ANOMALY DETECTION FOR VISION-BASED INSPECTION

CHEW YAN ZHE

UNIVERSITI TUNKU ABDUL RAHMAN

ANOMALY DETECTION FOR VISION-BASED INSPECTION

CHEW YAN ZHE

A project report submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering (Honours) Electrical and Electronic Engineering

Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

September 2022

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature	:	[illegible]
Name	:	Chew Yan Zhe
ID No.	:	1806357
Date	:	28 August 2022

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"ANOMALY DETECTION FOR VISION-BASED INSPECTION"** was prepared by **CHEW YAN ZHE** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Honours) Electrical and Electronic Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature	:	[illegible]
Supervisor	:	Dr. Chua Sing Yee
Date	:	3 September 2022

ACKNOWLEDGEMENTS

I would like to express my utmost gratitude towards Dr. Chua Sing Yee for her guidance and invaluable advice throughout the duration of the project, it would have been difficult to set proper goals and directions within the project without her guidance and patience. I would also like to thank my parents, who have been so supportive of my academic progress and endeavours, I will not be here without their unyielding support. Last but not least, I would like to thank my friends and course mates who have offered me their help and advice throughout this project.

ABSTRACT

The usage of Artificial Intelligence (AI) techniques in vision-based anomaly detection is gaining traction within the manufacturing industry for quality inspection purposes. The implementation of visual inspection in a production line can effectively detect defective products, while saving time and cost by increasing the efficiency through process automation. There are two approaches to visual inspection: the conventional approach which uses image processing techniques and the modern AI-based approach through deep learning. This study aims to implement visual inspection systems using both approaches and determine the suitability of each approach for visual anomaly detection. Tests were conducted on the MVTec Anomaly Detection dataset, which is a dataset for benchmarking anomaly detection methods with a focus on industrial inspection. The algorithms that were used for the conventional approach are Template Matching, Structural Similarity Index, Scale Invariant Feature-Transform and Oriented FAST and Rotated BRIEF, whereas the deep learning models that were used for the modern approach are Patch Distribution Modelling (PaDiM) and PatchCore. The results demonstrate that conventional approaches are not suitable for anomaly detection, whereas modern AI approaches are able to detect anomalies and segmentate the area with a high degree of accuracy. The average Image Area Under the Receiver-Operator Characteristic Curve (AUROC) and Image F1 Score for the PaDiM model is 0.9151 and 0.9033, whereas for the PatchCore model the scores are 0.9993 and 0.9979 respectively. Nevertheless, there are still some instances where AI models will fail to perform as intended, but generally the performance is good. Future work regarding visual anomaly detection should be focused on modern AI-based approach, but classical methods can be applied at other suitable areas where the effectiveness to complexity trade-off is warranted.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS / ABBREVIATIONS

AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
TM	Template Matching
SSIM	Structural Similarity Index
SIFT	Scale Invariant Feature-Transform
DoG	Difference-of-Gaussian
SURF	Speeded Up Robust Features
ORB	Oriented FAST and Rotated BRIEF
FAST	Features from Accelerated Segment Test
BRIEF	Binary Robust Independent Elementary Features
BFM	Brute-Force Matcher
FLANN	Fast Library for Approximate Nearest Neighbours Matcher
VAE	Variational Automatic Encoder
CNN	Convolutional Neural Network
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
OS	Operating System
PaDiM	Patch Distibution Modelling Framework
AUROC	Area Under the Receiver-Operator Characteristic Curve
TPR	True Positive Rate
FPR	False Positive Rate

CHAPTER 1

INTRODUCTION

1.1 General Introduction

Anomaly detection is the act of identifying peculiar data points or trends that deviates from the conventional behaviour of a system. These unexpected occurrences are also known as dataset outliers, and the emergence of these may indicate the presence of corrupted data, hardware malfunctions, fraudulent activity etc depending on the scenario. The traditional method to perform anomaly detection is being replaced with Artificial Intelligence (AI)-based techniques such as Machine Learning (ML) algorithms and Deep Learning (DL) neural networks, which enables a faster, more efficient and accurate detection process. Properly identified system anomalies helps to prevent and eliminate major system malfunctions.

The usage of ML in anomaly detection is gaining a lot of traction in recent years as there is an exponential increase of data generated across all industries. Processing and analysing the huge amount of unstructured data that is collected in a timely, efficient and accurate manner using conventional methods is no longer a viable option. Besides that, companies are also unwilling to allocate large financial and human resources to perform anomaly detection using inefficient and costly methods. This is where ML comes in to play as the most viable option to process large datasets.

Proper ML-based anomaly detection is essential in industries where the data needs to be processed in a structured and comprehensive manner while maintaining the integrity of data. As an example, anomaly detection is especially prevalent in industries where the detection and prevention of suspicious activity and fraud is of utmost importance, such as the banking, finance and insurance industries. The payment and spending habits of users can be profiled and fed into detection models, which can detect and even predict the occurrence of fraud or suspicious transactions.

The healthcare industry also benefits from ML-powered anomaly detection, where algorithms are used to effectively detect abnormalities from medical diagnoses. Such instances can be seen in the usage of in-depth X-ray image analysis, which helps doctors to obtain a more precise diagnosis, and thereby reducing human errors. Network security applications can also heavily benefit from the usage of anomaly detection, such as performing unauthorised access detection and suspicious request detection. The manufacturing industry also reaps in serious benefits from the usage of anomaly detection, such as collecting sensor data to predict when machines or hardware will fail, which is also known as predictive maintenance. Anomaly detection is also used to perform quality checks on manufactured products to detect faulty items (NIX United, 2021).

Vision-based inspection or visual inspection is a subset within the large domain of anomaly detection. This type of inspection is usually applied in the manufacturing industry for quality inspection purposes such as detecting flaws in products, ensuring that the positioning and measurement of components are correct, identifying missing or additional parts in a product and so forth. The usual set-up for a visual inspection system is having high quality cameras to capture pictures for inspection, and the images are transferred to the hardware setup. There, these images are fed into software algorithms to render and analyse the given data. A proper environment at the test site such as adequate lighting is essential to obtain accurate results.

The implementation of visual inspection in a production line can significantly uncover hidden defects of products. For example, inspecting automobile components for flaws or missing components, detecting defects on silicon wafers, checking construction raw materials for any cracks or dents etc. In recent times, DL approaches are frequently used to perform vision-based inspection as neural networks are extremely suitable to process image or video-based data and generate inferences, which saves significant time and effort as compared to conventional methods.

1.2 Importance of the Study

To design a vision-based inspection manually using conventional methods is difficult, inaccurate and time consuming. An inherent limitation that comes with the usage of manual labour is the requirement of human personnel to perform assessment of the products and making a judgement based on previous training and experience. However, the errors that comes from manual visual inspection usually range from 20% to 30% (Khan, 2021). A certain percentage of errors can be reduced through proper training and experience, but it cannot be completely removed. The main reason behind this is the undependability of the human vision, as well as the imprecision of eyesight. On top of these issues, hiring human operators to perform inspections are also costly. It is also nearly impossible for humans to go through a large number of products within given period of time. With all these considered, the usage of human operators for visual inspection simply cannot match the efficiency and accuracy of automated DL-powered systems (Khan, 2021).

Hence, the implementation progress of DL-based neural networks for vision-based inspection is growing at a very fast pace and will sooner or later be the de-facto industry standard when it comes to visual quality checks. With this technology, process automation is embedded into the system, whereby machine vision systems can be automated and process a large number of parts within a short time. These DL models can also self-learn, or the parameters of the model can always be fine-tuned down the road, which gives the model sort of an adaptive performance. The overall handling of the system would be easier as well due to the automated and systematic approach, which allows manpower to be distributed to tasks absolutely requires manual work. Some other benefits that come with DL-based inspection are performance increase of system, overall increase in system stability, and reduction in time and cost (NIX United, 2021).

However, there also exists more conventional computer vision methods for anomaly detection, such as by template and pattern matching, or similarity detection. Even though most companies are favouring the usage of AI techniques to perform visual inspection, conventional methods are not completely phased out as these are much easier to implement as compared to DL neural networks. Through this study, a comparison between conventional methods and AI-based methods for vision-based inspection will be made, and further analysis can be done to highlight the pros and cons of both systems.

1.3 Problem Statement

Traditionally, the process of quality checking in a production line is performed by human operators who will visually inspect the manufactured products for defects and flaws. However, this process is a repetitive one, which introduces errors in human judgement as the operators are unable to stay focused for long periods of time. Besides that, the sheer number of products that requires to be checked cannot be processed efficiently and accurately by humans. Therefore, the manufacturing industry has been looking for other alternatives to resolve this issue.

Machine vision methods can be implemented to automate and simplify the whole process, while increasing the performance and efficiency of the system. However, the machine vision domain consists of conventional methods and AI-based methods, which are usually DL neural networks when it comes to dealing with image or video related data. Although the DL method is gaining traction, conventional methods are still used in some cases because the deployment of DL algorithms is not a straightforward task. Therefore, there is a need to study the effectiveness and disadvantages of both the conventional and AI-based methods.

1.4 Aim and Objectives

The aim of this project is to perform anomaly detection for vision-based inspection by using conventional image processing methods and AI-based methods in order to make a comparison between these two methods. In order to do so, the following objectives should be met:

i. Develop a vision-based inspection system.
ii. Investigate and implement anomaly detection methods based on classical image processing algorithms and modern AI-based approaches.
iii. Analyse the performance of respective methods and making evaluations using suitable metrics.

1.5 Scope and Limitation of the Study

The main focus of this study is on the implementation of classical image processing methods as well as modern DL-based approaches to perform anomaly detection for vision-based inspection. The identified methods were developed and compared by using quantitative metrics to assess the performance of these methods.

This study is fully based on the MVTec Anomaly Detection dataset, where all the training and benchmarking process are solely based on this dataset. Therefore, the obtained results may not be the same when it is implemented on other image datasets. Besides that, the study is based on static images only, where real-time analysis based on dynamic images is not covered due to time constraints. Hence, the real-time performance of these implementations may differ from the obtained results.

1.6 Contribution of the Study

This study explores both classical and modern approaches for vision-based anomaly detection and provides a comprehensive comparison between the two. Current comparisons that were made only focuses on either the conventional approach or the modern approach, but comparison between the two approaches by using relevant examples or datasets for both methods have not been performed before. Hence, this study performs the necessary tests on both approaches to make a detailed comparison between conventional and modern approaches for visual anomaly detection.

1.7 Outline of the Report

This report contains five chapters in total. Chapter 1 provides an overview of this study while explaining the importance of this project. It also provides the problem statement, aim and objectives, scope and limitation of the study, and the contribution of the study. Chapter 2 then provides the literature review on previous works that are related to this project. Chapter 3 explains the methodology and work plan of the entire project and the process flows of the classical and modern approaches for visual anomaly detection. Chapter 4 presents the results that were obtained from this study, and the discussion and analysis regarding the obtained results. Chapter 5 provides a conclusion to the study and recommendations for future works.

CHAPTER 2

LITERATURE REVIEW

2.1 Computer Vision

Computer vision is the field of study that enables computers or machines to derive meaningful information and provide inferences and recommendations from images, videos and other types of visual inputs. Basically, computer vision enables computers to see and detect visual stimulus, observe and understand the situation, and extract information from visual data like humans are able to (IBM, n.d.). Figure 2.1 illustrates a simple computer vision program that is able to recognise what type of fruit is detected by a camera. When it comes to human vision, the eyes are the sensory organs that can detect visual stimulus, and these stimuli are converted into electrical signals that are sent to the brain for processing. In this case, the brain will be able to accurately recognise the fruits in the image provided that the person has seen those fruits before. For computer vision, the sensing device is usually a camera, and the images and videos that are captured by the camera are fed into a computer vision algorithm. This trained algorithm will be able to recognise all the fruits in the picture.

A lot of research has been done on computer vision and its applications, and it has also been implemented in a lot of real-world scenarios that are related to business, entertainment, transportation, healthcare, manufacturing etc. Computer vision technology is especially useful in tasks that are related to image classification, object detection, object tracking. As the name implies, image classification algorithms take in image inputs and classify them. Hence, it is able to accurately determine whether the input image is under a certain class or not. Object detection algorithms are used to detect specific objects from an image or video input, or even pinpoint minor differences between similar objects. These algorithms can be used to detect anomalies and defects from products on a production line. Object tracking is used to follow or track an object when detected and is often used with real-time video inputs (IBM, n.d.).

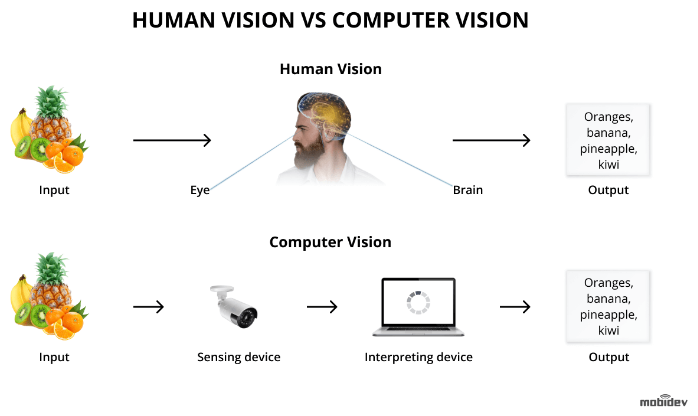
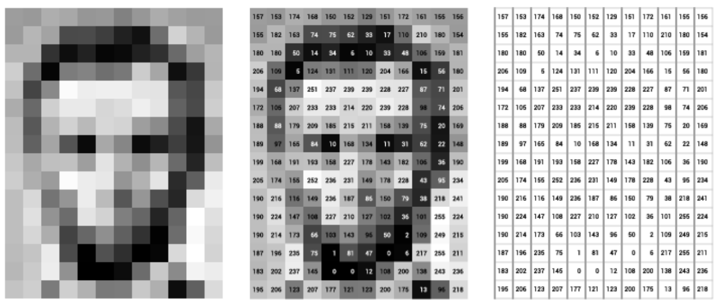


Figure 2.1: A Simple Fruit Recognition Computer Vision Program (Krasnokutsky, 2021).

The fundamental approach to developing computer vision technologies is to discover patterns within images in order to decode individual objects. This concept is used because it is similar to the way the human brain functions. Hence, pattern recognition is a huge part of today's computer vision algorithms. For example, if a computer vision model is being developed in order to accurately detect faces, then a huge amount of human face images will be used to train the model in order for it to identify patterns that are representative of all human faces. Through this training process, the model will then be able to identify human faces whenever images are fed into the system. For a computer to understand the input images, the images are usually decoded into pixels with specific colour values and brightness. From the image in Figure 2.2, the computer decodes the person's face into pixel grids. The value of each pixel represents the brightness at that particular point. These numbers are used by the machine to perform further analysis and inferencing (Babich, 2020).



157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

Figure 2.2: Pixel Data Diagram of an Image (Levin, n.d.).

2.2 Machine Learning and Deep Learning

ML is a subset of AI which deals with data and mathematical algorithms in order to mimic the way humans process information, which helps to increase the model's efficiency with time and more data. Although ML is not a field that was pioneered within these few decades, it was only until recently when the processing power of computers and machines have reached a certain prowess that the field of ML has started to really take off. It is now a major part of most companies and projects because the value and impact it brings to the table is immense. This is especially true in the growing field of data science, where algorithms are developed to provide accurate classifications and inferencing, which then generates impactful insights that has the ability to transform companies and organisations (IBM, 2020a).

ML is based upon mathematical models, statistics-based reasoning and optimisation methods that power machines to analyse huge datasets and identify important patterns to generate crucial insights. ML techniques can also be used to leverage upon historical data to provide predictions for the future. Three essential components are typically used to build up supervised ML algorithms. First, a decision process is usually present, which is basically feeding input data into a series of mathematical calculations to obtain a quantitative score. If the score is higher than a set threshold (or at times lower), then the specific input data is what the algorithm is trying to detect. After going through this decision process, an error function is used to measure the accuracy of the decision process by comparing it to known and tested examples. Once the error function evaluates the accuracy of the decision process, an updating and optimization process is applied to detect where errors occur from the decision process and further optimizes the model. The model is now able to make decisions that are more accurate. These processes are iterative, and they occur automatically without any human intervention, which allows these models to uncover hidden insights without being manually programmed (UC Berkeley, 2022).

The ever-increasing importance of ML techniques are due to the fact that these algorithms are able to extract insights from datasets that are too large to be processed manually by humans. On top of that, humans cannot deal with the velocity of data generation nowadays. With the constant improvement of processing power, these models can operate autonomously while discovering hidden details within huge datasets (UC Berkeley, 2022).

DL is a subset of ML, and these are usually neural networks with three or more layers. A neural network is an algorithm that is under ML, but it is also the backbone for DL algorithms. The depth of the neural network determines whether the algorithm is ML-based or DL-based. These neural networks mimic the way the human brain learns and interprets information, but up till this point these neural networks are still far from the capabilities of the human brain. The number of layers determines whether the model is able to make approximate guesses or make accurate predictions that are more optimised and refined. This is why when the number of layers is higher, the algorithms are under DL techniques. Of course, neural networks are only a part of the whole ML algorithm family, and what truly separates ML and DL algorithms are the types of input data these models can handle, and the way these algorithms learn from the given input data (IBM, 2020b).

ML algorithms works best with structured and labelled data to perform analysis and inferencing. This means that the important features are pre-labelled or annotated by human experts and then organised into structured tables. Therefore, if the input data on hand is unstructured data (e.g. texts, images, audio), it needs to be processed into structured data before further analysis. When it comes to DL, unstructured data can be utilised directly by DL models. This is because the feature extraction process is automated, so human experts will not need to manually perform this part. Let’s say if a model is being trained to differentiate between humans and animals, DL algorithms will be able to determine automatically which features are the most important to determine a human from an animal. For example, the gait of humans is different from animals as humans walk on two legs as compared to four legs for most animals, so this might be a differentiator between humans and animals (IBM, 2020b).

ML and DL algorithms as a whole can be classified into several different learning models, where the difference between these categories lies in the method each model uses to learn. The most common learning model is supervised learning, where a labelled dataset is used for the training process. Labelled dataset serves as a reference for the algorithm to know which are the important features that are important for the decision-making process, and for measuring the performance of the model as well. Basically, each data within the training dataset is tagged with the correct answer that the model is supposed to come up with itself. This is as if the machine is performing a task under the supervision of a human expert and is being told whether the obtained output is correct or not. Let’s say a model is being trained to differentiate species of birds, the input labelled data should label which images are eagles, hornbills, sparrows, owls etc. When the model is trained, new images are fed into the model and compared to the training data to predict the correct type of bird. Figure 2.3 shows the general process for supervised learning. From these processes, it is evident that supervised learning is best suited to problems where there exists reference answers or ground truth data to train the model. However, a lot of complex problems do not have readily available reference data for training data (Salian, 2018).

In contrast with supervised learning, unsupervised learning is able to utilise unlabelled data without human experts telling the model what to do with it explicitly. The collection of training data that is fed into the model comes without any specific feature or answer to be analysed. Instead, the model will perform feature extraction automatically and analyse the structure of the given data to generate useful predictions. Unsupervised learning is much more useful than supervised learning in a lot of real-world scenarios because a lot of data that are generated is unstructured data which are not labelled and cannot be organised in a structured manner. Besides that, human experts are not capable of sifting through humongous datasets to look for uncovered features or insights within a short amount of time. This is why unsupervised learning is very effective in tackling problems that are related to anomaly detection, where the anomalies within data are usually random and unpredictable (Salian, 2018).

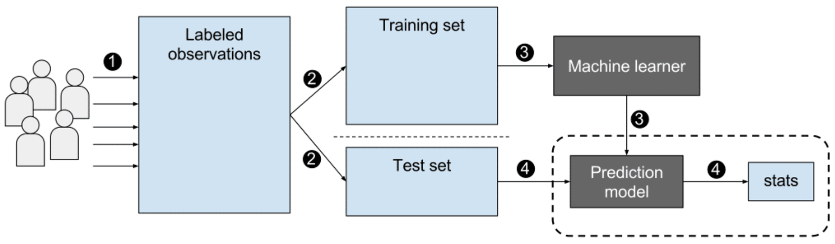


Figure 2.3: General Process Flow for Supervised Learning (Salian, 2018).

2.3 Conventional Methods for Vision-based Anomaly Detection

Prior research has been carried out for vision-based anomaly detection, and most of the state-of-the-art methods are based on DL models. However, there exists several classical image processing methods that can be used for anomaly detection. In this study, several classical image processing methods that are useful for visual anomaly detection will be explored.

2.3.1 Template Matching

Template Matching (TM) is a conventional method that is used to compute the similarity between a template image and an original input image by using mathematical algorithms. What this method basically does is identifying whether there is a specific object in an image, and where is the specific location in the said image. Figure 2.4 shows the original image on the left side, and the possible template images that can be used on the right side. If TM is used to detect the location of face in the image, the face will be cropped and used as the template image and slides through all pixels of the original image. The region that has the highest similarity score will be the location of the face in the original image (Hashemi, et al., 2016).

The template image is the object that needs to be detected from the original image, and it is located by sliding the template image through the original image while a similarity score is computed, which is illustrated in Figure 2.5. The template image, which is the head of a dog, is slid over the original image one pixel at a time. At each region in the original image, a metric or similarity score is calculated. The higher the similarity score, the more likely the region is where the target object is in the image (OpenCV, n.d.a). Hence, TM is a high-level computer vision technique that is used to detect objects from images given the specific image.

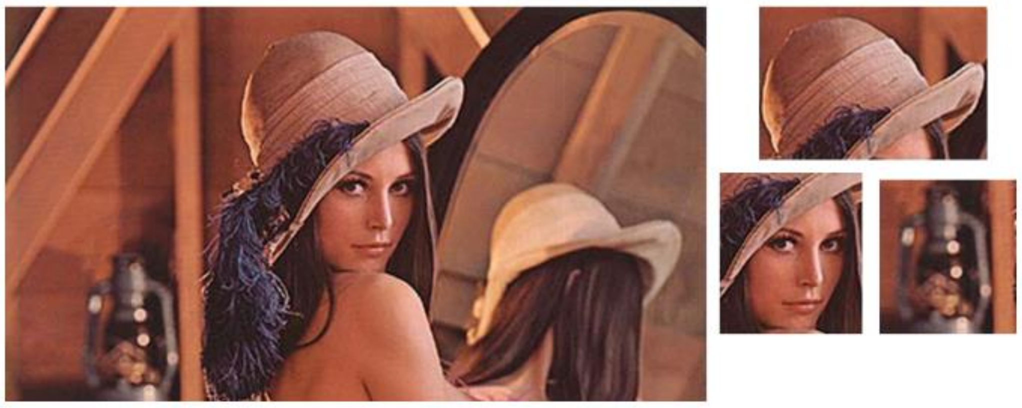


Figure 2.4: Original Image (Left) and Possible Template Images (Right) (Hashemi, et al., 2016).

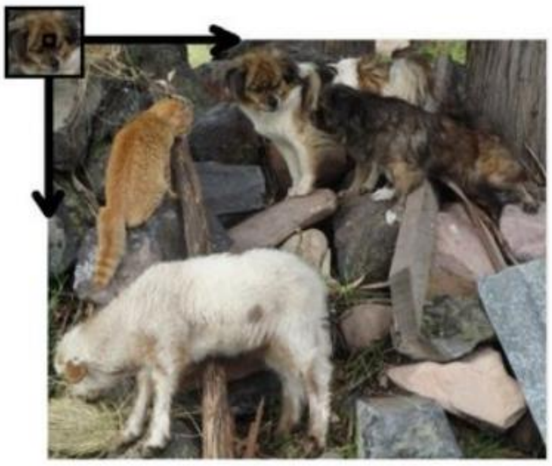


Figure 2.5: Sliding of Template Image Over Original Image (OpenCV, n.d.a).

There are a few approaches when it comes to the implementation of TM. The easier and simpler method is the template-based approach, which is suitable for cases when the templates do not have any strong features because the matches are determined by the intensity of the pixel values of both the original image and the template image. The sliding of the template image over the original image as shown in Figure 2.5 is a template-based approach, where the similarity value of the target object is measured pixel by pixel. To do so, usually both images are converted into grayscale images, and then TM algorithms such as cross-correlation or sum of squared difference are applied. While these algorithms are easy to implement, it is ineffective when the scale of the image or the orientation of the images are different, which means that this method is both scale and rotation invariant. To solve issues related to scale invariance, the original image can be resized to multiple scales and then compared to the template image. When different sizes of the original images are looped through, the scale which presents the highest similarity score is the most suitable scale to perform TM. However, problems that are related to rotation invariance are much more difficult to be solved (Luces, 2019).

The other approach is the feature-based approach, which is usually used when there exists more common features and control points between the original and template images. Some examples of features are points, curves, and surface models. These features provide a lot of information on the content of the image, and the usage of local features and their descriptors are the basis for many machine vision algorithms. Since the features are unique to each image, it is especially useful for object detection and classification and tracking purposes. The most useful part of this approach is that the features can be matched between two images even if they are scaled differently or rotated. This method is also more efficient when the resolution of the image is large. However, this approach is not desirable if the images have little common features, or when different types of objects share the same features. These will cause the algorithm to not detect the target object or detect objects that are not the target object. Figure 2.6 shows the different output results for template-based and feature-based approaches when the template image is rotated. It is demonstrated that template-based approaches are rotation invariant, whereas feature-based approaches can accurately detect the target object even when it is rotated, which makes it much more robust. However, the implementation process is much more complicated than template-based approaches (Luces, 2019).

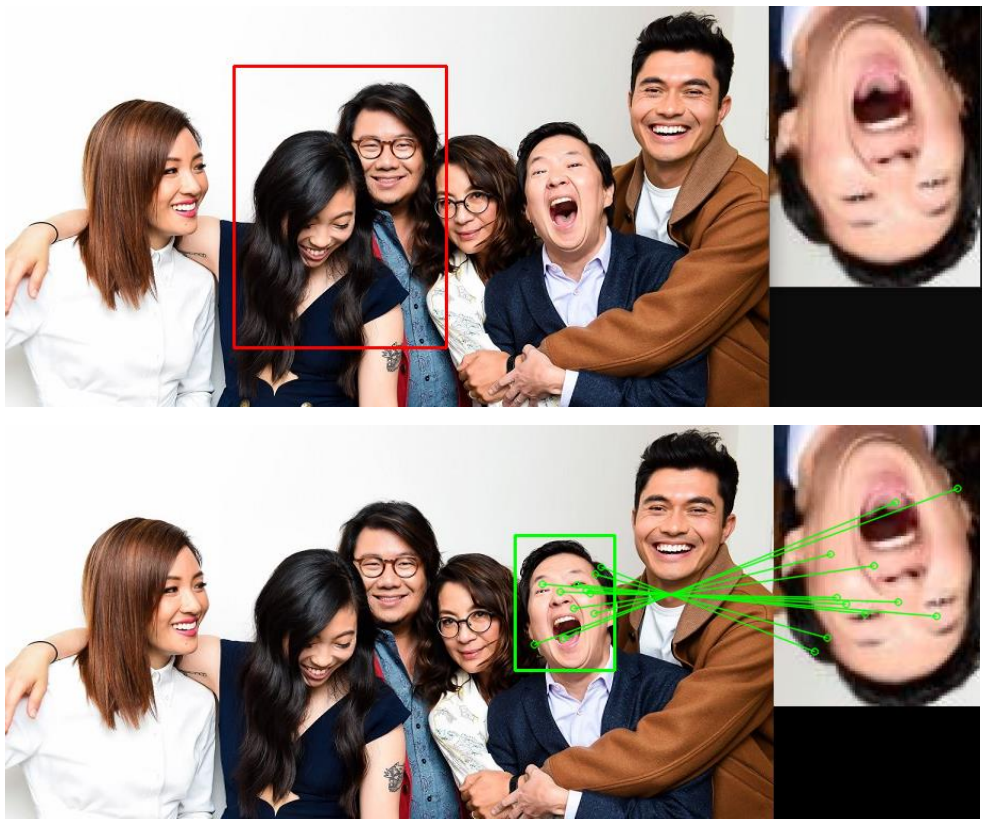


Figure 2.6: Results for Template-based Approach (Top) and Feature-based Approach (Bottom) for Rotated Template Image (Luces, 2019).

When it comes to template-based approaches, there are a few methods that are commonly used to measure the similarity between the template image and the original image. The basic method is known as naïve TM, which basically uses a template image that is directly obtained from the original image, and the similarity score is calculated by scanning the template through the original image. The template image is usually not scaled or manipulated, which makes this method to be very efficient for simple use cases. However, only regions that have a very high similarity score will be considered as a match, which renders it ineffective when there is a certain degree of variation between the template image and original image. The mathematical algorithm that is usually used is the sum of squared differences, but there are other algorithms such as root mean square distance, sum of absolute values differences etc that can be applied (Hashemi, et al., 2016).

Another method that is more robust to variations is the image cross-correlation technique, which enables the matching of images that have slightly larger deviations, but the images must be aligned properly for the results to be accurate. Cross-correlation is basically determining how similar two images are, the more similar the images are, the higher the degree of correlation between the images (Hashemi et al., 2016). Although a higher correlation value does indicate that two images are similar, this method is not completely robust as a change in the global brightness of the images will severely affect the accuracy of the outcome. This is where the normalised cross-correlation technique is introduced to combat this issue. The results from the normalised cross-correlation technique are invariant to changes in the global brightness of the image. Besides that, the output correlation values of the images are normalised to a range of [-1,1], where identical images will have a correlation of 1, while images with no correlation have a value of -1 (Adaptive Vision, n.d.). Figure 2.7 shows a comparison on the similarity scores that are obtained between the cross-correlation and normalised cross-correlation methods. Cross-correlation similarity scores are random and large, whereas scores for normalised cross-correlation are standardised to the range of [-1,1]. Not only does this makes it easier to gauge the similarity between two images, but it also has the added benefit from being invariant to global brightness changes of the images.

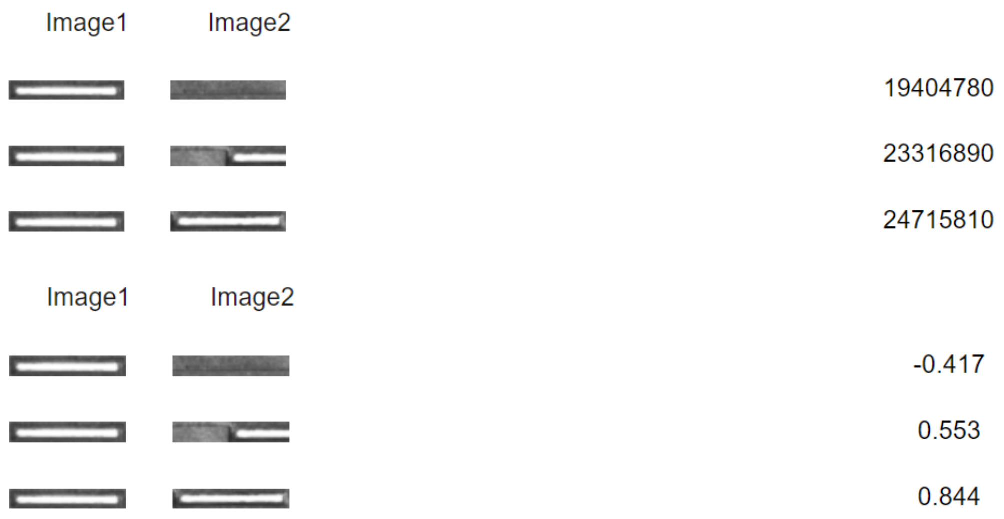


Figure 2.7: Comparison of Cross-Correlation (Top) and Normalised Cross-Correlation (Bottom) Outputs (Adaptive Vision, n.d.).

OpenCV, which is an open-source computer vision and ML software library, has provided built-in methods for template-based approaches for TM applications. Besides the aforementioned sum of squared difference and cross-correlation methods, OpenCV also has a correlation coefficient technique that provides more accurate results and is more robust as compared to the previously discussed methods. Figure 2.8 shows the list of methods provided by OpenCV, where SQDIFF stands for sum of squared differences, CCORR stands for cross-correlation, CCOEFF stands for correlation coefficient, and NORMED stands for normalised algorithm. From the mathematical equations, *I(x,y)* represents the source or original image, *T(x,y)* represents the template image, *R(x,y)* is the generated output result, and *(w,h)* is the width and height of the template image (OpenCV, n.d.a).

1. **method=TM_SQDIFF**

$$R(x,y)=\sum_{x',y'}(T(x',y')-I(x+x',y+y'))^2$$

2. **method=TM_SQDIFF_NORMED**

$$R(x,y)=\frac{\sum_{x',y'}(T(x',y')-I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'}T(x',y')^2\cdot\sum_{x',y'}I(x+x',y+y')^2}}$$

3. **method=TM_CCORR**

$$R(x,y)=\sum_{x',y'}(T(x',y')\cdot I(x+x',y+y'))$$

4. **method=TM_CCORR_NORMED**

$$R(x,y)=\frac{\sum_{x',y'}(T(x',y')\cdot I(x+x',y+y'))}{\sqrt{\sum_{x',y'}T(x',y')^2\cdot\sum_{x',y'}I(x+x',y+y')^2}}$$

5. **method=TM_CCOEFF**

$$R(x,y)=\sum_{x',y'}(T'(x',y')\cdot I'(x+x',y+y'))$$

where

$$T'(x',y')=T(x',y')-1/(w\cdot h)\cdot\sum_{x'',y''}T(x'',y'')$$
$$I'(x+x',y+y')=I(x+x',y+y')-1/(w\cdot h)\cdot\sum_{x'',y''}I(x+x'',y+y'')$$

6. **method=TM_CCOEFF_NORMED**

$$R(x,y)=\frac{\sum_{x',y'}(T'(x',y')\cdot I'(x+x',y+y'))}{\sqrt{\sum_{x',y'}T'(x',y')^2\cdot\sum_{x',y'}I'(x+x',y+y')^2}}$$

Figure 2.8: TM Methods from OpenCV (OpenCV, n.d.a).

2.3.2 Structural Similarity Index

Structural Similarity Index (SSIM) is an algorithm developed by Wang, et al. (2004), where this algorithm is mainly developed for image quality assessment. The SSIM algorithm essentially determines the visual perceptual difference between two images, but the score itself is not able to determine which image is "better". Hence, the user should have prior knowledge on which image is the original one and which is the modified one. Generally, the SSIM algorithm measures the luminance and contrast of two images and generates three comparisons, which are the luminance comparison, contrast comparison and structure comparison. The combination of these three comparisons gives the SSIM score. The SSIM measurement system is illustrated in Figure 2.9.

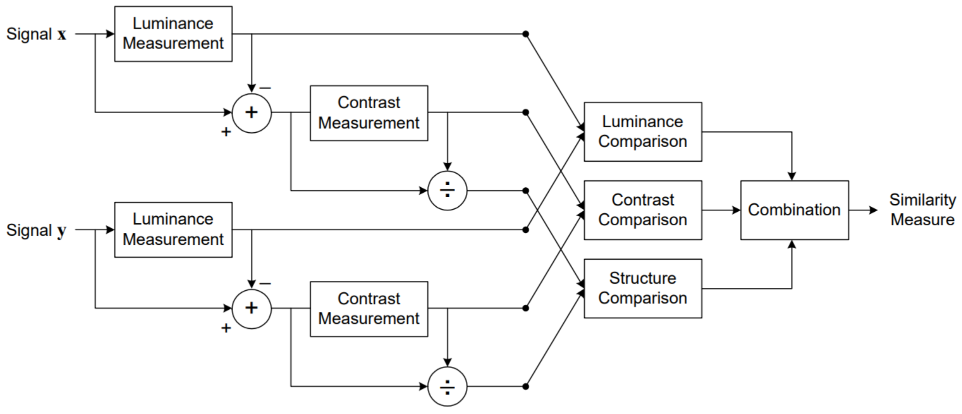


Figure 2.9: Measurement System of SSIM (Wang, et al., 2004).

The SSIM score has a range of -1 to 1, where a score of 1 indicates perfect similarity. Therefore, computing the SSIM score for identical images should yield a score of 1, and images that are very different from each other will have a score of -1. However, when it comes to the application of visual anomaly detection, this algorithm is unable to pinpoint where the anomaly is within an image. Besides that, SSIM is originally used for image quality assessment purposes instead of defect detection within images, but the SSIM algorithm can be combined with other image processing methods such as thresholding and contouring to visualise the differences based on the computed similarity score. Another issue with the SSIM algorithm is that it is rotation and scale invariant, which is similar to the TM algorithm. Therefore, both of the images that are being compared must have the same orientation and scale for this algorithm to work effectively.

2.3.3 Feature Matching

As discussed in Chapter 2.3.1, feature-based matching or feature matching is a more robust method to match images as this method is scale and rotation invariant, which means that it is able to match images even when the images are of different orientation and scale. However, this algorithm is only useful when there are enough common features between the images, else the algorithm might perform inaccurate matching between two images.

Features are information from specific areas or structures within an image such as points, edges or objects, and they can be divided into two categories. Some features that appear in specific location within images such as at the peak of a mountain top, corner of buildings or architectures, around the eyes of a person are called as keypoint features. These are localised features that are described by patches of pixels around a location. Another type of feature is called edges, which are matched based on the orientation and local orientation. These are good indicators of object boundaries. There are three main components for feature detection and matching, which are detection, description and matching. At the detection stage, feature points will need to be identified within the images. Then at the description stage, the surrounding local area within feature points are described in a way that is invariant to scale, rotation, translation and illumination using descriptor vectors. Lastly, descriptors between images are compared to match similar features at the matching stage (Tyagi, 2019).

The first feature matching algorithm to be widely used and reviewed is the Scale Invariant Feature-Transform (SIFT) algorithm, and it was proposed by Lowe (2004). The study develops a method to extract distinctive invariant features from image that can be utilised to perform matching between different images reliably regardless of rotation and scale variation, and even with a change in three-dimensional viewpoint, addition of noise and change in illumination. The obtained features are also highly distinctive, so it can be used for object recognition applications as well.

The four main steps in this algorithm are scale-space extrema detection, keypoint localisation, orientation assignment and keypoint descriptor. The first stage, which is scale-space extrema detection is basically the mathematical computations to search through the image matrix using a difference-of-Gaussian (DoG) method to determine potential points that are scale and orientation invariant. At the second stage, the potential points from the first stage are fit into a model to determine the location and scale. Keypoints will be selected from those potential points based on their stability, and scale invariance is achieved at this point. After that, one or several orientations are given to the keypoints based on local image gradient directions. Hence, later operations will be performed on keypoints that have been transformed relative to the assigned scale, orientation and location. Therefore, orientation invariance can be achieved. For the last step, a descriptor is computed for the local image region for each keypoint so that the keypoints are highly distinctive, hence providing invariance to change of viewpoint and illumination. When it comes to keypoint matching, features between two images are matched by identifying the nearest neighbours surrounding the feature. However, the second closest match might be too near to the closest match due to external factors such as noise. Therefore, Lowe proposed the ratio test of the closest distance to second closest distance neighbors, if this ratio is larger than 0.8, then the keypoint is rejected. Lowe suggests that this will remove 90% of incorrect matches while only discarding 5% of correct matches.

Although the SIFT algorithm have a relatively high accuracy, the high dimensionality of the generated descriptors during the matching stage imposes a large computational burden, which causes the matching speed to reach a bottleneck if the device has insufficient computational capability. Hence, Bay, Tuytelaars and Van Gool (2006) devised another algorithm known as the Speeded Up Robust Features (SURF) which has an improved accuracy and computational speed over SIFT and other state-of-the-art algorithms at that time. However, at this moment SURF is a patented algorithm, so it is not freely accessible. Later on, Rublee, et al. (2011) from OpenCV Labs created another new feature matching algorithm known as Oriented FAST and Rotated BRIEF (ORB), where FAST stands for Features from Accelerated Segment Test and BRIEF stands for Binary Robust Independent Elementary Features. Hence, ORB is a combination of the FAST keypoint detector and the BRIEF descriptor with some added modifications to increase the performance.

The ORB algorithm initially uses the FAST algorithm to detect keypoints, and then the Harris corner measure is used to find the top N points among all the keypoints. After that, a multiscale image pyramid is applied to the image to produce multiscale features. However, since FAST does not compute orientation, the authors of ORB added a modification to compute the intensity centroid of the patch with the located corner at the centre. The direction of the vector from the located corner to the centroid provides orientation to the image, which provides rotation invariance. Hence, this is now known as oriented FAST (OpenCV, n.d.b).

For the descriptor, ORB uses the BRIEF descriptor, but this descriptor has a poor performance when it comes to image rotation. Hence, ORB steers the BRIEF descriptors to follow the orientation of keypoints, where the rotation step is discretised to an angle of 12 degrees, and a lookup table is constructed with precomputed BRIEF formations. However, the orientation of the keypoints means that the variance of the features is reduced, which is not desirable to generate highly discriminative features. Therefore, a greedy search is performed among all binary tests to determine keypoints that have both a high variance and a mean value that is approximately 0.5. This is now known as rotated BRIEF (OpenCV, n.d.b).

Once the features are detected and the local descriptors are generated, matching algorithms are used to match similar features between images. Two algorithms are mainly used for feature matching, which are the Brute-Force Matcher (BFM) and the Fast Library for Approximate Nearest Neighbours Matcher (FLANN). The BFM is relatively simple as it takes all the descriptors of one feature from the first image and compares it with all the descriptors from one feature from the second image using a distance calculation, and the descriptor with the shortest distance is returned. For the FLANN matcher, it utilises algorithms to search for nearest neighbours in a much optimised and faster method within huge datasets and high dimensional features, hence it is much faster than BFM. However, its accuracy is lower than BFM as it only determines the approximate nearest neighbours, whereas BFM will determine the best match. Therefore, it is a trade-off between accuracy and speed between these two feature matching algorithms (OpenCV, n.d.c).

2.4 Modern AI based Methods for Vision-based Anomaly Detection

Modern methods for visual anomaly detection are usually based on AI techniques, which enables the processing of large datasets efficiently and effectively. As discussed in previous sections, there are several learning methods when it comes to the training of an AI model. The selection of learning method heavily depends on the availability of correctly labelled datasets. For anomaly detection, there are three main types of learning methods that are usually used. The first one is supervised anomaly detection, where the training and test datasets are fully labelled. However, this type of learning method is not practical for most real-life scenarios because it is very difficult to find and label every type of anomalies beforehand. This also indicates that it is hard to find or manually curate a dataset that encompasses all known anomalies for a given situation. Next, semi-supervised anomaly detection uses training data that comprises of normal data that are not anomalous. Once the model learns what constitutes of a normal situation from the training data, anomalies can be detected from the test data by categorising data that deviates from the normal behaviour of the training data. Thirdly, there is unsupervised anomaly detection, which is the most flexible learning method as the training and testing data does not require any labels. The model is able to distinguish abnormal regions from normal ones automatically, provided that sufficient data is fed to the model for training (Goldstein and Uchida, 2016). Figure 2.9 illustrates the training process for these three learning methods.

When it comes to modern approaches for vision-based anomaly detection, most studies are based on DL unsupervised or semi-supervised learning methods. In addition to that, the training method for unsupervised learning also resembles most practical application scenarios, where anomalies vary greatly and there is no standard pattern or trend on how anomalies will occur. Visual anomaly detection can be split into two categories, which are image-level and pixel-level detection. Image-level detection considers an image as a whole to detect whether it is normal or abnormal, whereas pixel-level detection dives deeper into the pixel data to determine the specific regions where there is an anomaly in the image (Yang, et al., 2021).

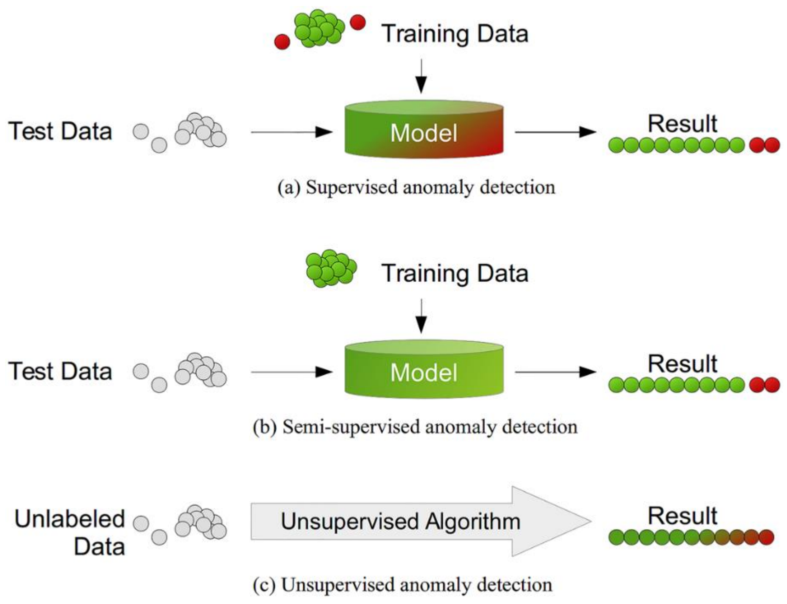


Figure 2.10: Different Types of Anomaly Detection Based on Learning Method of Models (Goldstein and Uchida, 2016).

Yang, et al. (2021) made a comprehensive survey that looks into unsupervised and semi-supervised methods for image-level and pixel-level visual anomaly detection. For image-level methods, the reviewed techniques are density estimation, one-class classification, image reconstruction, and self-supervised classification, however self-supervised classification will not be discussed here. Density estimation is basically generating a probability distribution of features that are obtained from normal training images. Then, the probability distribution of test images is generated and compared to the distribution obtained from original images. If a test image's probability distribution is not close enough to the probability distribution of normal images, then it is considered as an anomalous image. There are classical methods for density estimation such as Gaussian models, but these methods are not optimal for data with high number of dimensions such as images. Deep generative models are able to process high dimension data, but a large number of images are needed to train these models. On top of that, these models are not sufficiently robust and stable for anomaly detection. The findings suggest that popular deep generative models such as Variational Automatic Encoder (VAE) are not good enough for simple visual anomaly detection tasks.

One-class classification generates a binary decision on whether an image is anomalous or not. Since it does not take so many points of an image to generate a probability distribution, it does not require a huge number of training images. However, it faces problems such as dimension disaster and scalability issues. With the development of DL, researchers have looked into combining deep Convolutional Neural Networks (CNN) with conventional one-class classification methods. Another method for image-level anomaly detection is image reconstruction, which reduces an image to its latent space, which is the low-dimensional vector representation of the image that encompasses the essential features of the image. After reducing the dimensions of the image, the low-dimensional vector is then used to reconstruct the original image. The difference between normal images and anomalous images is that the error from reconstructing the original image is small for normal images, and larger errors ensue from the reconstruction of anomalous images. Autoencoders are frequently used for image reconstruction, which is a neural network that comprises of a narrow middle hidden layer. The narrowing of the hidden layers compresses the input data by removing redundant information while maintaining essential features of the images (Yang, et al., 2021).

For pixel-level methods, the reviewed techniques are image reconstruction and feature modelling. As mentioned above, image reconstruction is used to reduce the dimension of an image until the lowest possible size, where the vector contains all the essential information only. Then, this low-dimension vector is used to reconstruct the original image. After the original image is reconstructed, potential abnormal regions are detected by calculating the pixel difference between the input image and the reconstructed image by using methods such as the pixel-level l_2-distance or SSIM. Deep generative models such as VAE and generative adversarial networks (GAN) are frequently used as the model to reconstruct the images. This method is considered to be intuitive, and it is also expected to regenerate high quality original images because it detects anomalies within the pixel space. However, it is still difficult to regenerate high quality original images, where issues such as difficulty in generating sharp edges and complex texture structure of images causes reconstruction errors in edge or texture regions. This causes a large number of false positive alarms for anomaly detection (Yang, et al., 2021).

Whereas for feature modelling, anomalies are detected within the feature space instead of the image space. These features are usually handcrafted or determined by neural networks. Then, ML models such as sparse coding, Gaussian mixed model and K-means clustering can be used to model the distribution of the features pertaining to the images. If the regional feature of an image deviates from the modelled feature distribution from the training dataset, then the specific region can be considered as an anomalous region (Yang, et al., 2021).

2.5 Process to Deploy Deep Learning models

The majority of studies or real-world applications that are related to computer vision applications are based on DL technology, which is a subset under ML technology that is powered by neural networks. In general, a neural network is developed by feeding a huge amount of training data that is related for a specific application, so that it is able to decipher common traits from the input data and then generate a mathematical equation, which describes the behaviour of the neural network. This equation acts as a differentiator which is able to classify input images or videos into different categories, which is especially useful in visual inspection applications as the algorithm is able to separate defective images from normal ones (Krasnokutsky, 2021).

To develop a DL algorithm that can be used in visual inspection software, relevant training data must first be collected. Consider creating a visual inspection program for electronic components, the training data that needs to be gathered should be examples of defective components such as transistors with bent pins, integrated circuits with burnt surfaces etc. However, it needs to be specific to one component only. When sufficient data is fed to the neural network, it is able to differentiate defective and normal components without additional input by the user at a relatively high accuracy. The process to integrate DL visual inspection programs into a system is complicated and requires structured and well-planned steps to be performed (Krasnokutsky, 2021).

The whole process can be broken down into five main steps, where the first one is to define the main business problem that needs to be solved. So, if a company is manufacturing tyres, the problem to be solved is of course the detection of tyre defects. Besides that, there are a lot of important questions that needs to be asked, such as how the environment of the inspection site is, should the defects be categorised or is it just a binary defect/non-defect problem, is there any existing software that addresses the current problem, are there proper training data for the development of a DL model and so forth. After answering important questions revolving the project, the second step is to gather and prepare relevant data. High quality data is essential to train a DL algorithm, and these data can be obtained from open-source sites to client data, as well as gathering data from scratch. For data related to visual inspection, the collected images or videos must be of high resolution, or else the obtained results will be subpar (Krasnokutsky, 2021).

After obtaining the required data, the third step will be to start developing a DL model. There are multiple options when it comes to this step, where the company will have to choose between using DL model development services, using pre-trained models, or developing a DL model from scratch. DL model development services such as Google Cloud ML Engine and Amazon ML provides pre-defined templates that are non-customisable. So, if the defined problem can be solved using a provided template, this is a very viable option as significant time and cost can be saved because DL models do not need to be developed from scratch. For the second option, using a pre-trained model is doable if there is a DL model that is already created which solves similar issues that revolves around the main problem. Intel OpenVINO offers these models to be used by the public and using these enables companies to significantly shorten the deployment time and cost. The downside is that a pre-trained model will not completely comply with the defined problem, so the performance may not be as expected depending on the application. The third option is to develop a DL model from scratch, which is the way to go if the problem is extremely complex. High initial costs and long development hours are expected from this method, but the performance of the model is usually optimal as the model is tailor-made for the specific application. The fourth step is to train and evaluate the model, where the performance of the DL algorithm is evaluated and validated. After that, the last step is to deploy the visual inspection program and constantly look for ways to optimise and improve the overall model (Krasnokutsky, 2021).

2.6 Summary

Vision-based anomaly detection is an ongoing research topic within the CV community. It is applicable in a lot of practical areas, with the potential to transform the workflow of organisations into a much efficient and effective one. Studies on modern AI-based techniques for visual anomaly detection has been carried out for a relatively long period as well, and more research on this is still in progress. From reviewing previous works, it is suggested that unsupervised methods should be used for visual anomaly detection because it is difficult to curate a training dataset that covers all possible anomalies for a specific item. Besides that, unsupervised methods resemble real world scenarios, where anomalies are not known beforehand. However, there exists some open-source datasets that enables researchers to use supervised methods to perform visual anomaly detection as well.

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Introduction

In this study, the proposed approach to implement vision-based anomaly detection is to utilise both conventional and modern methods on relevant training images, and the performance of these methods will be evaluated. In order to properly implement these methods, relevant hardware devices and software tools are required. The flowchart in Figure 3.1 illustrates the general process flow for this project.

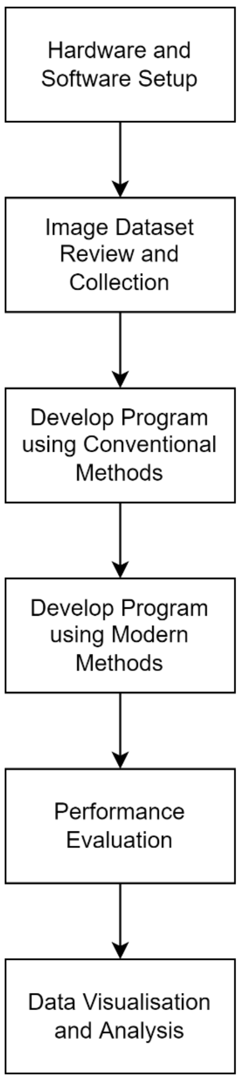

Hardware and Software Setup
Image Dataset Review and Collection
Develop Program using Conventional Methods
Develop Program using Modern Methods
Performance Evaluation
Data Visualisation and Analysis

Figure 3.1: General Process Flow.

3.2 Hardware Setup

The hardware setup for this project is relatively simple. To develop the required programs for visual anomaly detection, only a laptop was used to develop the whole program, where the Operating System (OS) is Windows 10. However, only the classical method was developed directly using the local platform, whereas the AI-based methods was run on Google Colaboratory's cloud computing system to run heavier workloads that requires Graphics Processing Unit (GPU) processing power. Table 3.1 shows the specification of the laptop used to develop the program.

Table 3.1: Specifications of Laptop Used in the Project.

CPU	Intel Core i7-7700HQ CPU @ 2.80GHz
CPU Cores	4
Memory	16 GB
Storage	256 GB SSD/1 TB HDD
GPU	Nvidia Geforce GTX 1050
OS	Windows 10

3.3 Software Setup

To run the programs locally on the device, an anaconda environment was set-up to ensure that the installed python modules will not interfere with the global python settings. Python is used as the programming language, which is preferred over other languages for its simplicity and readily available modules. More importantly, it supports essential libraries that are crucial to this project, such as OpenCV which is an open-source CV and ML software library, and also some other DL frameworks such as PyTorch, TensorFlow, Keras etc. The overall software setup for the local device is illustrated in Figure 3.2. The environment in Google Colaboratory is Linux based. Since the runtime is only available for a specific duration of time, the modules and frameworks can be installed directly without creating a virtual environment as the system will reset everything once the allocated duration is over.

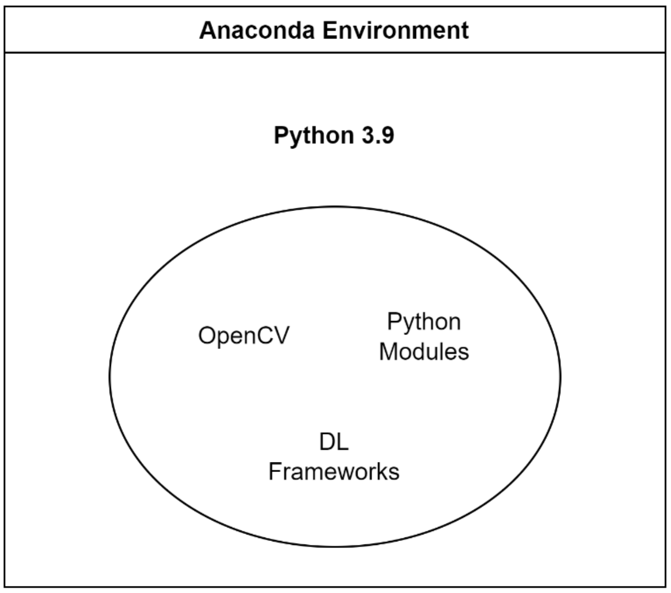


Figure 3.2: Software Setup.

3.4 Image Dataset

To train the AI models for visual anomaly detection, the open-source MVTec Anomaly Detection (MVTec AD) dataset will be used. This image dataset focuses on industrial inspection, and it has more than 5000 high-resolution images with various defects, as well as normal images. The pixel-precise annotations for all kinds of anomalies are also provided within this dataset (Bergmann, et al., 2021).

Five objects from the MVTec dataset were used for the training of AI models, which are bottle, transistor, metal nut, tile and hazelnut. The images for an image dataset are divided into training images and testing images. Training images are images of the objects in good condition, and these images will be used to train the AI models. Testing images comprises of both images of defective objects and good objects, which will be used to test whether the AI models are working as intended, and also for benchmarking purposes. Table 3.2 shows the number of training and testing images for each image dataset. For each image dataset, there will be several scenarios of defective objects, which are shown in Table 3.2. Figure 3.3 shows an example of a normal transistor and a defective transistor with a damaged case.

Table 3.2: Number of Training and Testing Images.

Image Dataset	**Num of Training Images**	**Num of Testing Images**
Bottle	209	broken large: 20 broken small: 22 contamination: 21 good: 20
Transistor	213	bent lead: 10 cut lead: 10 damaged case: 10 misplaced: 10 good: 60
Metal Nut	220	bent: 25 color:22 flip: 23 scratch: 23 good: 22
Tile	230	crack: 17 glue strip: 18 gray stroke: 16 oil: 18 rough:15 good: 33
Hazelnut	391	crack: 18 cut: 17 hole: 18 print: 17 good: 40

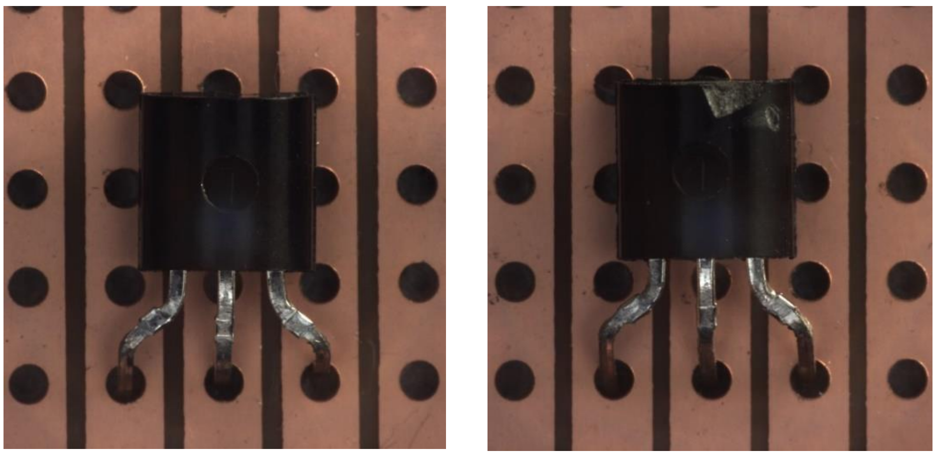


Figure 3.3: Normal Transistor (Left) and Defective Transistor (Right).

3.5 Template Matching

To implement TM, OpenCV already has several built-in methods to effectively implement this. The approach that was taken to implement the test program is to first obtain the original image, the template image and the threshold value for detection. This means that if an object has a detection score which is higher than the threshold value, then the object is a match with the template image. For this case, the template image is the specific anomaly that is required to be detected from the images.

To start the matching process, both the template and original images are first converted to grayscale. To make sure that both the original image and the template image are having the size, the original image will be looped through different scales to find the size that produces a high enough matching score to be considered as a good fit. This means that when the image is looped through a certain scale, then TM is performed using OpenCV's cv2.matchTemplate method. To obtain a more accurate result, the cv2.TM_CCOEFF_NORMED TM algorithm is used, which returns a normalised score with a maximum value of one. If the largest matching score for this image scale is higher than the threshold value that was set earlier, this means that this scale for the original image is able to generate an accurate match with template. Hence, this scale will be used for further matching and detection visualisation.

After the desired scale is set, then the result from the earlier TM process is used to determine which are the regions within the image where there is a possible anomaly. The obtained result from TM is a matrix, so the np.where method from Python’s numpy module is used to filter out (x,y) coordinates where the detection threshold is higher than the value that was initially set at the start of the program. As an example, if the detection threshold is set to be 0.8, then the locations where matching scores are higher than 0.8 within the result matrix will be recorded. After this, non-maxima suppression is applied to prevent the situation where multiple locations within the result matrix is actually referring to the same object. Once that is done, the relevant locations of possible anomalies are drawn on the original image using the cv2.rectangle method to visualise the detection of anomalies. The summarised process flow of the TM program is illustrated in Figure 3.4.

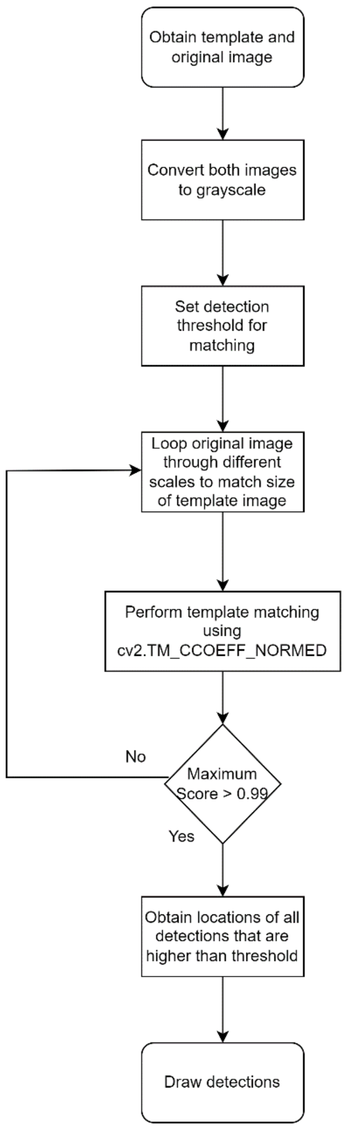

Obtain template and original image
Convert both images to grayscale
Set detection threshold for matching
Loop original image through different scales to match size of template image
Perform template matching using cv2.TM_CCOEFF_NORMED
Maximum Score > 0.99
No
Yes
Obtain locations of all detections that are higher than threshold
Draw detections

Figure 3.4: Process Flow for TM Program.

3.6 Structural Similarity Index

The SSIM algorithm is a built-in function within the scikit-learn python module which can be implemented directly. The program starts by taking two images as inputs, which is the first image and the second image. Since SSIM can only work if both images have the same dimensions, so the second image will be resized to have the same size as the first image if the size is different. After that, both images are converted to grayscale. Once that is completed, both images are fed into the compare_ssim function. This function returns the computed SSIM score, as well as the locations where there are differences between both images. The obtained locations are then fed into the opencv method cv2.threshold to perform thresholding on the areas where there are differences between the two images to highlight the differences, and the thresholded areas are then fed into the cv2.findContours to perform contouring. Therefore, the areas where both images differ can be shown clearly using thresholding and contouring. Lastly, bounding boxes are drawn at areas where there are differences between the two images. Figure 3.5 illustrates the process flow for the SSIM program.

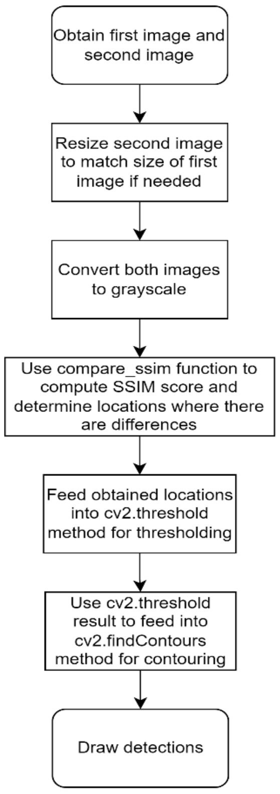

Obtain first image and second image
Resize second image to match size of first image if needed
Convert both images to grayscale
Use compare_ssim function to compute SSIM score and determine locations where there are differences
Feed obtained locations into cv2.threshold method for thresholding
Use cv2.threshold result to feed into cv2.findContours method for contouring
Draw detections

Figure 3.5: Process Flow for SSIM Program.

3.7 Feature Matching

When it comes to feature matching, the SIFT and ORB algorithms were implemented in this project. In addition to that, the BFM matcher is used over the FLANN matcher as the number of images that are dealt with in this project is not too large, so BFM will not be too slow, and it provides better accuracy as well. To start the program, two images are fed into the program as well, which are the first image and second image. Since feature matching algorithms are scale and rotation invariant, the scale of both images can be different from each other. Then, both input images will be fed into the sift_brute function and the orb_brute function for SIFT and ORB feature matching respectively.

To perform SIFT feature matching, a sift object needs to first be created using OpenCV's cv2.SIFT_create() method. Then, the sift object will be used to detect and compute the keypoints and descriptors of both images. After that, a BFM object is created using the cv2.BFMatcher() method. To perform matching, the BFM object will use K-nearest neighbours matching with a K value of 2 by using the descriptors from both images. All the macthes will be stored into a variable. Now, to filter out high quality matches, the ratio test from Lowe's paper was implemented. If the distance of a descriptor from the first image is less than the ratioed distance of a descriptor from the second image, then the match is considered as a good match. All prior detected matches will be filtered through this ratio test, and good matches are stored into a separate variable. Then, the good matches are visualised on both of the images. To compute a SIFT score to represent how similar both images are, the ratio v was calculated and returned by the function. The higher the SIFT score, the higher the similarity between two images. The process flow for SIFT feature matching is shown in Figure 3.6.

The process for ORB feature matching is very similar to the process for SIFT feature matching. An ORB object is first created using the cv2.ORB_create() method, and the keypoints and descriptors are detected and computed for both images. Then, the BFM object is created, and matching is performed as well. However, the K-nearest neighbours matching is not used in this case, so the direct BFM matching was used. After matching, the matches were sorted in order of their distance. Since ORB matches are different from SIFT matches, the ratio test was not used. Instead, a good match is defined as one with a distance of less than 50, note that ORB matches have a distance of 0 to 100. If a match has a distance of less than 50, it is considered as a good match, and it will be appended into another variable. All good matches will be visualised on both images, and the ORB score is also calculated by using the ratio between the number of good matches and the number of total matches. The higher the ORB score, the higher the similarity between two images. The process flow for ORB feature matching is shown in Figure 3.7.

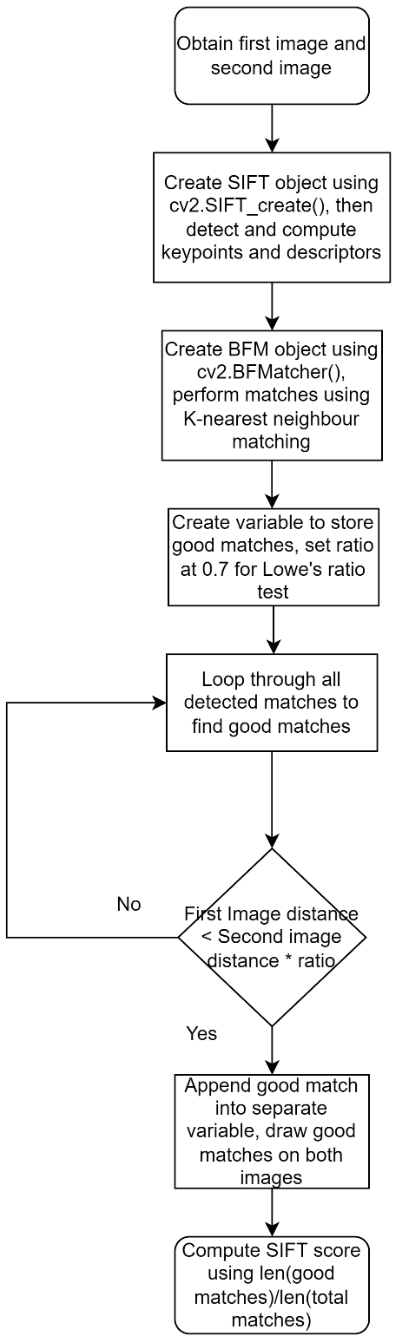

Obtain first image and second image
Create SIFT object using cv2.SIFT_create(), then detect and compute keypoints and descriptors
Create BFM object using cv2.BFMatcher(), perform matches using K-nearest neighbour matching
Create variable to store good matches, set ratio at 0.7 for Lowe's ratio test
Loop through all detected matches to find good matches
First Image distance < Second image distance * ratio
No
Yes
Append good match into separate variable, draw good matches on both images
Compute SIFT score using len(good matches)/len(total matches)

Figure 3.6: Process Flow for SIFT Feature Matching.

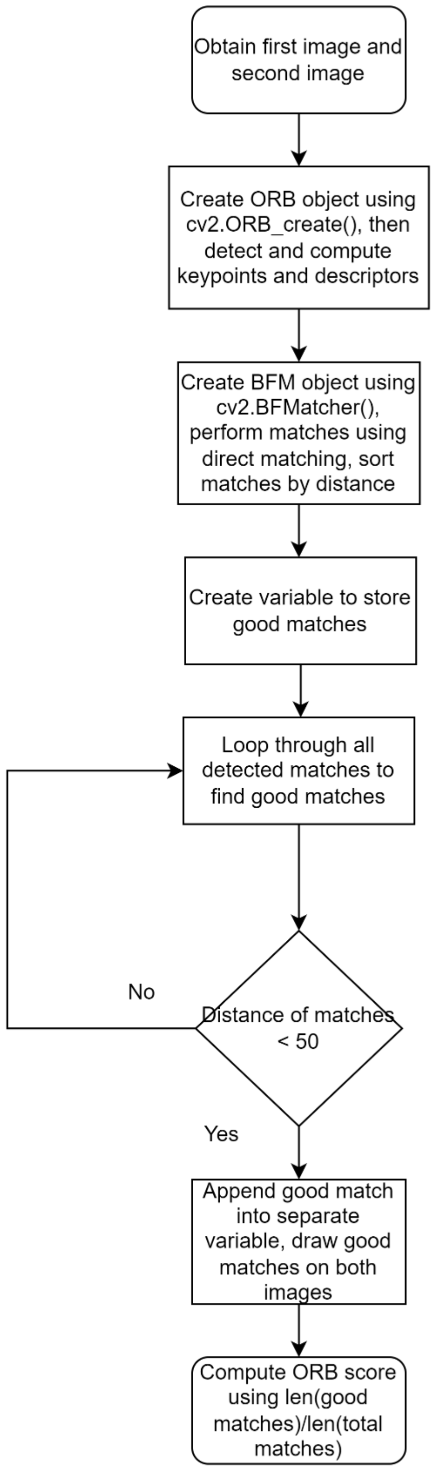

Obtain first image and second image
Create ORB object using cv2.ORB_create(), then detect and compute keypoints and descriptors
Create BFM object using cv2.BFMatcher(), perform matches using direct matching, sort matches by distance
Create variable to store good matches
Loop through all detected matches to find good matches
No
Distance of matches < 50
Yes
Append good match into separate variable, draw good matches on both images
Compute ORB score using len(good matches)/len(total matches)

Figure 3.7: Process Flow for ORB Feature Matching.

3.8 AI-Based Methods

When it comes to vision-based anomaly detection, the unsupervised learning approach is much more effective and practical than supervised learning, because it is nearly impossible to curate a dataset that covers all types of anomalies to train a model using supervised learning. Hence, unsupervised learning is the way forward for real life visual anomaly detection applications. In this project, the Anomalib DL library for anomaly detection will be used, which was developed by Akcay, et al. (2022). This open-source library focuses on unsupervised learning for anomaly detection purposes, and a set of state-of-the-art models and algorithms are included in the library. In addition to that, Anomalib provides an end-to-end process from the training dataset to deployment of model, and the architecture of this library is shown in Figure 3.8.

From the selection of provided AI models, the models that were implemented in this project are the Patch Distibution Modelling Framework (PaDiM) and the PatchCore model. PaDiM was proposed by Defard, et al. (2021), and it is a patch-based algorithm that relies on a pre-trained CNN feature extractor for embedding extraction. This means that an image will be divided into patches, and embeddings are extracted for each image patch. All layers of the pre-trained CNN will be utilised to ensure that all details are encapsulated. To reduce redundant information within the embeddings, a random selection method was used, which surprisingly performed well while reducing the complexity and training time of the model. A multivariate gaussian distribution is created for each patch embedding, and these distributions are modelled as a matrix of gaussian parameters. For the inference process, the Mahalanobis distance was used by the authors to compute an anomaly score for each patch of the image. The architectural overview of the PaDiM framework is illustrated in Figure 3.9.

The PatchCore model was proposed by Roth, et al. (2022), and the algorithm also divides an image into patches. The idea behind this algorithm is that an image is anomalous if any of the patches are anomalous. This model is also based on a pre-trained CNN, but only the middle layers are utilised as the authors believed that the lower layers are too broad, whereas the higher layers are too specific to the dataset the model is trained on. For the inferencing process, the memory bank is coreset subsampled, which creates a subset of the image dataset in order to reduce the search cost that is frequently seen in nearest neighbour search algorithms. The anomaly score is obtained by taking the maximum distance between the test patch in each collection to each respective nearest neighbour. Figure 3.10 shows the architectural overview of the PatchCore model.

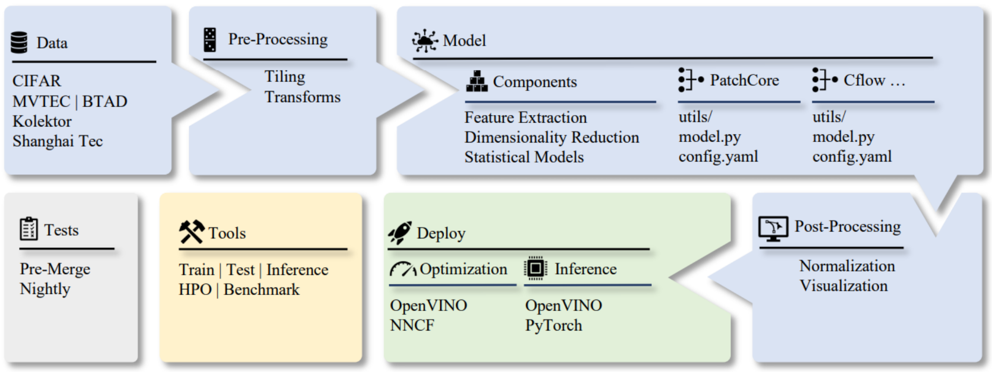


Figure 3.8: Architecture of the Anomalib Library (Akcay, et al., 2022).

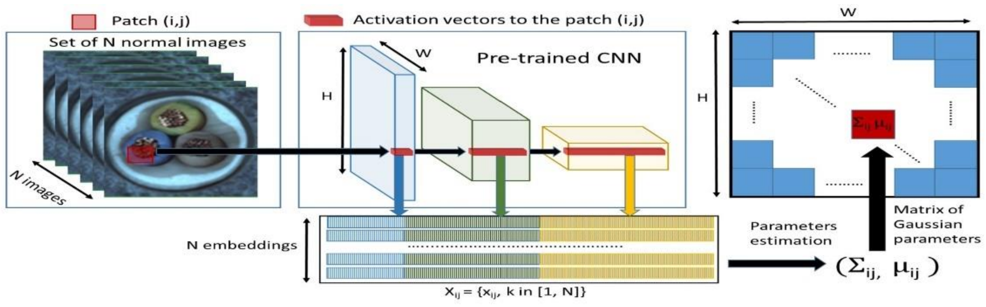


Figure 3.9: Architecture of the PaDiM Framework (Defard, et al., 2021).

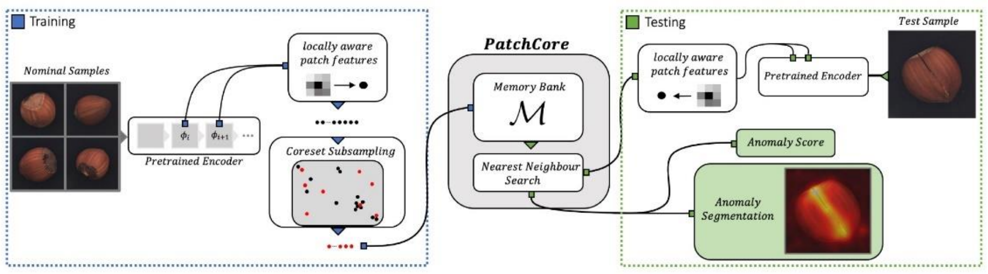


Figure 3.10: Architecture of the PatchCore Framework (Roth, et al., 2022).

The Anomalib library is available as a Python module, so Anomalib has to first be installed into the Python environment. After that, the python program to kickstart the training process needs to be run, while specifying the specific configuration file to be used in order to decide whether the PaDiM or PatchCore model is to be used, as well as which object from the MVTec AD dataset will be fed to the model for training. After the training is completed, the model will then be used in another program to perform inferencing on test images, in which the anomalous regions will be segmented and visualised on the test images.

To benchmark the AI models, two metrics will be used, which are the Area Under the Receiver-Operator Characteristic Curve (AUROC) and F1 Score. The AUROC curve is a benchmarking metric that is usually used for binary classification problems, and it is a probability curve. The area under this curve is essentially the measure of separability, so it tells how capable a model is at segregating the data into binary classes. The ROC curve is generated by plotting the True Positive Rate (TPR) of the data against the False Positive Rate (FPR). The TPR is defined in equation 3.1, whereas the FPR is defined in equation 3.2. The TPR indicates the proportion of positive class that was classified correctly, whereas the FPR indicates the proportion of negative class that was classified incorrectly. For this application, a high AUROC score means that the model is good at distinguishing between normal and anomalous images, so a high AUROC score is desirable.

$$TPR = \frac{TP}{TP+FN} \tag{3.1}$$

where

TP = Number of True Positive Cases

FN = Number of False Negative Cases

$$FPR = \frac{FP}{FP+TN} \tag{3.2}$$

where

FP = Number of False Positive Cases

TN = Number of True Negative Cases

The F1 Score is another benchmarking metric that is suitable for classification models. This score is the improvement of two simpler performance metrics, which are the Precision and Recall metrics, both of these are defined in equation 3.3 and 3.4 respectively. Precision is the measure of correct percentage of true positives, whereas recall is the percentage of detected true positives within all positive cases. In other words, precision is the question where out of all positive predictions, how many are truly positive? Whereas recall is the question where out of all real positive cases, how many are predicted as positive? The F1 score is a combination of precision and recall, which is defined in equation 3.5. Since the F1 score is the harmonic mean of precision and recall, it gives equal weight to both precision and recall. Hence, a high F1 Score means that both the precision and recall are high, which is desirable.

$$Precision = \frac{TP}{TP+FP} \tag{3.3}$$

where

TP = Number of True Positive Cases

FP = Number of False Positive Cases

$$Recall = \frac{TP}{TP+FN} \tag{3.4}$$

where

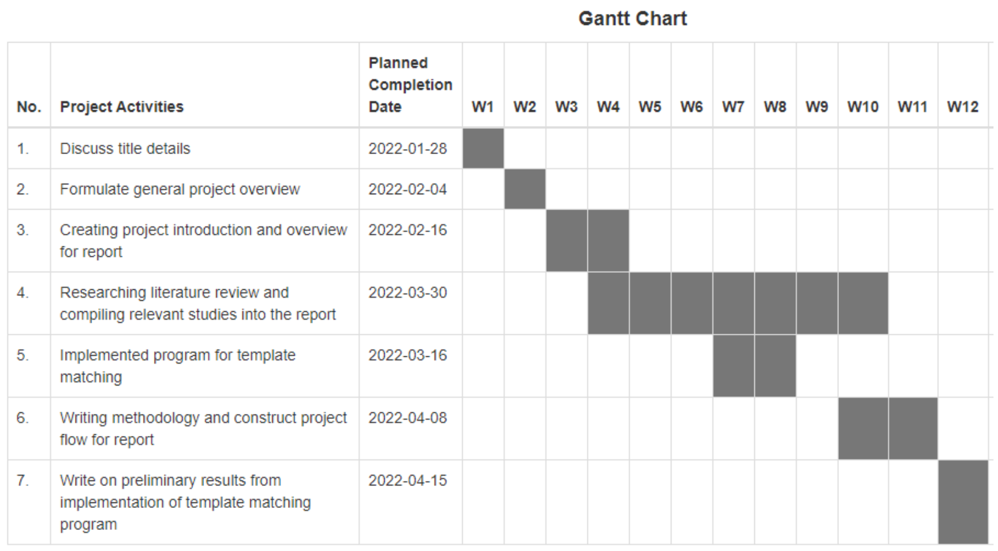
TP = Number of True Positive Cases

FN = Number of False Negative Cases

$$F1\,Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3.5}$$

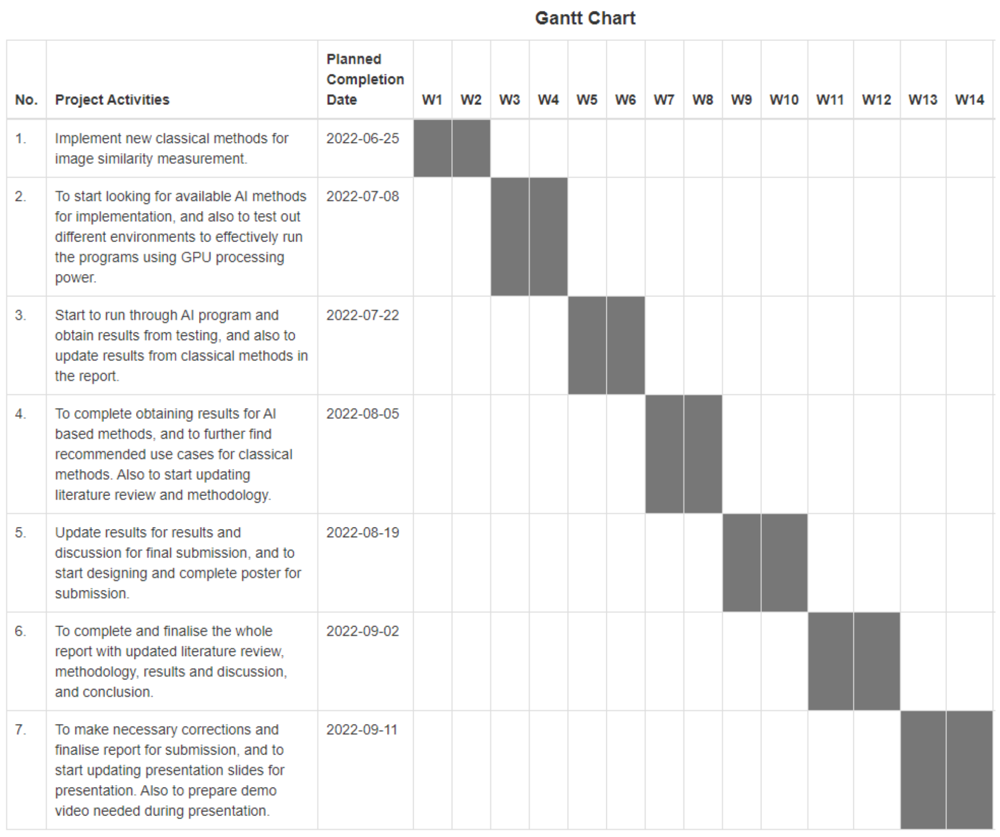
For both metrics, the image and pixel level scores will be calculated. The difference between image and pixel level scores is that the image level score takes the whole image as a whole for benchmarking purposes, whereas pixel level score considers each pixel within an image for benchmarking purposes. For example, determining whether an image contains an anomaly or not is an image level problem, whereas determining whether a single pixel within an image is considered as anomalous is a pixel level problem.

3.9 Project Work Plan



Gantt Chart

No.	Project Activities	Planned Completion Date	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12
1.	Discuss title details	2022-01-28	▨											
2.	Formulate general project overview	2022-02-04		▨										
3.	Creating project introduction and overview for report	2022-02-16			▨	▨								
4.	Researching literature review and compiling relevant studies into the report	2022-03-30				▨	▨	▨	▨	▨	▨	▨		
5.	Implemented program for template matching	2022-03-16							▨	▨				
6.	Writing methodology and construct project flow for report	2022-04-08										▨	▨	
7.	Write on preliminary results from implementation of template matching program	2022-04-15												▨



Gantt Chart

No.	Project Activities	Planned Completion Date	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14
1.	Implement new classical methods for image similarity measurement.	2022-06-25	▨	▨												
2.	To start looking for available AI methods for implementation, and also to test out different environments to effectively run the programs using GPU processing power.	2022-07-08			▨	▨										
3.	Start to run through AI program and obtain results from testing, and also to update results from classical methods in the report.	2022-07-22					▨	▨								
4.	To complete obtaining results for AI based methods, and to further find recommended use cases for classical methods. Also to start updating literature review and methodology.	2022-08-05							▨	▨						
5.	Update results for results and discussion for final submission, and to start designing and complete poster for submission.	2022-08-19									▨	▨				
6.	To complete and finalise the whole report with updated literature review, methodology, results and discussion, and conclusion.	2022-09-02											▨	▨		
7.	To make necessary corrections and finalise report for submission, and to start updating presentation slides for presentation. Also to prepare demo video needed during presentation.	2022-09-11													▨	▨

Figure 3.11: Project Gantt Chart for Semester 1 (Top) and Semester 2 (Bottom).

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

Results for conventional and modern AI-based methods were analysed and compared. In addition to that, benchmarking was also performed for AI-based methods to assess the performance and accuracy of the model to detect anomalies for the defective items.

4.2 Template Matching

The program for TM was tested using a simple template image and original image, which is as shown in Figure 4.1 below. It is shown that the template image is a yellow-coloured circular port, in which there are seven of them in the original image. Using TM algorithms, the goal is to detect all occurrences of this yellow port in the original image. Since the template image has the same scale with the original image, the best possible match occurs when the scale of the original image is maintained at the original size. Initially when the program was run with a threshold value of 0.9, only three occurrences of the template image were detected on the original image. This means that only three regions within the original image have a match that is 0.9 and above with the template image. When the threshold value is reduced to 0.8, four of the occurrences were detected accurately. However, when the threshold is reduced to 0.78, a false occurrence was detected in the image, but the remaining three occurrences that were not detected previously were still not detected. Figure 4.2 shows the results when the threshold value is 0.8 and 0.78.

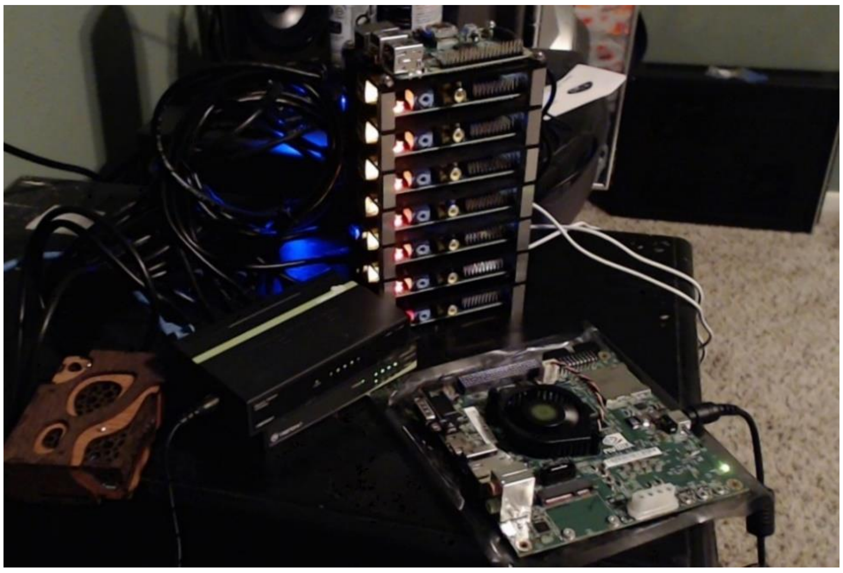
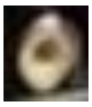
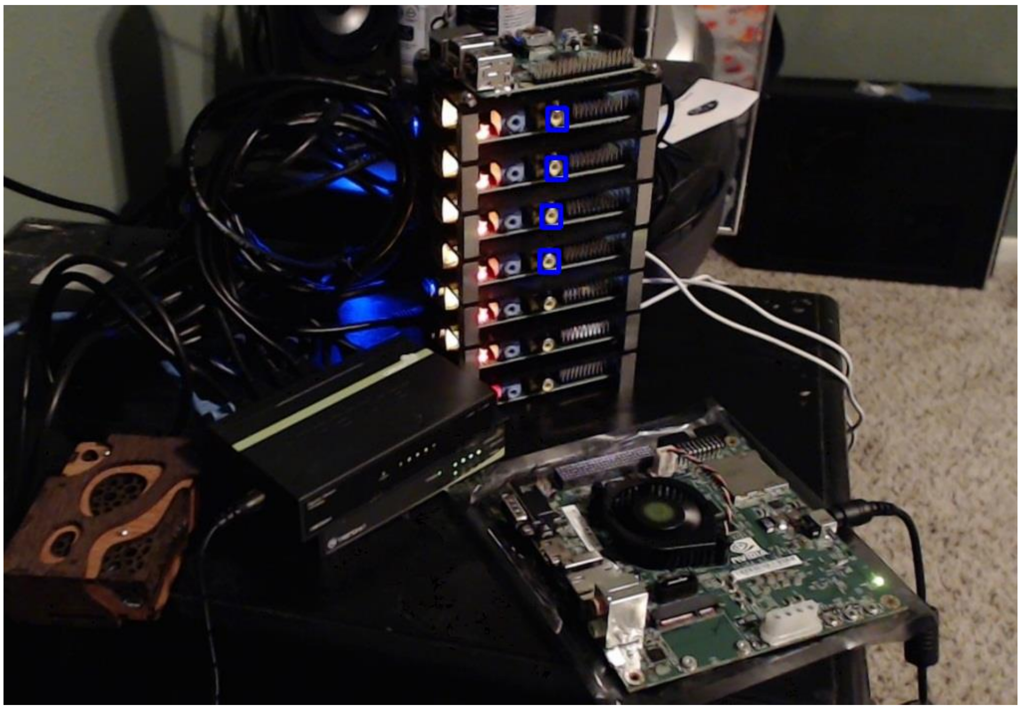



Figure 4.1: Original Image (Right) and Enlarged Template Image (Left) (PythonProgramming, n.d.).

Notice that there is a false positive detection on the left side of the image when the threshold value is 0.78. However, the three yellow ports at the bottom are still not detected. This suggests that the orientation of the ports within the image, as well as the brightness of the specific region will likely play a huge part in determining whether an occurrence can be accurately detected. When the threshold value is decreased to 0.5, more false positive occurrences were detected, which is shown in Figure 4.3. From these results, it can be observed that TM is accurate only if the region within an image is almost identical to the template image.

In addition to that, TM is only effective if the object to be detected is almost identical to the template image. In this above example, the yellow-coloured port is the exact object that exists in the original image, hence its' occurrence was detected, but not all occurrences were able to be detected accurately. However, situations like this are almost non-existent in practical scenarios for anomaly detection, where defects will occur in all shapes and sizes.



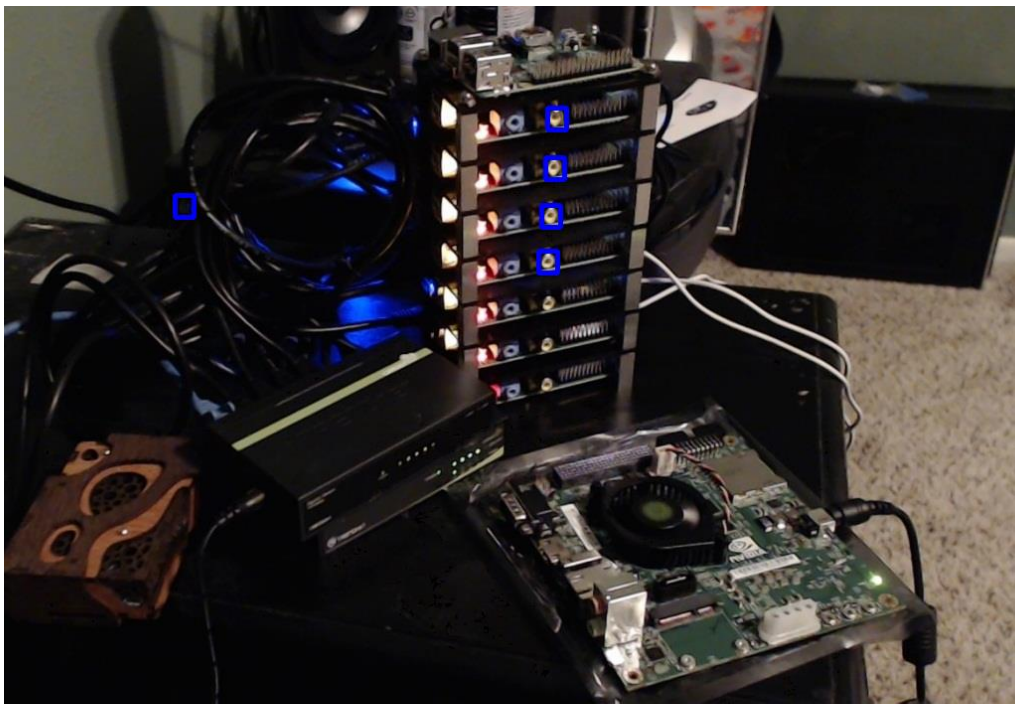


Figure 4.2: Results when Threshold Value Equals 0.8 (Top) and 0.78 (Bottom).

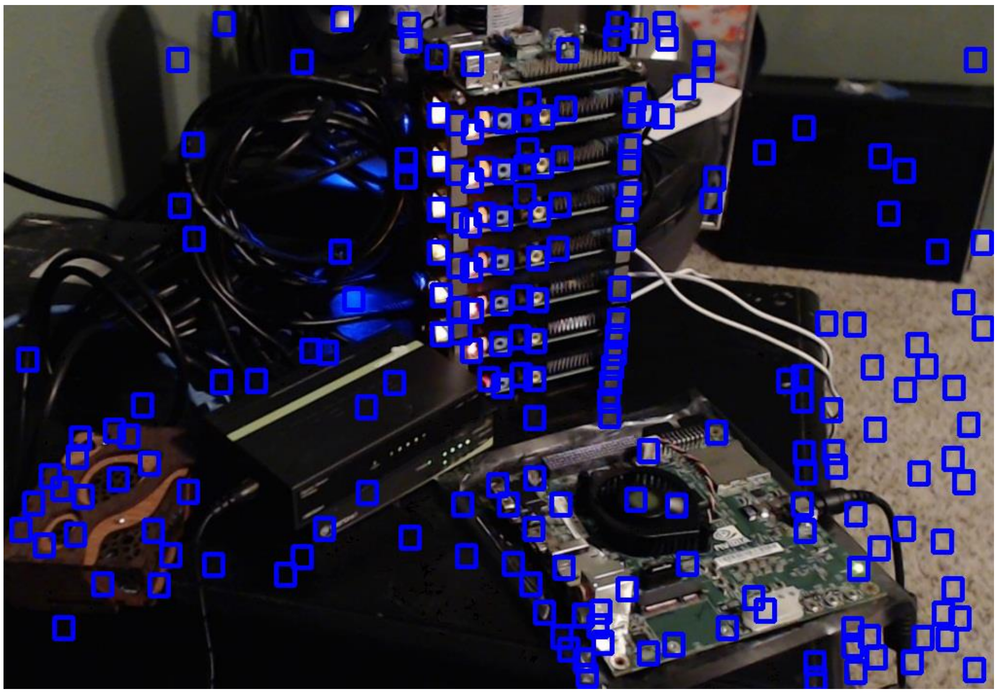


Figure 4.3: Results when Threshold Value Equals 0.5.

Another possible implementation for anomaly detection using TM is to use the image of a good item as the template image, and then trying to match the whole image to other images to see if the images will match. If the images match, this means that the item in the original image is not defective. Figure 4.4 shows a good transistor from the MVTec AD dataset being used as the template image, and a few good and defective transistors were used as the original image to be tested. The results show that not only is the algorithm unable to accurately differentiate good and defective transistors, but it also only matches with images that are very close to the original image in terms of placement and background of the image. Figure 4.5 shows a defective transistor with a damaged case that was matched with the transistor in Figure 4.4. Upon visual inspection, the placement of the transistor and the background of the original image is similar to the template image, which is why the algorithm detected this as a match with a detection threshold of 0.8. In another scenario where a bottle in good condition is used as the template image, all the tested good and defective bottles were matched to the template image, because all the bottles are positioned at the same exact position. Hence, the TM algorithm detects all the bottles as having a good match with the template image. The results are shown in Figure 4.6 and Figure 4.7.

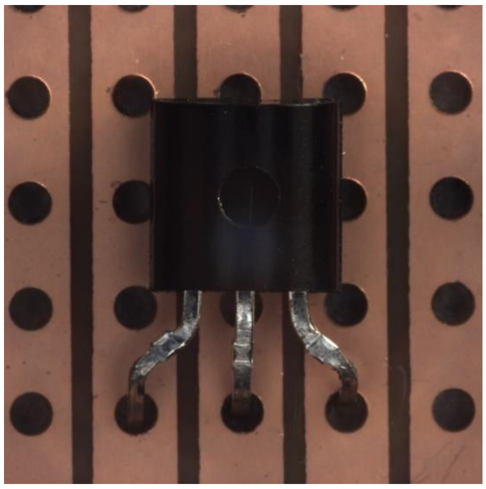


Figure 4.4: Good Transistor as Template Image.

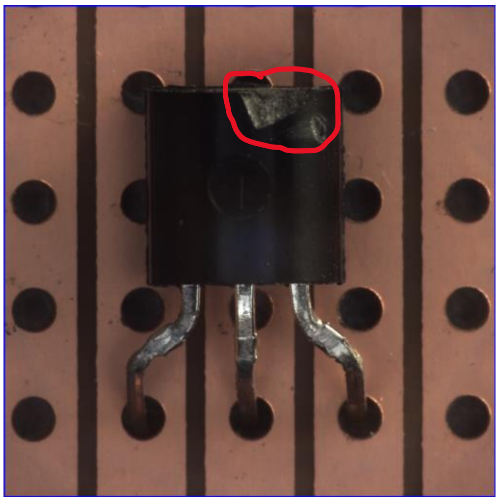


Figure 4.5: Defective Transistor Matched with Good Transistor.

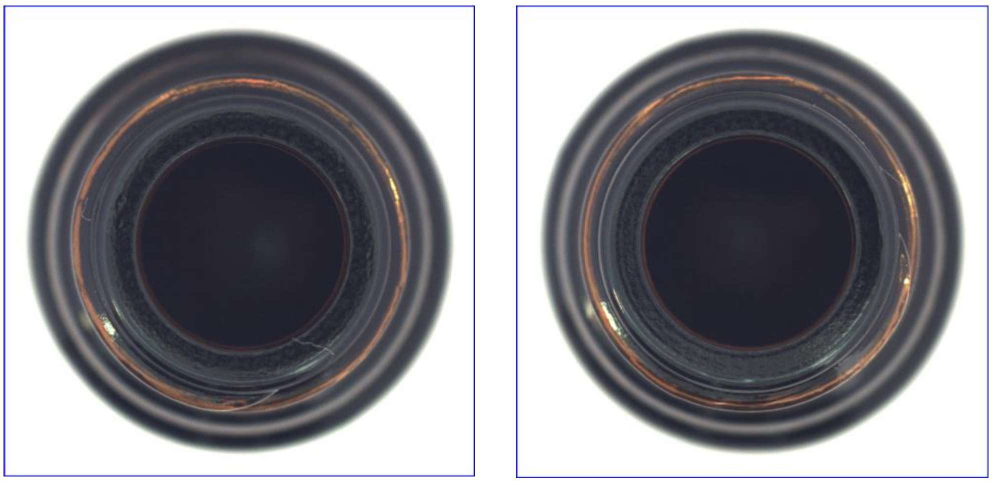


Figure 4.6: Good Bottles Matched with the Template Image of a Good Bottle.

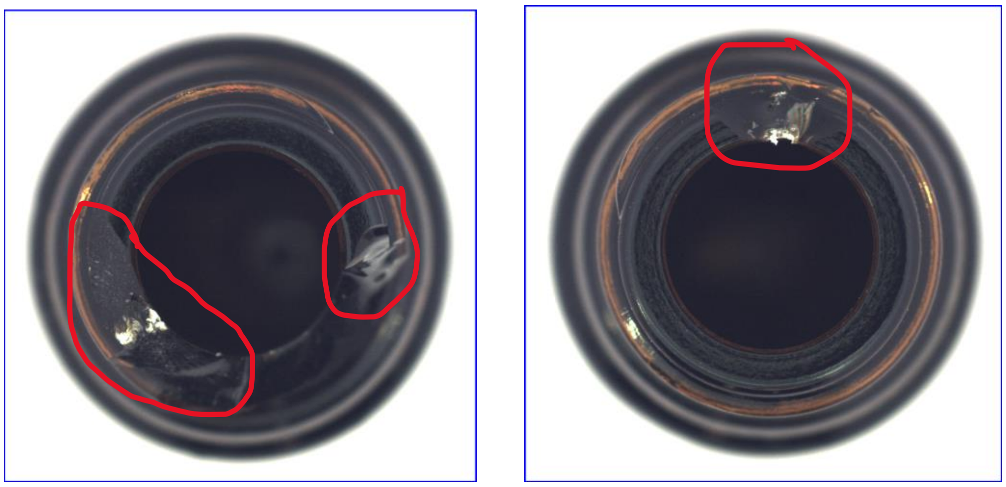


Figure 4.7: Defective Bottles Matched with a Good Bottle.

From the above results, it was shown that TM is not suitable for anomaly detection applications as it is impossible to set an anomaly as a template image. This is because anomalies and defects can occur in all types of shapes and sizes. It is futile to set different types of template images to detect anomalies as there are too many defect variations. When it comes to matching the whole image to the template image, the results have shown that this method is unreliable as TM takes into account of the difference in pixel properties within a region, and it cannot differentiate the defective object. Hence, there are scenarios where a defective object managed to match with the good template image because the position of the object and the background is similar to the one in the template image.

4.3 Structural Similarity Index

There are scenarios where SSIM can perform well to detect changes or outliers within an image, which is when the images follow a specific format or template. Figure 4.8 showcases some scenarios where the SSIM algorithm is able to pinpoint the differences or anomalies within the images. As shown on the images, SSIM can perform well if all the images to be compared are almost perfectly aligned and structured. Figure 4.9 illustrates the result on the test images from Figure 4.8, and the algorithm is able to successfully determine the modified areas accurately. However, a key part to the SSIM algorithm is the generated score, and the score for these two images is 0.9347.

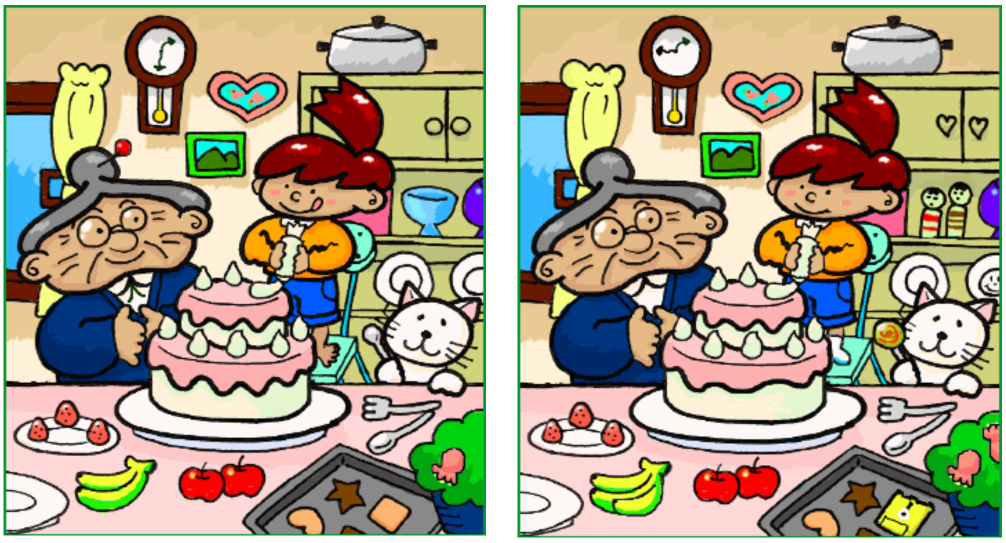


Figure 4.8: Spot the Differences between Original Image (Left) vs Modified Image (Right) (Wikipedia, n.d.).

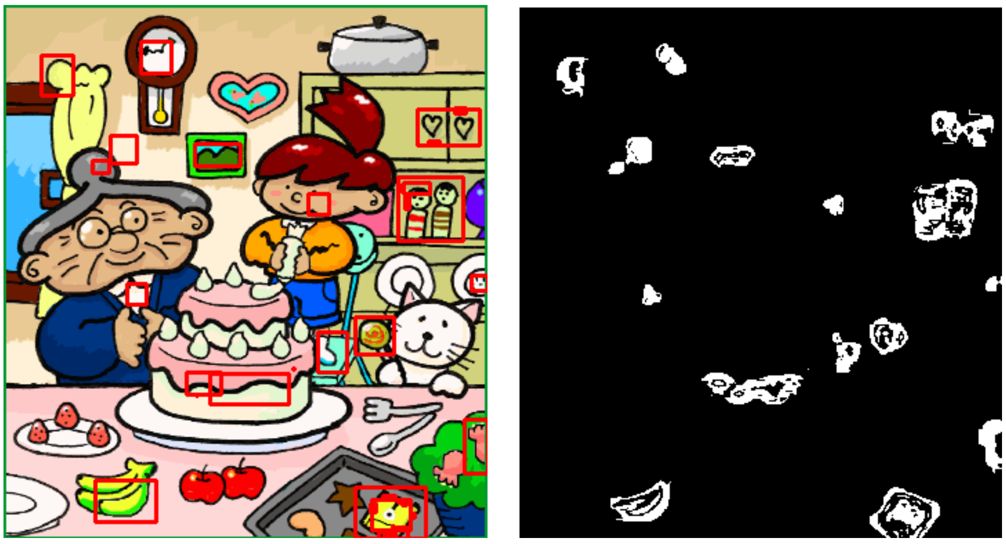


Figure 4.9: Bounding Boxes on Differences on Modified Image (Left) and Thresholding to Accentuate Modified Parts (Right).

Take note that the SSIM score has a range of 1 to -1, where a score of 1 represents perfect resemblance between the two images. Hence, a score of 0.9347 indicates that both images are almost identical to each other. However, this score is misleading because there are differences between the two images, and these small differences are in fact the defects that needs to be filtered out when it comes to anomaly detection. By conventional logic, a SSIM score of 0.9347 would suggest that there are no anomalies within the test images, but the result shows otherwise. Therefore, when it comes to real life anomaly detection, a high SSIM score as in this scenario cannot guarantee that there are no defects within the tested image.

The SSIM algorithm was also tested on the MVTec AD dataset, and the results are displayed in Table 4.1, which compares relative SSIM scores of different test subjects with respect to a good sample of the specific object. From the obtained results, the first observation is that there is not much difference in the computed SSIM scores between good and defective objects when they are compared to a good object as a reference. When the bottle is used as the reference object, it was observed that the overall SSIM score is higher than other items, which is because the image of all the bottles are positioned in a fairly similar manner. Therefore, there is very less variation of bottle placement between all the images, which means that there is less perceived difference of the test subject, which gives a relatively high SSIM score. The same could be said for the transistors, where the images all look fairly similar where the transistor is focused in the middle with a simlar background for all the test images. Hence, the SSIM algorithm is unable to discriminate between items that have defects and normal items.

The SSIM scores of metal nuts are lower because the images of metal nuts will have a different planar rotation, so the placement of the subject in one image will not be as similar to the placement of the subject in another image, which causes the reduction in SSIM scores. Figure 4.10 demonstrates the planar rotation of metal nuts in different images. Tiles have the lowest overall SSIM score because the patterns in tiles are completely randomised, so there is no fixed structure on how the patterns and dots will be aligned within the tile itself. Hence, SSIM classifies the difference in tile pattern as a difference in structure, hence the low computed scores. Figure 4.11 illustrates the random patterns on different tile images. Nevertheless, the overall score of all the items are not important if there is no clear distinction between scored for good and defective objects, which again shows that SSIM algorithms are not fit for detecting small and actual differences between test subjects under practical settings.

Table 4.1: SSIM Scores for Different Test Subjects.

Reference Object	**SSIM Scores**				
	Good Object 1	Good Object 2	Defective Object 1	Defective Object 2	Defective Object 3
Good Bottle	0.7636	0.7576	0.7922	0.7042	0.7572
Good Transistor	0.6904	0.7352	0.6853	0.7323	0.7545
Good Metal Nut	0.5535	0.5744	0.6149	0.5880	0.5471
Good Tile	0.2648	0.2525	0.2303	0.2516	0.2500

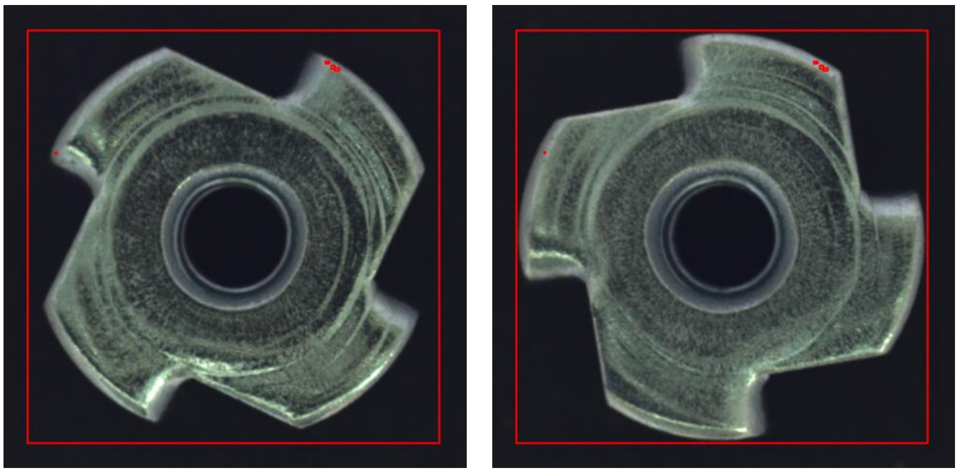


Figure 4.10: Planar Rotation of Metal Nuts Perceived as a Difference Between Images.

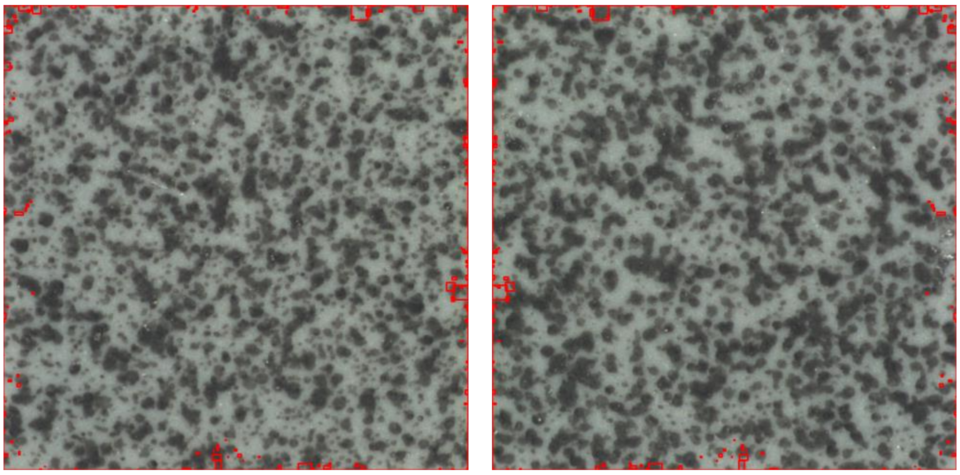


Figure 4.11: Differences in Tile Pattern for Good Tile Images.

4.4 Feature Matching

Feature-based matching should be more robust as compared to previously reviewed conventional methods as it is scale and rotation invariant. The tested feature matching algorithms are SIFT and ORB, and the BFM algorithm was used to match the descriptors as it is more accurate than the FLANN based matcher. Figure 4.12 illustrates an example of the matching result between a picture of Mona Lisa and its rotated counterpart by 45 degrees. As shown in the Figure 4.12, both SIFT and ORB algorithms were able to detect similar features from both images even when one of the images is rotated. To generate a similarity score, the ratio between the number of high-quality matches and the total number of matches was computed. The similarity score for SIFT is 0.48, and the score for ORB is 0.9849. From the similarity scores, it can be deduced that ORB performs exceptionally better than SIFT as it is able to produce a lot of high-quality matches that are low in distance.

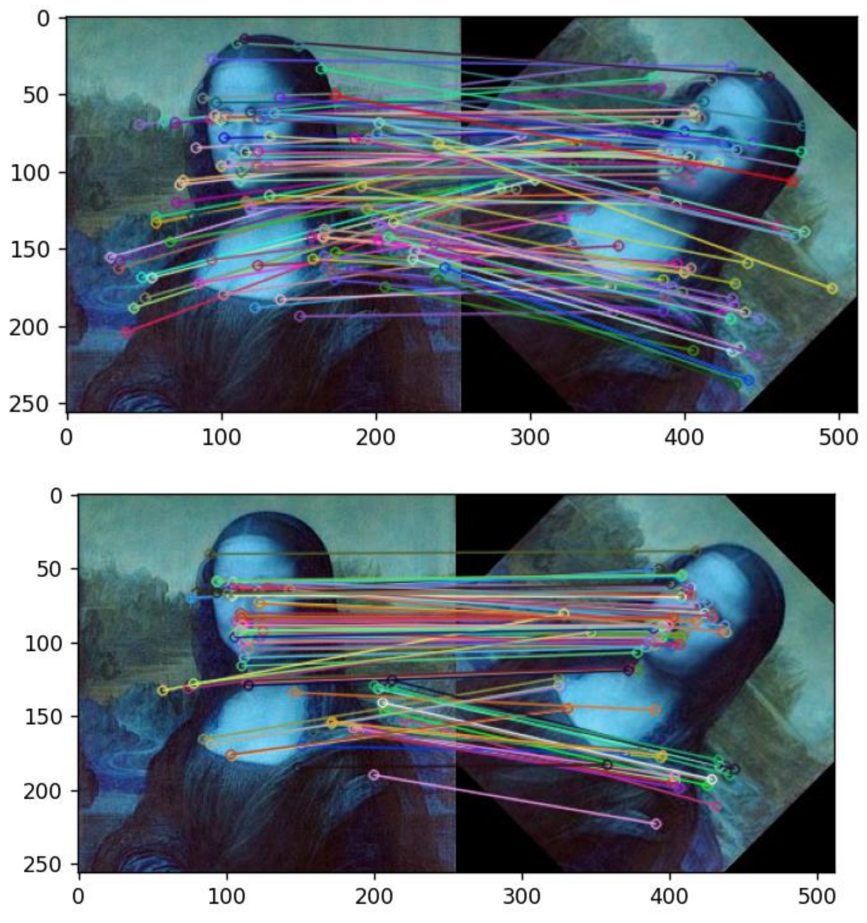


Figure 4.12: Matched Keypoints between Original and Rotated Image for SIFT (Top) and ORB (Bottom).

By using two other Mona Lisa test images where one of the images is rotated by 90 degrees to the left, and the other test image is translated by -45 x -45 pixels as shown in Figure 4.13, the similarity scores, (SIFT, ORB) for both of these images are (0.75, 1.0) and (0.505, 0.9342) respectively. Again, ORB has performed exceptionally better than SIFT as it was able to match strong features between the images with a high degree of accuracy. Therefore, the ORB algorithm will be used to perform tests on the MVTec AD dataset to determine if it is able to discriminate defective samples from good ones. The results are tabulated in Table 4.2 and note that an additional hazelnut test image was added in this test.

From the results in Table 4.2, it is obvious that the ORB algorithm is also unable to accurately differentiate good and defective products as the similarity scores for both good and defective images are relatively close to each other. An interesting occurrence for this test is the very low similarity scores for the tile and hazelnut images. This is caused by the lack of distinctive features within these images, which makes it difficult for the ORB algorithm to extract keypoints that are essential to perform matching. In addition to that, the three-dimensional rotation of the hazelnut test image which is shown in Figure 4.14 also drastically affects the ability of the algorithm to match keypoints, as different sides of the hazelnut may have different keypoints and descriptors. Nevertheless, feature-based matching algorithms are not able to accurately differentiate between good and anomalous test images.

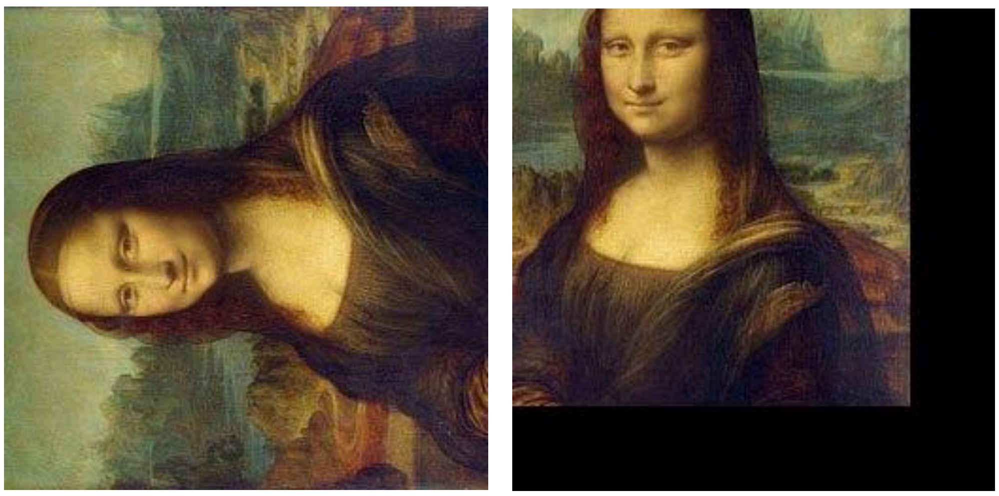


Figure 4.13: Rotated by 90 Degrees to the Left (Left) and Translated by -45 x -45 pixels (Right).

Table 4.2: ORB Scores for Different Test Subjects.

Reference Object	**ORB Scores**				
	Good Object 1	Good Object 2	Defective Object 1	Defective Object 2	Defective Object 3
Good Bottle	0.5528	0.6148	0.4824	0.4632	0.5932
Good Transistor	0.5508	0.4486	0.5636	0.5214	0.5000
Good Metal Nut	0.5128	0.4643	0.4597	0.3761	0.3852
Good Tile	0.3194	0.3007	0.0840	0.1151	0.2208
Good Hazelnut	0.0000	0.0000	0.0284	0.0286	0.0238

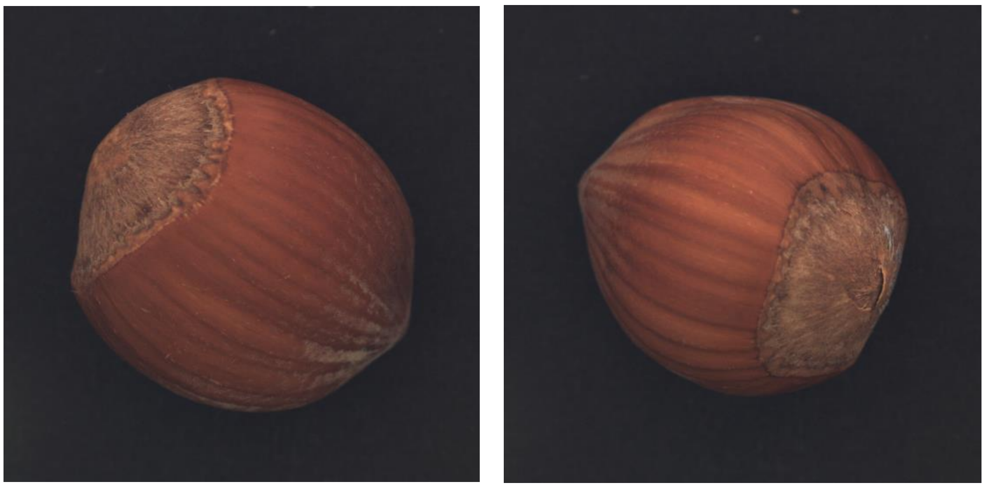


Figure 4.14: Three-Dimensional Rotation of Good Hazelnut Test Image.

4.5 AI-based Method

AI-based methods have shown very promising results when it comes to real-world anomaly detection scenarios. Two models were used from the Anomalib library to train and perform inferencing on the MVTec AD dataset, which are the PaDiM and PatchCore models. To benchmark the performance of these models, the AUROC and F1 Score metrics were used. The range for both of these metrics is [0,1], where a score of 1 is the best possible score, and a score of 0 is the worst possible score. Table 4.3, Table 4.4, Table 4.5 and Table 4.6

shows the image AUROC, image F1 Score, pixel AUROC and pixel F1 Score respectively for different objects. Note that the higher the score, the better the performance of the model at distinguishing between the good and defective objects.

Table 4.3: Image AUROC Scores for Different Models and Objects.

Model	**Image AUROC Scores**					
	Bottle	**Transistor**	**Metal Nut**	**Tile**	**Hazelnut**	**Average**
PaDiM	0.9937	0.9200	0.9614	0.9502	0.7504	0.9151
PatchCore	1.0000	1.0000	0.9966	1.0000	1.0000	0.9993

Table 4.4: Image F1 Scores for Different Models and Objects.

Model	**Image F1 Scores**					
	Bottle	**Transistor**	**Metal Nut**	**Tile**	**Hazelnut**	**Average**
PaDiM	0.9764	0.7957	0.9738	0.9341	0.8364	0.9033
PatchCore	1.0000	1.0000	0.9894	1.0000	1.0000	0.9979

Table 4.5: Pixel AUROC Scores for Different Models and Objects.

Model	**Pixel AUROC Scores**					
	Bottle	**Transistor**	**Metal Nut**	**Tile**	**Hazelnut**	**Average**
PaDiM	0.9830	0.9679	0.9696	0.9339	0.9779	0.9665
PatchCore	0.9844	0.9817	0.9895	0.9610	0.9874	0.9808

Table 4.6: Pixel F1 Scores for Different Models and Objects.

Model	**Pixel F1 Scores**					
	Bottle	**Transistor**	**Metal Nut**	**Tile**	**Hazelnut**	**Average**
PaDiM	0.7220	0.6316	0.7646	0.5397	0.5621	0.6440
PatchCore	0.7215	0.6688	0.8527	0.6271	0.6220	0.6984

As shown from the obtained results above, AI-based methods are able to achieve much better results as compared to classical methods. Most of the benchmarking scores are above 0.9, except for the pixel F1 Scores which are low for both models. To test the accuracy of the segmentation of anomalies from the test images, each of the test images were fed into an inferencing program. Figure 4.15a and Figure 4.15b shows the sample results for the PaDiM model, whereas Figure 4.16a and Figure 4.16b shows the sample results for the PatchCore model. These results clearly prove the ability of AI models to accurately point out anomalies from defective objects, which is much better when compared to classical methods that were discussed above. This indicates that the learning process AI models go through is essential for a program to accurately learn the differences between normal and anomalous objects.

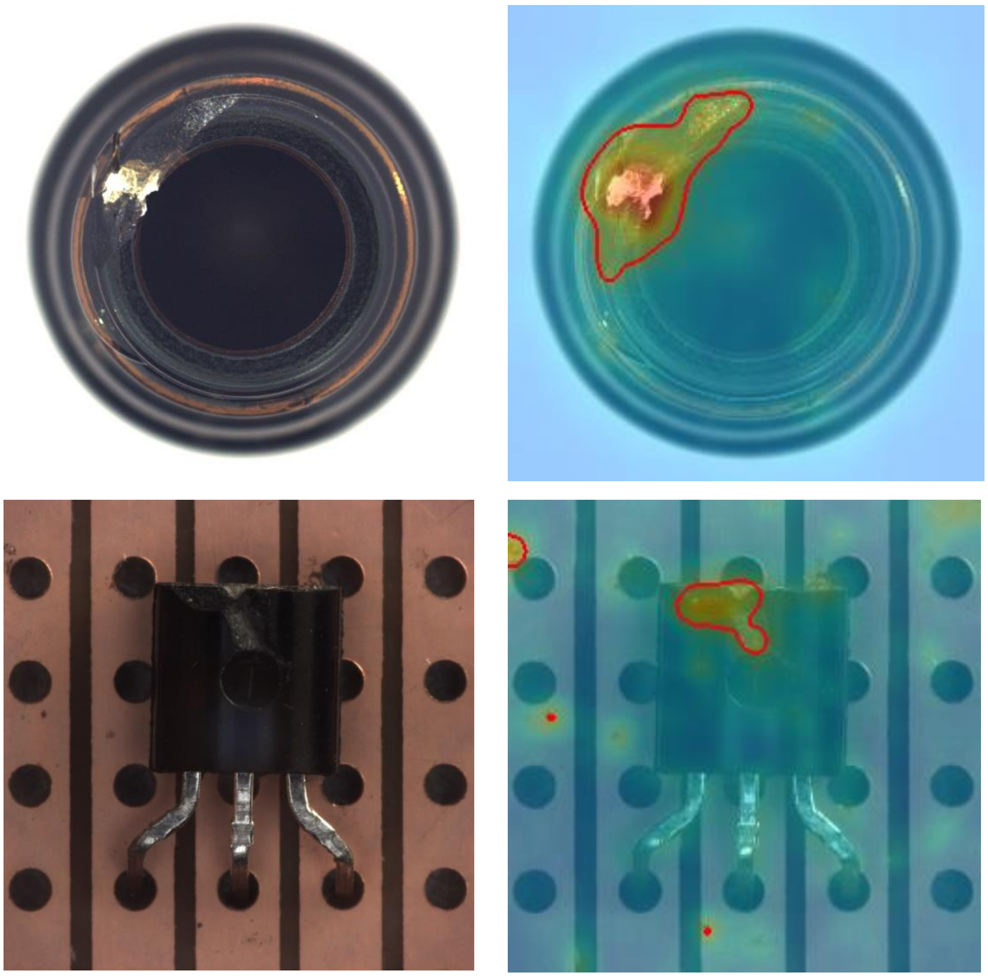


Figure 4.15a: Sample Inferencing Results for the PaDiM Model.

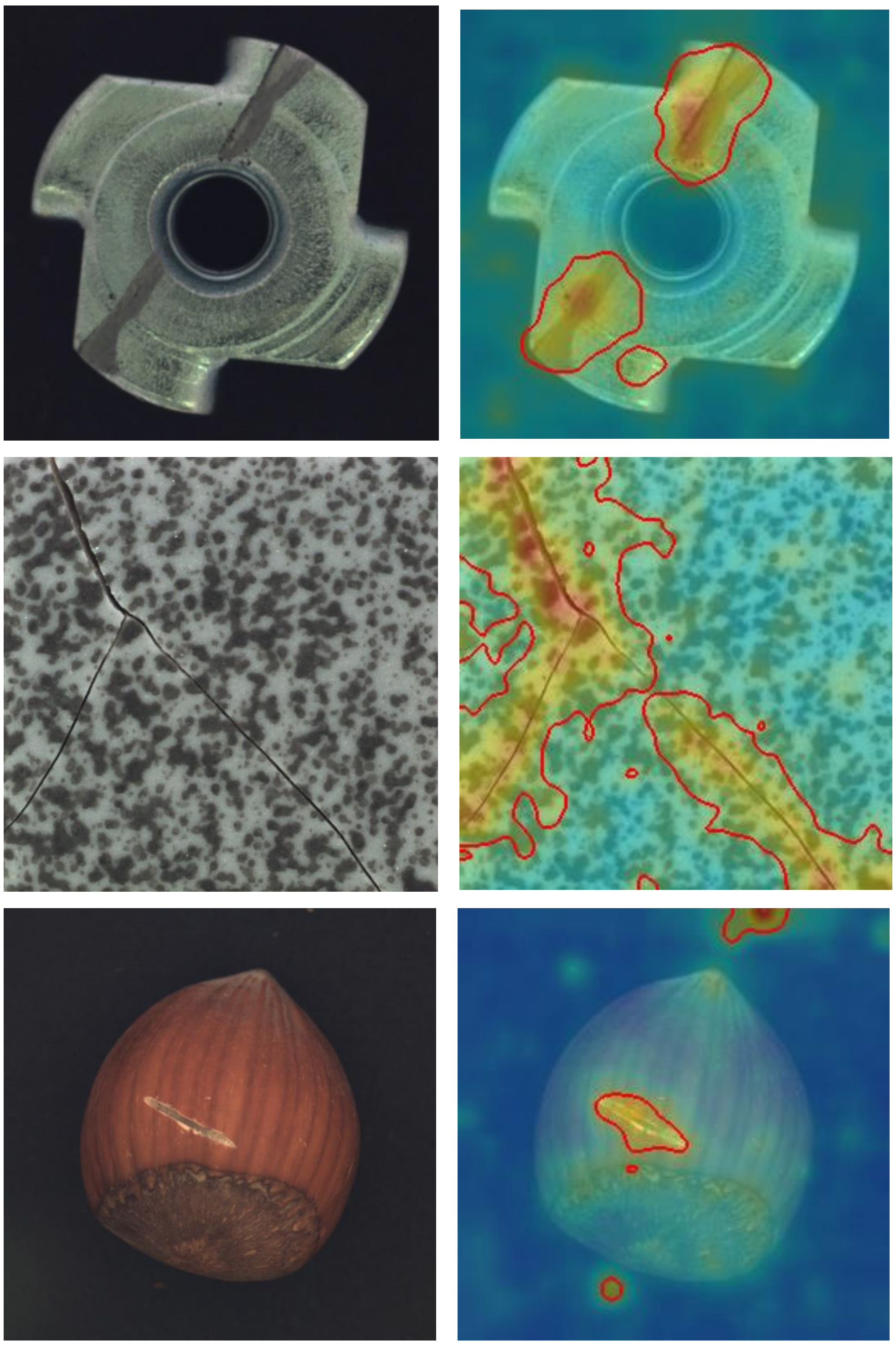


Figure 4.15b: Sample Inferencing Results for the PaDiM Model.

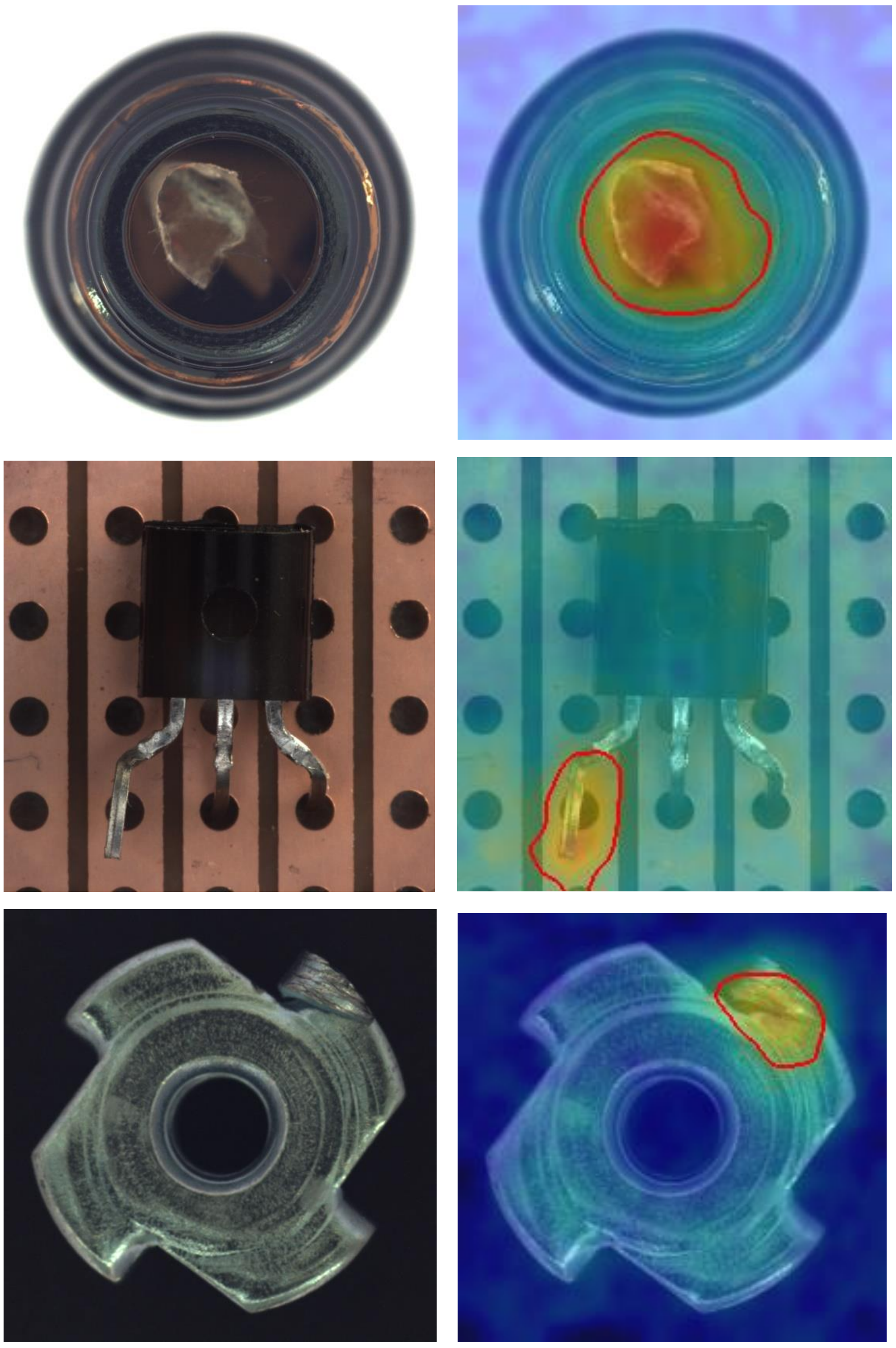


Figure 4.16a: Sample Inferencing Results for the PatchCore Model.

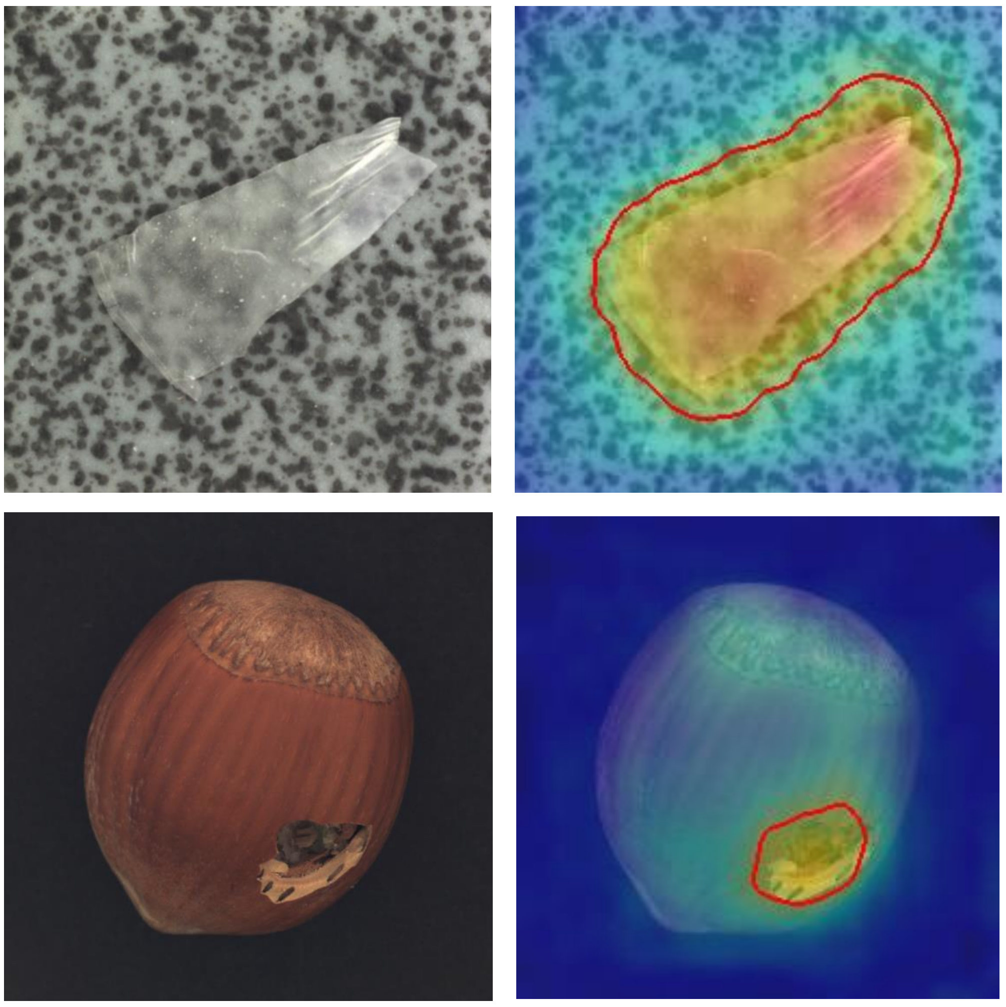


Figure 4.16b: Sample Inferencing Results for the PatchCore Model.

When comparing both of these AI models, it is evident that the PatchCore model is able to segmentalise anomalies with better accuracy, whereas the PaDiM model will actually segmentalise some regions that are not exactly anomalous. Further tests on this were conducted, and the results are shown in Figure 4.17. It was observed that the PatchCore model is able to determine anomalous regions more accurately as compared to the PaDiM model in general. However, there are also instances where the PaDiM model is able to determine anomalous regions with better precision. There are even occurrences, albeit very little, where the PatchCore model was not able to point out the anomalous region within a defective object. There is also a scenario where both models cannot accurately pinpoint the anomalous region, which is shown in Figure 4.18. Nevertheless, the PatchCore model still has a better overall performance than the PaDiM model.

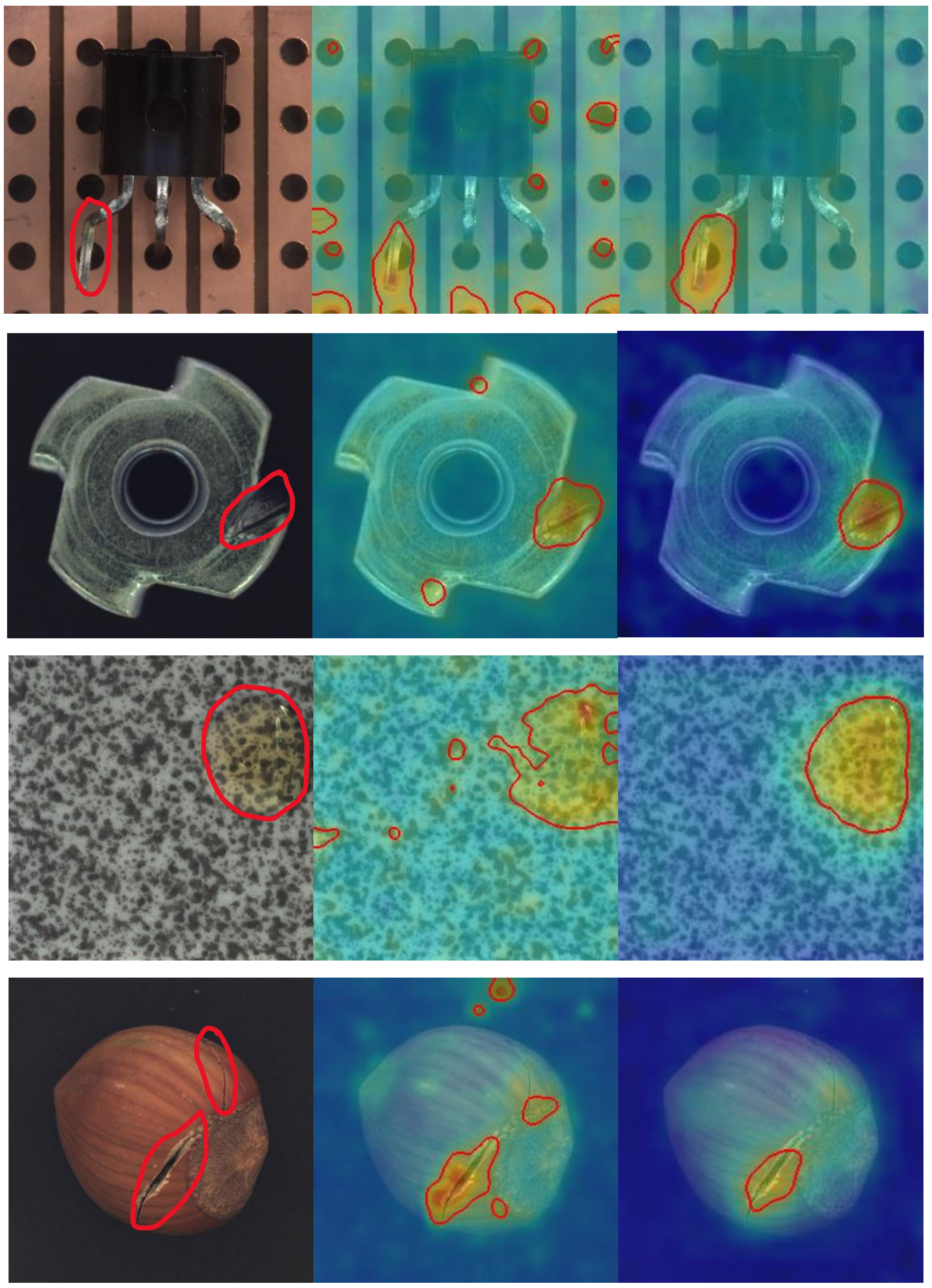


Figure 4.17: Comparison between Test Image (Left), PaDiM Results (Middle) and PatchCore Results (Right).

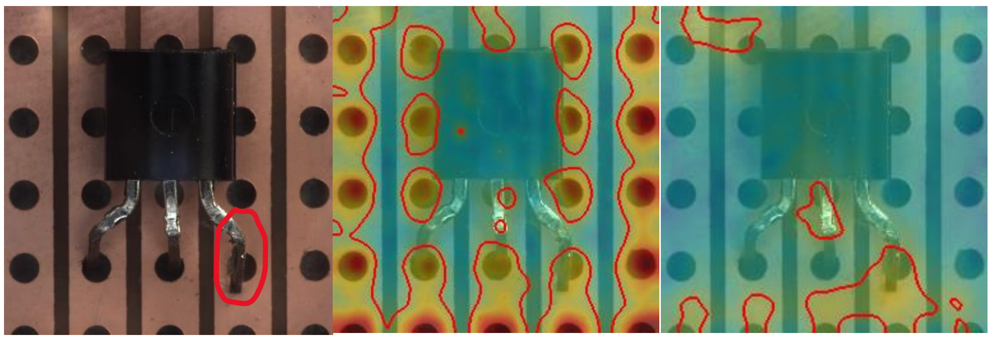


Figure 4.18: PaDiM (Middle) and PatchCore (Right) Not Able to Pinpoint Bent Lead Anomaly on Right Lead (Left).

4.6 Summary

After going through both classical and modern AI methods, it can be said that classical methods are not robust enough to accurately detect anomalies from defective objects, whereas AI-based methods outperform conventional methods in all aspects of this application and is able to pinpoint defects with a high degree of accuracy. Nevertheless, there are still other avenues of detection applications where these conventional methods are able to perform well. When it comes to TM, it can be used to detect whether specific objects are present in an image or not. For example, detecting specific components within a printed circuit board. Since the circuit board for a specific product is identical for each of those products, TM can be used to detect whether the desired components are detected within the circuit board, which is illustrated in Figure 4.19. However, for this to work effectively, the lighting conditions must be consistent for each testing as this will affect the detection accuracy of the algorithm.

For SSIM, it was demonstrated earlier that this algorithm is very useful in highlighting differences between images if the images have a defined structure. An example of an object with defined structure are barcodes, where the positioning of the numbers and lines are fixed. However, the SSIM score alone cannot be used as a metric to determine whether two images have differences or not, as it was shown in the results above that a high SSIM score does not mean that two images are the same. Instead, the algorithm should be paired with thresholding and contouring to visualise the area of defect, which is illustrated in Figure 4.20.

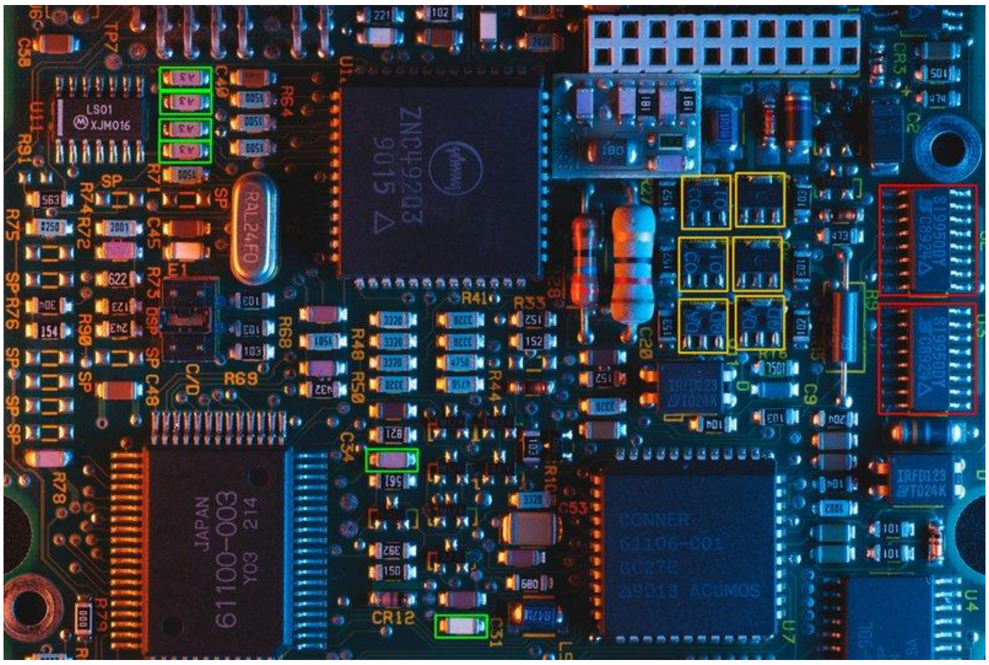


Figure 4.19: Using TM to Detect Components Within a Printed Circuit Board (Rovani, n.d.).

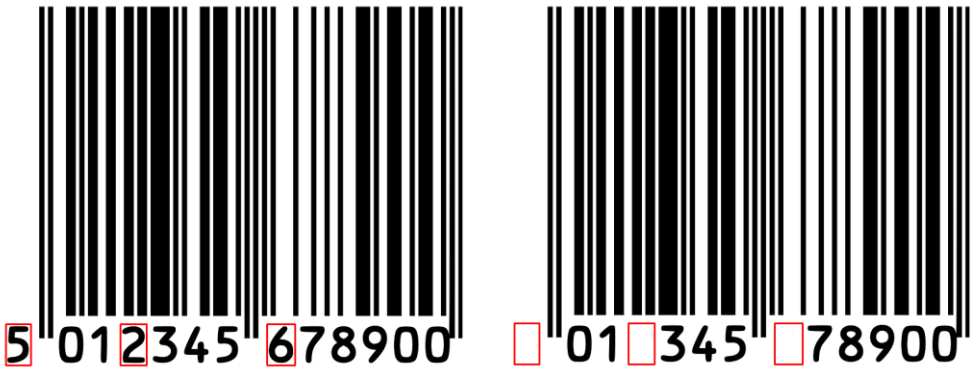


Figure 4.20: Using SSIM to Pinpoint Differences Between Barcodes.

When it comes to feature matching algorithms like SIFT and ORB, these algorithms can be used for object or even human detection applications in situations that require more robust detection as compared to TM or SSIM as these algorithms are not greatly affected by affine transformations. However, the detection object should have a considerable number of visual features that can be used as keypoints for detection. Another usage of feature matching is for panorama stitching. Since these algorithms are able to detect and match keypoints between images, several images can be stitched together based on similar keypoints to form a panoramic image as demonstrated in Figure 4.21.

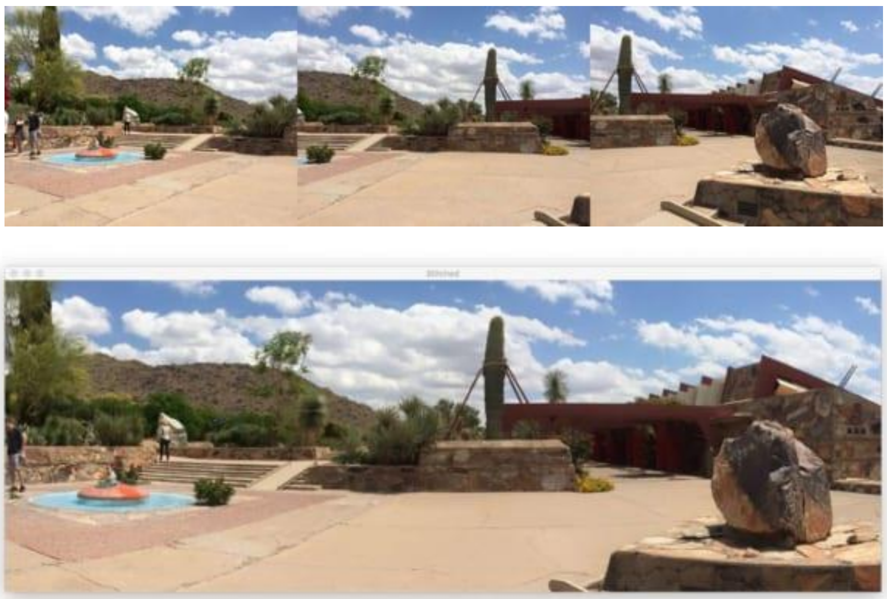


Figure 4.21: Stitching Three Images (Top) to Form a Panoramic Image (Bottom) (Rosebrock, 2018).

As a whole, the findings from this study are summarised in Table 4.7. It can be concluded that only modern AI approaches are suitable for accurate and robust anomaly detection, whereas conventional methods are not capable of doing so. However, there are other use cases for the aforementioned classical methods, which was thoroughly discussed in this section. Therefore, classical methods can be used for the right applications as these approaches are less complex and requires much lesser computational power.

Table 4.7: Summarised Findings from This Study.

Approaches / Algorithms	**Suitable for Anomaly Detection**	**Recommended Use-Cases**
TM	No	Specific Object Detection, Presence/Absence of an Object
SSIM	No	Highlighting Differences between Structured Images
Feature Matching	No	Object/Human Detection, Image Stitching
Modern AI	Yes	Various but model requires dataset to be trained

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

In this study, vision-based anomaly detection was performed using both conventional and modern AI approaches. The obtained result for the classical methods indicates that they were unable to discriminate between normal and defective objects. The average Image AUROC and Image F1 Score for the PaDiM model is 0.9151 and 0.9033, whereas for the PatchCore model the scores are 0.9993 and 0.9979 respectively. The results are in clear favour of modern AI methods being more suitable for this application, whereas classical methods were not able to produce substantial results for visual anomaly detection. However, there are other avenues where it is justifiable to implement classical methods over AI-based methods as classical methods have a better effectiveness to implementation complexity trade-off for other recommended use-cases, which was discussed in the previous chapter. Nevertheless, AI methods should be the de-facto standard for all visual anomaly detection research and application in the future.

5.2 Recommendations for Future Work

Vision-based anomaly detection is usually implemented in production lines in order to segregate the defective items. In a practical scenario, the processing and computational speed of a program should be as fast as possible increase the productivity, but the accuracy needs to be high as well to correctly filter out the defective items. Hence, real-time testing can be performed in the future to test the practical efficiency and accuracy of the AI models. Further improvements can also be made on the AI models by performing hyperparameter tuning to improve the accuracy. In addition to that, more anomaly detection datasets should be explored in order to cover more items that are not included within the MVTec AD dataset. More benchmarking metrics can also be explored in order to evaluate the AI models in a more non-biased and accurate manner.

REFERENCES

Adaptive Vision, n.d. *Template Matching*. [online] Available at: <https://docs.adaptive-vision.com/4.7/studio/machine_vision_guide/TemplateMatching.html> [Accessed 21 April 2022].

Akcay, S., Ameln, D., Vaidya, A., Lakshmanan, B., Ahuja, N. and Genc, U., 2022. *Anomalib: A Deep Learning Library for Anomaly Detection*. [online] Available at: <https://doi.org/10.48550/arXiv.2202.08341> [Accessed 10 August 2022].

Babich, N., 2020. *What Is Computer Vision & How Does it Work?.* An Introduction. [online] Available at: <https://xd.adobe.com/ideas/principles/emerging-technology/what-is-computer-vision-how-does-it-work/> [Accessed 21 April 2022].

Bay, H., Tuytelaars, T. and Van Gool, L., 2006. SURF: Speeded Up Robust Features. *European Conference on Computer Vision 2006*, [e-journal] pp.404-417. https://doi.org/10.1007/11744023_32.

Bergmann, P., Batzner, K., Fauser, M., Sattlegger, D. and Steger, C., 2021. The MVTec Anomaly Detection Dataset: A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection. *International Journal of Computer Vision 2021,* [e-journal] 129, pp. 1038-1059. https://doi.org/10.1007/s11263-020-01400-4.

Defard, T., Setkov, A., Loesch, A. and Audigier, R., 2021. PaDiM: A Patch Distribution Modeling Framework for Anomaly Detection and Localization. *Pattern Recognition. ICPR International Workshops and Challenges 2021,* [e-journal] pp. 475-489. https://doi.org/10.1007/978-3-030-68799-1_35.

Goldstein, M. and Uchida, S., 2016. A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data. *PLoS ONE,* [e-journal] 11(4): e0152173. https://doi.org/10.1371/journal.pone.0152173.

Hashemi, N.S., Aghdam, R.B., Ghiasi, A.S.B., and Fatemi, P., 2016. Template Matching Advances and Applications in Image Analysis. *American Academic Scientific Research Journal for Engineering, Technology, and Sciences*, [e-journal] 26(3), pp. 91-108. https://asrjetsjournal.org/index.php/American_Scientific_Journal/article/view/2378.

IBM, 2020a. *Machine Learning*. [online] Available at: <https://www.ibm.com/cloud/learn/machine-learning> [Accessed 21 April 2022].

IBM, 2020b. *Deep Learning*. [online] Available at: <https://www.ibm.com/cloud/learn/deep-learning> [Accessed 21 April 2022].

IBM, n.d. *What is computer vision?*. [online] Available at: <https://www.ibm.com/my-en/topics/computer-vision> [Accessed 21 April 2022].

Khan, J., 2021. *Everything you need to know about Visual Inspection with AI*. [online] Available at: <https://nanonets.com/blog/ai-visual-inspection/> [Accessed 21 April 2022].

Krasnokutsky, E., 2021. *AI Visual Inspection For Defect Detection*. [online] Available at: <https://mobidev.biz/blog/ai-visual-inspection-deep-learning-computer-vision-defect-detection> [Accessed 21 April 2022].

Levin, G., n.d. *Image Processing and Computer Vision*. [online] Available at: <https://openframeworks.cc/ofBook/chapters/image_processing_computer_vision.html> [Accessed 21 April 2022].

Lowe, D.G., 2004. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision,* [e-journal] 60, pp. 91-110. https://doi.org/10.1023/B:VISI.0000029664.99615.94.

Luces, R.N., 2019. *Template-based versus Feature-based Template Matching*. [online] Available at: <https://medium.datadriveninvestor.com/template-based-versus-feature-based-template-matching-e6e77b2a3b3a> [Accessed 21 April 2022].

NIX United, 2021. *MACHINE LEARNING FOR ANOMALY DETECTION: IN-DEPTH OVERVIEW*. [online] Available at: <https://nix-united.com/blog/machine-learning-for-anomaly-detection-in-depth-overview/> [Accessed 21 April 2022].

OpenCV, n.d.a. *Template Matching*. [online] Available at: <https://docs.opencv.org/4.x/de/da9/tutorial_template_matching.html> [Accessed 21 April 2022].

OpenCV, n.d.b. *ORB (Oriented FAST and Rotated BRIEF).* [online] Available at: <https://docs.opencv.org/4.6.0/d1/d89/tutorial_py_orb.html> [Accessed 10 August 2022].

OpenCV, n.d.c. *Feature Matching.* [online] Available at: <https://docs.opencv.org/4.6.0/dc/dc3/tutorial_py_matcher.html> [Accessed 10 August 2022].

PythonProgramming, n.d. *Template Matching OpenCV Python Tutorial*. [online] Available at: <https://pythonprogramming.net/template-matching-python-opencv-tutorial/> [Accessed 21 April 2022].

Rosebrock, A., 2018. *Image Stitching with OpenCV and Python*. [online] Available at: <https://pyimagesearch.com/2018/12/17/image-stitching-with-opencv-and-python/> [Accessed 10 August 2022].

Roth, K., Pemula, L., Zepeda, J., Schölkopf, B., Brox, T. and Gehler, P., 2022. *Towards Total Recall in Industrial Anomaly Detection*. [online] Available at: <https://doi.org/10.48550/arXiv.2106.08265> [Accessed 10 August 2022].

Rovani, n.d. *End-to-end Object Detection with Template Matching using Python.* [online] Available at: <https://www.sicara.fr/blog-technique/object-detection-template-matching> [Accessed 10 August 2022].

Rublee, E., Rabaud, V., Konolige, K. and Bradski, G., 2011. ORB: An efficient alternative to SIFT or SURF. *2011 Internatiopnal Conference on Computer Vision*, [e-journal] pp. 2564-2571. https://doi.org/10.1109/ICCV.2011.6126544.

Salian, I., 2018. *SuperVize Me: What's the Difference Between Supervised, Unsupervised, Semi-Supervised and Reinforcement Learning?.* [online] Available at: <https://blogs.nvidia.com/blog/2018/08/02/supervised-unsupervised-learning/> [Accessed 21 April 2022].

Tyagi, D., 2019. *Introduction To Feature Detection And Matching*. [online] Available at: <https://medium.com/data-breach/introduction-to-feature-detection-and-matching-65e27179885d> [Accessed 10 August 2022].

UC Berkeley, 2022. *What Is Machine Learning (ML)?.* [online] Available at: <https://ischoolonline.berkeley.edu/blog/what-is-machine-learning/> [Accessed 21 April 2022].

Wang, Z., Bovik, A.C., Sheikh, H.R. and Simoncelli, E.P., 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, [e-journal] 13(4), pp. 600-612. https://doi.org/10.1109/TIP.2003.819861.

Wikipedia, n.d. *Spot the difference*. [online] Available at: <https://en.wikipedia.org/wiki/Spot_the_difference> [Accessed 10 August 2022].

Yang, J., Xu, R., Qi, Z. and Shi, Y., 2021. *Visual Anomaly Detection for Images: A Survey*. [online] Available at: <https://doi.org/10.48550/arXiv.2109.13157> [Accessed 21 April 2022].