V. Capuano, A. Harvard, and S.-J. Chung, "On-board cooperative spacecraft relative navigation fusing GNSS with vision," *Progress in Aerospace Sciences,* vol. 128, p. 100761, 2022.
[343] E. Martin, D. Maharaj, R. Richards, J. W. Tripp, J. Bolger, and D. King, "RELAVIS: the development of a 4D laser vision system for spacecraft rendezvous and docking operations," in *Spaceborne Sensors*, 2004, vol. 5418: SPIE, pp. 69-80.
[344] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing ICP variants on real-world data sets: Open-source library and experimental protocol," *Autonomous robots,* vol. 34, pp. 133-148, 2013.
[345] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments," *The international journal of Robotics Research,* vol. 31, no. 5, pp. 647-663, 2012.
[346] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard, and J. McDonald, "Real-time large-scale dense RGB-D SLAM with volumetric fusion," *The International Journal of Robotics Research,* vol. 34, no. 4-5, pp. 598-626, 2015.
[347] R. Dominguez *et al.*, "A common data fusion framework for space robotics: architecture and data fusion methods," in *International Symposium on Artificial Intelligence, Robotics and Automation in Space Symposia*, 2018.
[348] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII 14*, 2016: Springer, pp. 483-499.
[349] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "LSD: A line segment detector," *Image Processing On Line,* vol. 2, pp. 35-55, 2012.
[350] C. Akinlar and C. Topal, "EDCircles: A real-time circle detector with a false

detection control," *Pattern Recognition,* vol. 46, no. 3, pp. 725-740, 2013.
[351] C. Akinlar and C. Topal, "Edlines: Real-time line segment detection by edge drawing (ed)," in *2011 18th IEEE International Conference on Image Processing*, 2011: IEEE, pp. 2837-2840.
[352] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," *IEEE Transactions on pattern analysis and machine intelligence,* vol. 30, no. 2, pp. 328-341, 2007.
[353] G. Taguchi, G. Taguchi, and R. Jugulum, *The Mahalanobis-Taguchi strategy: A pattern technology system*. John Wiley & Sons, 2002.