MACHINE LEARNING APPROACH FOR AUTOMATED OPTICAL INSPECTION OF ELECTRONIC COMPONENTS

LIM SIEW KEE

A project report submitted in partial fulfilment of the

requirements for the award of the degree of

Bachelor of Engineering (Honours) Electronic Engineering

Faculty of Engineering and Green Technology

Universiti Tunku Abdul Rahman

May 2019

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 : _____

Name : LIM SIEW KEE

ID No. : 14AGB05044

Date : 24th APRIL 2019

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"MACHINE LEARNING APPROACH FOR AUTOMATED OPTICAL INSPECTION OF ELECTRONIC COMPONENTS"** was prepared by **LIM SIEW KEE** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Honours) Electronic Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature : _____

Supervisor : Dr. Teh Peh Chiong

Date : _____

Specially dedicated to
my beloved mother, brother and all my friends

ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor, Dr. Teh Peh Chiong for his invaluable advice, guidance and his enormous patience throughout the development of the research.

In addition, I would also like to express my gratitude to my loving parent, friends and lecturers who had helped and given me encouragement to continue and complete my project.

MACHINE LEARNING APPROACH FOR AUTOMATED OPTICAL INSPECTION OF ELECTRONIC COMPONENTS

ABSTRACT

Rapid development in electronic industry causing a high volume of electronic components being manufactured on each day. Human visual inspection system has been traditionally used in electronic industry for quality control. However, human visual inspection system is affected by inconsistent between different inspectors, limitation of human ability, and is time consuming. (Smith and Adendorff, 1991). Human visual inspection system is insufficient for conducting quality control under such increased capacity. Thus, automated optical inspection system has been invented to replace human visual inspection system. Automated optical inspection system is a system that inspect the surface of the device under testing with the aid of machine vision under sufficient amount of light source. Automated optical inspection system become more crucial in the quality control area due to its inspection speed, inspection consistency and inspection accuracy. However, the technology used by automated optical inspection system keep on improving. This project proposed an automated optical inspection system implementing supervised machine learning algorithm and local features to inspect and classify defectiveness of surface mounted device light emitting diode into two classes: PASS and FAIL. PASS indicates that the product is a non-defective product while FAIL indicates that the product is a defective product. Supervised machine learning algorithm that work the best with the selected features for the inspection of surface mounted device light emitting diode is studied. The performance of the supervised machine learning is determined by the prediction accuracy. The factor that affecting the confidence level of the supervised machine learning algorithm is discussed.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS / ABBREVIATIONS

AOI	automated optical inspection
DUT	device under testing
LR	logistic regression
KNN	K-nearest neighbours
CART	decision tree
RF	random forest
NB	naïve bayes
SVM	support vector machine
TP	true positive
TN	true negative
FP	false positive
FN	false negative
SMD	surface mounted device
LED	light emitting diode
USB	universal serial bus
IDE	integrated development environment

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Background

In recent years, the role of human visual inspection system in quality control has been replaced by automated optical inspection system. AOI system has been widely used in the quality control process due to its inspection speed, accuracy and consistency.

AOI technology is a technology that enables the inspection of electronic assemblies to be done accurately within a short period of time. AOI technology ensures the product leaving the production line is in good condition and the items are built correctly according to the design without manufacturing faults.

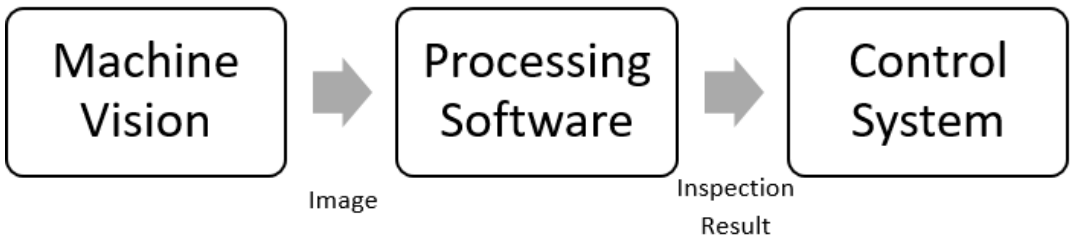

Figure 1.1: Flow of AOI System

All the inspections are done by visually scanning the surface of the DUT with the aid of machine vision. Customised software within the AOI system process the image obtained by the machine vision to determine defectiveness of the DUT. Control

system will then perform corresponding action according to the inspection result determined by the processing software.

There are many inspections methods being used in AOI area. Generally, the inspection methods that are commonly applied by the AOI system to carry out defect inspection can be categorised into three main types: referential method, non-referential method and Hybrid method. (Electronics-notes.com, n.d.)

1.2 Problem Statements

Generally, conventional AOI system implementing referential method, non-referential method and Hybrid method for defect inspection. However, the technology used by AOI system keep on improving. Learning based method has been introduced as one of the inspection method with the development of artificial intelligent and is said to be more intelligent than conventional AOI system.

Learning based method can be separated into machine learning and neural network. Machine learning learns from a set of features that is provided by user while neural network extracts and learns from the set of features extracted by itself. Both machine learning and neural network have high performance. However, neural network is expensive and impractical for the optical inspection of electronic components that do not have much features. High computation power is required for feature extraction part of the neural network. Thus, machine learning that requires low computation power is more preferable.

Machine learning can be further separated into supervised and unsupervised machine learning. Supervised machine learning is used for classification and regression problems while unsupervised machine learning is used for clustering and regression problems.

Supervised machine learning is more suitable to be implemented in AOI system since AOI system involve classification of DUT according to defectiveness.

However, there are many type of supervised machine learning algorithms. Supervised machine learning algorithm that perform the best with the selected feature should be implemented for the automated optical inspection of electronic components.

1.3 Aims and Objectives

The objectives of this thesis are shown below:

i. To build a system that is able to classify images of electronic components according to defectiveness using supervised machine learning algorithm.

ii. To study the best supervised machine learning algorithm to be implemented in real time AOI system for the optical inspection of electronic components according to prediction accuracy.

iii. To build a real time AOI system that is able to inspect and classify electronic components according to defectiveness using supervised machine learning algorithm.

iv. To determine whether the confidence level of classifier model is affected by the number of training data provided.

CHAPTER 2

LITERATURE REVIEW

2.1 Automated Optical Inspection System

2.1.1 Machine Vision

Machine vision of AOI system responsible to capture image of DUT. The image obtained by the machine vision will be passed to the processing software within the AOI system to perform image analysis. Machine vision of AOI system comprises of camera and lighting.

A machine vision may comprise of single or multiple cameras. The usage of single camera enables two dimensional image of DUT to be inspected while the usage of multiple cameras enables three dimensional image of DUT to be inspected. There are two main types of cameras being applied in the AOI system. They are streaming video camera and still image camera. Speed and accuracy is the main concern when selecting the type of camera for the machine vision of AOI system.

i. Streaming video camera

   Streaming video camera captures complete frame of moving DUT. Still image is generated from the captured frame to be analysed by the processing software. Streaming video camera provides a high speed image capturing. However, streaming video camera provides low accuracy image.

ii. Still image camera

Still image camera captures single frame of still DUT. Still image camera provides accurate image that can be directly analysed by the processing software. However, still image camera requires relatively high quality of lighting system to enable a clear image of DUT being captured.

Lighting ensures a clear image of DUT to be obtained by the camera. Care must be taken when selecting the type, position and amount of light source. Certain type of defections is only detectable by certain type of light source. Defection like stain, scratch and missing components is detectable by using front lighting while hole is only detectable by using back lighting. The amount of light source must be sufficient enough to prevent shading of image. Some defection may be blocked by the shading and are not detected. (Electronics-notes.com, n.d.)

2.1.2 Image analysis

AOI system determine the defectiveness of the DUT through image analysis. Generally, there are three main type of image analysis method being applied by the AOI system. They are referential method, non-referential method and Hybrid method.

Referential method is a method that perform comparison between reference image and inspected image by performing subtraction algorithm. This may result in zero image, positive image and negative image. For zero image, no defection is detected. For positive and negative image, defection is detected. Positive image indicates that there are extra features such as scratches exist while negative image indicates that there are missing component. Referential method can be further divided into pattern matching, statistical pattern matching and template matching.

i. Pattern matching

Reference image for both defective and non-defective image is stored by the processing software of the AOI system. AOI system perform comparison between the inspected image and both reference images. Inspected image is considered as defective image when the image matches with the defective

reference image while the inspected image is considered as non-defective image when the image matches with the non-defective reference image.

ii. Statistical pattern matching

Several defective and non-defective image are used as the reference image. Minor defections that will not causing threatening for the DUT is acceptable by using this method.

iii. Template matching

Only one non-defective image is used as the reference image. Defection considered occur whenever the inspected image does not match with the non-defective reference image.

Non-referential method is also known as design rule inspection method. Reference image is not required for this method. The required information is extracted from the inspected image. Design specification standard is used as a guideline to check the extracted information. Defections considered exist when there is measurement that does not follow the design specification standard.

Hybrid method is the combination of referential method and non-referential method. Basically, the inspected image is compared with reference image while checking of measurement according to the design specification standard is being carried out simultaneously. Hybrid method overcome the weaknesses of both referential and non-referential methods. (Taha, Emary and Moustafa, 2014)

2.2 Learning Based Method

Learning based method is a method that learns from features of ground truth data and generate decision rule that performs as a guideline for the classification of new unknown data. (Chang et al., 2017)

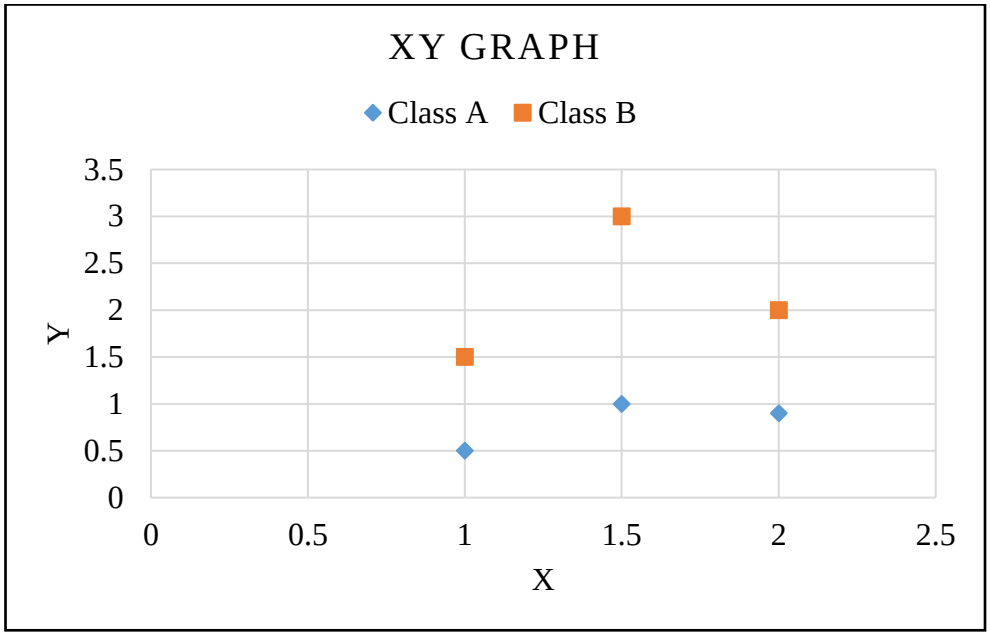

Figure 2.1: XY Graph for Six Samples

Figure 2.1 shows a XY graph for six samples. Three samples from Class A and three samples from Class B. For learning based method, the value of the sample is treated as the feature while the class of the sample is treated as label. Learning based method studies the relationship between the feature and label. Based on Figure 2.1, whenever the y-coordinate for feature of the sample is below 1.5, that sample is belong to Class A. On the other hand, whenever the y-coordinate for the feature of the sample is equal to or above 1.5, that sample is belong to Class B. A decision rule has been generated.

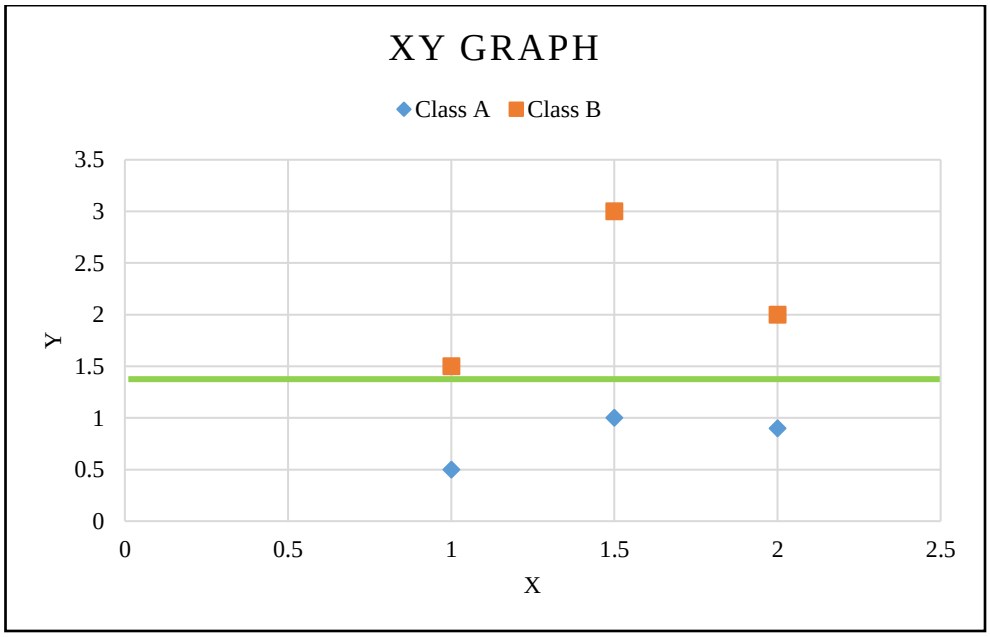

Figure 2.2: XY Graph for Six Samples with A Decision Rule Generated

Based on the Figure 2.2, a decision rule has been generated after learning based method is applied to the six samples. The green line represents the decision rule.

Prediction on new unknown data is made by referring to the decision rule. Whenever the y-coordinate for the value of the new unknown data is below 1.5, the method will automatically classify the unknown data into Class A. However, when the y-coordinate is equal to or above 1.5, the unknown data is classified into Class B.

2.3 Machine Learning

Machine learning is an application of artificial intelligent that enables system to learn from example and improve from experiences rather than through explicit programming. Machine learning algorithms can be divided into supervised machine learning and unsupervised machine learning.

Supervised machine learning is an algorithm that enable system to learn from labelled data. This algorithm is able to make prediction for new input data based on the pattern learnt. Supervised machine learning can be further separated into classification and regression. The output variable for classification is category while the output variable for regression is real value.

Unsupervised machine learning is an algorithm that enable system to learn from unlabelled data without human assistance. This algorithm is able to uncover unknown pattern embedded in unlabelled data. Unsupervised machine learning can be further separated into clustering and association. Clustering discovers subgroups within the dataset. Elements in the same cluster have similar characteristic. Association discovers relationship among the elements within a large database. Normally, association is used to find related items. (Sharma and Kumar, 2017)

Both supervised and unsupervised machine learning will generate a decision rule based on the understanding on the training data. The decision rule will be used by the algorithm as a guidance for them to perform prediction on new unknown data.

2.4 Supervised Machine Learning

Some of the well-known supervised machine learning algorithm are: Logistic Regression, K-Nearest Neighbours, Decision Tree, Random Forest, Naïve Bayes and Support Vector Machine.

Logistic Regression is an algorithm that make prediction for binary output. LR is restricted to a binary class classifier. LR make prediction based on logistic function. LR compare the probability of the event occurring to the probability of the event not occurring. Classification is made according to the class with higher probability.

K-Nearest Neighbours is one of the simplest classification algorithm. KNN is also known as lazy learning algorithm. KNN classifies new input data based on similarity without learning the pattern behind of the training data. KNN is useful when the knowledge level for the classes is low. (F.Y. et al., 2017)

Decision Tree is an algorithm that performs classification by forming decision tree. The final decision tree developed is tree consisting decision nodes, branches and leaf nodes. Decision node represent test on an attribute or feature. Branches represent outcome for the test. Leaf node represent the class label. (Muhammad and Yan, 2015)

Random Forest is an ensemble method that is made up of several individually trained supervised machine learning. Each of the supervised machine learning algorithm make prediction for the new input data. RF merges all the prediction made to make a final decision. RF has a higher predictive power even though the operation of RF is similar to CART. RF does not face with overfitting problem since features to be classified are randomly selected from the total features extracted. (Goel and Abhilasha, 2017)

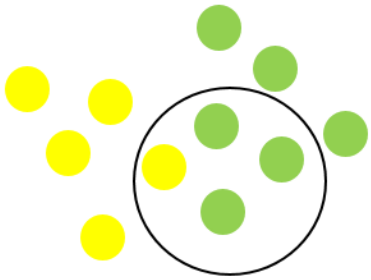

Figure 2.3: Naïve Bayes Example

Naïve Bayes performs classification based on Bayes' theorem. Naïve Bayes classification is performed by comparing the posterior probability of the circled area being class A or class B. Posterior probability is obtained by multiplying the prior probability of a class with the likelihood of the circled area being that class.

$$\text{Likelihood of X being YELLOW} = \frac{\text{number of YELLOW in X}}{\text{total number of YELLOW}} \quad \text{Eq. (2.1)}$$

$$Prior\ probability\ for\ YELLOW = \frac{number\ of\ YELLOW}{total\ number\ of\ objects} \quad \text{Eq. (2.2)}$$

$$Posterior\ probability\ of\ X\ being\ YELLOW = \frac{prior\ probability\ for\ YELLOW}{\text{likelihood of circled area being YELLOW}} \quad \text{Eq. (2.3)}$$

X indicates the circled area. Eq.2.1 to Eq.2.3 is repeated by replacing the YELLOW to GREEN. X will be classified as one of the classes according to the posterior probability. (Electronic Statistics Textbook, 2013) Naïve Bayes can be built easily with no complicated estimation of iterative parameter.

Support Vector Machine is an algorithm that performs classification by finding hyperplane with maximize margin. Hyperplane is a decision line that separate data into classes. The maximization of margin is performed with the help of support vectors. The accuracy level of SVM can be improved by using Kernel trick. (F.Y. et al., 2017)

2.5 Prediction Accuracy

For every prediction made by machine learning algorithm, there will be four possible outcomes: True Positive, True Negative, False Positive and False Negative.

Table 2.1: Cases of TP, TN, FP and FN

Case	Prediction Made	Actual Result	Correctness
TP	PASS	PASS	Correct prediction
TN	FAIL	FAIL	Correct prediction
FP	PASS	FAIL	Wrong prediction
FN	FAIL	PASS	Wrong prediction

For Table 2.1, assuming that only two type of prediction will be made, “PASS and “FAIL”. For TP and TN, the prediction made is the same as the actual result. Thus, a correct prediction has been performed. However, for FP and FN, the prediction made is different from the actual result. Thus, a wrong prediction has been performed.

Prediction accuracy is calculated according to the total number of correct prediction made out of the total number of prediction.

$$Prediction\ Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100\ \%$$

Eq. (2.4)

Higher prediction accuracy indicates that more number of correct prediction has been made. The higher the prediction accuracy, the better the performance of the machine learning algorithm.

2.6 Confidence Level

Confidence level is how likelihood the prediction result to be correct. In the other word, confidence level is how confident is the machine learning algorithm for the prediction made.

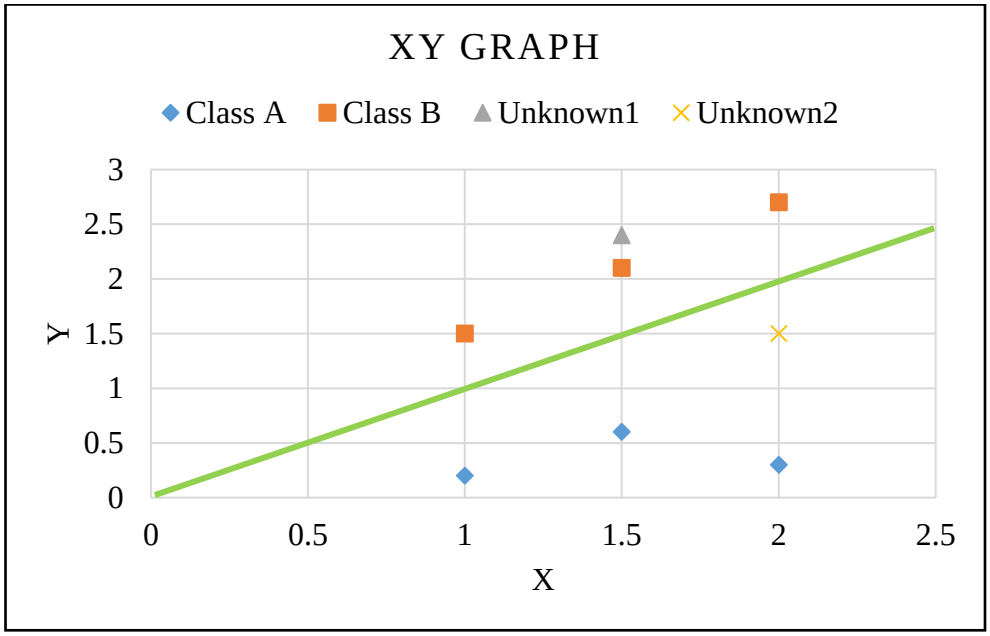

Figure 2.4: XY Graph for Six Samples with A Decision Rule Generated

Based on Figure 2.4, there are two unknown data namely Unkown1 and Unknown2. Unknown 1 is classified into Class B while Unknown2 is classified into Class A according to the decision rule generated by learning based method. The likelihood of the prediction result to be correct is determined by using confidence level.

Confidence level can be obtained by using mathematical calculation. Based on observation, the distance between Unknown 1 with Class B is shorter than its distance with Class A. Assume that the distance between Unknown1 with Class A is 30 units and the distance between Unknown1 with Class B is 10 units. The confidence level is calculated using Eq.2.5 and Eq.2.6. The confidence level of unknown data to be predicted as Class A is:

$$\left[1 - \left(\frac{30}{30 + 10}\right)\right] \times 100\ \% = 25\ \%$$

Eq. (2.5)

The confidence level of unknown data to be predicted as Class B is:

$$\left[1 - \left(\frac{10}{30 + 10}\right)\right] \times 100\ \% = 75\ \%$$

Eq. (2.6)

The distance between Unknown 2 with Class A is shorter than its distance with Class B. Assume that the distance between Unknown1 with Class A is 15 units and the distance between Unknown1 with Class B is 25 units. The confidence level is calculated using Eq.2.7 and Eq.2.8. The confidence level of unknown data to be predicted as Class A is:

$$\left[1 - \left(\frac{15}{25 + 15}\right)\right] \times 100\ \% = 62.5\ \%$$

Eq. (2.7)

The confidence level of unknown data to be predicted as Class B is:

$$\left[1 - \left(\frac{25}{25 + 15}\right)\right] \times 100\ \% = 37.5\ \%$$

Eq. (2.8)

The prediction made by the learning based method for both unknown data is correct. However, the confidence level for the prediction of Unknown1 is higher than Unknown2. The distance between Unknown1 with Class B is shorter than the distance between Unknown2 with Class A. The algorithm is more confident with the prediction made for Unknown1 compared to Unknown2. It can be observed that the confidence level for the prediction made increase when the distance between the unknown data and the decision line is further.

2.7 Journals Review

Muhammad and Yan (2015) have performed a survey to study about the strengths and weaknesses of different supervised machine learning algorithms. Strengths and weaknesses are studied in order to determine the best algorithm to be implemented in classification of data. However, selection of algorithm depends on the nature of task. An algorithm that perform well in one task may perform badly in another task.

Support Vector Machine and Neural Network perform better when the task assigned deals with continuous and multidimensional features. However, Decision Tree performs better when the task involved categorical or discrete features. Support Vector Machine and Neural Network perform better when large amount of samples is provided by the task. However, K-Nearest Neighbours only requires small set of samples.

The selection of algorithm for the classification of data should be done by evaluating the algorithm using prediction accuracy. Prediction accuracy is determined by calculating the number of correct prediction out of total number of prediction.

$$Prediction\ Accuracy = \frac{number\ of\ correct\ classification}{total\ number\ of\ classification\ done} \times 100\ \%$$

Eq. (2.9)

High prediction accuracy indicates that the algorithm performs well and is able to make prediction correctly. (Muhammad and Yan, 2015)

According to the work of Ravikumar, Ramachandran and Sugumaran (2011), they have implemented machine learning approach for automated visual inspection of machine components. Ravikumar, Ramachandran and Sugumaran state that there are a lot of features and classifiers available for the classification of data. However, only the best combination of feature and classifier can yield highest classification accuracy. Thus, they have carried out a study to determine the best machine learning algorithm between Decision Tree and Naïve Bayes to work with histogram feature in order to achieve highest classification accuracy for the inspection of machine components.

Training of classifier is done before best combination of classifier and feature can be determined.

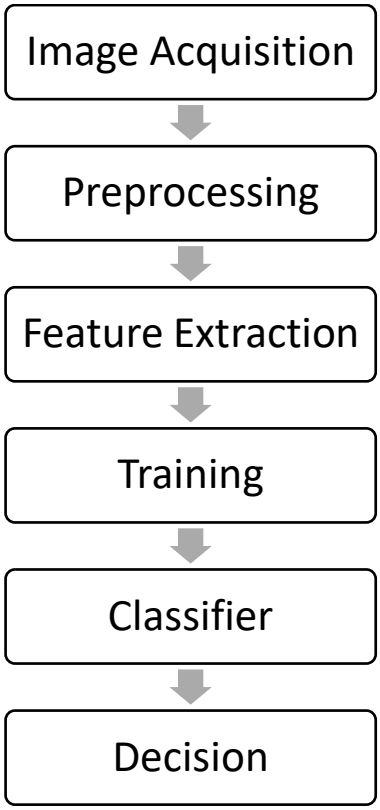

Figure 2.5: Methodology for Development of Visual Inspection System

Raw image of machine component is obtained by using a digital camera under diffused light source. Several type of lighting methods are explored before diffused lighting is chosen. Diffused lighting is chosen since a high resolution image of machine component can be obtained under the lighting of diffused light source. Equal number of samples for each classes must be collected and provided to the machine learning algorithm for training purpose. Three classes of image have been collected: good image, minor scratch image and deep scratch image.

Image pre-processing is performed on the captured image before feature extraction take place. Image is cropped to a fixed size of 500 x 500 and converted into grayscale image. Grayscale image is further converted into binary image using thresholding method. Histogram feature is extracted from the binary image.

Feature extracted for the three hundred images is used to train both Decision Tree and Naïve Bayes algorithms. New image of machine component is provided to the pre-trained classifier model to be classified. Prediction accuracy level for each algorithm is recorded.

Based on the working done, histogram feature work best with Decision Tree algorithm. The prediction accuracy of Decision Tree classifier model is higher than Naïve Bayes classifier model. There is no such thing of best machine learning algorithm. There is only machine learning algorithm that work best with selected features. (Ravikumar, Ramachandran and Sugumaran, 2011)

Chang et al. (2017) has proposed to implement one of the machine learning algorithm which is Support Vector Machine in the optical inspection system to inspect and classify defection of compact camera lens.

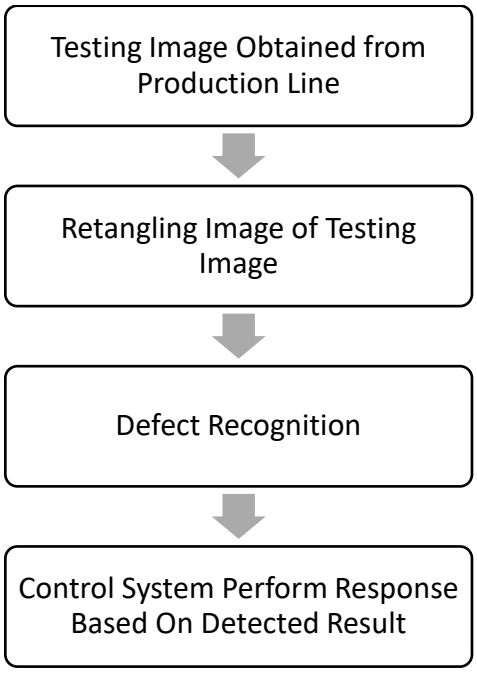

Figure 2.6: Overview of Proposed Inspection System

Figure 2.6 shows the overview of the inspection system proposed by Chang et al. The camera captures the testing image of the product, which is the compact lens while the product moving on the production line. The captured image act as the input

for the defect detection algorithm. The defect detection algorithm involves pre-trained Support Vector Machine classifier model. The algorithm returns the inspection result to the control system after inspection is completed. The returned inspection result determines the further action of the control system.

The inspection system has been successfully developed using the proposed algorithm with a high accuracy level. However, there is some limitations for the system. Visual inspection system deals with images. Low quality image affects the inspection result of the system. Care must be taken so that good quality of training and testing images are taken. Light source, distance of camera from the device under testing and magnification power of camera must be constant throughout the development and application of visual inspection system. (Chang et al., 2017)

CHAPTER 3

METHODOLOGY

3.1 Equipment

Hardware	Software
• USB-500 Camera • Microscope • LED Light Source	• Python • Spyder

Figure 3.1: Equipment List for the Development of Proposed AOI System

Figure 3.1 shows the list of equipment required for the development of proposed AOI system for this project. The equipment part can be divided into hardware and software. Hardware form the image capture system or the machine vision which is essential for data collection. Software form the analysis system for image pre-processing, feature extraction, image classification and decision.

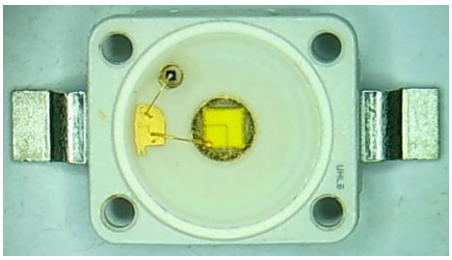

Figure 3.2: Surface Mounted Device Light Emitting Diode

SMD LED is chosen as the DUT for this project. SMD LED is one of the most popular surface mount electronic component that is widely used in electronic industry due to its smaller size and longer lifespan. SMD LED is made up of a base, LED chip and two terminal pads. (Patents.google.com, n.d.)

3.2 Hardware Configuration

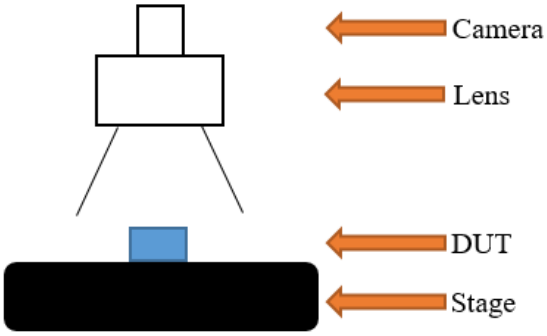

Figure 3.3: Hardware Configuration for Proposed AOI System

Figure 3.3 shows the hardware configuration for the proposed AOI System. It consists of one USB-500 camera and one microscope. The detail image of SMD LED under testing is obtained by the camera with the aid of microscope under sufficient amount of light source.

USB-500 camera is a video streaming camera that provides only basic video capturing function. However, USB-500 camera is used since a video capturing camera is more suitable for automated system when compared to a still image camera. Video capturing camera provides a high speed image capturing function. Since the size of SMD LED is small, microscope helps the USB-500 camera to obtain a detail image of SMD LED under testing. Two dimensional defect inspection is performed. Thus, only one camera is required for the AOI system.

There are two types of built-in light source come together with the microscope: fluorescent lighting and LED lighting. Both types of light sources are placed beyond

the stage of microscope. No back lighting from the bottom of the stage is required since hole is not the interest defect for SMD LED. (Torng, Maung and Fan, 2013). LED lighting is chosen instead of fluorescent lighting due to its position and level of lighting. LED lighting is located exactly above the stage of microscope. The light emitted by LED lighting is brighter than fluorescent lighting. This ensures the SMD LED under testing is well light and no shading is generated.

3.3 Programming Language and Environment

Programming language and programming environment form the software part of the proposed AOI system. Machine learning is selected as the algorithm for the proposed AOI system. Thus, programming language and environment must be chosen according to criteria that will optimise the performance of the selected algorithm.

Python is the preferable language for machine learning. Generally, machine learning deals with a lot of data. Thus, Python is more prefer to be used for machine learning algorithm since its data handling capacity is great. In addition, most of the readily available code libraries for machine learning algorithm is written in Python language.

Spyder is an open source cross-platform IDE. Spyder is designed specifically for scientific programming in Python language. Spyder consists of debugger that enable step-by-step execution. Coding error can be detected and solved easily by executing the code part-by-part.

3.4 Development of Machine Learning Based Classifier Model

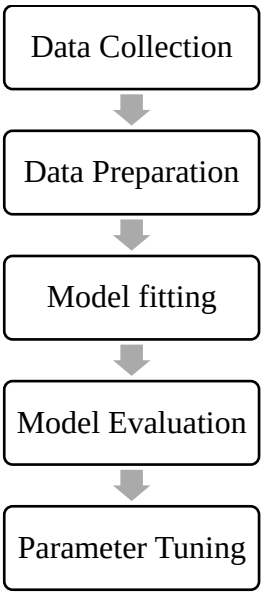

Figure 3.4: Methodology for the Development of Classifier Model using Supervised Machine Learning Algorithm

Figure 3.4 shows the whole process of developing classifier model using supervised machine learning algorithm. Five main steps are involved in the developing process: data collection, data preparation, model fitting, model evaluation and parameter tuning. (Edwards, 2018)

Data collection collect data that is required to train the algorithm into a classifier model. Data collection is performed with the aid of machine vision. Machine vision captures images using camera under sufficient amount of light source.

Data preparation extract information of the images into real value that can be understand by the algorithm. Data preparation is performed using three steps: image pre-processing, feature extraction and labelling. Image pre-processing process images to enhance the signals required by feature extraction process. Feature extraction extract feature of region under interest and convert the feature into feature vector. Proper label is given to the feature extracted according to the class of the image. Data preparation is a process that generate a set of question along with answer to be leant by the algorithm. Feature is the question while label is the answer. Muhammad and Yan

(2015) mentioned that the standard format of the data collected after data preparation should be in the form as shown in Table 3.1.

Table 3.1: Standard Format of Data Collected After Data Preparation

Image	Feature 1	Feature 2	…	Feature n	Label
1	x1	x2	…	n	A
2	x1	x2	…	n	B
3	x1	x2	…	n	C
…	…	…	…	…	…

Model fitting train the algorithm into a classifier model. Supervised machine learning algorithm learns from the labelled data that has been collected and prepared. Supervised machine learning algorithm generate a decision rule based on the relationship discovered between the feature and the label. Decision rule is used for the purpose of classification. Classifier model is developed after training process is completed.

Model evaluation evaluate the performance of the pre-trained classifier model. The performance of the classifier model is evaluated according to the prediction accuracy of the model. Prediction accuracy is determined by calculating number of correct prediction made out of the total number of prediction.

$$Prediction\ Accuracy = \frac{number\ of\ Correct\ Prediction}{total\ Number\ of\ Prediction} \times 100\ \%$$

Eq. (3.1)

High prediction accuracy indicates the classifier model performs well in classification while low prediction accuracy indicates the classifier model performs badly in classification.

Parameter tuning tune the hyperparameter of the supervised machine learning algorithm to optimise the performance of the algorithm. Tuning can be done by using test and trial method or GridSearch method.

1000 non-defective images and 1000 defective images of SMD LED are captured and stored in image database by using machine vision. Defective images are obtained through fault creation process. Three kinds of defect are created: disconnection of inner components, dimensional error and scratch.

Figure 3.5: Sample Image of SMD LED with Disconnection of Inner Components, Dimensional Error and Scratch

Images are resized to a fixed size of 400 x 400 and converted into grayscale image. Features of the region of interest are extracted using colorthresholding, edge detection and binarization techniques. Local features selected for this project is area of the SMD LED, connection between the inner component of SMD LED and texture of the surface of SMD LED. Label is given to the features extracted according to the class of the image. 0 is assigned to defective image while 1 is assigned to non-defective image.

1400 sets of data are separated from the total data collected for the training of the classifier model. However, restriction is applied to the separation process. The training data must contain both class of data in equal ratio. 700 sets of data must be data from defective class and 700 sets of data must be data from non-defective class. This is to prevent the classifier model trained have more knowledge on certain type of class only.

There are a total of six classifier models namely LR, KNN, CART, RF, NB and SVM developed after the training process is completed.

Performance of each classifier model is evaluated by testing the model with the training data. However, only feature of the training data is provided. Each classifier models make prediction for the feature consists in the training data. Label is then compared with the prediction made in order to determine the prediction accuracy. Prediction and comparison process is repeated for 12 times using same set of data. Prediction accuracy for each classifier model is collected and discussed in Chapter 4.

3.5 Project Management

The schedule of the project is shown in the Gantt chart as below:

Table 3.2: Gantt Chart for FYP 1

Week / Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Project selection														
Literature review														
Methodology research														
DUT and hardware selection														
Access USB camera														
Features selection for DUT														

Table 3.3: Gantt chart for FYP 2

Week / Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Obtain image of DUT														
Extract features from images														
Generate data for training data														
Train classifier models														
Classifier models selection														
AOI system implementing supervised machine learning algorithm														

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Prediction Accuracy

Table 4.1: Prediction Accuracy in Decimal Value Obtained by Each Classifier Model for 12 Times of Classification Using Same Set of Testing Data

Classifier Model	LR	KNN	CART	RF	NB	SVM
Trial 1	0.897436	0.982906	0.991453	1	0.982906	0.487179
Trial 2	0.854701	1	0.991453	0.991453	0.982906	0.521368
Trial 3	0.846154	0.982906	0.991453	1	0.965812	0.42735
Trial 4	0.880342	0.991453	1	1	0.948718	0.452991
Trial 5	0.837607	0.965812	1	1	0.931624	0.504274
Trial 6	0.905983	0.974359	0.982906	0.991453	0.940171	0.452991
Trial 7	0.888889	0.974359	0.982906	0.991453	0.965812	0.401709
Trial 8	0.897436	0.957265	0.982906	0.982906	0.965812	0.529915
Trial 9	0.844828	0.974138	0.991379	1	0.956897	0.525862
Trial 10	0.887931	0.982759	1	0.991379	0.982759	0.517241
Trial 11	0.913793	0.991379	0.991379	1	0.931034	0.491379
Trial 12	0.862069	0.991379	0.991379	0.991379	0.956897	0.413793

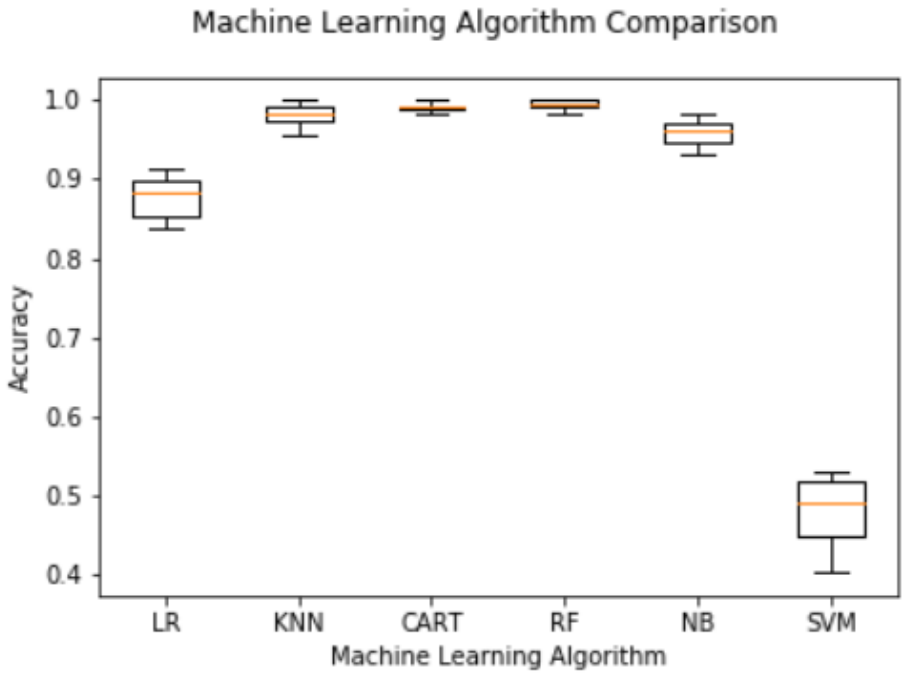

Figure 4.1: Boxplot Graph of Prediction Accuracy for Each Classifier Model Based on the Prediction Accuracy Obtained Through 12 Times of Classification

Table 4.2: Average Prediction Accuracy for Each Classifier Model for 12 Times of Classification

Classifier Model	Average Prediction Accuracy (%)
LR	87.64
KNN	98.07
CART	99.14
RF	99.50
NB	95.92
SVM	47.71

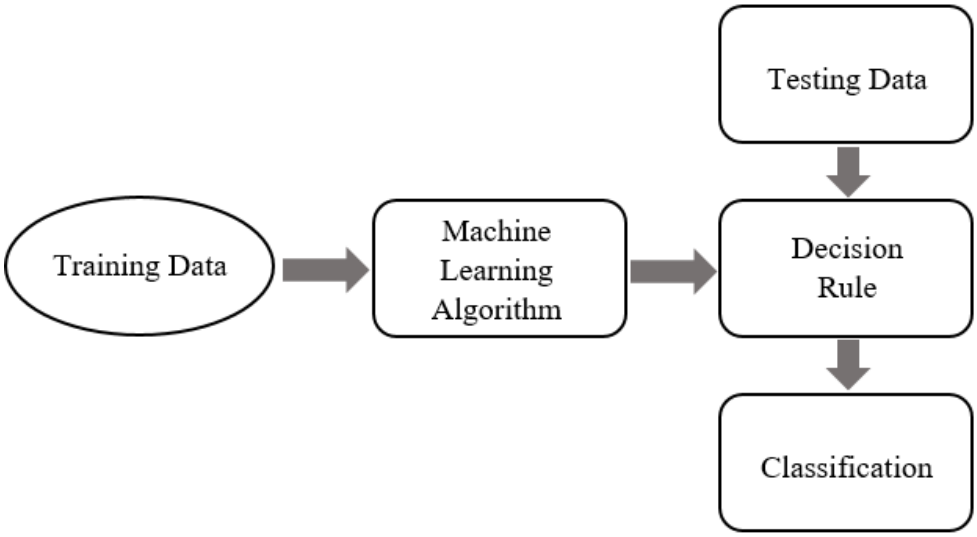

Figure 4.2: Methodology for the Operation of Classifier Model in Performing Classification

Supervised machine learning algorithm learn and understand the relationship between the feature and label of the training data to generate a decision rule. Decision rule is used as a guideline for the classification of new testing data. Prediction accuracy for providing the classifier model with exactly the same data as training data is able to evaluate the understanding of the supervised machine learning algorithm on the relationship between the feature and label of training data. High prediction accuracy indicates that the algorithm work best with the selected local features, has a high understanding on the relationship between feature and label of training data, and has generate a good decision rule based on the understanding.

The evaluation for the understanding of the algorithm is carried out by using cross-validation method. Cross-validation method is a method that provide same set of testing data to the classifier model more than 1 time to make prediction. The average prediction accuracy obtained indicates whether the algorithm performs well with the features selected or whether a good decision rule has been generated by the algorithm for the classification of classes. The range of the prediction accuracy for 12 times indicates whether the prediction made by the classifier model is consistent.

Table 4.1 shows the prediction accuracy of each classifier model for 12 times of classification. Based on observation, the prediction accuracy of SVM classifier

model is the lowest compared to other classifier model. SVM classifier model only able to make 562 to 741 correct predictions out of 1400 predictions made. The prediction accuracy of CART and RF classifier model is the highest. The highest number of wrong prediction made by CART and RF classifier model is 24 predictions.

Based on Table 4.2, the average prediction accuracy of the SVM classifier model is 47.71 %. However, the average prediction accuracy for other classier model is 85 % and above. The prediction accuracy of SVM classifier model is relatively low when compared to other and is not acceptable since the number of correct prediction made is less than half of the total prediction. Low prediction accuracy indicates that the number of wrong prediction made is high. The understanding of SVM algorithm on the relationship between the feature and label of the training data is low. SVM algorithm fails to generate a decision rule that can clearly separate the feature of the two label. In the other word, SVM algorithm does not work best with the selected local feature.

Figure 4.1 is a boxplot graph that is plotted by using the prediction accuracy obtained by each classifier model after performing 12 times of classification using same set of data. Boxplot graph is plotted by using the first quartile, median, third quartile and range of the set of data provided. The size of boxplot represents the range of the set of data provided. Large size boxplot indicates that the range of the set of data is large.

Based on Figure 4.1, it can be observed that the size of boxplot for LR, KNN, NB and SVM is large. Large size boxplot indicates that the range of the prediction accuracy obtained by each classifier models for 12 times is large. Large range prediction accuracy means that the prediction made by the classifier model is not consistent even though same question is being asked to the classifier model. Different answer is given by the classifier model even though same set of question is being asked to them. Prediction that is not consistent affecting the reliability of the prediction made.

SVM classifier model has low prediction accuracy and low consistency. LR, KNN and NB classifier models have high prediction accuracy but low consistency. CART and RF classifier models have high prediction accuracy and high consistency.

CART and RF classifier models have the best performance among the six classifier models.

4.2 Model Selection

CART and RF classifier models have the best performance among the six classifier model. RF is chosen as the supervised machine learning algorithm to be implemented in the real time AOI system instead of CART due to its higher predictive power.

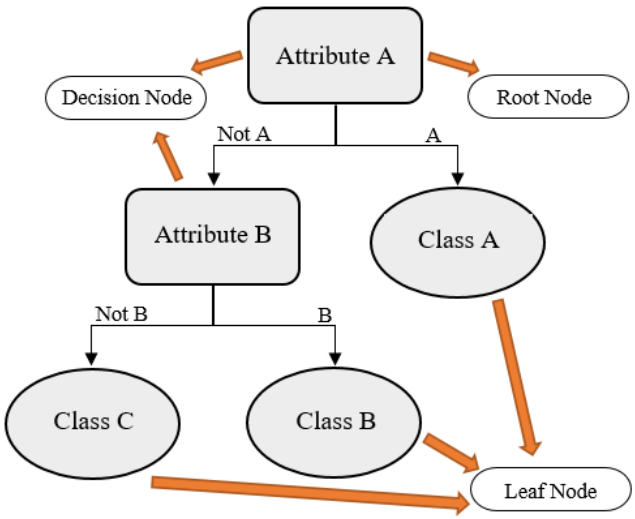

Figure 4.3: Decision Tree Algorithm

Decision Tree algorithm is an algorithm that performs classification by building a decision tree as shown in Figure 4.3. A decision tree consisting decision nodes, branches and leaf nodes is formed during the training process of Decision Tree algorithm. Decision nodes represent test on an attribute or features. Branches represent outcome of the test. Leaf node represent the class of label or decision. Root node is a node that consists best attribute of the dataset. Decision tree start to form from that attribute. Best attribute is attribute that can clearly differentiate two classes.

The way Decision Tree algorithm operate when performing classification is similar to logic of human thinking when making decision. It can be easily understood

and operate without the need of tuning any hyperparamter. However, high probability of overfitting is the main limitation of Decision Tree algorithm.

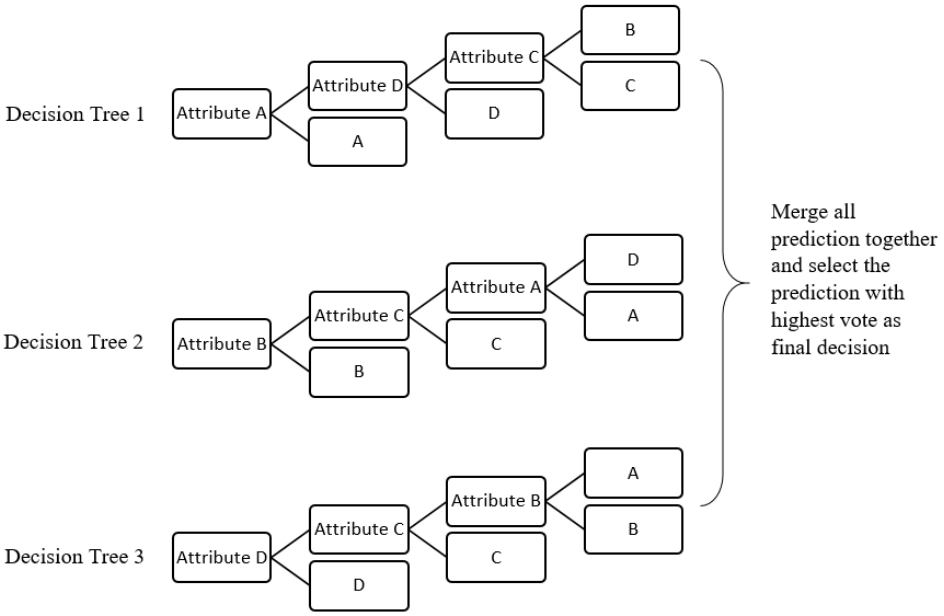

Figure 4.4: Random Forest Algorithm

Random Forest algorithm is an ensemble algorithm that consists of several individually trained decision tree. It performs classification by using similar operation as Decision Tree. However, the root node of Random Forest algorithm is not necessary the best attribute of the dataset provided.

Several decision tree is trained individually by the Random Forest algorithm by using randomly selected attribute inside the dataset. Random Forest performs classification by providing testing data to the decision tree that are individually trained. Each decision tree make prediction on the testing data. Random Forest then make final decision on the classification of testing data by selecting label with the highest occurrence level as the classification result. Overfitting can be avoided since the attribute selected is randomly selected. Random Forest has a higher predictive power since it combines result of several individually trained supervised machine learning algorithm.

4.3 Confidence Level

Confidence level describe how likelihood the prediction made by the classifier model is a correct prediction. High confidence level indicates that the classifier is certainly sure for the prediction made. The confidence level for the prediction made must be high to ensure the classification result is reliable.

Table 4.3: Confidence level for Prediction Made by RF Classifier Model

Test Image	Prediction	Confidence Level with 1400 Training Data (%)	Confidence Level with 2000 Training Data (%)
0	PASS	98	100
1	PASS	100	99
2	PASS	99	100
3	PASS	95	98
4	PASS	99	99
5	PASS	99	99
6	PASS	100	100
7	PASS	100	99
8	PASS	97	99
9	PASS	99	99
10	PASS	99	99
. . .	. . .	. . .	. . .

300 new testing data has been provided to the RF classifier model in order to determine the confidence level of the classifier model for the prediction made when the classifier model is trained with 1400 training data and when the classifier model is trained with 2000 training data. The prediction accuracy for the classifier model is the same when the classifier model is trained with 1400 training data and when the

classifier model is trained with 2000 training data. It achieved a prediction accuracy of 93.33 % for both cases. Only two data are being predicted wrongly.

Table 4.3 shows the confidence level for 10 predictions out of total for 300 predictions made. Based on Table 4.3, it can be observed that for the same image predicted by the RF classifier model, RF classifier model classify the image into same class, which is “PASS”. However, the confidence level for the prediction increased when the number of training data increased from 1400 training data to 2000 training data. RF is more confident to make prediction after it has more knowledge about each classes.

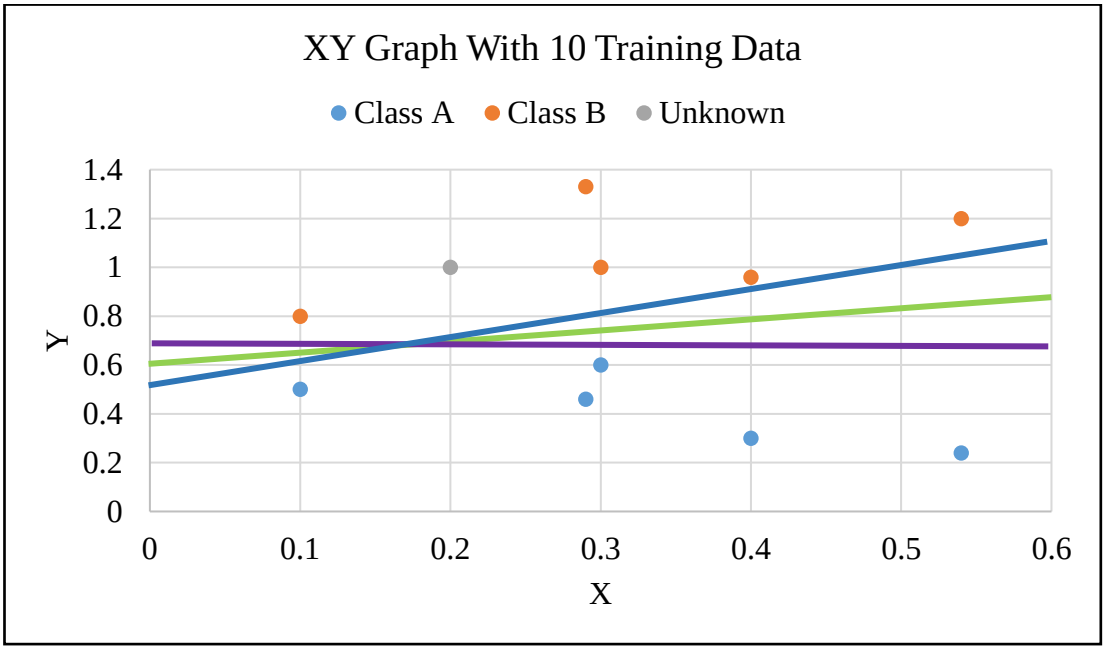

Figure 4.5: XY Graph with 10 Training Data and Three Possible Decision Rule Generated

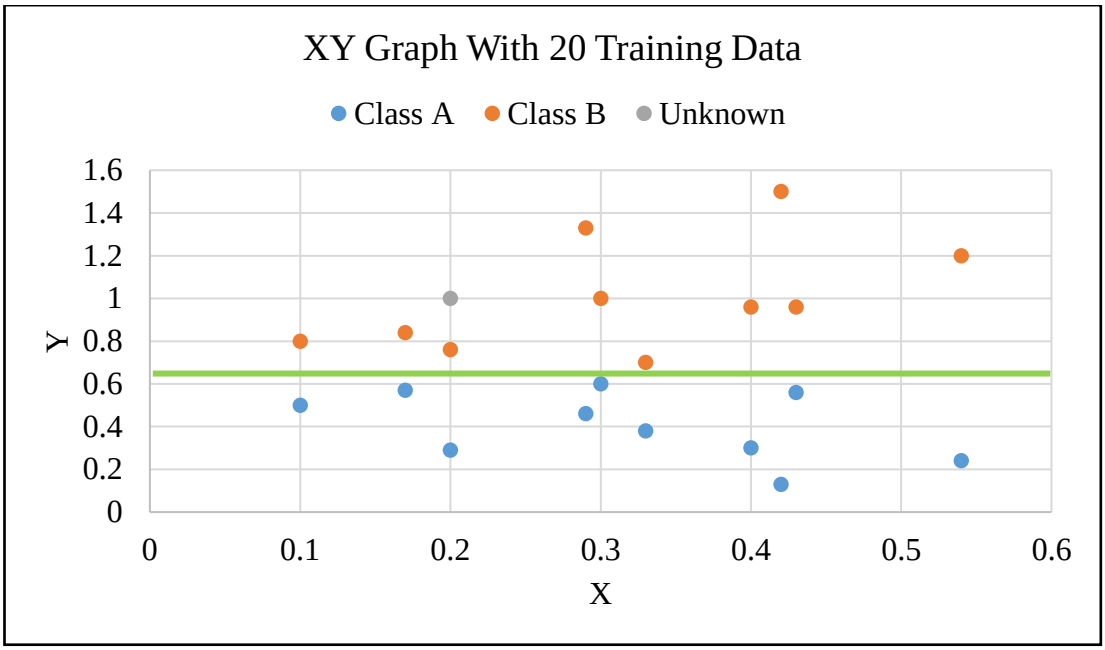

Figure 4.6: XY Graph with 20 Training Data and A Decision Rule Generated

Based on Figure 4.5 and Figure 4.6, it can be observed that the decision rule generated by the machine learning algorithm is more precise when the number of training data provided increased.

Same unknown data is being provided to both classifier model that is trained with 10 and 20 training data respectively. Both classifier model is able to classify the unknown data into Class A based on the decision rule that has been generated. However, according to the concept of confidence level mentioned in Chapter 2 Section 6, the distance between the unknown data and Class A significantly reduce when the number of training data increase. Thus, the confidence level increase. An increase in the confidence level indicates that the classifier model is more confident with the prediction made.

4.4 AOI System Implementing Machine Learning Based Image Classifier

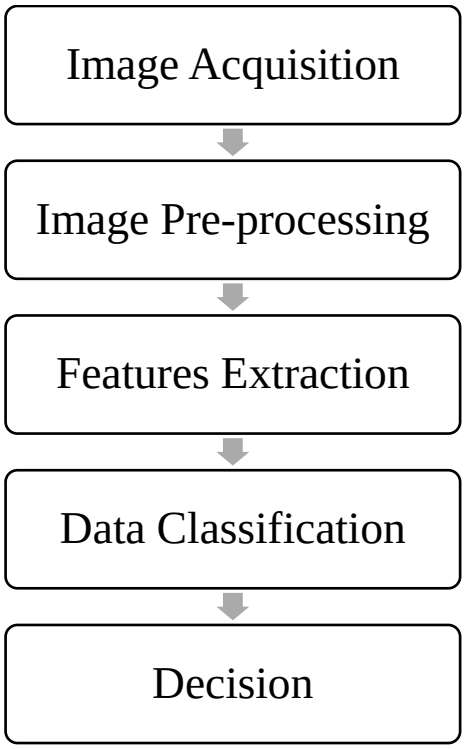

Figure 4.7: Methodology for the Operation of Real Time AOI System

Based on the studied performed, high prediction accuracy indicates that RF algorithm work best with the selected local features to generate a good decision rule that can separate features of two labels clearly. Confidence level for the prediction made increase when the number of training data increase. Thus, RF classifier model that is pre-trained using 2300 training images is implemented in the operation of proposed real time AOI system in the data classification stage as shown in Figure 4.7. 1150 training data is from defective image and 1150 training data is from non-defective image. This is to prevent the algorithm to have more knowledge on certain type of label only. RF classifier model is chosen due to its prediction accuracy, consistency and reliability. The proposed AOI system has been developed successfully and is able to inspect the surface of SMD LED.

Image of SMD LED under the machine vision is obtained and passed to the processing software of the AOI system. The processing software resize the image to a fixed size of 400 x 400 and convert the image into grayscale image. Features are

extracted using the same method as the method performed in training operation. The features extracted is passed to the pre-trained RF classifier model to perform data classification. RF classifier model classify the data extracted into either class 0 or class 1. Class 0 indicates that surface defection is inspected while class 1 indicate that no surface defection is inspected. The software then performs adequate response according to the classification result given by the classifier model. For this project, the software displays the result together with the confidence level. “FAIL” is displayed when class 0 is predicted while “PASS” is displayed when class 1 is predicted.

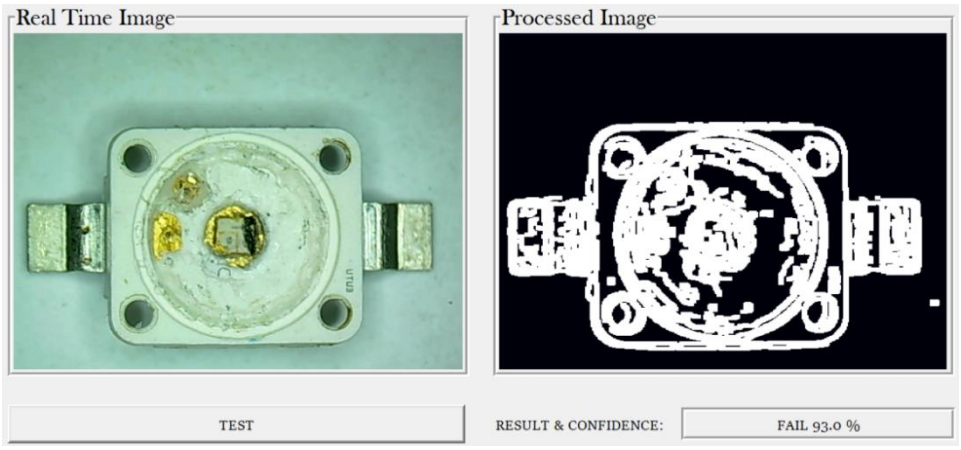

Figure 4.8: GUI for the Real Time AOI System

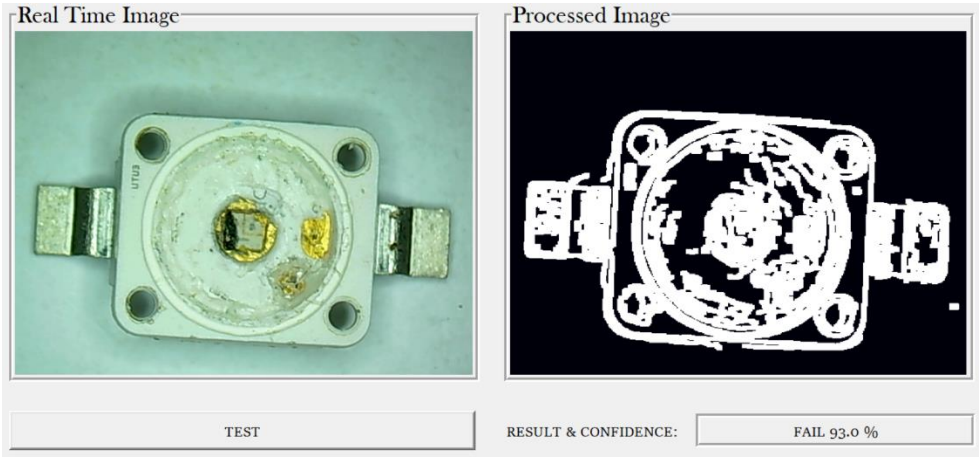

Figure 4.9: GUI for the Real Time AOI System

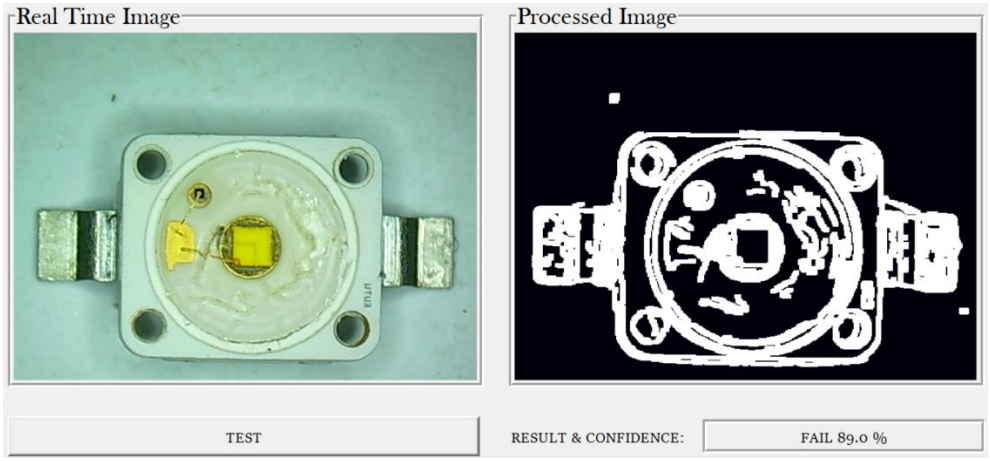

Figure 4.10: GUI for the Real Time AOI System

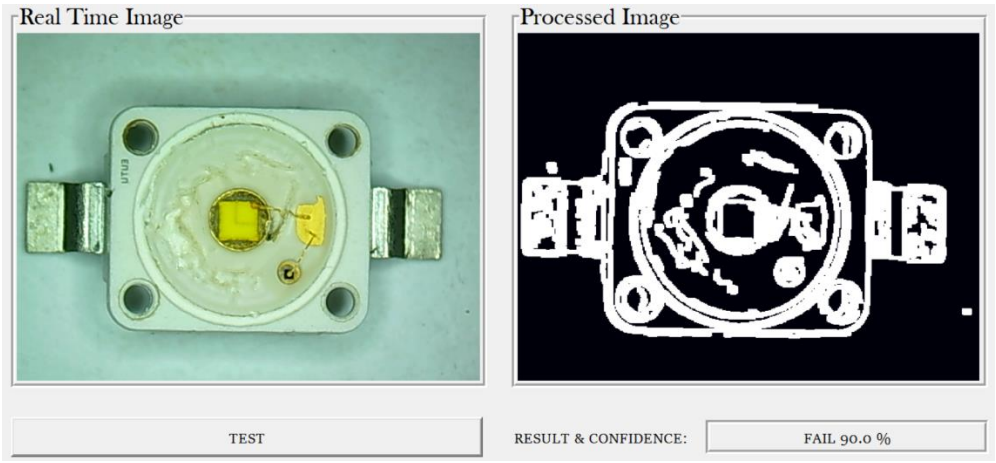

Figure 4.11: GUI for the Real Time AOI System

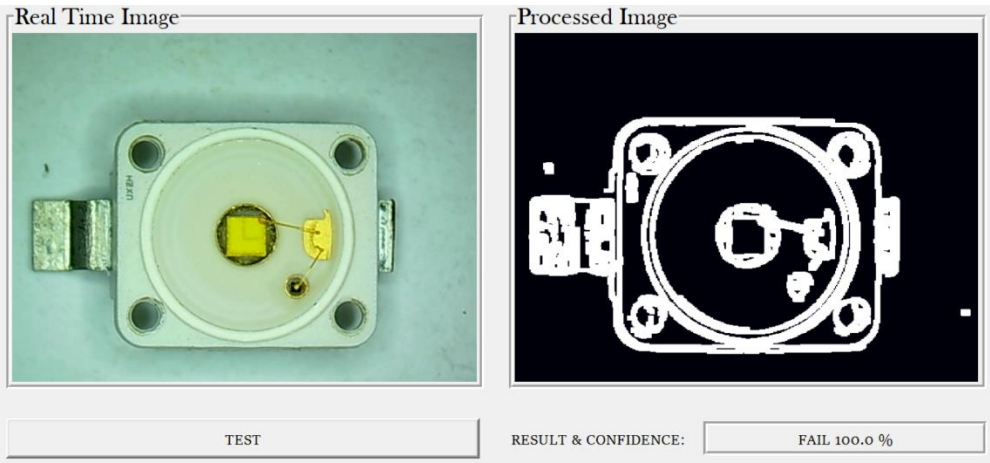

Figure 4.12: GUI for the Real Time AOI System

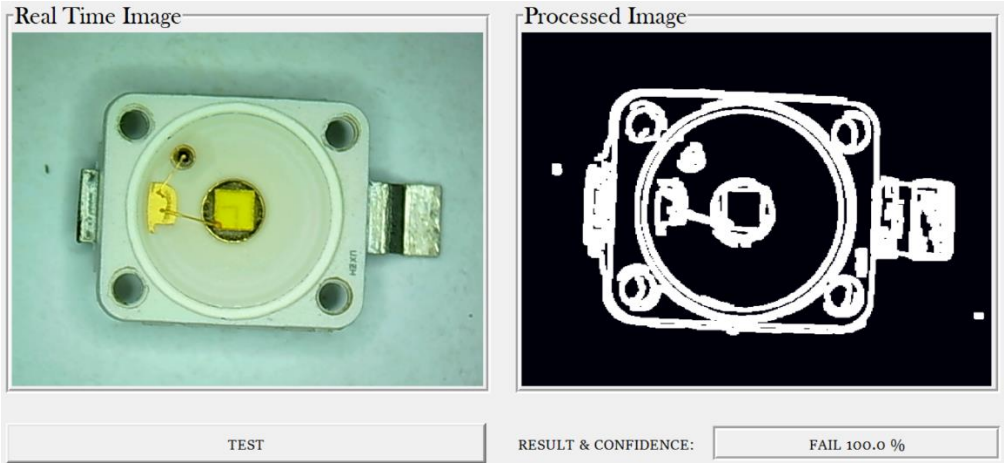

Figure 4.13: GUI for the Real Time AOI System

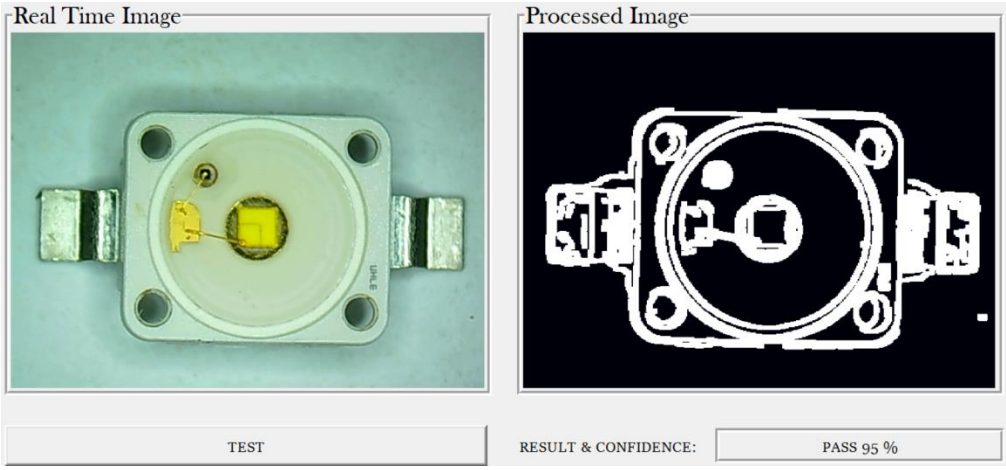

Figure 4.14: GUI for the Real Time AOI System

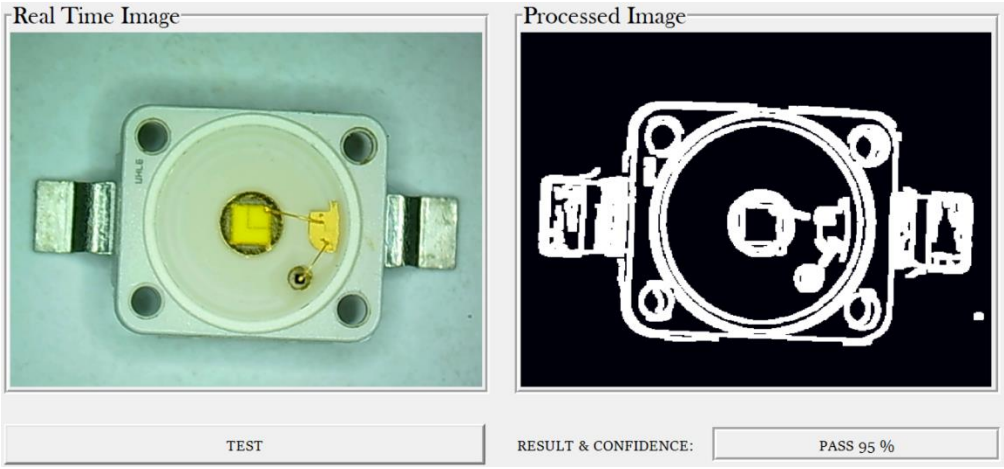

Figure 4.15: GUI for the Real Time AOI System

Figure 4.8 to Figure 4.15 show some example of the real time inspection made. The AOI system is able to inspect and classify the DUT despite of orientation and position of the DUT.

Table 4.4: Confidence Level and Prediction Correctness for Each Real Time Inspection

Trial	Confidence Level for Prediction Made (%)	Prediction Correctness
1	90	Correct
2	88	Correct
3	62	Wrong
4	98	Correct
5	99	Correct
6	100	Correct
7	78	Correct
8	94	Correct
9	98	Correct
10	76	Correct
11	99	Correct
12	100	Correct
13	76	Correct
14	81	Correct
15	84	Correct
16	93	Correct
17	100	Correct
18	77	Correct
19	89	Correct
20	90	Correct

$$Prediction\ Accuracy = \frac{number\ of\ correct\ prediction}{total\ number\ of\ prediction} \times 100\ \%$$

Eq. (4.1)

$$Average\ Confidence = \frac{sum\ of\ confidence\ for\ prediction\ made}{number\ of\ prediction \times 100} \times 100\ \%$$

Eq. (4.2)

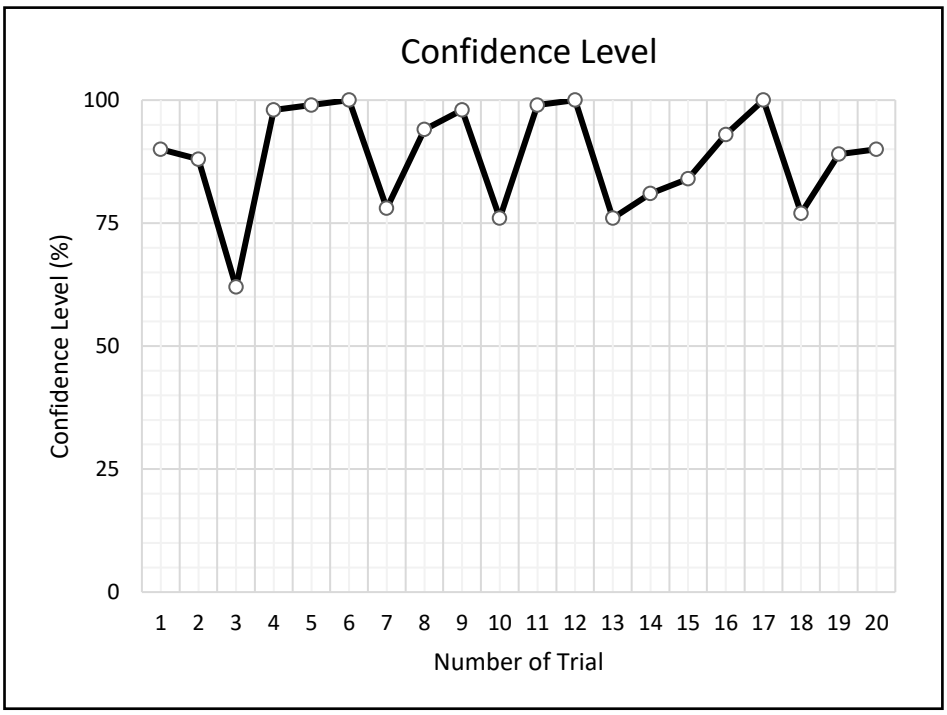

Figure 4.16: Line Graph of the Confidence Level for Each Real Time Testing

The performance of the proposed AOI system is determined by performing 20 times of real time inspection. The proposed AOI system achieved prediction accuracy level of 95 % with average confidence level of 88.6 %. The prediction accuracy level and average confidence level is calculated using Eq.4.1 and Eq.4.2.

4.5 Limitation of Implementing Supervised Machine Learning Approach

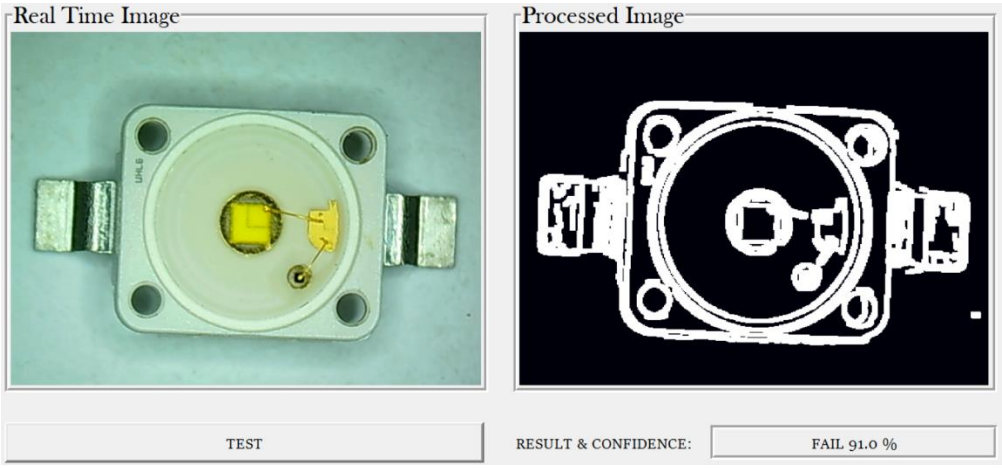

Figure 4.17: GUI for the Real Time AOI System with Correct Prediction Made

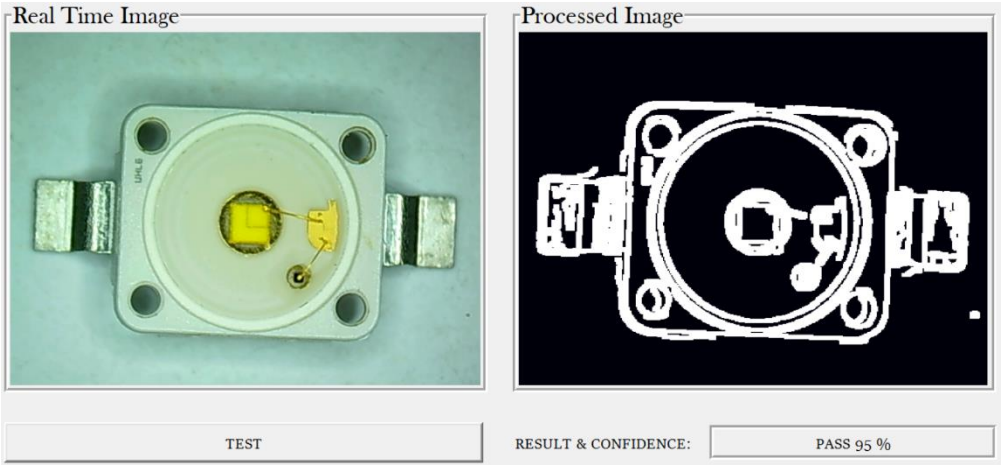

Figure 4.18: GUI for the Real Time AOI System with Wrong Prediction Made

Machine learning algorithm is an algorithm that deals with the pixel value of image. Information are extracted directly from the input image and classification is made based on the decision rule that is generated by the classifier model during the training stage.

Light is the main limitation for this project. The inspection environment for the training and testing process of the classifier model must be exactly the same. Slight different in the inspection environment affecting the classification result. The inspection environment is affected by the distance between the DUT and the camera,

magnification power of microscope, amount of light and position of light. All these four factors must be constant for both training and testing process of the classifier model.

The type of light source used for these project is LED light source. It has a better lighting that can provide high quality image. However, reduction of light output occurs over time. The light is not stable enough. Images obtained under different amount of light source will affect the feature extraction process and causing the classifier model to make wrong prediction.

Figure 4.17 and Figure 4.18 show that the classifier model make different prediction under same amount of LED light source provided. This is because the quality of light degrade over time. A more stable light source should be used.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

Learning based algorithm is an intelligent approach to be used in AOI area since the system is able to generate decision rule without human assistance. The interest of implementing learning based algorithm in the AOI area for the inspection of various type of product increase with the development of artificial intelligent technology.

A system that is able to classify images of electronic components according to defectiveness has been successfully developed by using supervised machine learning. Random Forest algorithm has the highest performance among the six types of supervised machine learning algorithms trained. The performance of supervised machine learning algorithm is evaluated based on prediction accuracy, consistency and reliability. Random Forest has been implemented in the real time operation of proposed AOI system. The AOI system is able to inspect and classify surface defect of SMD LED despite of the position and orientation into two classes: PASS and FAIL. All the objectives of this project is achieved.

The main challenges for this project is the preparation of training data to be learnt by the supervised machine learning algorithm. Local features extracted from each image must be labelled properly to avoid confusion of the algorithm in the learning process. Large set of data must be prepared manually with the number of samples for each classes in equal ratio. Equal number of training data for each classes

must be provided to the algorithm to avoid the classifier model to have more knowledge on certain type of class only. This will affect the decision rule generated by the algorithm for classification of new unknown data.

Further improvement can be made by training the classifier model into multiple class classifier. This can be achieved by sorting images into more classes and provide the classes with proper label. With such improvement, the AOI system will not only just able to classify the DUT into an error product that consist of defection or a good product that does not consist of defection. The system can further identify the type of defection consisted by the product. Care must be taken when training the classifier model into multiple class classifier. Proper label must be assigned to each classes.

Light source is the main concern for the implementation of machine learning approach in AOI system. Further research should be carried out in order to reduce the effect of light on the classification result.

REFERENCES

Smith, C. and Adendorff, K. (1991). Advantages and Limitations of an Automated Visual Inspection System. The South African Journal of Industrial Engineering, 5(1), pp.27-36.

Electronics-notes.com. (n.d.). What is AOI | Automatic Optical Inspection Systems | Electronics Notes. [online] Available at: https://www.electronics-notes.com/articles/test-methods/automatic-automated-test-ate/aoi-optical-inspection.php [Accessed 15 Apr. 2019].

Taha, E., Emary, E. and Moustafa, K. (2014). Automatic Optical Inspection for PCB Manufacturing: a Survey. International Journal of Scientific & Engineering Research, 5(7), pp.1095-1102.

Chang, C., Wu, J., Chen, K. and Hsu, M. (2017). A Hybrid Defect Detection Method for Compact Camera Lens. Advances in Mechanical Engineering, 9(8).

Sharma, D. and Kumar, N. (2017). A Review on Machine Learning Algorithms, Tasks and Applications. International Journal of Advanced Research in Computer Engineering & Technology, 6(10), pp.1548-1552.

F.Y., O., J.E.T., A., O., A., J.O., H., O., O. and J., A. (2017). Supervised Machine Learning Algorithms: Classification and Comparison. International Journal of Computer Trends and Technology, 48(3), pp.128-138.

Muhammad, I. and Yan, Z. (2015). Supervised Machine Learning Approaches: A Survey. ICTACT Journal on Soft Computing, 05(03), pp.946-952.

Goel, E. and Abhilasha, E. (2017). Random Forest: A Review. International Journal of Advanced Research in Computer Science and Software Engineering, 7(1), pp.251-257.

Electronic Statistics Textbook. (2013). [ebook] Tulsa, OK: StatSoft. Available at: http://www.statsoft.com/textbook/naive-bayes-classifier [Accessed 15 Apr. 2019].

Ravikumar, S., Ramachandran, K. and Sugumaran, V. (2011). Machine Learning Approach for Automated Visual Inspection of Machine Components. Expert Systems with Applications, 38(4), pp.3260-3266.

Patents.google.com. (n.d.). US5331512A - Surface-mount LED - Google Patents. [online] Available at: https://patents.google.com/patent/US5331512 [Accessed 10 Apr. 2019].

Torng, J., Maung, K. and Fan, K. (2013). Development of an Automated Optical Inspection System for Mobile Phone Panels. Journal of the Chinese Society of Mechanical Engineers, 34(2), pp.103-108.

Edwards, G. (2018). Machine Learning | An Introduction. [online] Towards Data Science. Available at: https://towardsdatascience.com/machine-learning-an-introduction-23b84d51e6d0 [Accessed 10 Apr. 2019].

APPENDICES

APPENDIX A – Coding

Data Collection

```
import os
import cv2
import time
import tkinter
import PIL.Image
import PIL.ImageTk

##-----------------------------------------------------------
---------------##

import warnings
warnings.filterwarnings('ignore')

##-----------------------------------------------------------
---------------##

CollectData = 'C:\\Users\\USER\\Desktop\\CollectData'
os.chdir(CollectData)

##-----------------------------------------------------------
---------------##

class App:
    def __init__(self, window, window_title, video_source=1):
        self.window = window
        self.window.title(window_title)
        self.video_source = video_source

        self.vid = Video(self.video_source)

        self.frame = tkinter.LabelFrame(window, text="Image",
width=638, height=480, bd=5)
        self.frame.grid(row=0, column=0, columnspan=2, padx=10,
pady=10)

        self.canvas = tkinter.Canvas(self.frame, width=638,
height=480)
        self.canvas.grid()
```

```
        self.btn_capture=tkinter.Button(window, text="Capture
and Save", width=60, command=self.capture, bd=5)
        self.btn_capture.grid(row=1, column=0, padx=30, pady=
10)
        self.btn_capture.config(font=("Bookman Old Style", 15))

        self.delay = 15
        self.update()

        self.window.mainloop()

    def capture(self):
        ret, frame = self.vid.get_frame()

        if ret:
            frame = cv2.resize(frame, (638,480))
            cv2.imwrite("frame-" + time.strftime("%d-%m-%Y-%H-%
M-%S") + ".jpg", cv2.cvtColor(frame, cv2.COLOR_RGB2BGR))

    def update(self):
        ret, frame = self.vid.get_frame()

        if ret:
            self.photo = PIL.ImageTk.PhotoImage(image =
PIL.Image.fromarray(frame))
            self.canvas.create_image(0, 0, image = self.photo,
anchor="nw")

        self.window.after(self.delay, self.update)

##-------------------------------------------------------------
---------------##

class Video:
    def __init__(self, video_source=1):

        self.vid = cv2.VideoCapture(video_source)
```

```
        if not self.vid.isOpened():
            raise ValueError("Unable to open video source",
video_source)

    def get_frame(self):
        if self.vid.isOpened():
            ret, frame = self.vid.read()

            if ret:
                frame = cv2.resize(frame, (638,480))
                return (ret, cv2.cvtColor(frame,
cv2.COLOR_BGR2RGB))

            else:
                return (ret, None)

        else:
            return (ret, None)

    def __del__(self):
        if self.vid.isOpened():
            self.vid.release()

##-----------------------------------------------------------------
---------------##

App(tkinter.Tk(), "Collect Data")
```

Features Selection

```
import os
import cv2
import mahotas
import numpy as np
from skimage import feature
import matplotlib.pyplot as plt
from scipy import ndimage as ndi

##---------------------------------------------------------------
---------------##

path = 'C:\\Users\\USER\\Documents\\FYP_TehTarik
\\FeatureImage'
os.chdir(path)

##---------------------------------------------------------------
---------------##

fileset = os.listdir(path)

for i in range(len(fileset)):
    file = np.array(fileset)
    img = cv2.imread(file[i])
    rgb = cv2.resize(img, (400,400))

    hsv = cv2.cvtColor(rgb, cv2.COLOR_BGR2HSV)

    lower_limit = np.array([10,100,100])
    upper_limit = np.array([45,255,255])
    mask = cv2.inRange(hsv,lower_limit,upper_limit)

    inner_kernel = np.ones((4,4),np.uint8)
    inner_close = cv2.morphologyEx(mask, cv2.MORPH_CLOSE,
inner_kernel)
    inner_dilate =
cv2.dilate(inner_close,inner_kernel,iterations = 1)

    image = inner_dilate.astype('uint8')
    nb_components, output, stats, centroids =
```

```
cv2.connectedComponentsWithStats(image, connectivity=4)
    sizes = stats[:, -1]
    max_label = 1
    max_size = sizes[1]
    for i in range(2, nb_components):
        if sizes[i] > max_size:
            max_label = i
            max_size = sizes[i]
    ROI = np.zeros(output.shape)
    ROI[output == max_label] = 255

    fig = plt.figure()
    plt.imshow(ROI, cmap="gray")
    plt.title('Connectivity of Inner Component')
    plt.show()

    inner_area = cv2.countNonZero(ROI)
    inner_area2 = np.array(inner_area)
    inner_area3 = inner_area2.reshape((1,-1))

    print ("Connectivity of Detected Component:
{}\n".format(inner_area3))

    gray = cv2.cvtColor(rgb, cv2.COLOR_BGR2GRAY)

    filtered_image = ndi.gaussian_filter(gray, 1)
    edge = feature.canny(filtered_image)

    edge = np.array(edge, dtype=np.uint8)

    outer_kernel = np.ones((5,5),np.uint8)
    outer_close = cv2.morphologyEx(edge, cv2.MORPH_CLOSE,
outer_kernel)
    outer_dilate =
cv2.dilate(outer_close,outer_kernel,iterations = 1)

    fig = plt.figure()
    plt.imshow(outer_dilate, cmap="gray")
    plt.title('Dimension of SMD LED')
   plt.show()

   outer_area = cv2.countNonZero(outer_dilate)
   outer_area2 = np.array(outer_area)
   outer_area3 = outer_area2.reshape((1,-1))

   print ("Dimension of Detected Component:
{}\n".format(outer_area3))

   texture = mahotas.features.haralick(gray).mean(axis=0)
   texture2 = texture.reshape((1,-1))

   print ("Texture of Detected Component:
{}\n".format(texture2))
```

Classifier Model Training and Evaluating

```
import os
import cv2
import joblib
import mahotas
import numpy as np
from skimage import feature
import matplotlib.pyplot as plt
from scipy import ndimage as ndi
from sklearn.model_selection import KFold
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression

##-------------------------------------------------------------
---------------##

import warnings
warnings.filterwarnings('ignore')

##-------------------------------------------------------------
---------------##

def feature_extraction(rgb):
    hsv = cv2.cvtColor(rgb, cv2.COLOR_BGR2HSV)

    lower_limit = np.array([10,100,100])
    upper_limit = np.array([45,255,255])
    mask = cv2.inRange(hsv,lower_limit,upper_limit)

    inner_kernel = np.ones((4,4),np.uint8)
    inner_close = cv2.morphologyEx(mask, cv2.MORPH_CLOSE,
inner_kernel)
    inner_dilate =
cv2.dilate(inner_close,inner_kernel,iterations = 1)
```

```
    image = inner_dilate.astype('uint8')
    nb_components, output, stats, centroids =
cv2.connectedComponentsWithStats(image, connectivity=4)
    sizes = stats[:, -1]
    max_label = 1
    max_size = sizes[1]
    for i in range(2, nb_components):
        if sizes[i] > max_size:
            max_label = i
            max_size = sizes[i]
    ROI = np.zeros(output.shape)
    ROI[output == max_label] = 255

    inner_area = cv2.countNonZero(ROI)
    inner_area2 = np.array(inner_area)
    inner_area3 = inner_area2.reshape((1,-1))

    gray = cv2.cvtColor(rgb, cv2.COLOR_BGR2GRAY)
    texture = mahotas.features.haralick(gray).mean(axis=0)
    texture2 = texture.reshape((1,-1))

    filtered_image = ndi.gaussian_filter(gray, 1)
    edge = feature.canny(filtered_image)

    edge = np.array(edge, dtype=np.uint8)

    outer_kernel = np.ones((5,5),np.uint8)
    outer_close = cv2.morphologyEx(edge, cv2.MORPH_CLOSE,
outer_kernel)
    outer_dilate =
cv2.dilate(outer_close,outer_kernel,iterations = 1)

    outer_area = cv2.countNonZero(outer_dilate)
    outer_area2 = np.array(outer_area)
    outer_area3 = outer_area2.reshape((1,-1))

    features_extracted = np.hstack([inner_area3, texture2,
outer_area3])
```

```
    return features_extracted

##---------------------------------------------------------------
---------------##

train_path = 'C:\\Users\\USER\\Documents\\FYP_TehTarik
\\TrainingImage'
os.chdir(train_path)

##---------------------------------------------------------------
---------------##

results = []
data = []
names = []

fileset = os.listdir(train_path)

print ("\n")
print ("Feature Extraction Started...")

for i in range(len(fileset)):
    file = np.array(fileset)
    img = cv2.imread(file[i])
    image = cv2.resize(img, (400,400))

    if i <= 999:
        label = 0
    else:
        label = 1

    labels = np.array(label)
    labels = labels.reshape((1,-1))

    info = feature_extraction(image)
    features = np.array(info)
    features= features.reshape((1,-1))

    labeled_features = np.hstack([labels,features])
```

```
    data.append(labeled_features)

print ("\n")
print ("Feature Extraction Completed...")

data = np.array(data)

data = data.reshape(len(fileset),16)

X = data[:,1:16]
y = data[:,0]

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.30, random_state = 0)

##-------------------------------------------------------------
---------------##

print ("\n")
print ("Training Started...")

models = []
models.append(('LR', LogisticRegression(random_state=0)))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier(random_state=
0)))
models.append(('RF', RandomForestClassifier(n_estimators=100,
random_state=0)))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC(random_state=0)))

##-------------------------------------------------------------
---------------##

for name, model in models:
    kfold = KFold(n_splits=12, random_state=0)
    cv_results = cross_val_score(model, X_train, y_train,
cv=kfold, scoring="accuracy")
    results.append(cv_results)
    names.append(name)

print ("\n")
print ("Training Completed...")

fig = plt.figure()
fig.suptitle('Machine Learning Algorithm Comparison')
ax = fig.add_subplot(111)
graph = plt.boxplot(results)

ax.set_xticklabels(names)
plt.xlabel("Machine Learning Algorithm")
plt.ylabel("Accuracy")
plt.show()
fig.savefig("Machine Learning Algorithms Comparison.png")
```

Confidence Level Evaluation

```
import os
import cv2
import joblib
import mahotas
import numpy as np
from skimage import feature
import matplotlib.pyplot as plt
from scipy import ndimage as ndi
from sklearn.model_selection import train_test_splitfrom
sklearn.ensemble import RandomForestClassifier

##-------------------------------------------------------------------
---------------##

import warnings
warnings.filterwarnings('ignore')

##-------------------------------------------------------------------
---------------##

def feature_extraction(rgb):
    hsv = cv2.cvtColor(rgb, cv2.COLOR_BGR2HSV)

    lower_limit = np.array([10,100,100])
    upper_limit = np.array([45,255,255])
    mask = cv2.inRange(hsv,lower_limit,upper_limit)

    inner_kernel = np.ones((4,4),np.uint8)
    inner_close = cv2.morphologyEx(mask, cv2.MORPH_CLOSE,
inner_kernel)
    inner_dilate =
cv2.dilate(inner_close,inner_kernel,iterations = 1)

    image = inner_dilate.astype('uint8')
    nb_components, output, stats, centroids =
cv2.connectedComponentsWithStats(image, connectivity=4)
    sizes = stats[:, -1]
    max_label = 1
    max_size = sizes[1]
```

```
    for i in range(2, nb_components):
        if sizes[i] > max_size:
            max_label = i
            max_size = sizes[i]
    ROI = np.zeros(output.shape)
    ROI[output == max_label] = 255

    inner_area = cv2.countNonZero(ROI)
    inner_area2 = np.array(inner_area)
    inner_area3 = inner_area2.reshape((1,-1))

    gray = cv2.cvtColor(rgb, cv2.COLOR_BGR2GRAY)
    texture = mahotas.features.haralick(gray).mean(axis=0)
    texture2 = texture.reshape((1,-1))

    filtered_image = ndi.gaussian_filter(gray, 1)
    edge = feature.canny(filtered_image)

    edge = np.array(edge, dtype=np.uint8)

    outer_kernel = np.ones((5,5),np.uint8)
    outer_close = cv2.morphologyEx(edge, cv2.MORPH_CLOSE,
outer_kernel)
    outer_dilate =
cv2.dilate(outer_close,outer_kernel,iterations = 1)

    outer_area = cv2.countNonZero(outer_dilate)
    outer_area2 = np.array(outer_area)
    outer_area3 = outer_area2.reshape((1,-1))

    features_extracted = np.hstack([inner_area3, texture2,
outer_area3])
    return features_extracted

##-----------------------------------------------------------------
---------------##

train_path = 'C:\\Users\\USER\\Documents\\FYP_TehTarik
\\TrainingImage'
```

```
os.chdir(train_path)

##-------------------------------------------------------------
---------------##

results = []
data = []
names = []

fileset = os.listdir(train_path)

print ("\n")
print ("Feature Extraction Started...")

for i in range(len(fileset)):
    file = np.array(fileset)
    img = cv2.imread(file[i])
    image = cv2.resize(img, (400,400))

    if i <= 999:
        label = 0
    else:
        label = 1

    labels = np.array(label)
    labels = labels.reshape((1,-1))

    info = feature_extraction(image)
    features = np.array(info)
    features= features.reshape((1,-1))

    labeled_features = np.hstack([labels,features])

    data.append(labeled_features)

print ("\n")
print ("Feature Extraction Completed...")

data = np.array(data)
```

```
data = data.reshape(len(fileset),16)

X = data[:,1:16]
y = data[:,0]

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.30, random_state = 0)

classifier = RandomForestClassifier(n_estimators = 100,
random_state=0)

model1 = classifier.fit(X_train, y_train)
model2 = classifier.fit(X, y)

filename1 = "RandomForestClassifier_1400.sav"
filename2 = "RandomForestClassifier_2000.sav"

joblib.dump(model1,filename1)
joblib.dump(model2,filename2)

##-----------------------------------------------------------------
---------------##

test_path = 'C:\\Users\\USER\\Documents\\FYP_TehTarik
\\ValidationImage'
os.chdir(test_path)

##-----------------------------------------------------------------
---------------##

print ("\n")
print ("Feature Extraction Started...")

test_data = []

test = os.listdir(test_path)

for i in range(len(test)):
    file = np.array(test)
```

```
    test_img = cv2.imread(file[i])
    test_image = cv2.resize(test_img, (400,400))

    if i <= 149:
        test_label = 0
    else:
        test_label = 1

    test_labels = np.array(test_label)
    test_labels = test_labels.reshape((1,-1))

    test_info = feature_extraction(test_image)
    test_feature = np.array(test_info)
    test_feature= test_feature.reshape((1,-1))

    test_features = np.hstack([test_labels,test_feature])

    test_data.append(test_features)

test_datas = np.array(test_data)
test_datas = test_datas.reshape(len(test),16)

X_test = test_datas[:,1:16]
y_test = test_datas[:,0]

##-------------------------------------------------------------
---------------##

print ("\n")
print ("Feature Extraction Completed...")

file1 = "C:\\Users\\USER\\Documents\\FYP_TehTarik
\\TrainingImage\\RandomForestClassifier_1400.sav"
file2 = "C:\\Users\\USER\\Documents\\FYP_TehTarik
\\TrainingImage\\RandomForestClassifier_2000.sav"

loaded_model1 = joblib.load(file1)
loaded_model2 = joblib.load(file2)


print ("\n")
print ("Prediction Started...")

prediction1 = loaded_model1.predict(X_test)
accuracy1 = (loaded_model1.score(X_test, y_test))*100
confidence1 = loaded_model1.predict_proba(X_test)

prediction2 = loaded_model2.predict(X_test)
accuracy2 = (loaded_model2.score(X_test, y_test))*100
confidence2 = loaded_model2.predict_proba(X_test)

print ("\n")
print ("Prediction Completed...")
```

Random Forest Model

```
import os
import cv2
import joblib
import mahotas
import numpy as np
from skimage import feature
from scipy import ndimage as ndi
from sklearn.ensemble import RandomForestClassifier

##-----------------------------------------------------------------
------------##

import warnings
warnings.filterwarnings('ignore')

##-----------------------------------------------------------------
------------##

def feature_extraction(rgb):
    hsv = cv2.cvtColor(rgb, cv2.COLOR_BGR2HSV)

    lower_limit = np.array([10,100,100])
    upper_limit = np.array([45,255,255])
    mask = cv2.inRange(hsv,lower_limit,upper_limit)

    inner_kernel = np.ones((4,4),np.uint8)
    inner_close = cv2.morphologyEx(mask, cv2.MORPH_CLOSE,
inner_kernel)
    inner_dilate = cv2.dilate(inner_close,inner_kernel,iterations
= 1)

    image = inner_dilate.astype('uint8')
    nb_components, output, stats, centroids =
cv2.connectedComponentsWithStats(image, connectivity=4)
    sizes = stats[:, -1]
    max_label = 1
    max_size = sizes[1]
    for i in range(2, nb_components):
        if sizes[i] > max_size:
```

```
            max_label = i
            max_size = sizes[i]
    ROI = np.zeros(output.shape)
    ROI[output == max_label] = 255

    inner_area = cv2.countNonZero(ROI)
    inner_area2 = np.array(inner_area)
    inner_area3 = inner_area2.reshape((1,-1))

    gray = cv2.cvtColor(rgb, cv2.COLOR_BGR2GRAY)
    texture = mahotas.features.haralick(gray).mean(axis=0)
    texture2 = texture.reshape((1,-1))

    filtered_image = ndi.gaussian_filter(gray, 1)
    edge = feature.canny(filtered_image)

    edge = np.array(edge, dtype=np.uint8)

    outer_kernel = np.ones((5,5),np.uint8)
    outer_close = cv2.morphologyEx(edge, cv2.MORPH_CLOSE,
outer_kernel)
    outer_dilate = cv2.dilate(outer_close,outer_kernel,iterations
= 1)

    outer_area = cv2.countNonZero(outer_dilate)
    outer_area2 = np.array(outer_area)
    outer_area3 = outer_area2.reshape((1,-1))

    features_extracted = np.hstack([inner_area3, texture2,
outer_area3])
    return features_extracted

##-----------------------------------------------------------------
------------##

final_train_path = 'C:\\Users\\USER\\Documents\\FYP_TehTarik
\\2300TrainingImage'
os.chdir(final_train_path)
```

```
##-----------------------------------------------------------------
------------##

results = []
database = []
names = []

fileset = os.listdir(final_train_path)

print ("\n")
print ("Feature Extraction Started...")

for i in range(len(fileset)):
    file = np.array(fileset)
    img = cv2.imread(file[i])
    image = cv2.resize(img, (400,400))

    if i <= 1149:
        label = 0
    else:
        label = 1

    labels = np.array(label)
    labels = labels.reshape((1,-1))

    info = feature_extraction(image)
    features = np.array(info)
    features= features.reshape((1,-1))

    labeled_features = np.hstack([labels,features])

    database.append(labeled_features)

print ("\n")
print ("Feature Extraction Completed...")

database = np.array(database)

database = database.reshape(len(fileset),16)

X2 = database[:,1:16]
y2 = database[:,0]

##-----------------------------------------------------------------
------------##

classifier = RandomForestClassifier(n_estimators = 100,
random_state=0)

print ("\n")
print ("Training Started...")

trained_model = classifier.fit(X2, y2)

filename = "RandomForestClassifier_2300.sav"
joblib.dump(trained_model,filename)
```

Real Time Automated Optical Inspection System Implementing Random Forest Model with Graphical User Interface (Main Program)

```
import tkinter
import cv2
import os
import PIL.Image
import PIL.ImageTk
import time
import mahotas
import joblib
import random
import numpy as np
from skimage import feature
import matplotlib.pyplot as plt
from scipy import ndimage as ndi

##-----------------------------------------------------------------
------------##

import warnings
warnings.filterwarnings('ignore')

##-----------------------------------------------------------------
------------##

Realtime = 'C:\\Users\\USER\\Desktop\\RealTime'
os.chdir(Realtime)

##-----------------------------------------------------------------
------------##

def feature_extraction(rgb):
    hsv = cv2.cvtColor(rgb, cv2.COLOR_BGR2HSV)

    lower_limit = np.array([10,100,100])
    upper_limit = np.array([45,255,255])
    mask = cv2.inRange(hsv,lower_limit,upper_limit)

    inner_kernel = np.ones((4,4),np.uint8)
    inner_close = cv2.morphologyEx(mask, cv2.MORPH_CLOSE,
inner_kernel)
```

```
    inner_dilate = cv2.dilate(inner_close,inner_kernel,iterations
= 1)

    image = inner_dilate.astype('uint8')
    nb_components, output, stats, centroids =
cv2.connectedComponentsWithStats(image, connectivity=4)
    sizes = stats[:, -1]
    max_label = 1
    max_size = sizes[1]
    for i in range(2, nb_components):
        if sizes[i] > max_size:
            max_label = i
            max_size = sizes[i]
    ROI = np.zeros(output.shape)
    ROI[output == max_label] = 255

    inner_area = cv2.countNonZero(ROI)
    inner_area2 = np.array(inner_area)
    inner_area3 = inner_area2.reshape((1,-1))

    gray = cv2.cvtColor(rgb, cv2.COLOR_BGR2GRAY)
    texture = mahotas.features.haralick(gray).mean(axis=0)
    texture2 = texture.reshape((1,-1))

    filtered_image = ndi.gaussian_filter(gray, 1)
    edge = feature.canny(filtered_image)

    edge = np.array(edge, dtype=np.uint8)

    outer_kernel = np.ones((5,5),np.uint8)
    outer_close = cv2.morphologyEx(edge, cv2.MORPH_CLOSE,
outer_kernel)
    outer_dilate = cv2.dilate(outer_close,outer_kernel,iterations
= 1)

    fig = plt.figure()
    fig.set_size_inches(5,5)
    ax = plt.Axes(fig, [0., 0., 1., 1.])
    ax.set_axis_off()
```

```
    fig.add_axes(ax)
    plt.set_cmap('hot')
    ax.imshow(outer_dilate, aspect='equal')
    fig.savefig("frame-" + time.strftime("%d-%m-%Y-%H-%M-%S") +
".png", dpi=300)

    outer_area = cv2.countNonZero(outer_dilate)
    outer_area2 = np.array(outer_area)
    outer_area3 = outer_area2.reshape((1,-1))

    features_extracted = np.hstack([inner_area3, texture2,
outer_area3])
    return features_extracted

##-----------------------------------------------------------------
------------##

class VideoCapture:
    def __init__(self, video_source=1):
        self.vid = cv2.VideoCapture(video_source)
        if not self.vid.isOpened():
            raise ValueError("Unable to open video source",
video_source)

        self.width = self.vid.get(cv2.CAP_PROP_FRAME_WIDTH)
        self.height = self.vid.get(cv2.CAP_PROP_FRAME_HEIGHT)


    def get_frame(self):
        if self.vid.isOpened():
            ret, frame = self.vid.read()
            if ret:
                frame = cv2.resize(frame,(640,480))
                return (ret, cv2.cvtColor(frame,
cv2.COLOR_BGR2RGB))
            else:
                return (ret, None)
        else:
            return (ret, None)
```

```
    def __del__(self):
        if self.vid.isOpened():
            self.vid.release()
        self.window.mainloop()

##-----------------------------------------------------------------
-----------##

class App:
    def __init__(self, window, window_title, video_source=1):

        self.window = window
        self.window.title(window_title)
        self.video_source = video_source

        self.vid = VideoCapture(video_source)

        self.frame1 = tkinter.LabelFrame(window, text="Real Time
Image", width=640, height=480, bd=6)
        self.frame2 = tkinter.LabelFrame(window, text="Processed
Image", width=640, height=480, bd=6)
        self.frame3 = tkinter.LabelFrame(window, width=638, bd=6)

        self.frame1.grid(row=0, column=0, columnspan=2,
sticky="W", padx=90, pady=30)
        self.frame1.config(font=("Baskerville Old Face", 25))
        self.frame2.grid(row=0, column=1, columnspan=2,
sticky="W", padx=780)
        self.frame2.config(font=("Baskerville Old Face", 25))
        self.frame3.grid(row=1, column=1, columnspan=2,
sticky="W", padx=1047)

        self.canvas1 = tkinter.Canvas(self.frame1, width = 640,
height = 480)
        self.canvas1.grid()

        self.canvas2 = tkinter.Canvas(self.frame2, width = 640,
height = 480)
        self.canvas2.grid()
```

```
        self.btn_test = tkinter.Button(window, text="TEST",
width=53, bd=4, command=self.test)
        self.btn_test.grid(row=1, column=0, columnspan=2,
sticky="W", padx=90, pady=10)
        self.btn_test.config(font=("Sitka Text", 15))

        self.label = tkinter.Label(window, text="RESULT &
CONFIDENCE:")
        self.label.grid(row=1, column=1, columnspan=2,
sticky="W", padx=780, pady=10)
        self.label.config(font=("Sitka Text", 15))

        self.predict = None

        self.delay = 15
        self.update()

        self.window.mainloop()

    def update(self):
        ret, frame = self.vid.get_frame()

        if ret:
            self.photo = PIL.ImageTk.PhotoImage(image =
PIL.Image.fromarray(frame))
            self.canvas1.create_image(0, 0, image = self.photo,
anchor="nw")


        self.window.after(self.delay, self.update)

    def test(self):
        ret, frame = self.vid.get_frame()

        if ret:
             cv2.imwrite("frame-" + time.strftime("%d-%m-%Y-%H-%
M-%S") + ".png", cv2.cvtColor(frame, cv2.COLOR_RGB2BGR))

        data = []
```

```
        image_files = os.listdir(Realtime)
        latest = max(image_files, key=os.path.getctime)

        test_image = cv2.imread(latest)
        test_image = cv2.resize(test_image, (400,400))

        info = feature_extraction(test_image)
        features = np.array(info)
        features= features.reshape((1,-1))

        data.append(features)

        data = np.array(data)

        data = data.reshape(1,15)

        filename = "C:\\Users\\USER\\Documents\\FYP_TehTarik
\\2300TrainingImage\\RandomForestClassifier_2300.sav"
        loaded_model = joblib.load(filename)

        level = random.randint(90, 100)
        rand = "{} %".format(level)

        prediction = loaded_model.predict(data)

        confi_level =
(np.amax(loaded_model.predict_proba(data)))*100
        confi_percent = "{} %".format(confi_level)

        if (prediction == 0):
            if (confi_level>70):
                result = "FAIL {}".format(confi_percent)
            else:
                result = "PASS {}".format(rand)
        else:
            result = "PASS {}".format(rand)

        if self.predict:
            self.predict.destroy()
```

```
        self.predict= tkinter.Label(self.frame3, text=result,
width=31)
        self.predict.grid(row=1, column=1, columnspan=2,
sticky="W")
        self.predict.config(font=("Sitka Text", 15))

        processed_files = os.listdir(Realtime)
        latest_processed = max(processed_files,
key=os.path.getctime)

        dilate = cv2.imread(latest_processed)
        dilate = cv2.resize(dilate, (638,480))

        self.photo3 = PIL.ImageTk.PhotoImage(image =
PIL.Image.fromarray(dilate))
        self.canvas2.create_image(0, 0, image = self.photo3,
anchor="nw")

##-----------------------------------------------------------------
------------##

App(tkinter.Tk(), "Optical Inspection System")
```