Identification of sticker printing defects in glove manufacturing process using Computer Vision techniques

A dissertation submitted for the Degree of Master of Computer Science

W. M. C. Wickramathunga

University of Colombo School of Computing

2021

DECLARATION

I hereby declare that the thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis. This thesis has also not been submitted for any degree in any university previously.

Student Name: W M C Wickramathunga

Registration Number: 2018/MCS/099

Index Number: 18440997

___ _____29/11/2021_____

Signature of the Student & Date

This is to certify that this thesis is based on the work of Mr. /Ms. _____ under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Certified by,

Supervisor Name: MGNAS Fernando

_____

Signature of the Supervisor & Date

I would like to dedicate this thesis to my parents and my beloved wife for their heartily support.

ACKNOWLEDGEMENTS

At the outset, I wish to express my sincere gratitude to my supervisor Dr. M. G. N, A. S. Fernando, senior lecturer of the University of Colombo School of Computing-UCSC, for his unceasing guidance, constant supervision throughout my research.

Also, I would like to convey my utmost gratitude to all the other academic members of the University of Colombo School of Computing-UCSC for the knowledge they passed throughout the course.

Finally, I would like to thank my family for their valuable support and encouragement given endlessly to make this research successful.

ABSTRACT

This research aims to provide an automatic and real-time defect detection framework for the glove manufacturing industry using computer vision techniques. This research concerns detecting defects on specific sticker printed on the glove at the end of the production. The whole approach is divided into two operational modes: Teaching mode and Inspection mode. The teaching mode contains time complex tasks that can be performed before the actual inspection. The inspection mode does the actual inspection to find the defects.

An image of a printed sticker will be processed in inspection mode using three levels to identify defects. Lower levels contain naïve computer vision algorithms and detect high-degree errors only, whereas higher levels contain complex algorithms that could detect more sophisticated errors. It is an efficient technique to identify defects in the early stages of the defect inspection process.

The significance of sticker's content to its domain will be calculated for every object in the sticker by combining the visibility and domain importance of that specific content. The visibility of content is measured using size and density. A decision function is proposed to decide whether to accept or reject the glove by considering the calculated error and the significance. Finally, a quality measurement model is proposed to calculate the printed sticker's quality for each accepting glove.

The visibility calculation model proved to be valid and consistent with perceptual visibility. The significance calculation model also provides reliable and consistent results according to testing. The defect inspection process is also efficient and performs as expected. However, inspection level-3 provide inconsistent result in some situations, and that algorithm needs to be improved.

Keywords: Computer Vision, Defect Detection, Object Visibility, Object Significance, Quality Measurement.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

BLOB	Binary Large Object
BP net	Backpropagation Network
CV	Computer Vision
FSIM	Feature Similarity Measure
GPU	Graphics Processing Unit
KSM	Kurtosis and Skewness Measure
MISM	Moment Invariant based Similarity Measure
PNG	Portable Network Graphics
QMI	Quality Measurement Index
ROI	Region of Interest
SSIM	Structural Similarity Measurement
SIFT	Scale Invariant Feature Transform
ZM	Zernike Moments

CHAPTER 1

INTRODUCTION

In any manufacturing process, it is difficult to find a production line with zero rejections due to the defects identified in the product. Usually, these defects are identified at the quality assurance stage using manual inspection by workers. The manual inspection process involves many issues, such as missed identifications or false rejections, and it requires many man-hours.

This research focuses on a specific production stage of the glove manufacturing process. There is a sticker printing process at the end of the glove production. This sticker contains several contents that hold vital information specific to the glove, such as branding information, model, size, and the world standard glove parameters. They use popular printing technologies to print this sticker on the glove; Heat Transfer Printing, Pad Printing. Because of the nature of the printing method and the material, it is inevitable to have several defects on the printed sticker. This research aims to provide a solution to identify defects on the sticker by concerning each content's importance.

Identifying sticker printing defects before shipping the product is crucial because discovering a considerable amount of such defects in a shipment by the customer may result in the whole shipment being rejected. The quality assurance division of the company is highly concerned about identifying defects in every glove.

However, it is impossible to print the sticker precisely; hence, each printed sticker may have slight variations. Such variations are not affecting to the readability or the identification of the content. Due to the high production cost, impossible to reject every glove that has printing variations. Therefore some reliable method is required to assess the actual impact of the printing error on the glove before rejection.

The current way of identifying the defects is solely a manual method where an employee checks each glove to find defects. There may be a few dozens/hundreds of gloves printed per hour in a single production line. Therefore, it is challenging to find some defects by looking through the naked eye, and there could be a tendency to miss some defects. In addition to the inspection's labor cost, it is impossible to provide a quantitative measure that reveals the printed sticker's quality. The proposed solution is an automatic defect detection mechanism that is more accurate and efficient than manual inspection. It will be able to mitigate human

errors as well as reduce labor costs. In addition, a quantitative measurement can be provided at the end of the inspection.

The primary computer science area used in this research is Computer Vision. The sticker images captured from the camera will be processed to identify defects by applying several computer vision techniques. Then decide whether to reject or accept the glove after evaluating the error caused by the defect. While evaluating the error, the actual impact of the defect on the glove will be considered. For each accepting glove, define a quality measurement by concerning the errors present in the sticker.

Even if this research is built around the glove sticker printing area, the technique proposed here can be easily incorporated into defect detection in other printing-like domains containing the 2D arbitrary shapes that comply with the constraints in the proposed solution.

1.1 Motivation

It would be beneficial for any manufacturing process with a similar printing task to have an automatic defect detection system to eliminate human error and reduce false rejections in quality inspection. Any manufacturing industry demands real-time solutions to reduce bottlenecks at the production line due to manual inspection slowness at a high production rate. A reliable method is required to assess the actual impact of a defect on the glove because it is impossible to reject every glove with minor variations in printed sticker due to the high production cost.

1.2 Statement of the problem

As mentioned in the introduction, this research mainly concerns the specific sticker containing several objects or content that holds valuable information relevant to the glove. Usually, the glove material will be knitted cotton or rubber. For printing this sticker, they use two popular printing methods called Heat Transfer Printing and Pad Printing. *Figure 1.1* shows two examples of a printed sticker. Each unique sticker has artwork that contains all the contents which must be printed on the sticker. The artwork can be considered as the precise and error-free version of the sticker, *Figure 1.2*. The artwork usually is in the black foreground and white background.

Figure 1.1: Printed sticker on a glove

(A) (B) (C)

Figure 1.2: Artwork Images

Due to the nature of the printing method and surface material, several defects are possible. In heat transfer printing, it is possible to have missing contents due to the uneven heat spread. Unwanted contents can also be found after print if the sticker contains such garbage. In the pad printing process, ink spillings, blurry content are possible. It is more frequent to have displacement and rotation defects, and minor scaling defects could also be possible. Several such defects are presented in *Figure 1.3*.

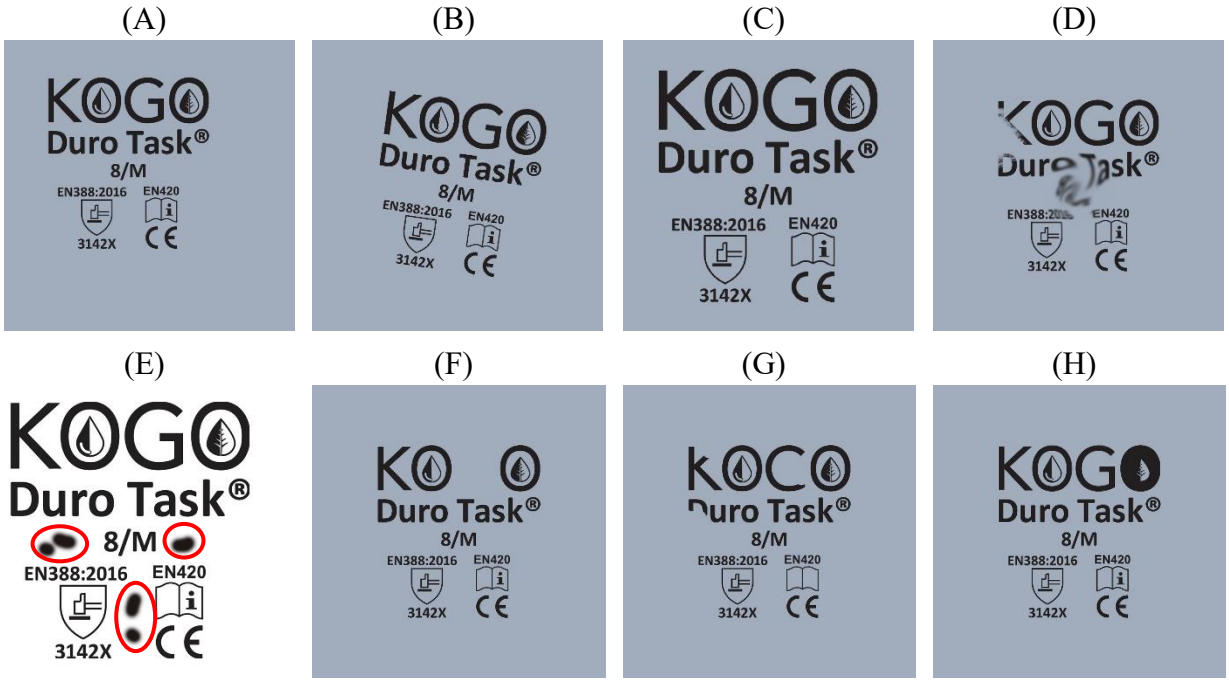


Figure 1.3: Type of Defects; (A) Displacement Defect, (B) Rotational Defect, (C) Scaling Defect, (D) Smudge objects, (E) Foreign bodies/objects, (F) Missing object, (G) Missing object's part, (H) Inkblots

The defects can be presented in different magnitudes. High-magnitude defects cause more damage to the content than minor defects. As an example, missing a whole component cause severe damage than missing part of a component. High magnitude defects may distort the content, hence challenging to identify. Sometimes a minor printing defect may not affect the content's visibility.

The current method of defect identification is manual. People at the production line inspect the sticker to identify defects soon after the printing process is completed. However, the printing process operates at a high rate, and it is challenging for a person to identify delicate defects at such a pace. There is a tendency to miss some defects and take much time. Therefore, there is a high demand for an automatic defect detection mechanism to identify more delicate defects in real-time and provide quantifiable measures.

In any manufacturing process, it is impossible to replicate something precisely similar. Hence some amount of dissimilarity is always possible. As mentioned above, there could be minor defects that are not affecting the content much. Similarly, it is impossible to obtain the print result precisely similar to the original sticker/artwork. Due to the high production cost of each glove, it is not practical to reject every glove with a defect. So it is essential to measure the impact of the defect on the product to decide whether to reject it or not. Therefore it is vital to have a defect detection technique capable of providing a realistic value that must reflect the actual impact on the product.

A simple defect detection method could provide the magnitude of the defect, which is used to define the quality measure of the product. However, the following phenomena described in the sticker printing domain are not captured by such quality measures. Some content on the sticker is less important compared to others. The label showing the size of the glove is more important since it has frequent access by the customer. A chemical resistivity information of a chemical glove is also important.

In comparison, a trademark registered label is less important. This research refers to this criticism as domain importance. The user must be aware of the importance of each content in the sticker to its domain. A defect on more important content may cause significant damage to the glove sticker. Therefore the domain importance must be considered when evaluating the impact of a defect.

The visibility level of content in the sticker is also a vital factor when considering the impact of a defect. This is because the customer could easily capture a defect on a highly visible component. Therefore the proposed solution must be capable of measuring the visibility level of content in the sticker.

A quantitative quality measure for a printed sticker is required for each accepting glove. By consolidating such measures, the production company could define the quality of the entire shipment.

1.3 Research Aims and Objectives

1.3.1 Aim

Develop automatic and efficient defect detection and quality measurement techniques for the glove sticker printing process by considering both the visibility and domain importance of the content.

1.3.2 Objectives

The project objectives can be listed down as follows,

1. Provide an automatic defect detection framework that can identify certain defects in real-time.

   The defect detection method should be fully automatic without user involvement. However, user involvement might need to set some criteria and parameter values. The

primary concern of this research is to introduce a framework capable of detecting defects in a minimum time duration. The defect detection algorithm in the proposed framework should be able to be replaced by new and more sophisticated defect detection algorithms in the future.

2. Identify the defects in a printed sticker in the early stage of the defect inspection process.

   The proposed solution should detect high magnitude defects early as possible and therefore decide to accept or reject the glove with less time.

3. Define the significance of the content/objects in the sticker.

   As stated in the introduction and problem statement, some reliable measure is required to evaluate the defect's actual impact on the glove. In such a case, the significance of defected object must be considered. This research aims to find the factors contributing to the object's significance and combine them to create a model that outputs a numerical measure.

4. Identify the acceptance and rejection of a glove using the specified criteria concerning calculated error and significance measure.

   A mechanism is required to accept or reject the glove by assessing calculated error with respect to the significance of the defective object in the sticker.

5. Define the quality of the printed sticker on an accepted glove.

   Finally, create a mathematical model to measure the quality of the printed sticker of each accepting glove where the model should reflect the errors in the sticker found in each inspection stage.

1.4 Scope

In this research, we develop an automatic framework for real-time defect detection in glove stickers. In order to make the process real-time, the choice of algorithms must be less time complex while being robust. Additionally, the chosen algorithms should be arranged appropriately to make the framework efficient. It is beneficial to identify the complex operations that can be performed before the defect inspection and store results for later use. The framework will be designed in two stages: one containing a complex operation that is pre-performed and stored in a database for real-time operations. The other stage contains real-

time operations that need to be performed just after the sticker printing is done.

Since this research deals with images, the fundamental operations in computer vision techniques must be adopted. There are two types of images considered in this research; the artwork image and the inspection image. The user uploads the artwork image, and it is in RGB color space and JPEG/PNG format, see *Figure 1.2*. The inspection image is acquired automatically for the defect inspection soon after the printing is done. The image preprocessing techniques such as grayscaling, thresholding, morphological operations, geometric correction, and image smoothing must be performed on images. Image segmentation, feature extraction, shape representation & description techniques will also be required. The selection of appreciate method is challenging because it is a real-time solution.

The defect detection process should be able to identify specific defects listed in *Figure 1.3*. The process will be designed to have several defect detection levels, where the complexity of algorithms will increase from top to bottom. Therefore, the high magnitude defects like displacements, scaling, rotations can be captured at an upper level and stop the inspection early.

Another main concern in this research is to model the object's significance that is used to decide the actual impact of a defect on a particular object. The model should combine the object's visibility and domain importance specified by the user. In this project, the domain importance (range 0.0-1.0) denotes that each content in the sticker holds some information related to the product (such as brand, model, size, chemical resistance level, etc.) and that information will be helpful for the customer to pick up the right product for him. The importance of each content varies according to the context. For example, the chemical resistivity information of a chemical glove is more important than the manufactured country information. Therefore, the user should define the importance level for each content before the inspection process starts.

Many facts affect the visibility of a printed object, such as size, density/solidity, contrast, and surrounding obstacles. The visibility of a printed object improves as its size and density grow. Additionally, when the contrast increases, printed things show out more against the background. The visibility of a printed object decreases when many surrounding objects are available close to the object itself. From the factors mentioned above, the essential factors for the current situation had to be determined and modelled accordingly while maintaining consistency.

After identifying a defect and calculating the error, there should be a reliable mechanism to decide whether to accept or reject the glove by considering the object's significance. Therefore a decision function must be proposed that should take the significance as an input parameter.

Finally, a mathematical model needs to be proposed to calculate the quality of each accepting glove. The model must consider all the errors identified and their impact when calculating the quality. The consistency and validity of proposed mathematical models and capabilities of defect detection algorithms will be evaluated using appropriate techniques at the end of the research.

1.5 Structure of the Thesis

Here is the outline of the main points of the thesis. Chapter one outlined the essential details of the problem, the objectives to be achieved, and the research scope. The introduction section under chapter one describes the problem environment, the requisite of the solution, and the computer science approach. The characteristics of the problem, constraints imposed by the environment are discussed under the problem statement section—the aim and objectives to be achieved are listed next. The scope section clearly described the concerns that needed to be covered to achieve the objectives, hence defining the boundary for the research.

The literature review chapter outlines similar researches are carried out for defect detection on printed materials. Here briefly discuss their approach, capabilities, and limitations of each research. Then discuss several computer vision techniques, how they work, and their suitability to the current problem.

The methodology chapter discusses the proposed solution with more technical details, figures, equations, etc. The methodology section is arranged in a staged manner, which follows the order to perform operations. Inside each stage, the order of steps is shown with flow charts. The techniques used to implement are thoroughly described, with figures containing results. The references are also given when appropriate.

The evaluation chapter discusses how well the given objectives are achieved from the proposed methodology with test results. The evaluation is experiment-based, and the test results are provided with images and comparison tables. Tests are done separately for each main concept discussed in this research.

Finally, the conclusion chapter emphasizes the strengths and weaknesses of the proposed approach and where it should be improved. The future research section describes several techniques in this research that can be improved as independent research in the future. The appendixes contain more test results with figures and comparison tables that follow the same format described in the Evaluation chapter.

CHAPTER 2

LITERATURE REVIEW

2.1 A Literature Review

Numerous researches have been carried out to identify printing defects in many industries using computer vision techniques, especially on paper-based materials, and not particularity was done to identify the printing defect on the sticker of a glove. However, in this research, we are considering an image of the printed sticker, and after binarizing the color image, the material that is printed on does not matter.

A real-time defect detection method for printed images is proposed by (Yangping et al., 2018) based on a combined approach of grayscale and gradient differences. A template-matching-based difference algorithm and more concerned about the high speed than the high accuracy. First, divide the inspection image into the edge and non-edge areas to reduce the influence of artifacts on defect detection. This algorithm uses different methods to identify the defect in edge and non-edge areas. The grayscale and gradient difference methods are joined to reduce false contours in the edge area without affecting the defective image. A quadratic image difference method is used to eliminate the edge artifacts in the nonedged area. Then combine both resulting images and perform BOLB analysis to detect the defect's size, area, and shape. However, this research provides GPU (CUDA) based parallel implementation to achieve real-time performance. Our approach also concerns real-time and efficient solutions with acceptable accuracy but without a GPU. This research focuses on designing non GPU-based defect detection techniques since many sticker printing stations are available in production houses. It is not cost-effective to have a high-end GPU for each printing device. Additionally, it is inefficient and unscalable to have one centralized system that performs all defect detection tasks from each printing station.

SSIM and Chromatism based techniques proposed by Zhou et al. presented a defect detection method of printing images on cans (Zhou et al., 2017). They have divided the defects into geometric defects and color defects. The geometric defects include scratches, stains, blur, and other line defects evaluated by SSIM. The color defects are the difference between the print colors and use the most common CIEDE2000 color formula to measure the color difference, which is detected via setting threshold. Since our printing domain has no vivid color output, the color difference method is not required and only concerns structural defects. Here they have used SSIM to detect structural defects. This Structural Similarity Measurement (SSIM)

index is a framework introduced by (Wang et al., 2004). It measures the structural similarity between two images. SSIM index is calculated by combining three similarity measures: Luminance, Contrast, and Structure. The luminance, contrast, and structural (shape) details are extracted from the two images, then compared individually, and that comparison output will be combined to yield an overall similarity measure. This method is more suitable for situations where photographs like images are compared to find similarities. However, in the research of (Premaratne and Premaratne, 2014) shows that the SSIM does not contain any notion of structure as structure in an image because their structural similarity calculation is simple cross-correlation. Cross-correlation would only capture similarity of pixels and not structure. Since the images of the our problem have no significant changes in luminance and contrast within the image, but more shape-based error. Therefore current problem requires a method that focused more on shape dissimilarities.

Wang et al. proposed a color printed image defect detection based on the image feature match (Wang et al., 2014). The defect detection algorithm is based on SIFT and YIQ models. The SIFT (Scale Invariant Feature Transform) is invariant in zooming, rotating, and brightness variation. This method is efficient and suitable for detecting offsets and color defects of paper printing methods.

In the study of (ou et al., 2007), a vision inspection system with high speed and online, is proposed to detect defects in printed matter. They have used the morphological preprocessing method to eliminate false defects brought by slight distortions. This research identifies the type of defect from an artificial neural network, trained by using a back-propagation algorithm, that can classify defects such as smudges, doctor streaks, pinholes, character misprints, foreign matters, hazing, and wrinkles, etc. The BP network inputs have six defect characteristics: position, area, length and breadth ratio, density, chromaticity, and shape. Their experimental results verify the speed, reliability, and accuracy of the proposed system. However, the detection accuracy of the system depends on the resolution. This research's scope does not include identifying the type of defect; therefore, it is unnecessary to implement a neural network.

Despite how many efficient and robust defect detection methods are in the literature, none considers the significance of the content to its domain when assessing the defect. That is our aim to model in this research.

As mentioned in chapter 1, this study is associated with many methodologies in computer vision. Each step in the proposed solution is required to have a different set of techniques.

Therefore it is mandatory to do a broad literature review over the correlated areas, which would be highly beneficial when selecting the most appropriate method in the implementation phase.

Since the computer vision research area has been out there for quite some time, numerous research articles can be found in literature, even for a single step in this study. Therefore we referred to both traditional and recent research articles to find the most appropriate techniques that are good enough for the current problem. A vital overview of the researches done related to this study will be discussed in the following paragraphs.

In any computer vision project, a preprocessing step involves making the input image suitable for the approach. There are multiple techniques available for preprocessing, and the choice depends on what kind of input image is expected by the subsequent steps. One of the most important steps is to convert the color image into a grayscale image and then into a binary image since many algorithms like feature extraction demand a binary image. Several image binarization and thresholding techniques are available, such as global thresholding, adaptive thresholding, Otsu's thresholding, and more. Each technique has different characteristics and is suitable for different problems. The adaptive thresholding method is suitable when the image has different lighting conditions in different areas, where the algorithm calculates the threshold for small regions of the image. In global thresholding, the pixels above and below a specific threshold value are assigned new values. The threshold value is a constant for the entire image. This method is suitable if we have prior knowledge about the intensity distribution and histogram. Otsu's method is an automatic threshold selection technique, and it calculates the threshold value from an image histogram of a grayscale image; however, it needs to be used with a global thresholding method (Otsu, 1979). Therefore with Otsu's method, we need to have at least some control over lighting conditions. In our situation, the inspection images are captured in a light-controlled environment; hence different illuminations on the image are not possible. Moreover, since we do not know the intensity distribution of the histogram, Otsu's method is best for our approach. It is an ancient method but is being used in many computer vision projects.

We could perform geometric corrections on the image as a preprocessing technique if the image is deformed in translation, scaling, or rotation. Bilinear Interpolation and Affine Transform are such geometric correction methods available in the literature. Several morphological operators such as erode, dilation, open, and close are performed on a binary image to remove unwanted objects, noisy pixels, fill small holes, etc. These methods have been implemented in many image processing libraries. In the proposed approach, some of

these preprocessing techniques will be used where it is necessary.

Suzuki and Abe have proposed two algorithms for topological analysis of images using border following techniques (Suzuki and Abe, 1985). These algorithms are capable of extracting contours of connected components in a binary image. These contours are useful for shape analysis and object recognition. In addition to the contours, it provides useful information such as the boundary and its surrounding relationships, i.e., hierarchical relationships of a component (the component inside the component, a hole inside a component, etc.). These algorithms can be used as the primary approach of segmentation in the proposed solution.

Sklansky proposed an algorithm to find the convex hull of any simple polygon (Sklansky, 1982). As they claim, when a sequence of m vertices is available, their algorithm can compute the convex hull with complexity O(m). However, later this algorithm, its predecessors and successors, has proven incorrect in certain situations. However, in the vast majority of cases, it produces the correct result. Even if this is an old algorithm, it is still prevalent in much recent research. (Fu et al., 2018) proposed an approach to measure shape similarity for areal entities in a geographical map where the convex hull is used to extract boundary features. It has been implemented in many image processing libraries. For our research, this convex hull approach would be beneficial for calculating the approximate area of a specific object. In order to calculate the hull area, we can use the green formula with the extracted convex hull.

Since the current problem has artwork as a template image, we can use image or shape similarity measurement techniques to find dissimilarities or defects in the inspection image. There are many such shape similarity measurement methods available in the literature. Some methods are robust but time complex therefore not suitable for this research. Therefore, the choice of the method is challenging because achieving high performance while preserving acceptable accuracy.

The current properties of human shape similarity judgments are not well understood yet, but some algorithms are developed to archive it to some extent. It is known that humans do not measure similarity in pixel by pixel. Therefore the object similarity measure should be calculated considering the object as whole or part based. The methods in literature and can be broadly categorized as region-based and contour-based approaches.

The moment representations describe image components as a probability density function of a 2D random variable and can be considered a region-based method. The properties of this random variable can be described using statistical characteristics called Moments. These moments, such as area, the centroid of a region, orientation, etc., can be used for region-based

shape representation in images. M K Hu has first introduced moment invariants, where he derived seven rotation, translation, and scale-invariant moment characteristics (Ming-Kuei Hu, 1962). It has been used as a foundation for many other similar types of research as well. These seven moments can be used for feature extraction and similarity matching of the objects in the proposed solution. Later research has found that Hu moment invariants have some limitations on image scaling and rotation because the images are not continuous function and can be polluted by noise. Additionally, the fluctuations are increased with the low-resolution images (Huang and Leng, 2010). Since this study uses good resolution and less noisy images, the above limitations can be ignored. Even if this is an old algorithm, it is proven to be a powerful tool and still a popular algorithm used in recent research. (Premaratne and Premaratne, 2014) presented a Moment Invariants-based Similarity Measure (MISM) where they have used the first two moments of seven moments invariants.

An image similarity measure based on moment invariants (MISM) was proposed by (Premaratne and Premaratne, 2014). This algorithm first normalizes two images by its own standard deviation, such that the two images being compared have unity standard deviation. Then approximate two images using wavelet decomposition and detect edges using the canny operator. Finally, calculate the first two moments invariant and compare using the equation proposed by them. They claim that MISM provides more accurate results than SSIM.

However, Hu's geometric moments are not orthogonal, thus resulting in information redundancy. (Khotanzad and Hong, 1990) proposed Zernike Moments (ZM), which are based on the orthogonal Zernike radial polynomials. The ZM are orthogonal and have no information redundancy; therefore, they have good shape representation capabilities. ZM moments are robust to noise. The ZM are rotational invariant but not scale and translation. Additional work has to be carried out to make the ZM scale and translate invariant. However, compared to other advantages, it would not be a problem. ZM has been used in shape representation and image retrieval in numerous researches and is still used. (Kim et al., 2000) proposed a modified ZM shape descriptor and similarity-based image retrieval method invariant to the translation, rotation, and scaling. (Hwang and Kim, 2006) proposed a novel approach to compute the Zernike moments. (Singh et al., 2013) given a method for accurately calculating the ZM. They proposed two algorithms to eliminate the geometric error and numerical integration error in the digitization process. (Castro-Ortega et al., 2019) proposed a hand vein pattern description from biometric data. Therefore, it is inevitable that Zernike moments are a powerful technique.

(Shnain et al., 2018) came up with a face recognition approach based on high-order statistics

and Zernike moments. Here they discussed the limitation of Structural Similarity measure (SSIM), Feature Similarity Measure (FSIM) in the face recognition context and proposed a method that combines Kurtosis and Skewness Measure (KSM) (Shnain et al., 2017), and Zernike moment for face recognition. They evaluate each method and prove their method is accurate than other methods. Even though it is proposed in the context of face recognition, it is suitable for other shape similarity contexts. We can use this method for our defect detection solution.

Basri et al. presented an approach to measure the shape similarity of deformable shapes using contour-based elastic matching (Basri et al., 1998). In this paper, they have identified several possibly desirable properties of a shape similarity method and determined the extent to which these properties can be captured by approaches that compare local properties of the contours of the shapes. Here, they have mentioned the importance of the part structure of an object; since parts appear to play a significant role in visual recognition, it is difficult to determine the part structure stably. They have mentioned that similarities of part structure can be captured without the explicit computation of part structure. As they have mentioned in the paper, this approach has some limitations. This approach has not considered the role of global properties of shapes, such as whether two shapes are symmetric or related by a single affine transformation. In our study, the sticker contains mostly arbitrary 2D shapes like logos, texts; hence this approach can also be appropriate. However, the choice of region-based or contour-based method would be made by analyzing the shape structures, required accuracy, and performance level.

For measuring 2D shape dissimilarity (Bribiesca and Wilson, 1997) proposed a solution. Two shapes are mapped to a representation invariant under translation, rotation, and area. This approach measures the similarity based on the transformation of one object to the other. First, perform maximum correlation on two shapes by taking one as a template. It is noticeably slow since this method superimposes one shape on another for all possible discrete transformations and rotations. After finding the best overlapping position, calculate the difference between the two shapes. For the remaining pixels, calculate the Euclidean distance to each other and make a weighted complete bipartite graph. Then, by using the *Kuhn-Munkres* algorithm, calculate the minimum distance that has to be made by each pixel to become the other shape. This measure will be taken as the dissimilarity between the two shapes. This algorithm has proven to be inefficient. However, with a preprocessed shape, this algorithm is yet to be robust.

Tu and Yuille presented an algorithm for shape matching and recognition based on a generative model for how one shape can be generated by the other (Tu and Yuille, 2004). This

approach uses a modified variant of the EM algorithm. Their work is limited by the types of representations they use and the transformations they allow. For example, it gives poor results for shapes composed of parts that can deform independently, such as human figures. However, in our case, the sticker does not contain part structures, and it is unnecessary to concern about independent deformations. Since the nature of Heat transfer printing and Pad printing methods, the independent deformations do not happen naturally.

The contour-based methods are more time complex than region-based methods. The contour-based methods are more sensitive to noise compared to the region-based methods.

As of the author's knowledge, no method has been proposed for defect detection, which concerns the significance of each printed content to its domain. Our primary goal is to propose a model that measures the significance of each content in the sticker. Moreover, provide a real-time defect detection approach that does not require to have GPU.

CHAPTER 3

METHODOLOGY

3.1 Introduction

The methodology section described how the research was conducted and how the objectives in chapter 1 were achieved, using knowledge gained from literature review and experimentations. As mentioned before, the majority of the scope is based on Image processing and Computer Vision techniques. This chapter provides a comprehensive overview of the design and implementation steps that have been carried out to achieve the goal.

The whole solution is divided into two operational modes based on the occasion it will be performed; Teaching mode and Inspection mode. The user initiates the teaching mode when a new glove type arrives for production, where the user must create a template in the database by uploading the necessary information required for the defect inspection later at the production house. The teaching mode operations are performed only once for a unique type of sticker. All computer-intensive tasks (time-consuming work) that produce essential outputs required for defect identification and decision making are done in the teaching mode. Therefore, the inspection time can significantly reduce and output the decision in real-time. Hence, the teaching mode plays a significant part when achieving the first objective, the real-time solution. The technical details of the teaching mode operations are discussed under subsection 3.2.

The inspection mode is a fully automatic process where the user has no involvement. These operations are performed for each glove after the print is done at the production house. As mentioned in the introduction, sticker printing is done at a significantly high speed. Therefore the inspection process must efficient as possible to eliminate the bottleneck. By considering this nature, the defect inspection framework is designed to decide whether to accept or reject the glove in the early stages. The framework consists of three defect detection levels, where the first layer contains naïve and less complex algorithms; therefore, it can only detect high-magnitude defects. The second layer is quite complex than the first layer and takes time to operate. The third layer contains more complex algorithms, and it can detect sophisticated errors; hence it would take more time than the other two levels. This leveled approach can detect high magnitude defects at a beginning level; hence it is possible to decide whether to accept or reject the glove early. The first two objectives can be achieved from the framework mentioned above. The technical details about inspection mode implementation are described

in subsection 3.3.

As stated in objective three, the significance of each object in the sticker is calculated by concerning the object's visibility and domain importance. A mathematical model is proposed to combine the two factors. A separate mathematical model is proposed to calculate the object's visibility which combines the object's size and density. These measurements are precalculated in teaching mode and stored in a database. More technical details are discussed under subsection 3.2.3.

A mechanism is proposed to evaluate error and make the decision, from which objective four will be achieved. This mechanism comprises of decision function, which calculates a maximum expected error for a specific object. The implementation and its use will be discussed in subsection 3.3.7.

Finally, a mathematical model to measure quality for each accepting glove is newly proposed in this research, discussed under subsection 3.3.8.

3.2 Teaching Mode

The primary purpose of this mode is to create a template in the database for a specific sticker type. When a new glove type arrives at production, the user initiates this mode by uploading several pieces of information. Then several automatic operations are performed on uploaded data to extract the information required in defect inspection and store it in the database.

The main tasks of Teaching Mode include

1. Collecting user inputs
2. Process the artwork to extract features
3. Calculate object visibility and significance
4. Calculate the maximum expected error for each object

At the end of this mode, a template database for this specific glove is created. The steps and the results of each operation are thoroughly explained in subsequent sections.

3.2.1 Collect User Inputs

At the beginning of the teaching mode, the user should import the artwork image and input some information into the system.

The user must provide an image of the artwork into the system *(Figure 3.1)*. The artwork image is usually high resolution and in RGB color space with jpg or png file format. The background is white (255, 255, 255), and the foreground is black (0, 0, 0). The user must also input the following information along with the images.

1. *Sticker-placement-region (Figure 3.2)* – the sticker is supposed to be printed inside of this region. If it is printed on or outside from this region, the glove will be rejected.
2. *Size-threshold* – The user-accepting limit of a scaling defect. The use of this value will be discussed in the inspection mode.
3. *Angle-threshold* - The user-accepting limit of a rotational defect. The use of this value will be discussed in the inspection mode.
4. Select regions of artwork and define the domain importance for each region *(Figure 3.3).* As stated in the problem statement, each label or object in the sticker holds some information relevant to the glove. Not every label is equally important to the domain. The user must be aware of how important each label is to a particular glove type, and he/she must provide that information to the system. The domain importance is a discrete variable ranging from 0 to 1 with 0.1 increments, where 0 indicates least importance and 1 indicates the highest importance. A simple desktop application is provided to the user for this purpose *(Figure 3.4),* where the user can select a label and specify the importance. The specified domain importance value will be applied to every individual object in the selected label.

Figure 3.1: Example of Artwork

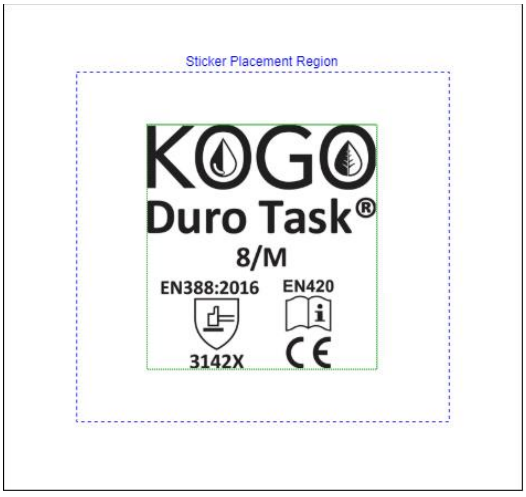


Figure 3.2: Artwork with sticker placement region

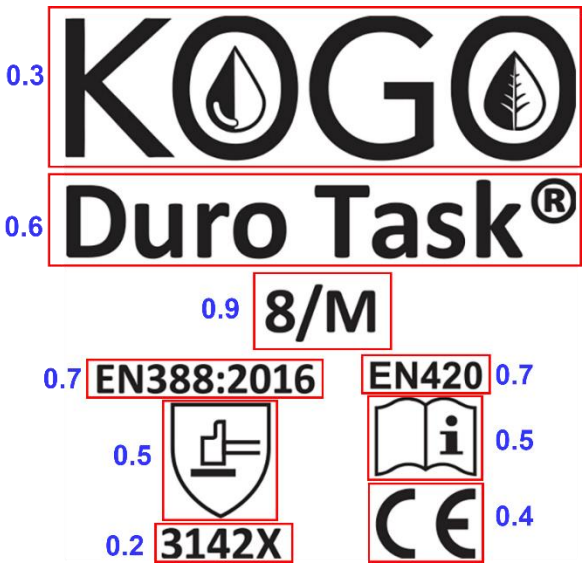


Figure 3.3: Artwork with Domain Importance (0.0-1.0) for each content (red box)

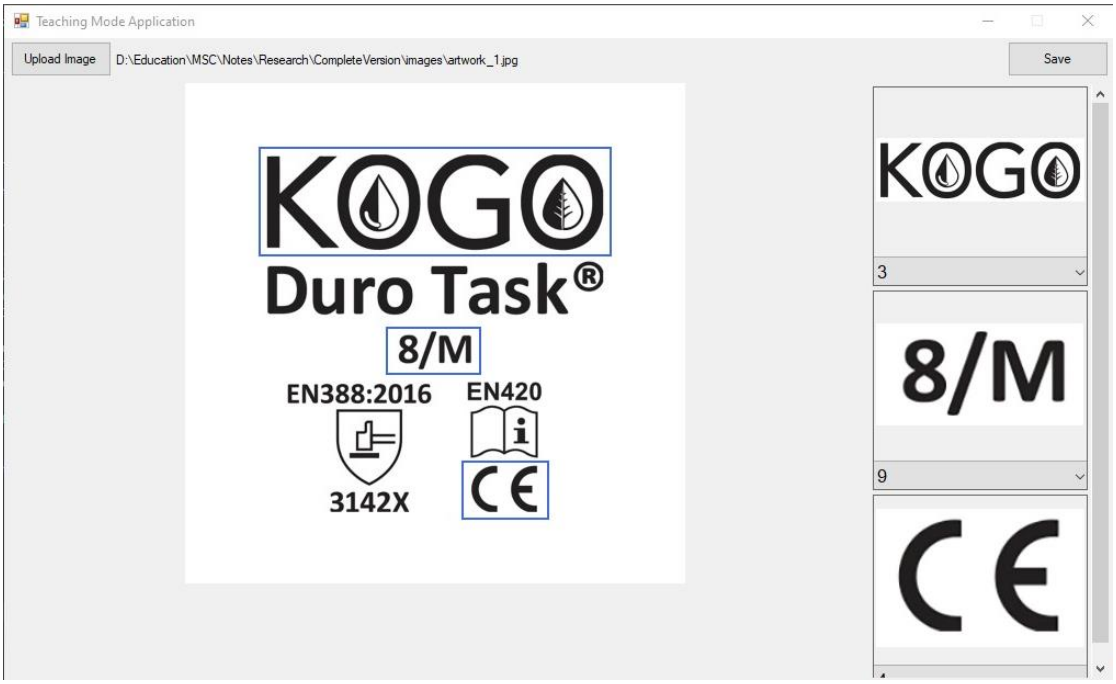


Figure 3.4: Domain importance selection application

3.2.2 Processing of Artwork

The artwork image will be going through several processing steps to extract features and information essential for the inspection process. The steps of artwork processing are shown in *Figure 3.5,* and a detailed explanation of each step is as follows.

The artwork image is converted from 24-bit RGB color space into 8-bit grayscale (0-255) format to reduce unnecessary complexity using Eq.*(1)*. It is a well-known method in the

literature called the weighted method or luminosity method, (Burger and Burge, 2009).

$$Y \leftarrow 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \qquad (1)$$

The grayscale image is then converted into a binary image using the inverse of the fixed-level threshold method *(2)*. The inverse function has to be used here because the foreground is low-intensity and the background is high-intensity.

$$g(x,y) = \begin{cases} 0 & \text{if } f(x,y) > \text{threshold} \\ 1 & \text{otherwise} \end{cases} \qquad (2)$$

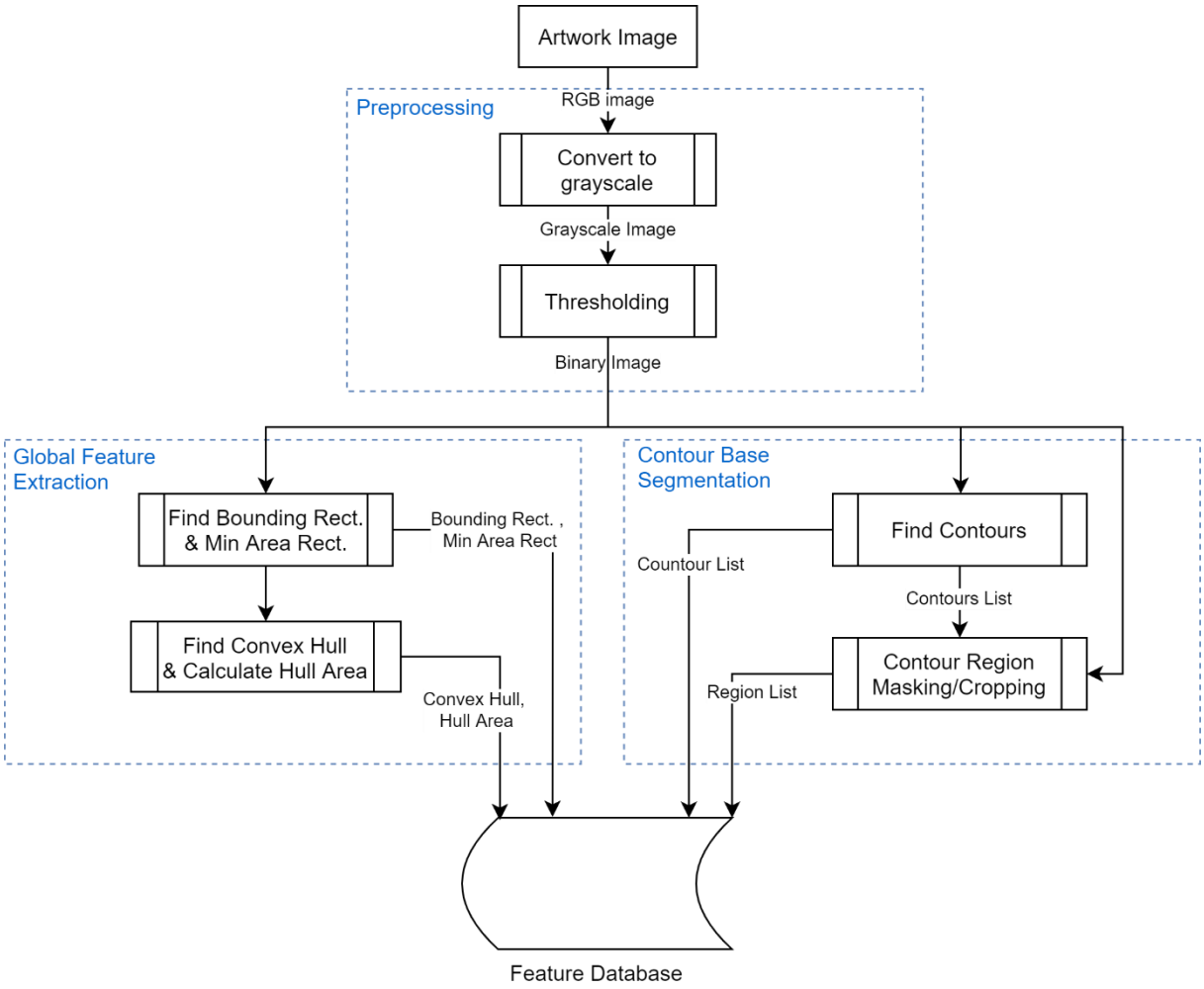


Figure 3.5: Artwork processing

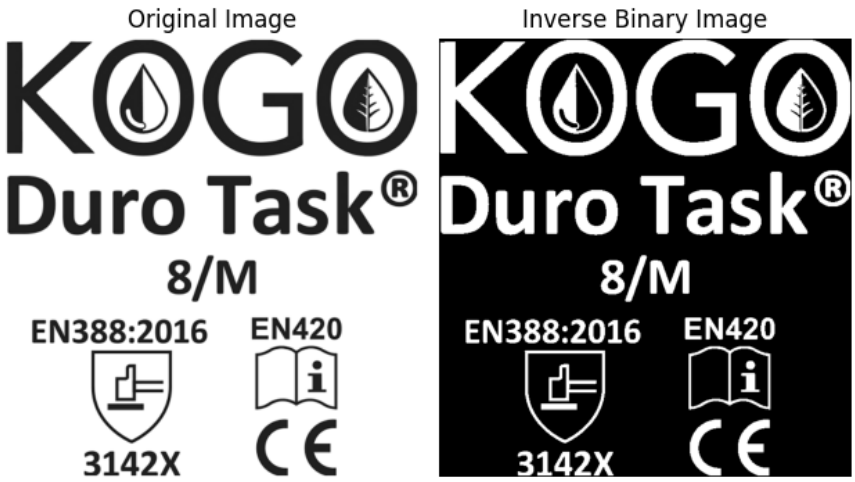


Figure 3.6: Original image vs. Inverse binary image

The binary image is then used to extract several global features where the image is considered whole, not as the individual components. The global features extracting here are bounding rectangle, minimum area rectangle, and convex hull. The bounding rectangle function calculates the x, y, width, height of the up-right bounding rectangle of non-zero pixels in the binary image. That can be used to remove the padding around the non-zero region and get the actual dimension of the sticker. The minimum area rectangle function calculates the rotated rectangle enclosing non-zero pixels, resulting in useful information such as *center*, *width*, *height,* and *angle* of rotation. The algorithm proposed by (Sklansky, 1982) is used to find the convex hull of the whole image. The resulting convex hull points are used to calculate the *hull area* using the *Green formula,* and the *hull area* is used for size comparison in inspection mode. The extracted features are shown in *Figure 3.7.*

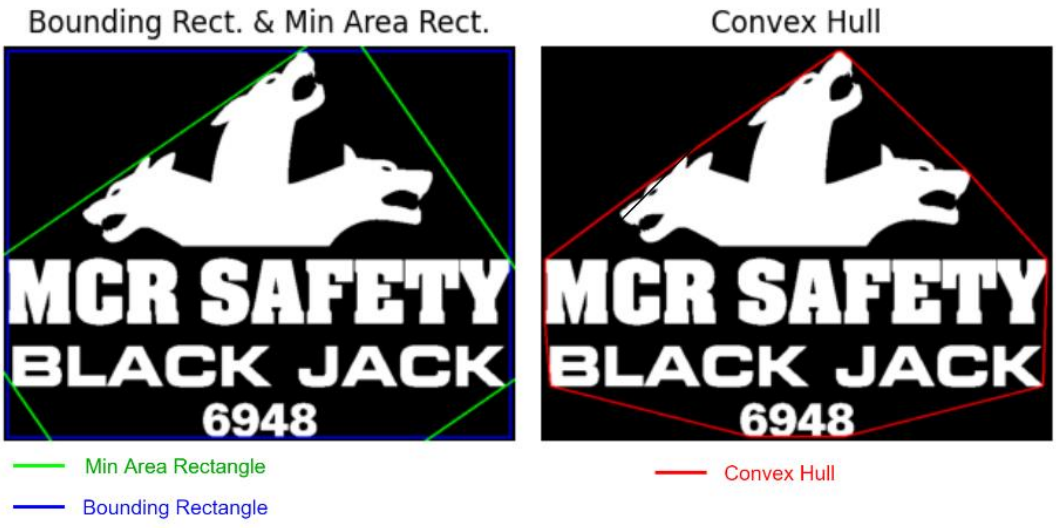


Figure 3.7: Bounding Rectangle, Min Area Rectangle & Convex Hull

The binary image is then used for contour-based segmentation, which decomposes the whole image into isolated components. The contours in binary images are extracted using the algorithm proposed by (Suzuki and Abe, 1985). For the implementation and testing, the *findContours()* function in the OpenCV library is used. At this stage, the hierarchical arrangement of the contours is not concerned, and only the external contours are retrieved using the *RETR_EXTERNAL* parameter in OpenCV. Each isolated contours set is considered an individual object, and using these contour points, the objects are cropped out from the whole binary image. The contour points and cropped-out region of an individual object are stored in the feature database for later use.

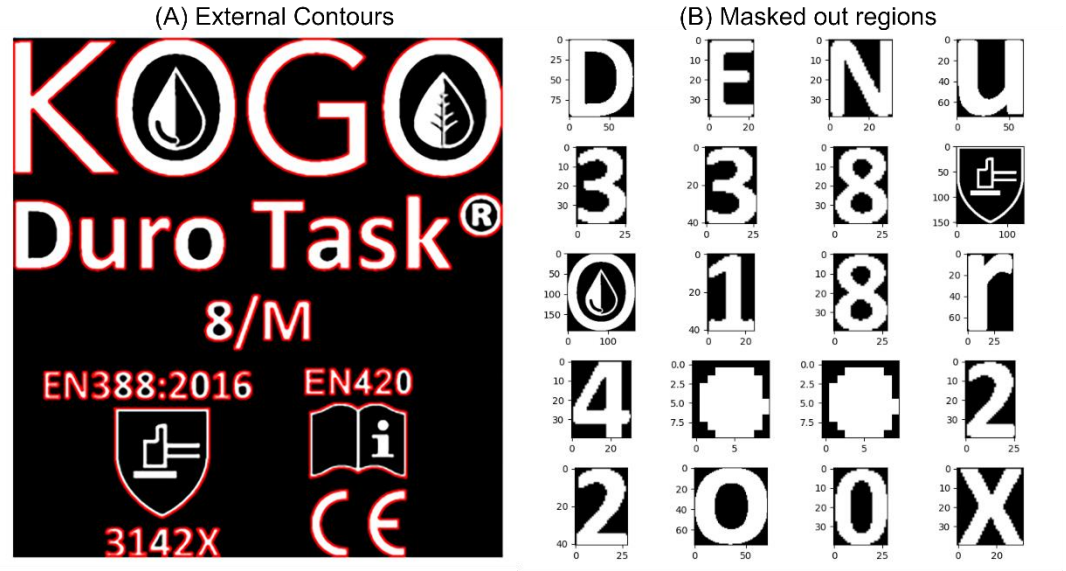


Figure 3.8: (A). External Contours in red color, (B). Masked out/cropped contour regions

All the features and information extracted from the artwork will be stated as template features in subsequent sections. In addition, the binary artwork image refers to as the template image, and each object in the artwork will be referred to as a template object.

3.2.3 Calculate object visibility and significance

This research newly proposed two mathematical models to measure the visibility and significance of each object that is extracted from the artwork.

Object Visibility

Multiple factors determine the visibility of a 2D shaped object, such as size, density, contrast, surrounding obstacles, the complexity of the shape, convexity, etc. However, by closely examining the nature of the problem, it is decided to choose only two factors, size and density, into the research scope.

The size is calculated using Eq.*(3).* The hull area is computed using (Sklansky, 1982) method followed by the *green formula* as stated above. This value means having the notion that any object with an area beyond the upper bound value is visible anyway. Therefore the upper bound is chosen by considering the pixel dimension of the whole sticker with respect to a standard unit. The hull area greater than the upper bound is set to maximum size 1; otherwise, get the ratio *(Figure 3.9).* The validity of the model is discussed under the evaluation section with relevant test results.

$$size\ (s_o) = \begin{cases} 1 & hull\ area > A \\ \dfrac{hull\ area}{A} & otherwise \end{cases} \qquad (3)$$

$$A\ -upper\ bound\ for\ area$$

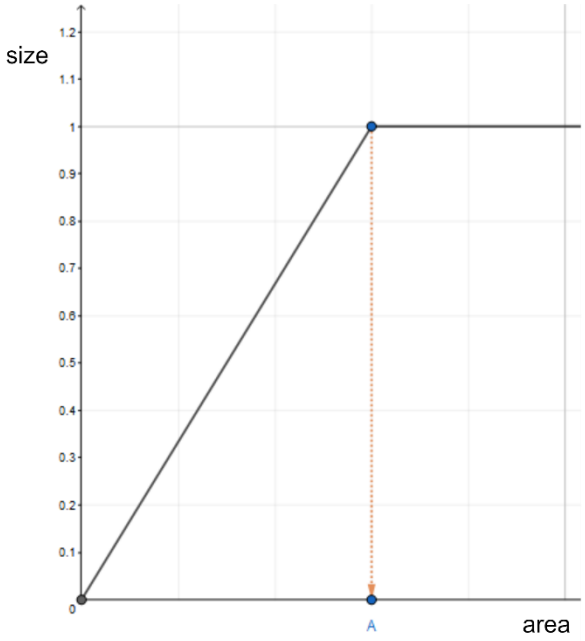


Figure 3.9: Graph of the size calculation function

The density is calculated using a well-known method in the literature (Burger and Burge, 2009), Eq. *(4)*. The $\boldsymbol{area}$ is number of white pixels in object.

$$density\ (d_o) = \frac{area}{hull\ area} \qquad (4)$$

Then the proposed visibility model combines size and density with respect to their weight values, as in Eq. *(5)*; where the $\boldsymbol{w_1} + \boldsymbol{w_2} = 1$. The influence of each factor to the final result can be changed by changing the weight values. The consistency and validity of this model will be discussed in the evaluation section with test results.

$$Object\ Visibility\ (V_o) = w_1 \times s_o + w_2 \times d_o$$
$$s_o \text{ - } size$$
$$d_o - density \qquad (5)$$
$$w_1, w_2 - weight\ values$$

Object Significance

A new model is proposed in this research to calculate the significance of each object in the sticker. It combines the calculated object visibility and domain importance specified by the user with respect to their weight values, as in Eq. *(6),* where the $\boldsymbol{w_3 + w_4 = 1}$. The influence of both factor can be changed by changing the weight values. These values are stored in the database for later use in inspection mode.

$$object\ significance\ (f_{sig}) = w_3 \times I_o + w_4 \times V_o$$
$$V_o - Object\ Visibility$$
$$I_o - Domain\ Importance \qquad (6)$$
$$w_3, w_4 - weight\ values$$

The consistency and validity of this model will also be discussed in the evaluation section with test results.

3.2.4 Calculate the maximum expected error

The maximum expected error is calculated for each object in the artwork. It is the maximum error acceptable for a specific object. The object's significance value calculated previously is taken as a parameter to calculate this value. This function is called the decision function because it can decide whether to accept or reject the glove by evaluating errors in the inspection. This measure can also be precalculated in the teaching mode and stored in the database to use in inspection mode, hence saving inspection time. A more detailed explanation about the implementation with figures and equations is given in subtopic 3.3.7 in inspection mode.

3.3 Inspection Mode

The inspection mode is a fully automatic and real-time process. It is triggered soon after the sticker is printed on a glove. Then follows series of operations and decide whether to accept or reject the glove. If the glove is accepted, the quality of the printed sticker will be calculated.

3.3.1 Image Acquisition

A draft version of the actual physical setup is shown in *Figure 3.10*. There are multiple metal hand formers attached to a conveyor chain with fixed gaps. When the conveyor chain receives a signal, it begins moving formers to one side for a predetermined distance. The setup is pre-calibrated; therefore, the setup ensures that the formers are precisely aligned with the printing equipment and cameras. The glove with no sticker will be attached to a metal former on one side of the setup. (left side of *Figure 3.10*). Once the wearing process is completed, the former is moved to the printing unit, where the sticker is printed on the glove's surface. After printing is complete, the printing unit transmits a signal to the conveyor chain instructing it to begin moving. The former will perfectly align with the camera when the moving is completed. Soon after the move is completed, the conveyor chain system will initiate the defect inspection process by sending a signal to the defect inspection system.

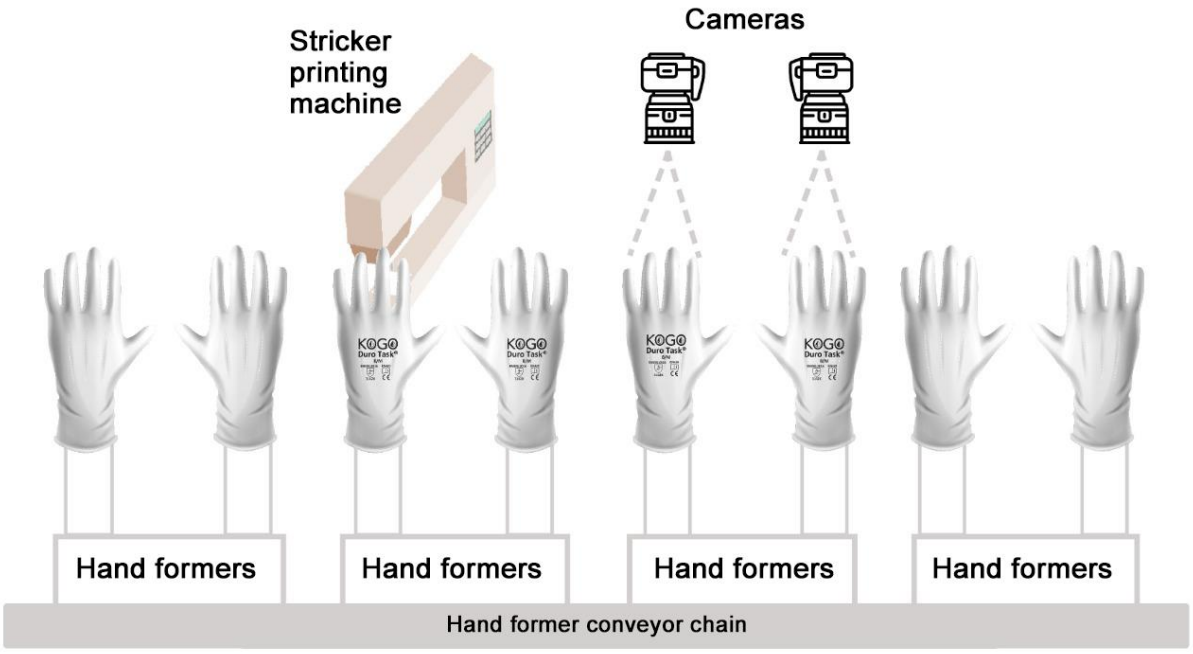


Figure 3.10: Physical Setup on the production house

The defect inspection system acquires the inspection image from a high-resolution camera mounted on top of the former. Since the camera's position is not changing, the distance and angle from the camera to the printing glove are always the same. Therefore, the camera's region of interest (ROI) is fixed, and the output image has an approximately fixed dimension in pixels for every inspection. The captured image is in RGB color space and fixed resolution. The whole setup is in a light-controlled environment; therefore, uneven illumination in the captured image is rare. Soon after the image acquisition is made, the system starts to inspect the image.

Because the camera's distance and Field Of View (FOV) are constant, the glove's position in the acquired image is approximately constant. Therefore it is possible to crop the area that sticker should be printed correctly and then perform inspection only on that area. As a result, it is easy to crop the region on which the sticker is supposed to be printed and then perform inspection only for that region. Cropping the region with a fixed ROI is possible since the dimensions are always fairly identical. This method simplifies the complexity of extracting the glove from the background and locating stickers by examining the entire glove. Additionally, it considerably improves the algorithm's efficiency.

After the inspection, the system provides a result command indicating whether the glove should be accepted or rejected. The inspection system will generate a signal and transmit it to the physical setup, which will take the glove from the former and place it in the proper location based on the outcome. Then the conveyor chain is triggered to start moving again.

3.3.2 Preprocessing

After acquiring the inspection image, several preprocessing steps need to be followed, as stated in *Figure 3.11*. The inspection image is in RGB color space, and it is converted into Grayscale format using the equation *(1)*. As the nature of this problem, since it has a high-contrast factor, the foreground objects are clearly separable from the background; this can be seen in the histogram diagram in *Figure 3.12*. Therefore the grayscale image is converted into a binary image using Otsu's threshold method. It is proven that the high-contrast factor provides better segmentation results, which significantly benefits the accuracy and performance of similar solutions.

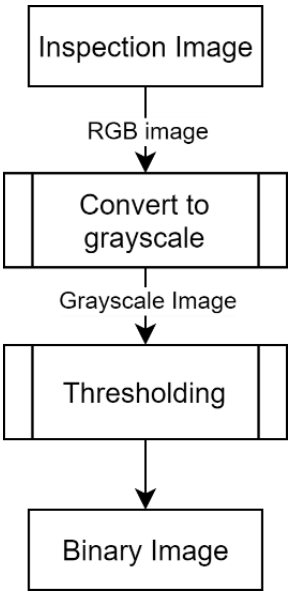


Figure 3.11: Preprocessing steps

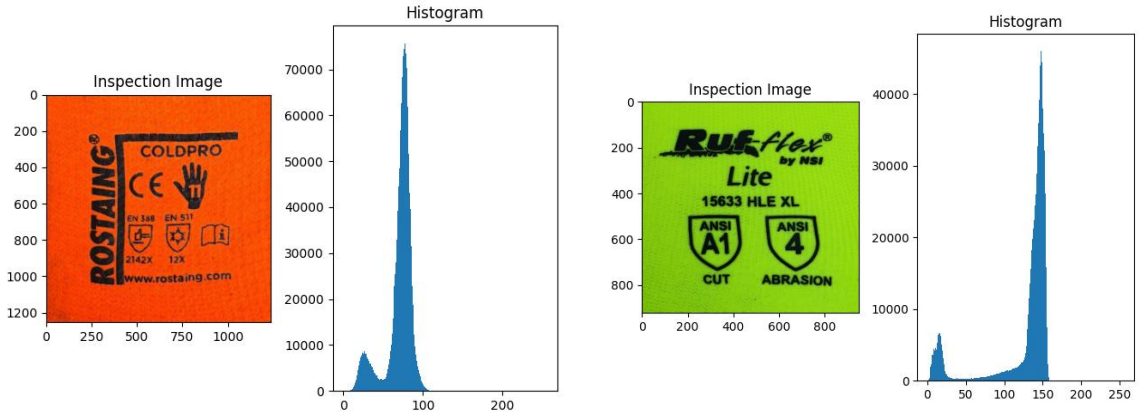


Figure 3.12:Example Histogram of Inspection images

3.3.3 Inspection Process

The inspection process is broken down into three defect detection levels; in each level, out of three decisions (i.e., ACCEPT, REJECT and PROCEED), one decision will be taken. The inspection image is evaluated at each level to determine either the glove should be rejected or accepted. If the current level cannot make a decision, then proceed to the next level. This architecture leads the system to make an early decision by analyzing the data processed by current or previous levels; hence, the operation is efficient and effective. Different criteria are checked at each level to find the defects/errors in the sticker. The complexity of the algorithms and the restriction of criteria are increasing at each level. Some sticker defects can be captured in early levels; therefore, the system will decide to reject the glove and stop the process without proceeding to subsequent complex levels. A high-level abstraction of the inspection process is depicted in *Figure 3.13*.

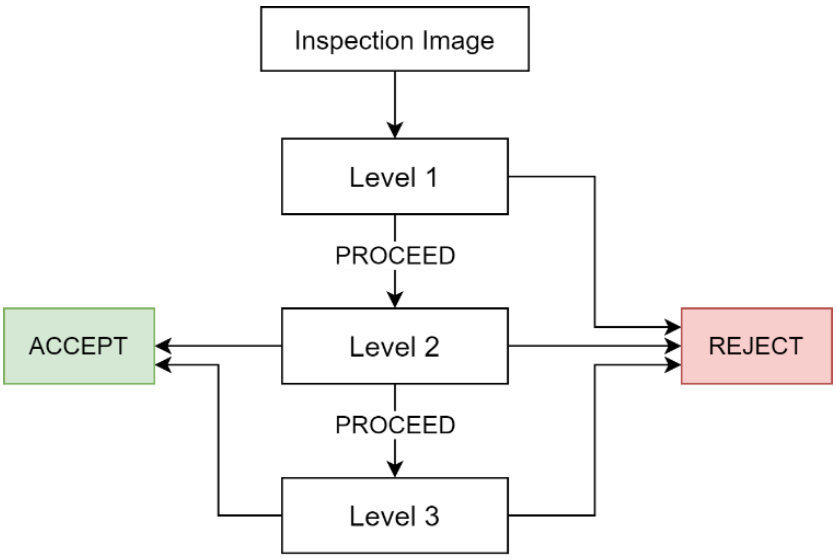


Figure 3.13: Levels of Inspection Mode

3.3.4 Level 1 – Inspection Process

In this section, the level-1 inspection process is discussed in detail. The entire process flow is shown in *Figure 3.14*. The main concern of level-1 is identifying the geometric deformations on the whole sticker resulting from the printing process. The whole sticker is considered as a single component here.

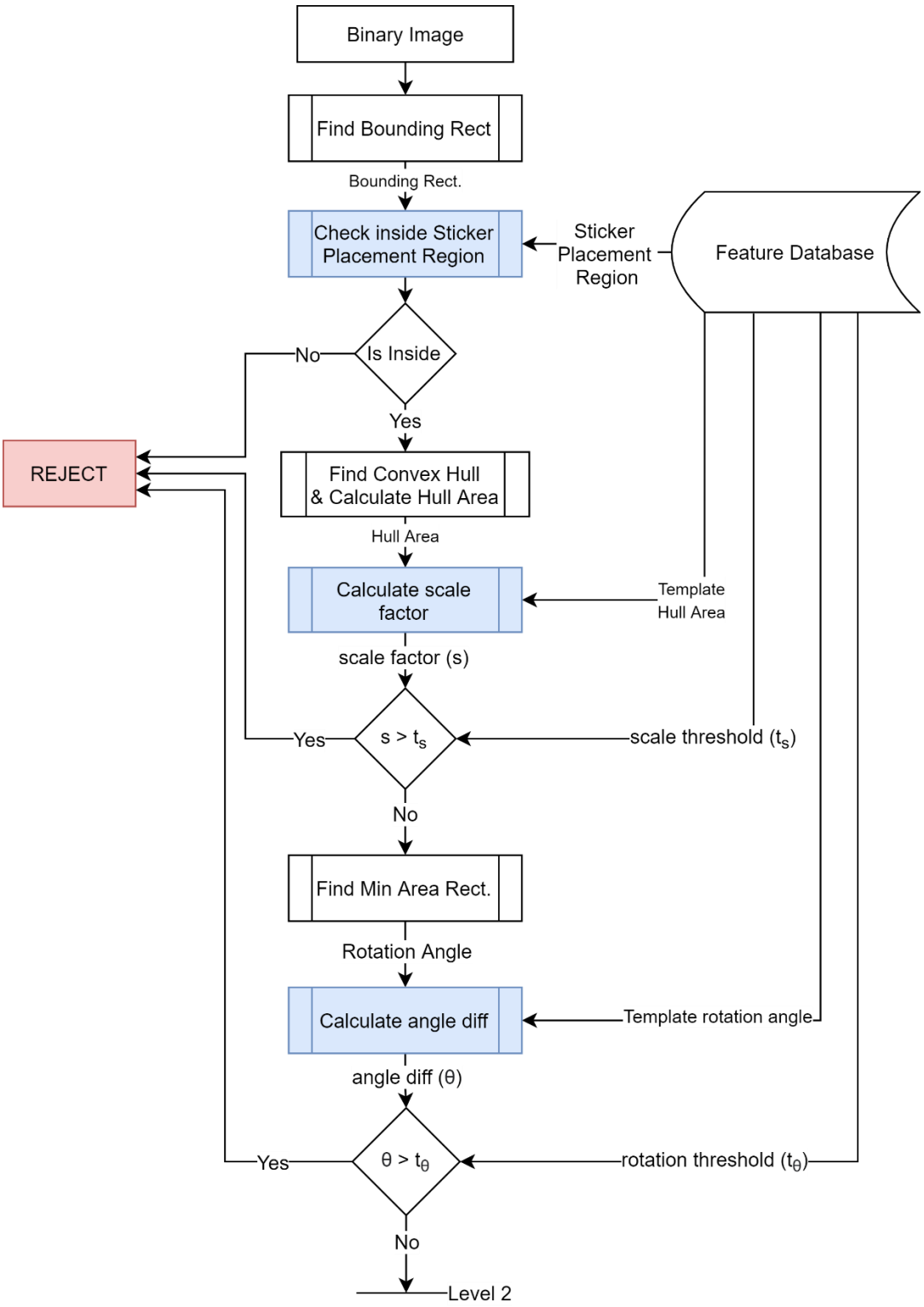


Figure 3.14: Inspection process - level-1

The inspection image is compared with the template image to identify and quantify the degree of geometric deformations such as translation, scale, and rotation. Suppose the degree of deformation is higher than the corresponding threshold specified by the user. In that case, the system will decide to reject the glove. Otherwise, the system will decide to proceed to the next level for more inspection.

The binary image that is resulted from preprocessing will be provided to series of global feature extractions and criteria evaluations, as shown in *Figure 3.14*. First, obtain the bounding rectangle *(x, y, width, height)* of the whole foreground using the same method in artwork processing. Then retrieve the *sticker-placement-region (Figure 3.2)* specified by the user in teaching mode from the feature database and check whether the bounding box is placed inside the region *(Figure 3.15)*. The glove is rejected if the bounding box is either on top or outside the specified region *(Figure 3.16)*. Otherwise, move to the next step.

Figure 3.15: Sticker placed correctly within the region

Figure 3.16: Sticker placed incorrectly, on the region

In the next step, find the convex hull of the whole foreground as the same method performed in artwork processing (Sklansky, 1982). Then compute the hull area (A_i) using the *Green formula*. Retrieve *template hull area* (A_t) and *scale-threshold* (t_s) from the feature database and calculate the scale factor *(s)* using Eq. *(7)*, and an example is depicted in *Figure 3.17*.

$$scale\ factor\ (s) = \frac{|A_t - A_i|}{A_t} \qquad (7)$$

If the scale factor is higher than the *scale-threshold* specified by the user, the glove is rejected. Otherwise, move to the next step.

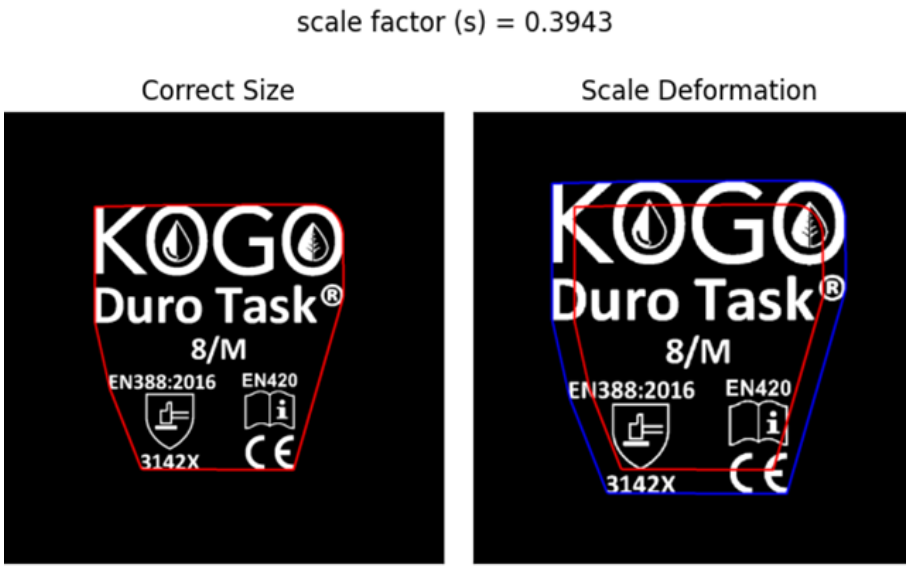


Figure 3.17: scale-factor calculation

In the next step, find the minimum area rectangle to obtain the rotation angle (a_i) of the whole image. Retrieve *template's rotation angle* (a_t) and *rotation-threshold* (t_θ) from the feature database and calculate the angle difference (θ), *Figure 3.18*.

$$angle\ diff\ (\theta) = |a_t - a_i| \qquad (8)$$

If the angle difference is higher than the *rotation-threshold* (t_θ) specified by the user, the glove is rejected. The level-1 processing is finished, and if no decision has been made, the system moves to the next level, i.e., level 2.

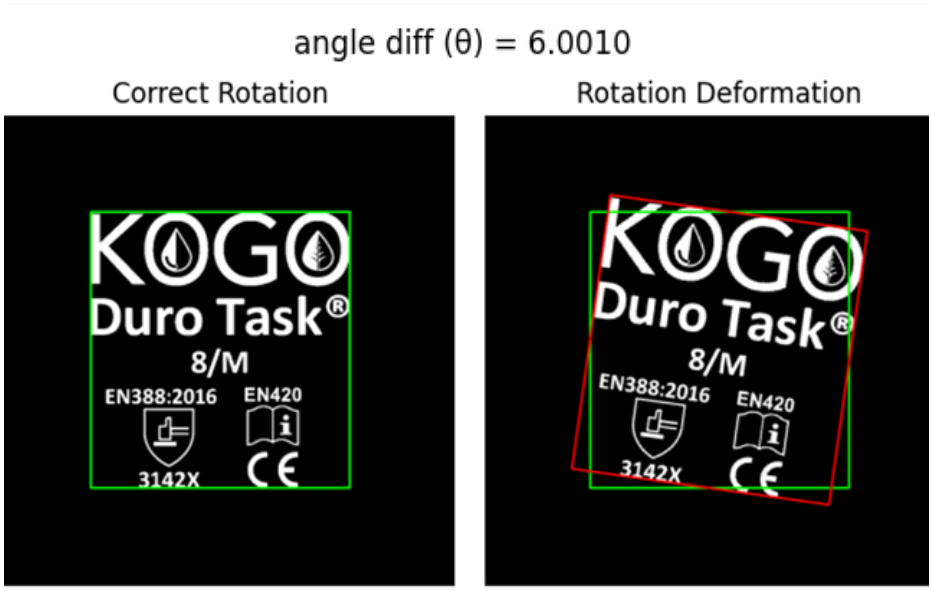


Figure 3.18: Rotation difference calculation

The level-1 operations are capable of detecting most of the high-degree geometric deformations on the sticker. If such deformation is present in the sticker, it can be captured

early and stop the process without proceeding to complex levels. Since the level-l checks for high-degree deformations using naïve algorithms, it is impossible to make an Accept decision, but multiple Reject decisions are possible.

3.3.5 Level 2 – Inspection Process

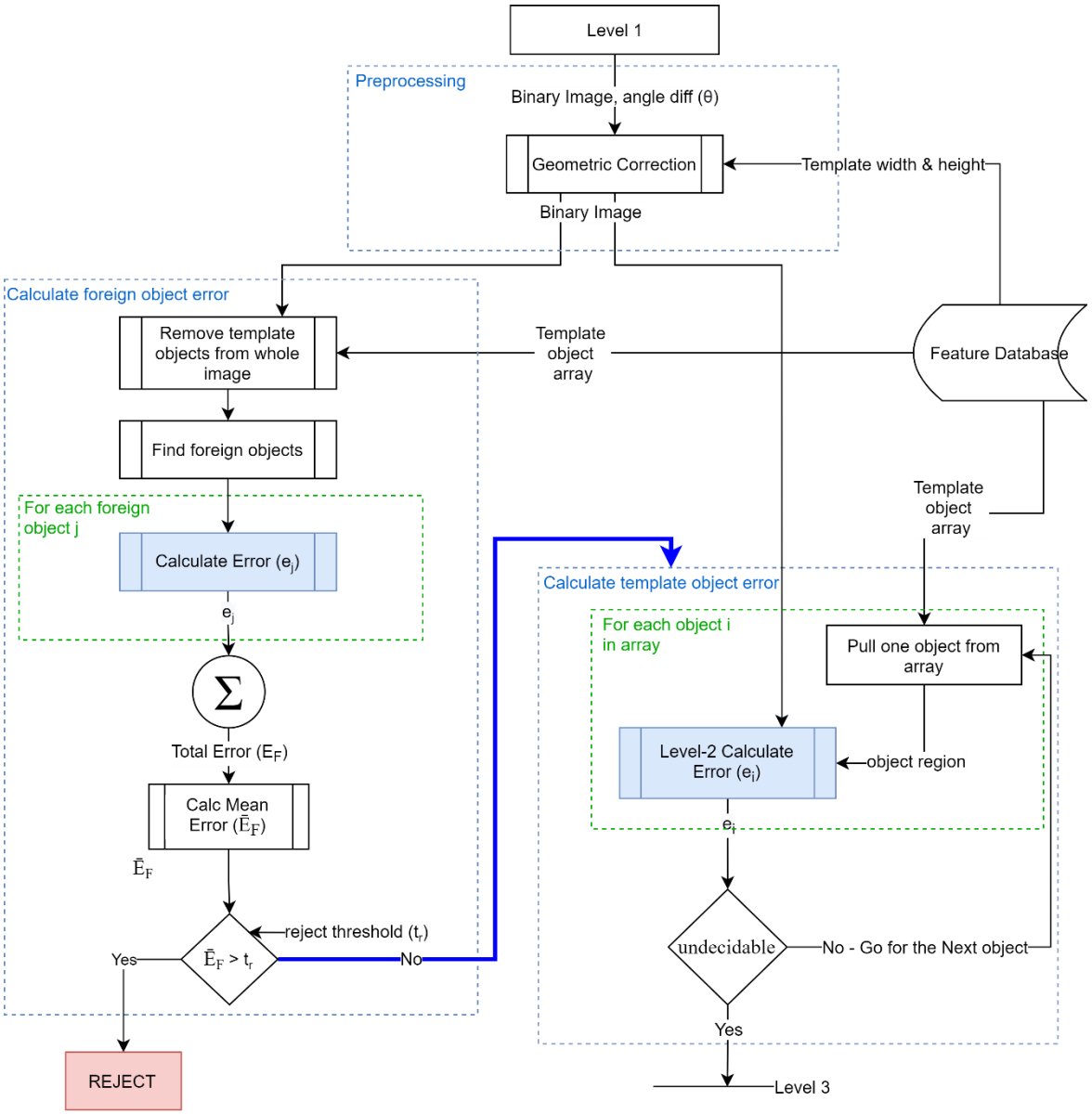


Figure 3.19: Inspection process – Level-2

The system proceeds to level-2 if a decision has not been made in level-1. The complete process flow of level-2 is shown in *Figure 3.19*. In level-2, the system tries to find defects in an individual object on an inspection image, unlike level-1 operations where consider the whole sticker as a single object. This level starts with a preprocessing step where several geometric corrections will be applied to the binary image. Then the corrected binary image is

passed into two separate subprocesses; (1). Calculate the total error of the foreign objects, (2). Calculate the total error of template objects. These operations are discussed in more detail under subsequent sections.

Geometric Correction

Level-1 can proceed with accepting some degree of deformations; thus, the binary image passed to level-2 might present some degree of geometric deformation such as translate, scale, and rotate. These deformations must be corrected before using the binary image for other operations.

The first step is correcting the rotation deformation into the correct angle as in the template image. The affine transform method is used to correct the rotation of the binary image. The *angle difference (θ)* calculated in the previous level can be taken as an input for the rotation function. Then, calculate the minimum bounding rectangle (x, y, w, h) for the corrected image and crop the non-zero area from the entire binary image, eliminating the translation deformation. Retrieve *template bounding rectangle* (x_t, y_t, w_t, h_t) from feature database and then correct the scale deformation by changing the Inspection image's scale to Template image's scale using bilinear interpolation. All the deformations in the inspection image are corrected. Both the template image and inspection image now have the same dimensions. The steps are shown in *(Figure 3.20)*

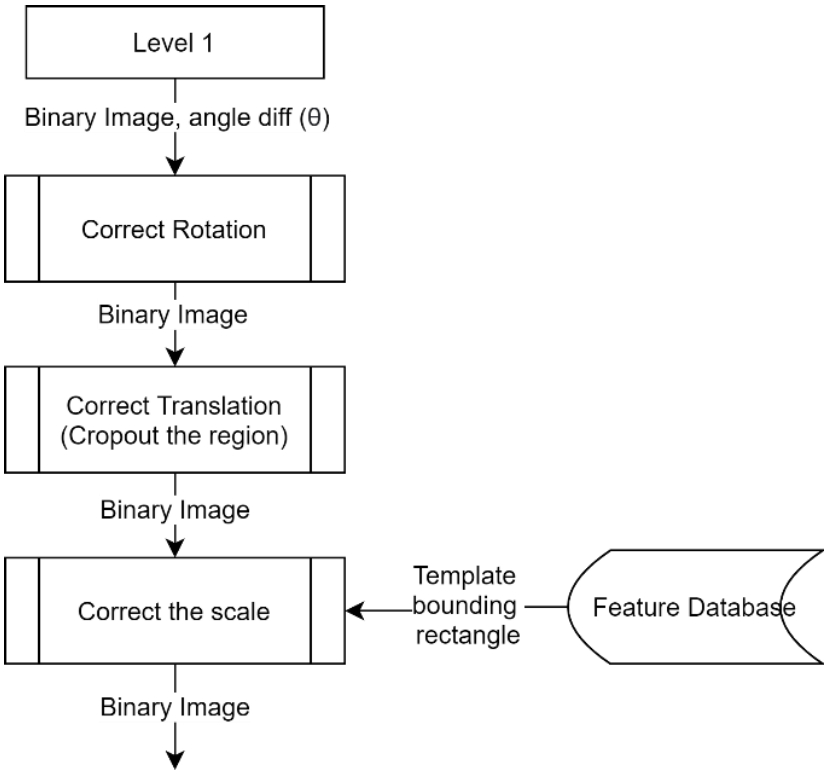


Figure 3.20: Preprocessing steps of level-2

(1). <u>Calculate the total error of foreign objects</u>

A foreign object is an object detected in the background area of the inspection image, which does not belong to the same place on the original artwork *(Figure 3.21)*. These objects are possible when ink is spilled on the glove, smudge an object after printing, or a displaced object, etc. The foreign objects can be identified as printing defects; when a considerable degree of error is presented, the glove must be rejected. The identification of foreign objects and calculating error is discussed in this section. The steps are depicted in *Figure 3.22*.

Figure 3.21: Foreign Objects

First, remove all template objects from the inspection image. Then, use the contour extraction algorithm (Suzuki and Abe, 1985) to find objects in the resulting image *(Figure 3.23)*. If the resulting image is empty and no contours are detected, there are no foreign objects available. If there are foreign objects available, extract local features such as the size and density of each object using the same method explained in *(3),(4)*.

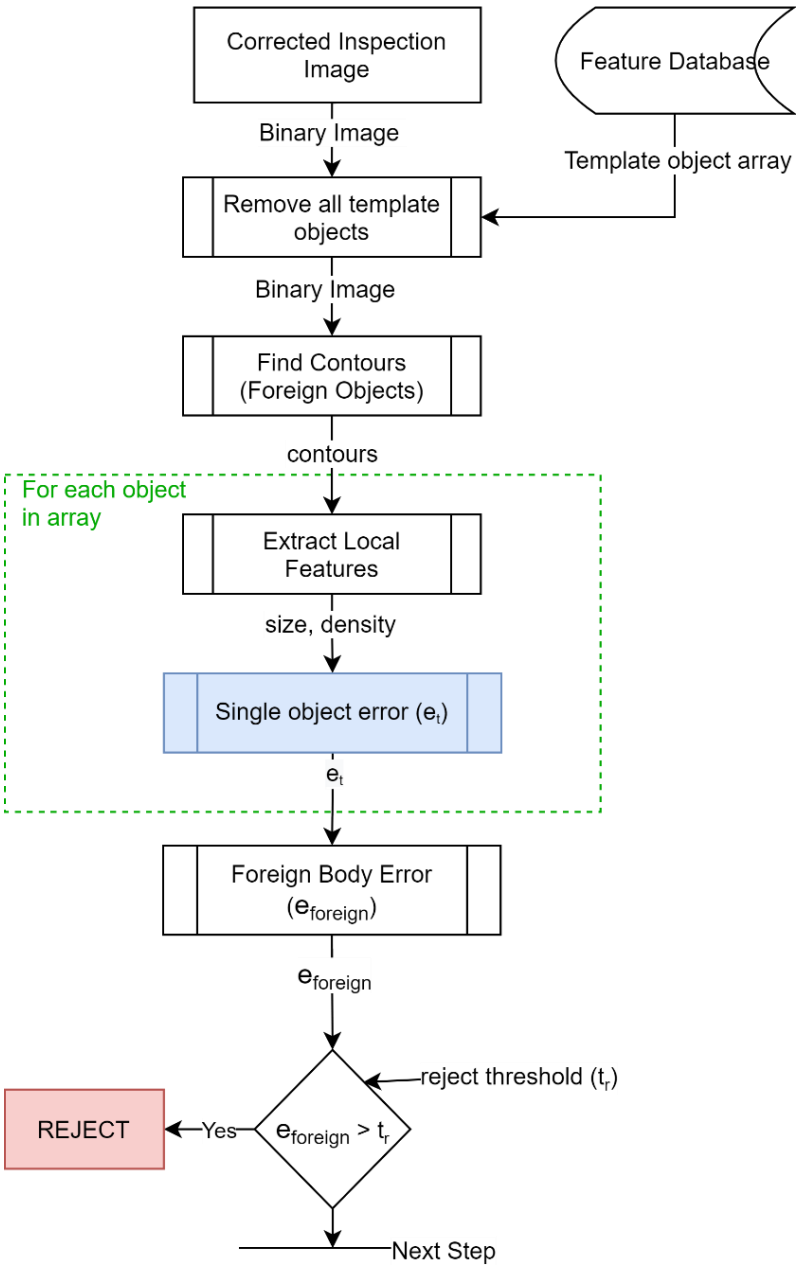


Figure 3.22: Process of calculating foreign object error

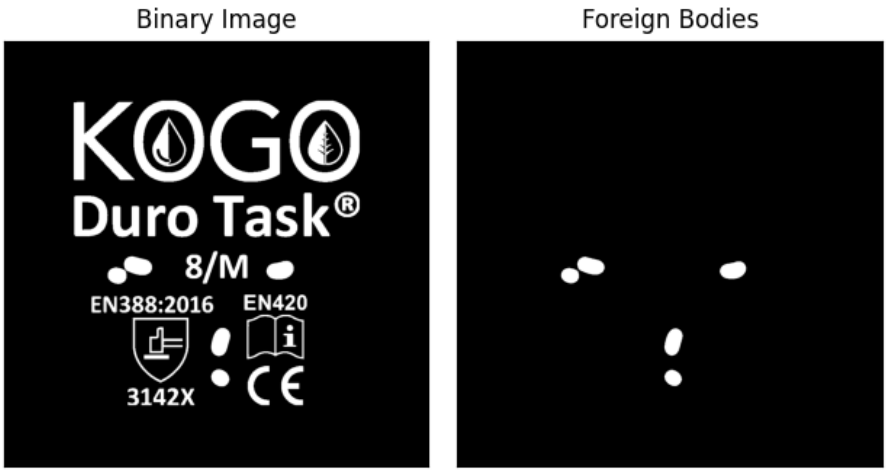


Figure 3.23: Extracted foreign bodies

First, calculate the single foreign object error (e_t) by combining size and density as in Eq. *(9),* where the $w_5 + w_6 = 1$.

$$Single\ object\ error\ (e_t) = w_5 \times s_i + w_6 \times d_i$$
$$d_i - density\ \ s_i - size$$
$$w_5, w_6 - weight\ values \qquad (9)$$

Then calculate the mean error using Eq. *(10)* and max error using Eq. *(11)*. The *N* represent the number of foreign objects.

$$Mean\ Error\ (E_{mean}) = \frac{1}{N}\sum_{i}^{N} e_t \qquad (10)$$

$$Max\ Error\ (E_{max}) = max(e_t) \qquad (11)$$

Finally, combine mean and max error as in Eq. *(12)*, where the $w_7 + w_8 = 1$. Here, the max value is used because to mitigate the information loss that happens when averaging in Eq. *(10)*. For example, suppose there is one sizeable foreign object and many small dotted like foreign objects; in that case, by calculating mean, the impact from the sizable object would fade away. Adding the max value will eliminate this issue. The consistency and validity of the model will be discussed in the evaluation section with test results. The N^{th} root of mean calculates here to introduce the notion of foreign object count that is faded away when calculating the mean.

$$Foreign\ body\ Error\ (e_{foreign}) = w_7 \times (E_{mean})^{\frac{1}{N}} + w_8 \times E_{max}$$
$$w_7, w_8 - weight\ values \qquad (12)$$

(2). <u>Calculate the total error of template objects</u>

The main concern of this subprocess is to calculate the amount of error in the legitimate objects in the artwork. The legitimate objects are the objects we extracted from the artwork in the Teaching mode. Retrieve all legitimate objects from the feature database. Following operations are applied to one object at a time to calculate the error. The process flow is shown in *Figure 3.24*.

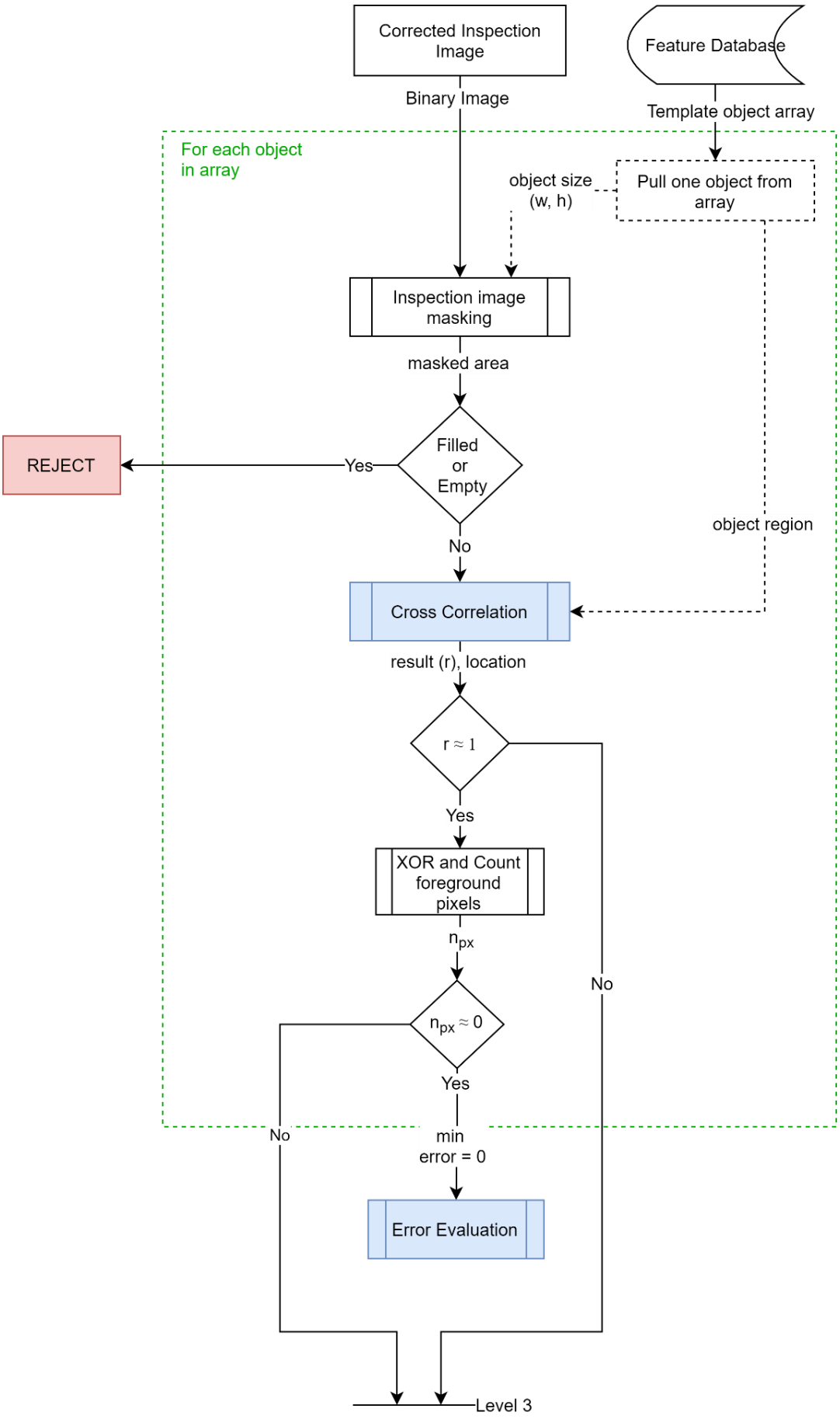


Figure 3.24: Process of calculating template object error

Pull a legitimate object from the database and obtain its bounding rectangle (x_i, y_i, w_i, h_i). Add fixed padding to the bounding rectangle to create a new rectangle.

$$(x_i', y_i', w_i', h_i') = (x_i - \alpha,\ y_i - \alpha, w_i + \alpha, h_i + \alpha) \qquad (13)$$

Then use the new rectangle (x_i', y_i', w_i', h_i') to define a region of interest (ROI) in the inspection binary image as depicted in *Figure 3.25*. In the next step, the system examines the ROI to determine either it is empty or filled, as in *Figure 3.26*. The empty ROI means that the legitimate object has not been printed on the glove. The filled ROI can mean many things, such as the object may have printed defectively, ink can be spiled, another displaced object covers the area, etc. In either situation, the system will decide to reject the glove. Otherwise, the system will move to the next step, cross-correlation, as depicted in *Figure 3.24*.

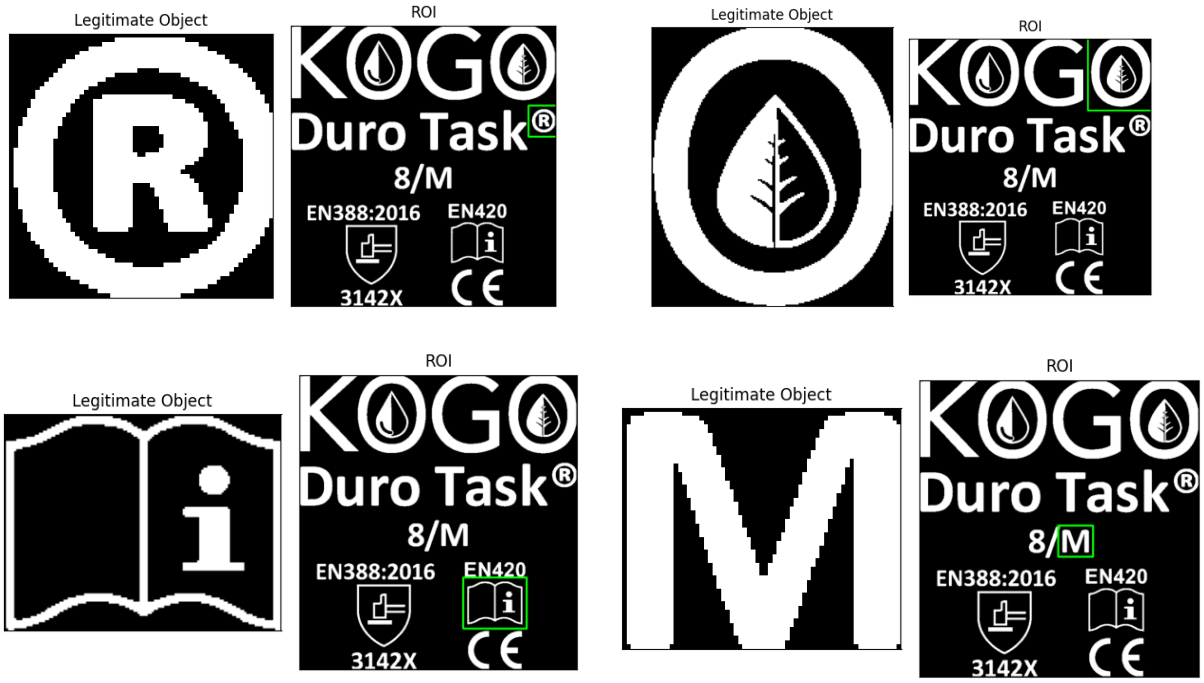


Figure 3.25: Define ROI around the Inspection object

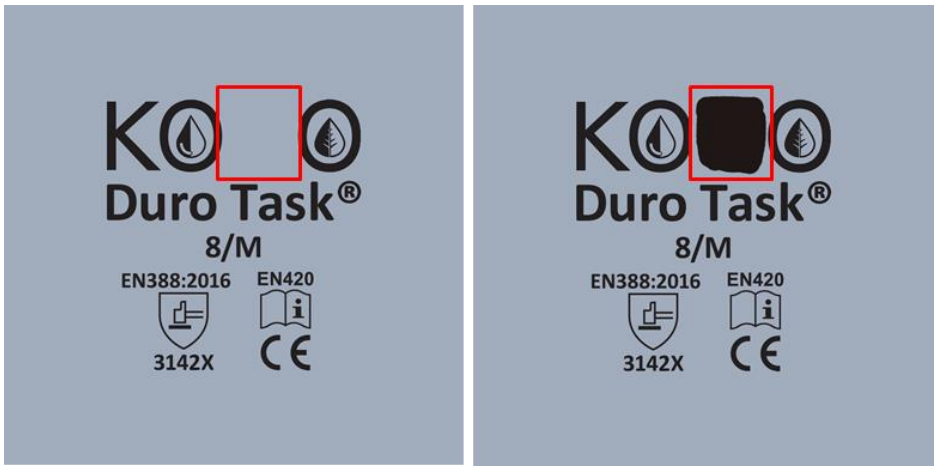


Figure 3.26: Empty or filled areas result in a missing object

In the next step, perform the normalized cross-correlation function over the inspection image ROI with the legitimate object from the template. The performance can significantly improve by applying the cross-correlation function only to the ROI than applying it to the entire image. The cross-correlation function returns the best overlapping location and value. If the resulting value is approximately 1, the two images overlap and perform the *XOR operation*. After the XOR operation, perform the morphological erode operation with a 3x3 kernel to eliminate the noisy edges. Then count the white pixels in the resulting image; if the number of white pixels is close to zero, then the inspection image object is very similar to the template image object. In such a case, set the error value for the object as *min_value (zero)* and continue to the next object. If the number of white pixels is not close to zero, then send the object to defect detection level-3.

Moving back to the cross-correlation result, if it is not close to 1, then send the object to defect detection level-3 to check for any defect.

3.3.6 Level 3 – Inspection Process

The level-3 operations are more sophisticated than previous levels; hence the time complexity is high. This level will be able to capture more delicate errors in objects. This level also inspects object by object. The Zernike moments proposed by (Khotanzad and Hong, 1990) is used for detecting the shape dissimilarity between template object and inspection object. The main steps are shown in *Figure 3.27*.

The Zernike moments are not invariant to the scale and translation. Therefore both template object and inspection object need to be normalized first. The non-pixel area is cropped out to make the object translation-invariant. Then make both template object and inspection object the same size using the bilinear interpolation, making the object scale-invariant.

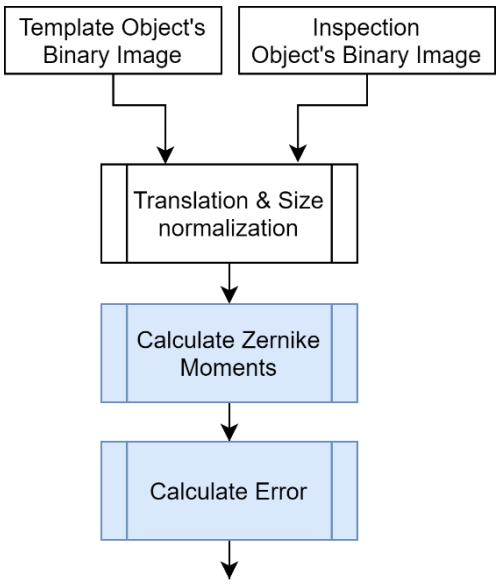


Figure 3.27: Level-3 Defect detection process

After normalization, calculate Zernike moments for both objects by providing their minimum enclosing circle's radius *(Figure 3.28)* as the disc radius to the Zernike function.

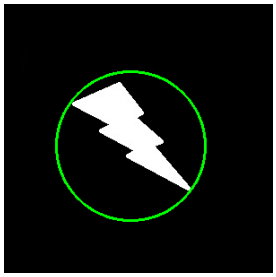


Figure 3.28: Minimum enclosing circle

Cite: https://docs.opencv.org/3.4/dd/d49/tutorial_py_contour_features.html

Then calculate dissimilarity between two shapes by measuring the sum of Euclidean distance between the first 20 Zernike moments, Eq. *(14)*.

$$Dissimilarity\ (D) = \sqrt{\sum_{i}^{20} (Z_i^T - Z_i^I)^2} \qquad (14)$$

$$Z_i^T - i^{th}\ moment\ of\ template\ object$$

$$Z_i^I - i^{th}\ moment\ of\ inspection\ object$$

The calculated dissimilarity value will be considered as the error for that object. This error will send to the error evaluation process to determine whether to accept or reject the object. The functionality of the error evaluation process is described in the following section.

The discussion of the defect inspection process ends here.

3.3.7 Error Evaluation and Decision Making

It is possible to reject or accept the glove in every defect detection level by comparing the defect with user-specified threshold values. However, in defect detection level-2 and level-3, there are some situations where the calculated error needs to be evaluated to assess the actual impact. To satisfy this purpose, the research proposes an error evaluation mechanism that takes the calculated error from level-2 or level-3 and the object's significance as inputs and outputs a decision to accept or reject the glove, as depicted in *Figure 3.29*.

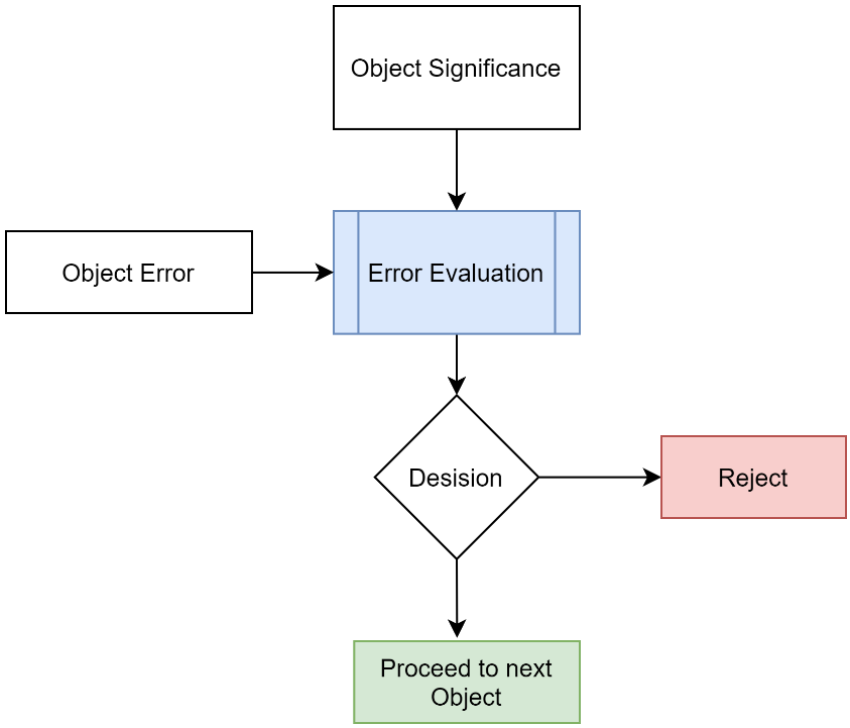


Figure 3.29: Error Evaluation

The error evaluation process comprises a decision function, which is the exponential decaying function, Eq. *(15)*. This function calculates the maximum expected error for each object with respect to its significance value. If the object's significance increases, the acceptable error that an object can hold is strictly reduced. This behavior can represent a continuous function that decays exponentially *(Figure 3.30)*.

This value is the maximum error that an object can deform if it needs to be accepted. If the calculated error of the object is beyond this value, it will be considered fatal to the glove and, therefore, decide to reject it.

$$expected\ max\ error\ (e_{max}) = \alpha e^{-\lambda f_{sig}} \qquad (15)$$

$$f_{sig}\ - object\ significance$$

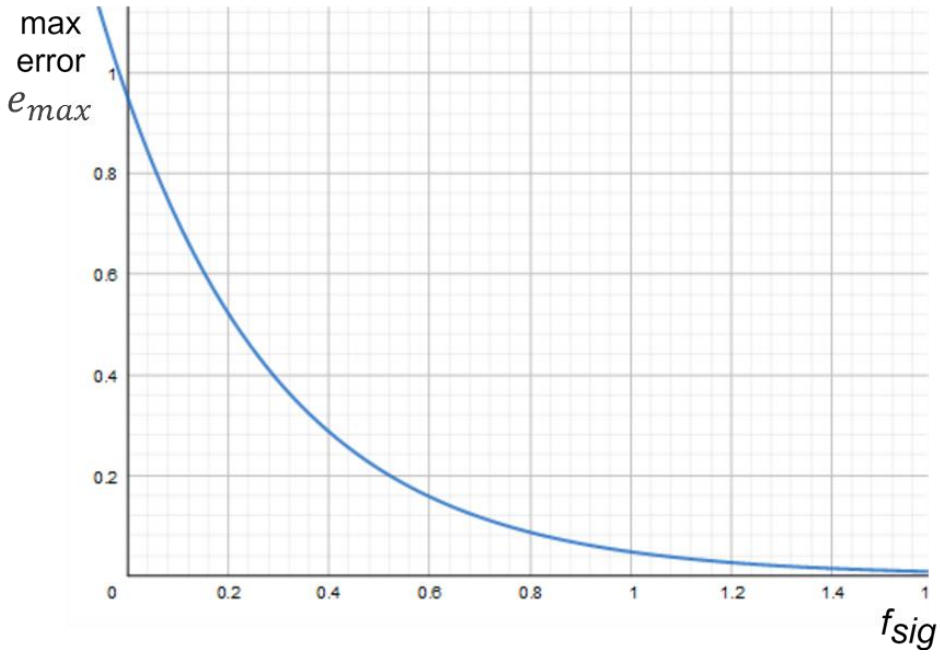


Figure 3.30: Decision function

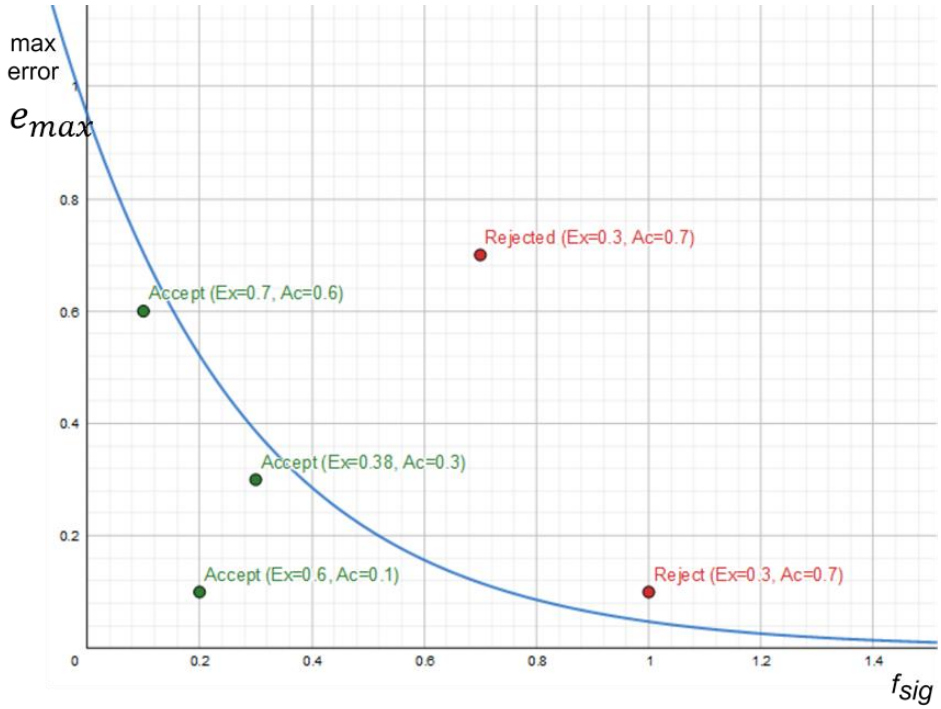


Figure 3.31: Error is higher than the expected error, reject the glove. Accept otherwise.

After the inspection complete for all objects, the glove could either be rejected or accepted. If the glove is accepted, then calculate the quality of sticker printing for that glove. The quality measurement method is explained in detail in the next section.

3.3.8 Quality Measurement Index (QMI)

At the end of the defect inspection, a quality value is calculated for each accepting glove. A mathematical model is newly proposed in this research to calculate the printing quality of the glove, where combine the error calculated in level-1, foreign body error, and the calculated error for each object in level-2 or level-3, as shown in *Figure 3.32*.

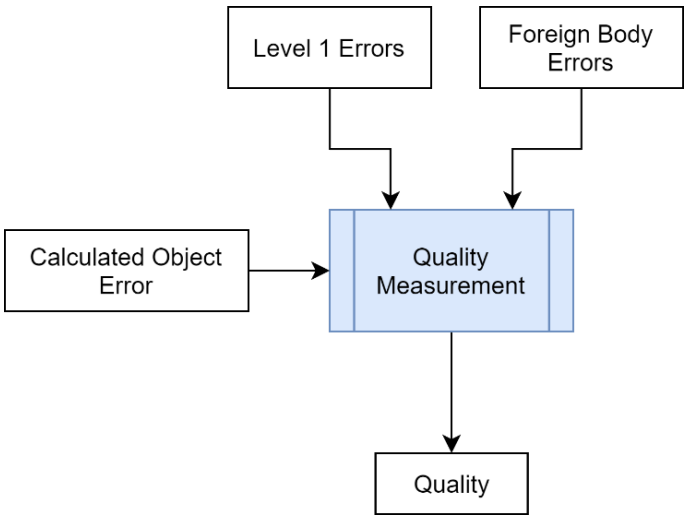


Figure 3.32: Quality Measurement

First, iterate through all objects in the sticker and filter out defective objects results from inspection. Then calculate the average error from all defective object's errors, Eq. *(16)*.

Finally, combine the result from Eq. *(16),* foreign body error, and the level-1 error with respect to their weight values to obtain the quality measure, Eq. *(17).* By changing the weight values, the influence from each factor can be changed. The consistency and validity of the mathematical model are proved experimentally in evaluation by comparing test results.

$$Avg.\ objects\ error, (E) = \frac{1}{N_d}\sum_{i}^{N_d}\left[\ \left(\frac{e_i}{e_{max}} \times f_{sig(i)}\right)\right] \qquad (16)$$

$$e_i\ - calc.\ error\ in\ level - 1\ or\ -2;\ e_{max} - expected\ \max error$$

$$f_{sig(i)} - object's\ significance;\ N_d\ - No.\ of\ defective\ objects$$

$$QMI = 1\ - \left[w_1 \times E + w_2 \times e_{foreign} + w_3 \times e_{level1}\ \right] \qquad (17)$$

$$e_{level1}\ - level\ 1\ errors$$

$$e_{foreign} - foreign\ body\ errors$$

$$w_1, w_2, w_3 - weight\ values$$

The inspection mode operations are finished after calculating the QMI value for the glove.

CHAPTER 4

EVALUATION AND RESULTS

4.1 Introduction

The proposed methodology in the previous section comprises a computer vision-based defect detection approach and several mathematical models to calculate numerical measures like visibility, significance, and QMI. In order to evaluate the consistency and validity of the proposed solution, different types of evaluation methods will be used. The overall evaluation method is experimental based on test results produced by providing images into algorithms. The test results are compared from several perspectives to prove the consistency and validity.

Since the proposed solution is based on computer vision techniques, the dataset is images. As described in the methodology, two types of images are used, template image and inspection image. The template image is the binarised artwork, and the inspection image is the binarised printed sticker. The artwork images are actual and provided by the production company.

However, the inspection images used for testing and evaluation are not the actual sticker images printed on a glove. Instead of actual printed sticker images, a set of synthetic images that are derived from manipulating the artwork are used for testing. The most common and possible defects happen when printing is artificially created by changing the particular artwork image. A series of images are obtained for evaluation by changing the magnitude of a specific type of defect. It is possible to obtain systematic results by providing synthetic defect images that are created with known magnitudes. On the other hand, if the algorithms are provided with random defective images, the results are also random. It is difficult to evaluate the consistency of a model by comparing the result obtained from random inputs. Therefore, synthetic defect images with known magnitudes better evaluate a model scientifically than randomly selected actual printed stickers on a glove.

The proposed methodology is implemented using python language for testing. A set of precompiled computer vision algorithms in OpenCV, Numpy, Scipy, Mahotas libraries are used. The test result images are generated using the matplotlib library. All the tests are performed on a computer with an Intel Core i7-8750H 2.20GHz processor and 8GB RAM.

For the evaluation, the artwork image shown in *Figure 4.1* is used. The inspection images used in this evaluation have the dimension of 1000px ×1000px. The list of different objects shown in *Figure 4.2* is used for comparison-based evaluations. More test results are presented

in APPENDIX A.

Figure 4.1: Artwork of selected sticker for the evaluation

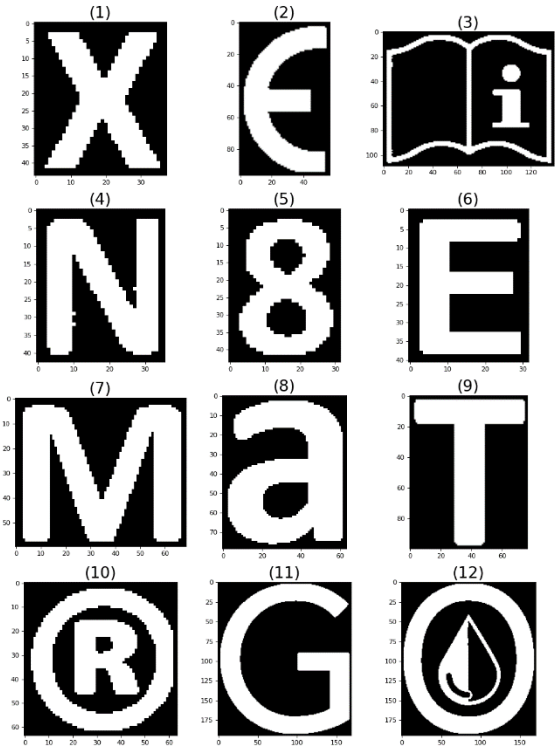


Figure 4.2: Selected artwork objects

4.2 Test Results

The test results discussed under the following sections are obtained using *Figure 4.1* artwork and the list of objects in *Figure 4.2*. The essential features of *Figure 4.2* objects required in the following sections are listed in *Table 4.1*.

Table 4.1: Object Features

Id	*Area (No. of white pixels)*	*Hull Area*
1	672	1178.5
2	2317	3811
3	2968	13305
4	757	1106
5	667	870
6	614	908.5
7	2206	3392.5
8	2929	3894
9	2717	4497.5
10	1960	2793.5
11	12201	26570
12	14468	25465.5

4.2.1 Object Visibility

This section presents the test results of object visibility measures obtained by applying Eq. *(5)* for each object in *Figure 4.2*. Here, we used $\boldsymbol{w_1}$ = 0.5, $\boldsymbol{w_2}$ = 0.5 as weight values, therefore the size and desity have equal influence on the visibility. To calculate the size using Eq. *(3),* we choose 10000 for the upper bound (A) value. The density is calculated using Eq. *(4)*, where the area and hull area values are calculated as in *Table 4.1.*

The test results of visibility measurement are listed in *Table 4.2*. Three objects (3, 11, 12) have the highest size value equals to 1.0 because their hull area is higher than the upper bound (A) value; check *Table 4.1*. Among objects-3, -11, -12, object-12 has given the highest visibility due to its high-density value compared to the other two. The object-11's visibility is slightly behind object-12 since it has low density than object-12. On the other hand, object-3's visibility is relatively low compared to object-11 and -12 since its density value is very low. Object-1 has the lowest visible value since its size and density are small. Likewise, the calculated visibility values reflect the perceptual visibility difference of each object.

Table 4.2: Test Result of Object Visibility

Id	*Size*	*Density*	*Visibility*
1	0.1179	0.5702	0.3440
2	0.3811	0.6080	0.4945
3	1.0000	0.2231	0.6115
4	0.1106	0.6844	0.3975
5	0.0870	0.7667	0.4268
6	0.0909	0.6758	0.3833
7	0.3393	0.6503	0.4948
8	0.3894	0.7522	0.5708
9	0.4498	0.6041	0.5269
10	0.2794	0.7016	0.4905
11	1.0000	0.4592	0.7296
12	1.0000	0.5681	0.7841

4.2.2 Object Significance

The object significance test results are obtained by applying Eq. *(6)* on each object in *Figure 4.2*. The weight values are chosen as $\boldsymbol{w}_3$ = 0.5, $\boldsymbol{w}_4$ = 0.5, hence the visibility and domain importance have equal influence to the significance value. The test results are shown in *Table 4.3*.

The user sets the *domain-importance* value according to the importance of the object's information to its domain. For example, object-7 is given the highest importance value since it shows the glove's size, which is the most frequently accessing customer information. The

visibility of object-7 is also moderate; therefore, it has been given the highest significance value compared to other objects. Object-1 has the lowest visibility and domain importance; therefore, its significance value is also low. *Table 4.3* shows the comparative results.

The weight values w_3, w_4 can be changed according to the requirement of the user.

Table 4.3: Test Result of Object Significance

Id	*Visibility*	*Domain Importance*	*Significance*
1	0.3440	0.2000	0.2720
2	0.4945	0.4000	0.4473
3	0.6115	0.5000	0.5558
4	0.3975	0.7000	0.5488
5	0.4268	0.7000	0.5634
6	0.3833	0.7000	0.5417
7	0.4948	0.9000	0.6974
8	0.5708	0.6000	0.5854
9	0.5269	0.6000	0.5635
10	0.4905	0.6000	0.5452
11	0.7296	0.3000	0.5148
12	0.7841	0.3000	0.5420

4.2.3 Defect Inspection

The defect detection methods in each level are tested separately by providing images containing artificially created defects. Series of images containing the different magnitudes of the same defect will be tested and compared.

Defect Inspection Level-1

Scale Factor

In level-1, the three most common defects can be detected; displacement, scaling, and rotation. The scale factor is calculated using Eq. (7) to detect scaling defects. *Figure 4.3* depicts the comparative scale-factor calculation results.

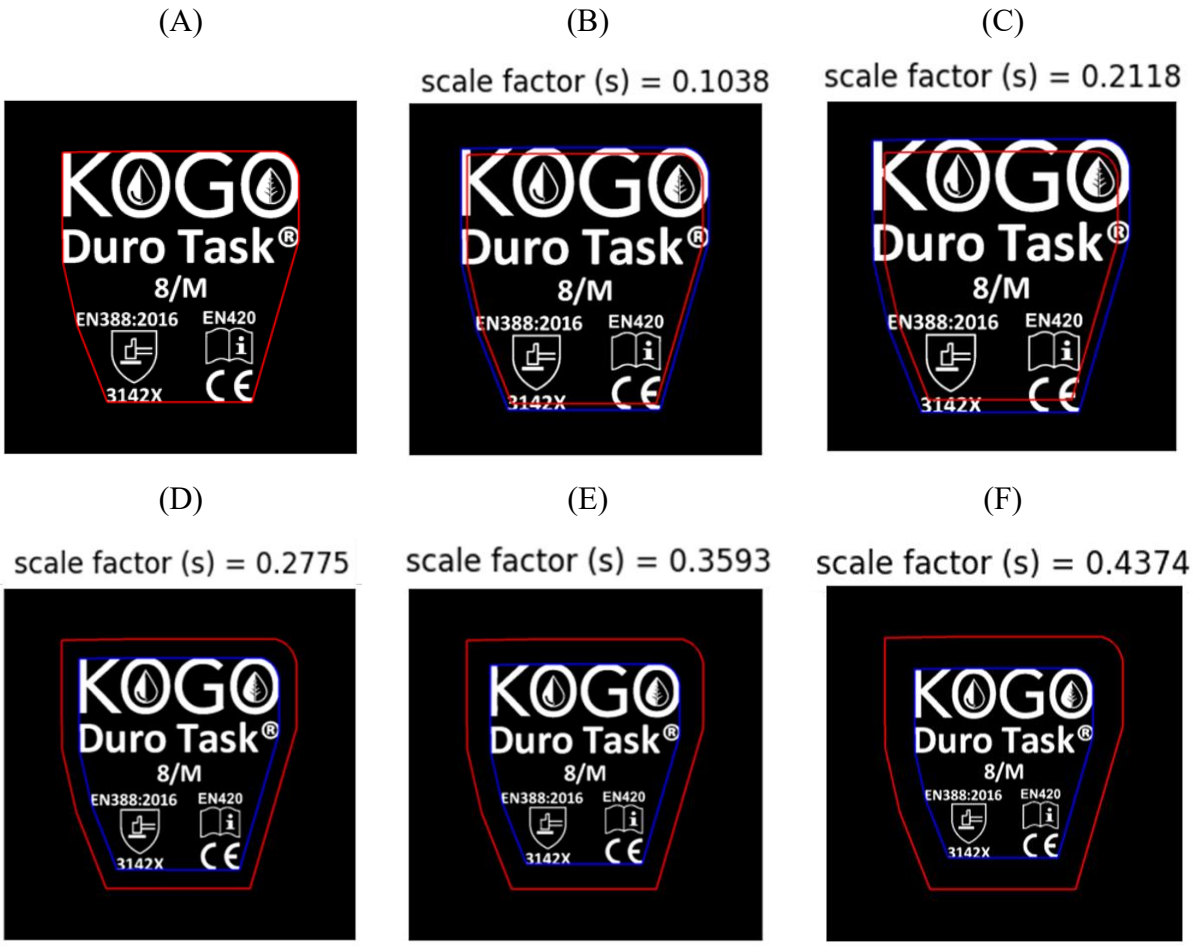


Figure 4.3: Scale factor calculation; (A) Correct Scale, (B)-(F) Scale deformations

Figure 4.3 (A) shows the original size of the sticker. The calculated template hull area (A_t) is 423924.0. *Figure 4.3 (B)-(F)* shows the five sticker images that have scaling defects. Both Image *(B)* and *(C)* are scaled-up, and Image *(D),(E),* and *(F)* is scaled-down in increasing order. The calculated scale factor is consistent with perceptual scale differences when each image is compared with the original image and with each other. *Table 4.4* shows the calculated scale factor for each image.

The rejection threshold can be set as the preference of the user. If the calculated value is greater than the threshold value, the glove will be rejected.

Table 4.4: Test result of scale factor calculation

template hull area (A_t) = 423924.0		
	inspection hull area (A_i)	*scale factor (s)*
(B)	467947.0	0.1038
(C)	513711.0	0.2118
(D)	306299.0	0.2775
(E)	271618.5	0.3593
(F)	238519.5	0.4374

Angle Difference

The rotational defects can be identified by calculating the angle difference using Eq. *(8)*. *Figure 4.4* depicts the comparative angle-difference calculation results.

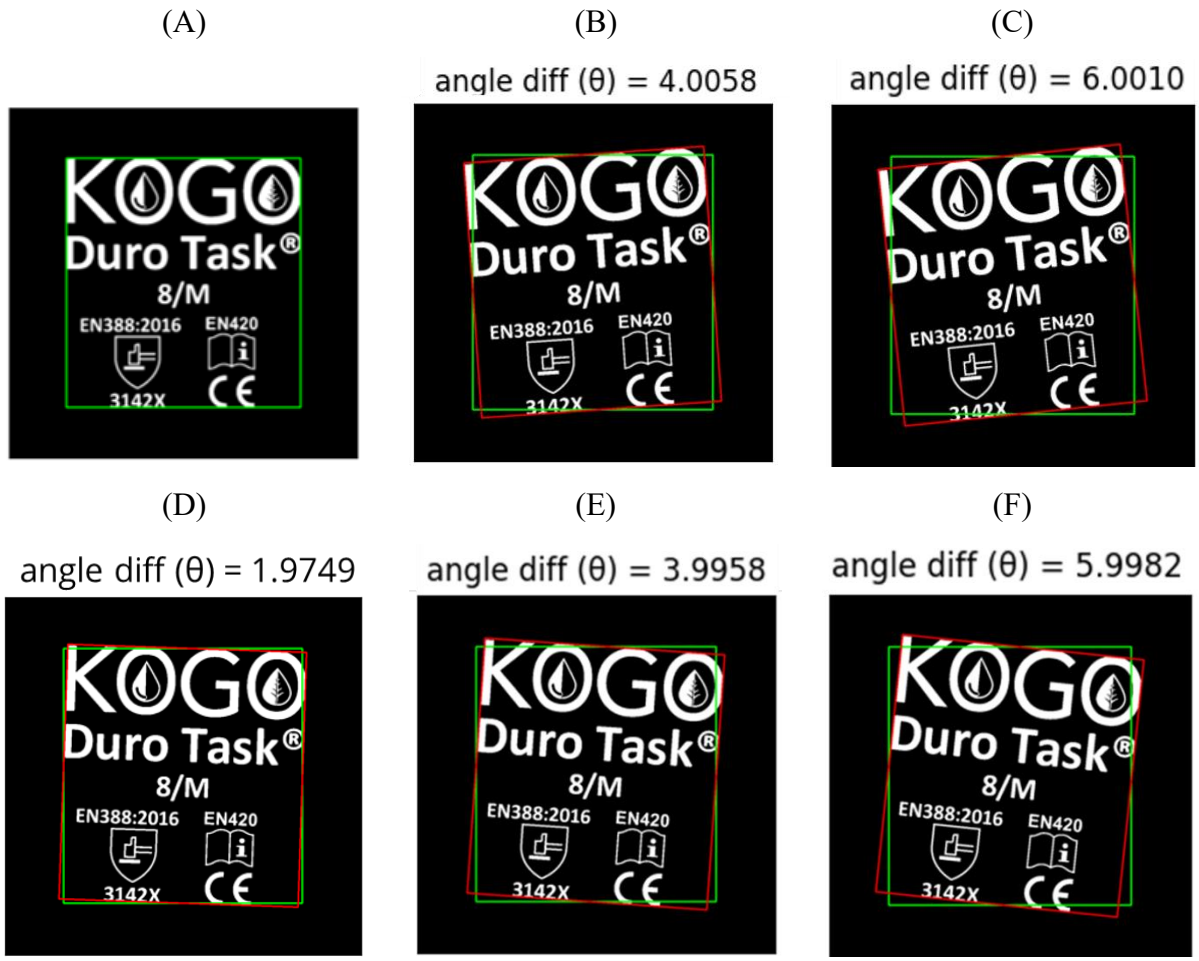


Figure 4.4: Angle-diff calculation; (A) Correct Angle, (B)-(F) Rotational deformations

Figure 4.4 (A) shows the correct orientation of the original sticker. The rotation angle of the template image (a_t) is 90^o. *Figure 4.4 (B)-(F)* shows the five sticker images that have rotational defects. Both images *(B)* and *(C)* are rotated counter-clockwise direction in 4º and 6º, respectively. Image *(D), (E),* and *(F)* are rotated clockwise in 2^o,4^o, *and* 6^o, respectively. By observation, the calculated angle-diff is approximately equal to artificially deformed values, and it is also consistent with perceptual rotational differences. *Table 4.5* shows the calculated angle difference for each image.

Here also, the rejection threshold can be set as the preference of the user. The glove will be rejected if the calculated value is greater than the threshold value.

Table 4.5: Test result of rotation diff. calculation

template rotation angle a_t = 90°		
	Inspection angle (a_i)	*angle diff (θ)*
(B)	-85.99°	4.0058
(C)	-83.99°	6.0010
(D)	88.02°	1.9749
(E)	86.00°	3.9958
(F)	84.00°	5.9982

Table 4.6 shows the approximate time taken to inspect and reject three defect types. The level-1 completes the inspection around 25ms. The testing is done in the environment described in the introduction.

Table 4.6: Performance of Level-1

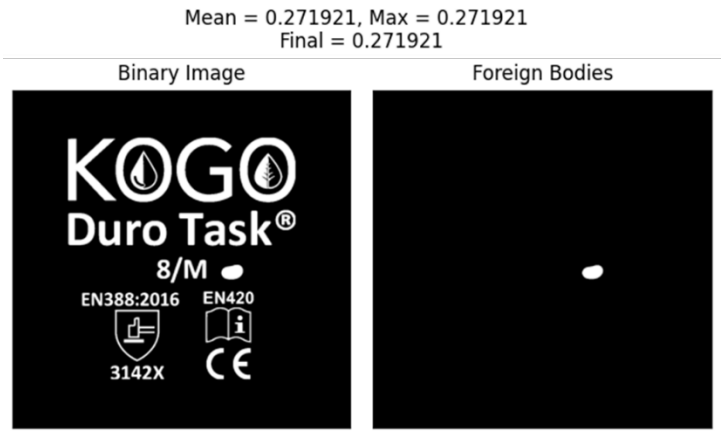
Defect Type	*Aprrox. Inspection Time (ms)*
Displacement	≈ 12 ms
Scaling	≈ 25 ms
Rotational	≈ 27 ms

Defect Inspection Level-2

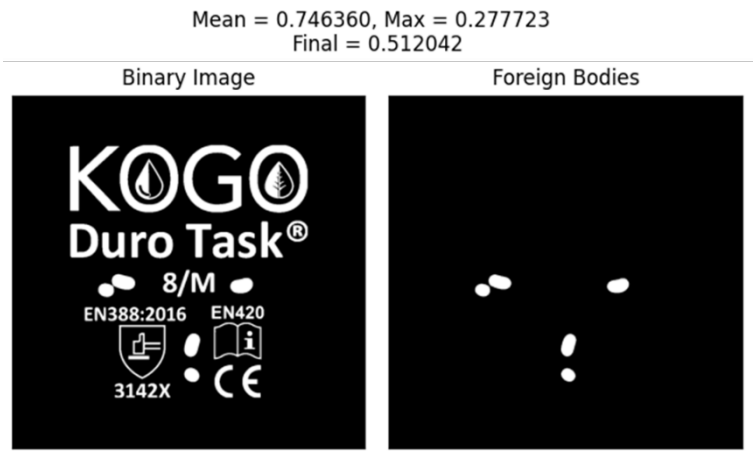
Foreign Object Errors

One primary purpose in level-2 is identifying foreign or unwanted objects and quantify the error. *Figure 4.5* shows comparative test results of foreign body error calculation using Eq. *(9),(10),(11)*, and *(12)*. The weight values in Eq. *(9)* are chosen as $\boldsymbol{w}_5 = (1 - \boldsymbol{w}_6)$, $\boldsymbol{w}_6 = \text{size}(s_i) / 2.0$, after the experiment. The weight values in Eq. *(12)* are chosen as $\boldsymbol{w}_7 = 0.5$, $\boldsymbol{w}_8 = 0.5$, hence have equal influence to the final result.

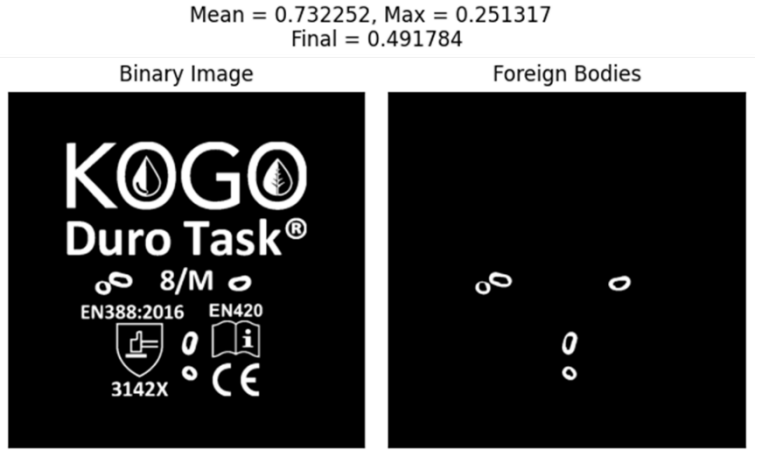
(A)



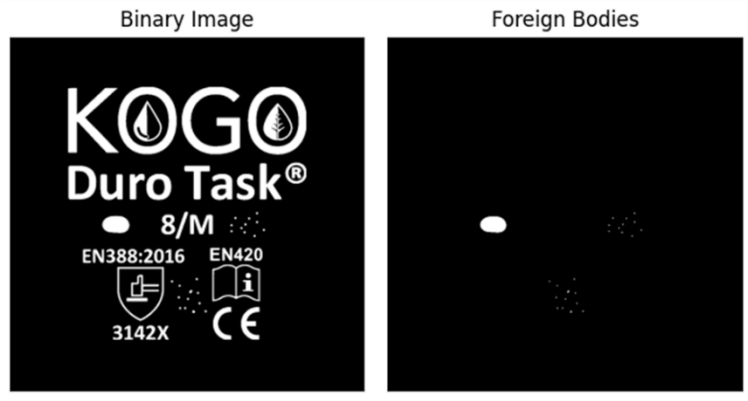
(B)


Mean = 0.746360, Max = 0.277723
Final = 0.512042
Binary Image
Foreign Bodies
KOGO
Duro Task®
8/M
EN388:2016
EN420
3142X

(C)


Mean = 0.732252, Max = 0.251317
Final = 0.491784
Binary Image
Foreign Bodies
KOGO
Duro Task®
8/M
EN388:2016
EN420
3142X

(D)


Mean = 0.397618, Max = 0.374075
Final = 0.385847
Binary Image
Foreign Bodies
KOGO
Duro Task®
8/M
EN388:2016
EN420
3142X

(E)

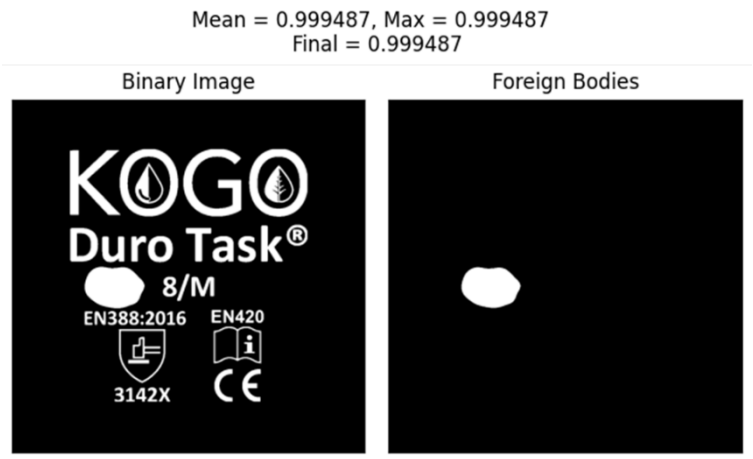


Figure 4.5: Test figures and results of foreign body errors

Image (A) contains only one foreign body, and Image (B) contains five approximately similar-sized foreign bodies. By comparing the image *(A)* and *(B),* see *Table 4.7*, the mean and max value of both images are similar, but the $(E_{mean})^{\frac{1}{N}}$ value in image *(B)* brings the notion of object count and elevates the final result. The image *(C)* contains the same foreign objects as image *(B)* but is not filled; hence, each object's density value is reduced while size remains intact. This situation is reflected in the final result as its final result is slightly smaller than the image *(B)* result. The image *(D)* contains one moderate-sized foreign object and several small objects. Since the small objects are negligible, we are not taking those for mean calculation. Image *(E)* contains one large object, and the final result reflects its impact on it.

Table 4.7: Test results of foreign body calculation

	Significant Size Objects (N)	*Mean* (E_{mean})	$(E_{mean})^{\frac{1}{N}}$	*Max* (E_{max})	*Foreign Body Error* $(e_{foreign})$
(A)	1	0.2719	0.2719	0.2719	0.2719
(B)	5	0.2316	0.7463	0.2777	0.5120
(C)	5	0.2105	0.7322	0.2513	0.4917
(D)	1	0.3976	0.3976	0.3740	0.3858
(E)	1	0.9994	0.9994	0.9994	0.9994

The user can define some rejection threshold as their requirement. If the calculated foreign body error is greater than the threshold, the glove will be rejected.

Table 4.8 shows the approximate time taken to calculate the foreign body error. The inspection time will increase as the number of objects increases.

Table 4.8: Performance of foreign body error calculation

	Aprrox. Inspection Time (ms)
(A)	≈ 2.6 ms
(B)	≈ 9 ms
(C)	≈ 9 ms
(D)	≈ 30 ms
(E)	≈ 2.6 ms

Template Object Errors

If there are no foreign bodies or no significant impact on the sticker, level-2 will look for high magnitude errors such as missing objects. As mentioned in the methodology, this is done by examining the template object area to see whether it is empty or filled out, like in *Figure 4.6*. The approximate time range is taken to find such defects is shown in *Table 4.9*.

(A) (B)

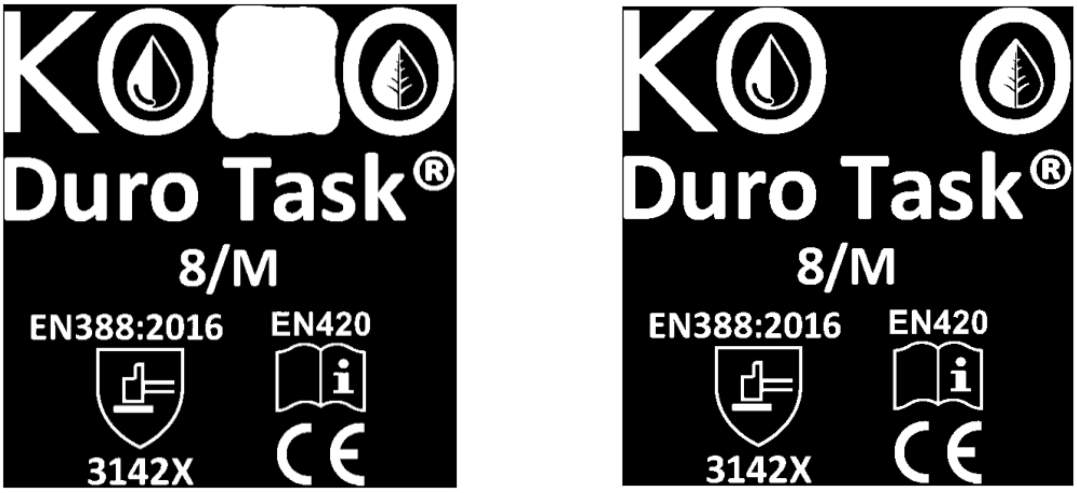


Figure 4.6: High-level object defects; (A) object missing due to inkblot, (B) object missing

Table 4.9: Performace of high-magnitude template object errors

	Aprrox. Inspection Time (ms)
(A)	≈ 5-10 ms
(B)	≈ 5-10 ms

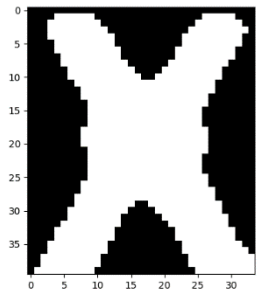
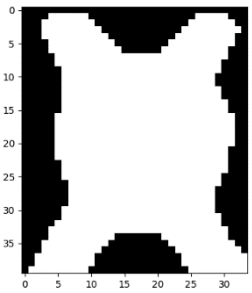
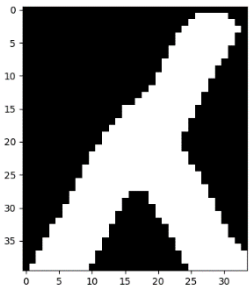
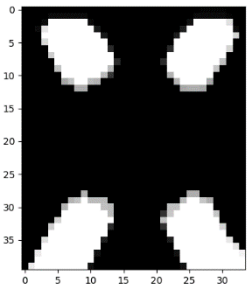
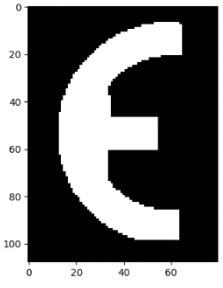
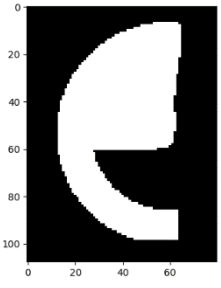
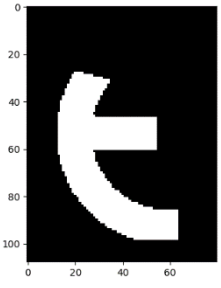
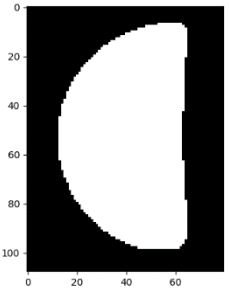
If there are no missing objects, then perform cross-correlation followed by Bitwise XOR for each object. If the similarity result is not approximately equal to 1, then send the object to Level-3 for further insection. Suppose the inspection image is very similar to the template image, and the inspection can be stoped in level-2 without going to level-3. In that case, it will take approximately 7ms to perform cross-correlation over 41 objects in the specific sticker we

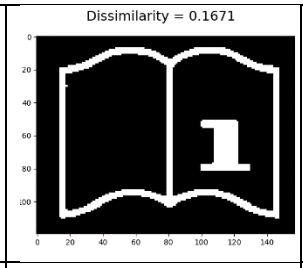
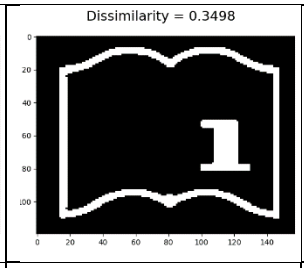
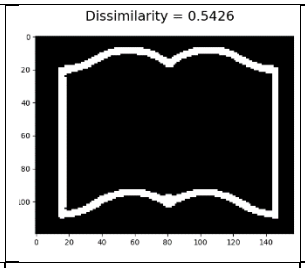
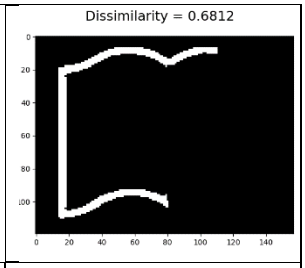
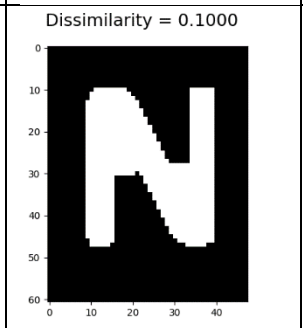
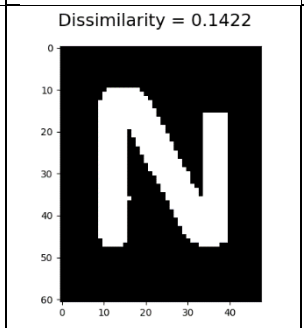
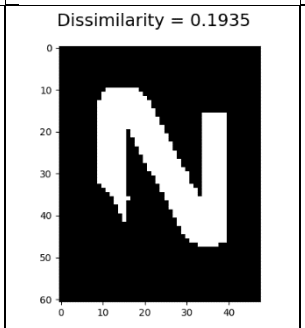
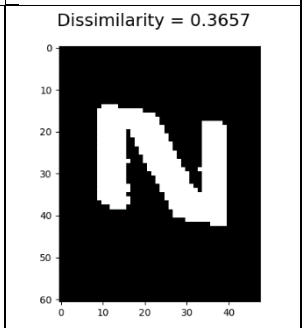
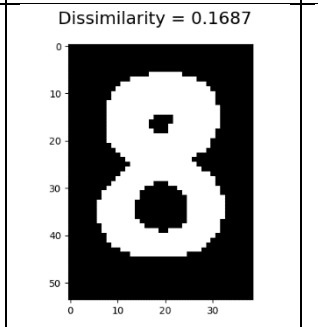
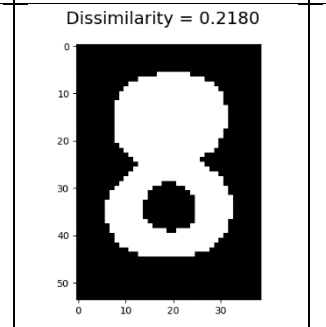
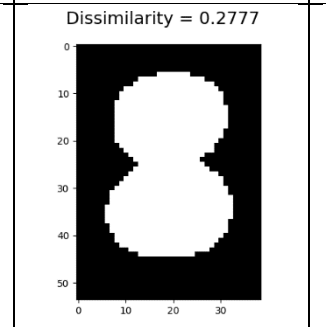
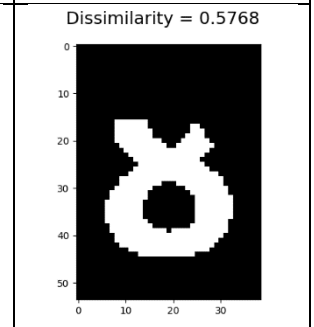
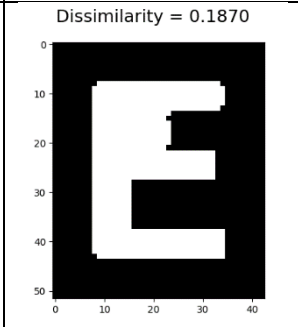
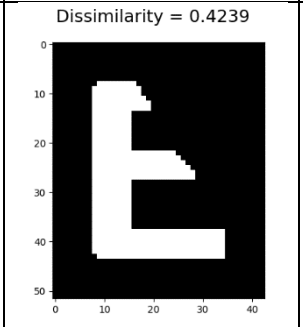
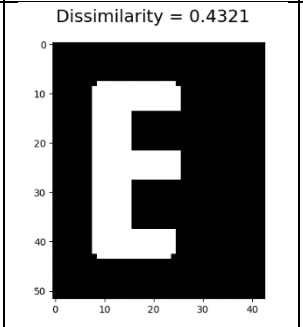
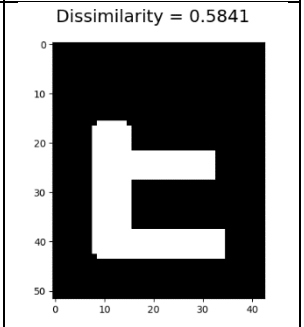
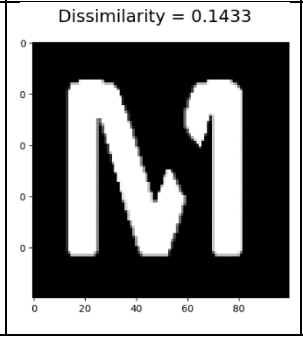
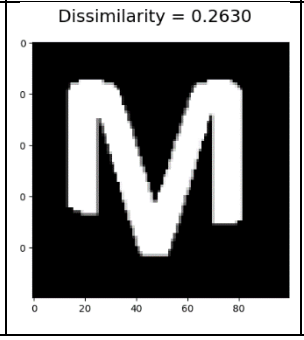
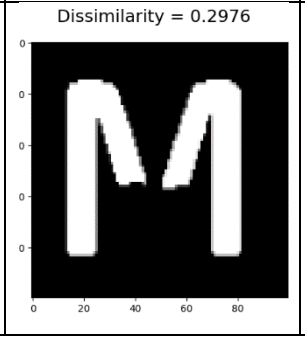
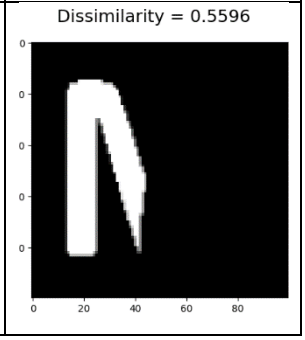
have chosen for evaluation. Naturally, the cross-correlation operation is inefficient with images that have high dimensions. As explained in the methodology, the cross-correlation operation performs only within ROI around the object rather than the whole image, which significantly reduces time.

Defect Inspection Level-3

The inspection process proceeds to level-3 if it cannot decide about a particular object in level-2. Level-3 uses Zernike moments to measure the dissimilarity between template object and inspection object. The comparison table in *Figure 4.7* shows the dissimilarities calculated in level-3 for each object in *Figure 4.2,* where one row contains four different defects of a single object in different magnitudes. The dissimilarity values are shown at the top of each figure. By comparison, the dissimilarity values in the same row have some level of consistency. The dissimilarity value does not reflect the actual magnitude of the defect in some objects; see the figure in the 2nd row and 2nd column. It is also inconsistent with perceptual dissimilarity in some situations; see the figure in the 12th row and 3rd column. Moreover, the dissimilarity value is not consistent across different objects; two different objects with the same dissimilarity value might not have the same perceptual dissimilarity.

Therefore, according to the test results, the Zernike moments with Euclidean distance measure alone are insufficient to measure reliable perceptual dissimilarity.

	A	B	C	D
1	Dissimilarity = 0.1968 	Dissimilarity = 0.3774 	Dissimilarity = 0.7064 	Dissimilarity = 0.8440 
2	Dissimilarity = 0.0878 	Dissimilarity = 0.2629 	Dissimilarity = 0.4517 	Dissimilarity = 0.5828 

3	 Dissimilarity = 0.1671	 Dissimilarity = 0.3498	 Dissimilarity = 0.5426	 Dissimilarity = 0.6812
4	 Dissimilarity = 0.1000	 Dissimilarity = 0.1422	 Dissimilarity = 0.1935	 Dissimilarity = 0.3657
5	 Dissimilarity = 0.1687	 Dissimilarity = 0.2180	 Dissimilarity = 0.2777	 Dissimilarity = 0.5768
6	 Dissimilarity = 0.1870	 Dissimilarity = 0.4239	 Dissimilarity = 0.4321	 Dissimilarity = 0.5841
7	 Dissimilarity = 0.1433	 Dissimilarity = 0.2630	 Dissimilarity = 0.2976	 Dissimilarity = 0.5596

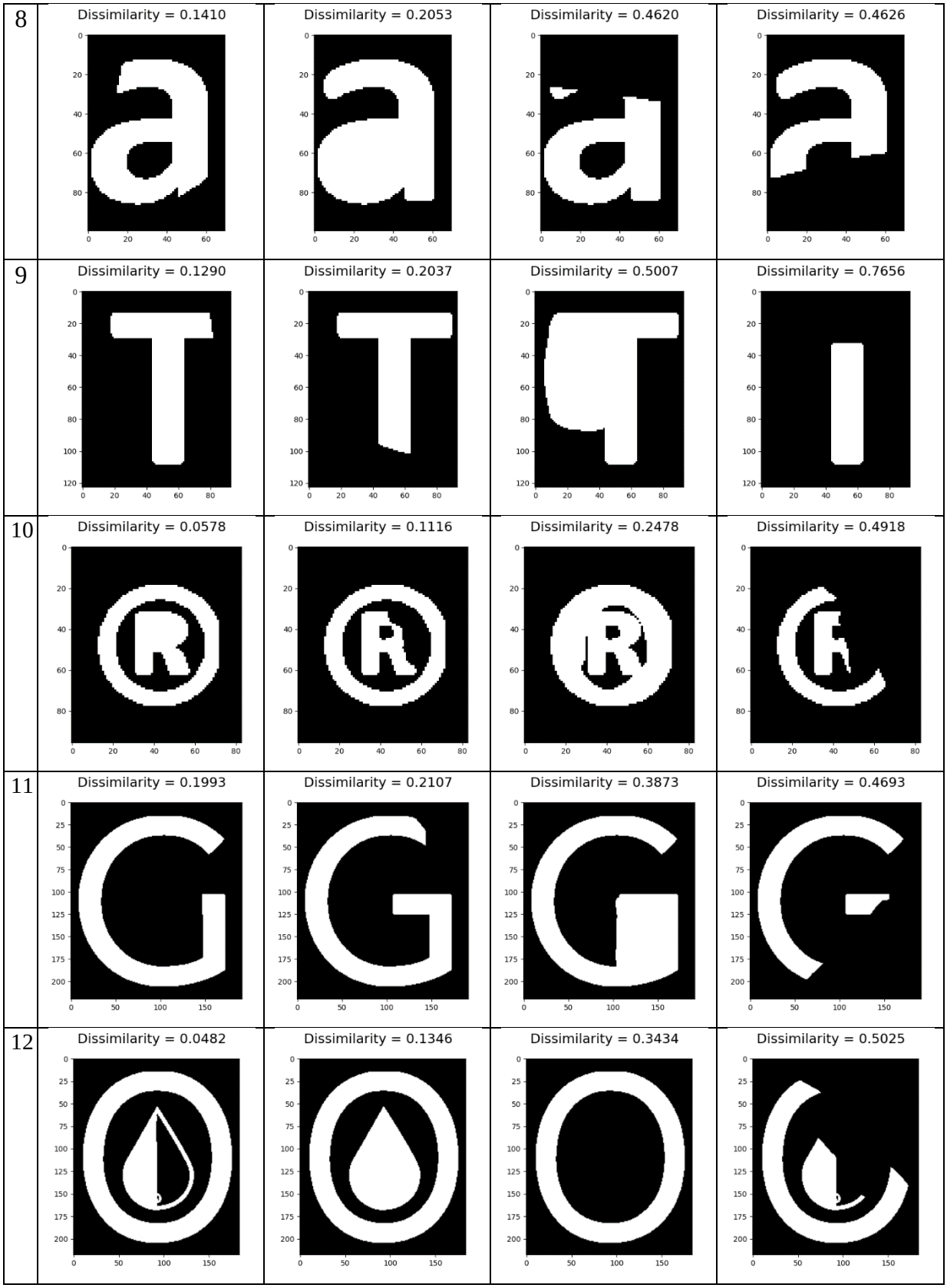


Figure 4.7: Level-3 defect detection test results

4.2.4 Maximum Expected Error

The maximum expected error for each object is calculated using Eq. *(15)*. The input parameters were chosen as $\alpha = 0.9$ and $\lambda = 2.5$. *Figure 4.8* shows the resulting graph of the function. According to the graph, even if the significance of an object is 0, we cannot expect the maximum error of 1. That is true because every object in the sticker has some value; therefore, it must be present in the final output at least recognizable. On the other hand, even if the object's significance is very high, like close to 1, we cannot expect it to have 0 error because it is impossible to print exactly as same as the artwork every time.

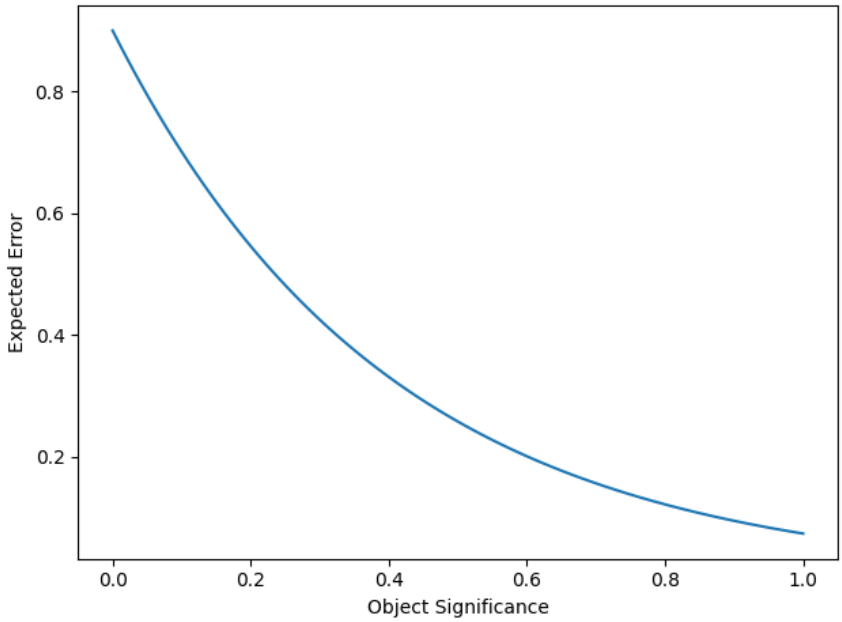


Figure 4.8: Graph of the decision function; α=0.9 and λ=2.5

Table 4.10 depicts the calculated maximum expected error for each object in *Figure 4.2*. The object-7 is the most significant object in the list; hence the maximum error we can expect from it is low. Object-1 can tolerate relatively high error since it is a less significant object compared to others.

Table 4.10: Test results of maximum expected error calculation

Id	*Significance*	*Expected Error*
1	0.2720	0.4559
2	0.4473	0.2942
3	0.5558	0.2243
4	0.5488	0.2283
5	0.5634	0.2200
6	0.5417	0.2323
7	0.6974	0.1574
8	0.5854	0.2083
9	0.5635	0.2200
10	0.5452	0.2303
11	0.5148	0.2485
12	0.5420	0.2321

The decision function decided after evaluating the calculated error in *Figure 4.7* is listed in *Table 4.11*.

Table 4.11: Decision Results

Id	*A*	*B*	*C*	*D*
1	a	a	r	r
2	a	a	r	r
3	a	r	r	r
4	a	a	a	r
5	a	a	r	r
6	a	r	r	r
7	a	r	r	r
8	a	a	r	r
9	a	a	r	r
10	a	a	r	r
11	a	a	r	r
12	a	a	r	r

4.2.5 Quality Measurement Index (QMI)

The sticker printing quality is calculated for every accepting glove using Eq. *(17)*. Here we calculate the quality value for each sticker shown in *Figure 4.9*.

Figure 4.9: Test figures of quality measurement

Table 4.12 lists down the quality measurement result obtained from *Figure 4.9*.

Table 4.12: Test results of quality values

	Quality
A	97.94%
B	89.96%
C	91.00%
D	95.51%
E	90.32%
F	90.85%
G	82.87%
H	78.00%

In *Figure 4.9 (A),* only one object has a minor defect, and that object has moderate significance around 0.5; hence less impact from that defect, and the quality is high. *Figure 4.9 (B)* shows two defects in two different objects, and their defect magnitude is higher than *(A)*. Both objects have significance around 0.5 as in *(A)*; however, quality is given as 89.96% due to the defect magnitude.

Figure 4.9 (C) contains minor defects in many objects. Since they are minor defects, they do not affect the readability of the object; thus, the quality is high. *Figure 4.9 (D)* has a minor defect in a single object as in *(A)*, but the significance of the defective object in *(D)* is higher than the defective object in *(A)*. Therefore, the quality of the *(D)* is less than *(A)*, even if their defect magnitudes are similar. The defect in *Figure 4.9 (E)* can be identified as a high magnitude defect since the object is barely recognizable. However, the object has the least significance; hence impact from the defect is low, and a high-quality value is given. *Figure 4.9 (F)* only contains foreign body defects, and other legitimate objects are intact; therefore, the quality value is 90.32%. *Figure 4.9 (G), (H)* have multiple defects; therefore, the quality value is very low.

4.3 Evaluation

The defect detection framework and its operations proposed in the methodology are thoroughly tested, and the results are presented and explained in the above section.

As test results have clearly shown, the object visibility measure is valid and consistent with perceptual visibility. The significance measuring model is also giving valid and consistent results.

According to the tests performed in the specified computer, the overall inspection process

takes about 300ms to complete the worst-case scenario. For high magnitude defects, inspection time is significantly reduced using the leveled approach. Therefore the performance of the proposed framework is high and undoubtedly real-time.

The defect detection level-1 and -2 can detect almost all defects as intended. However, the level-3 algorithm based on Zernike moments has given inconsistent results in some instances. Therefore, the Zernike moments with euclidean distance dissimilarity measurement alone are not accurate enough. This can be clearly seen in the comparison table in *Figure 4.7*. The level-3 algorithm is required to be upgraded in future research.

The test result shows that the decision function accurately models the requirement and works as intended. The function's output can be changed and fine-tuned as the user's requirements by changing the parameters.

The quality measure is also valid and consistent according to test results. However, the consistency of quality measurement depends on the results of defect detection algorithms, mainly level-3 algorithms. Since the level-3 algorithms shown inconsistencies in some situations, the quality value might also be inconsistent.

An overall, the defect detection framework acceptably achieves all objectives.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Introduction

This chapter wraps up this document by providing the final quest comments, thoughts, and future works to extend this study. This research proposed an automatic and real-time defect detection framework for the glove manufacturing industry that can detect certain sticker printing defects. We proposed a model to measure the significance of each object in the glove sticker by combining object visibility and domain importance specified by the user. The object's visibility is measured by combining the size and density. We introduce a decision function that calculates the maximum expected error for each object using the object's significance, which can then be used to decide whether to accept or reject the glove. Finally, measure the quality of the printed sticker of each accepting glove.

5.2 Findings, Contribution & Limitations

The first objective of creating an automatic and real-time process and the second objective of identifying the defects in an early stage is satisfied successfully. The time-consuming operations are done in the teaching mode and store result in a database for later use. The almost all the operations performed on artwork image is done at the teaching mode. Moreover, the calculation of visibility, significance, and maximum expected error for each object is done in the teaching mode. Therefore the time for the inspection process has significantly been reduced.

Since the defect detection framework is broken down into several levels, it could defect high magnitude errors in the early stage, therefore output the decision as early as possible. This also reduces the inspection time of a particular sticker hence the production bottleneck.

The level-1 and level-2 defect detection algorithms can detect all types of defects that are assigned to them. However, the level-3 algorithm based on Zernike moments produces inconsistent results in some cases. Therefore level-3 needed to be upgraded with a more sophisticated algorithm.

The evaluation of the defect detection algorithm is done with the synthesized defect images. Artwork images were used to construct a series of artificial defect images that are as near to the natural defect as possible. The use of synthetic images with established defect magnitudes was expected to produce more systematic findings. In most cases, the result produced by the

algorithm is shown to be consistent with the artificial magnitude of the defect on the synthetic image.

In the worst-case scenario, the defect inspection process takes approx—300ms to complete the inspection. In the best-case scenario, such as a high magnitude defect, the defect inspection process takes approx. 30ms. Therefore the framework can be considered as real-time.

In this research, we consider the size and density of the object to calculate the object's visibility. The properties of human visibility factors are not well understood yet. However, many other aspects could affect the visibility of a printed sticker, such as contrast, surrounding obstacles, shape simplicity, and complexity. In order to make the model consistent and straightforward, we only use size and density factors in this research. The evaluation of the object visibility model shows that it is consistent with perceptual visibility. Nevertheless, the weight values can be more fine-tuned as relevant to the domain.

The consistency of the object significance measurement is shown in the evaluation. However, the weight values can be fine-tuned to change the influence of domain importance and visibility to satisfy the user’s requirement.

The evaluation of the quality measurement shows that it is valid with perceptual defect magnitudes and their impact. The comparison table shows that it is consistent in the range of different defects.

Even if this research is built around a requirement of the glove sticker printing industry, the technique proposed here can be easily incorporated into defect detection in other printing-like domains containing the 2D arbitrary shapes that comply with the constraints in the proposed solution.

5.3 Future Work

The object visibility model can be improved by incorporating more visibility factors. The contrast factor is essential when recognizing an object from its background. The contrast factor is unimportant in this domain since the contrast is chosen to be high and uniform throughout the image. However, if this model is used in another domain, the contrast factor will be required. The simplicity of an object is also essential for visibility because a defect in a simple shape notice quickly than a defect in a complex shape. The complexity of an object can be measure using its convexity, Euler number, etc. The number of obstacles around an

object and how close those obstacles are to the object could determine the visibility. This can quantify by making the particular object the center and examine the circular area with some known radius.

The defect detection algorithm is implemented using Zernike moments and uses Euclidean distance to measure dissimilarity. However, this algorithm has given inconsistent results in the evaluation. In order to make the algorithm robust and consistent with perceptual dissimilarity, we can still use Zernike moments with other high-order statistical methods like Kurtosis and Skewness. Also, in order to make the dissimilarity measure more accurate, other distance calculation methods can be tested on Zernike moments, such as cosine, city block, Minkowski, etc.

Moreover, the defect detection algorithms needed to be tested by collecting actual gloves images from production houses.

REFERENCES

Bribiesca, E., Wilson, R., 1997. A Measure of 2D Shape-of-Object Dissimilarity. https://doi.org/10.1016/S0893-9659(97)00113-4

Burger, W., Burge, M., 2009. Principles of digital image processing. Core algorithms. Springer.

Castro-Ortega, R., Toxqui-Quitl, C., Padilla-Vivanco, A., Solís-Villarreal, J.F., Orozco-Guillén, E.E., 2019. Zernike moment invariants for hand vein pattern description from raw biometric data. JEI 28, 053019. https://doi.org/10.1117/1.JEI.28.5.053019

Fu, Z., Liang, F., Yu, Z., Zhou, K., 2018. A Moment-Based Shape Similarity Measurement for Areal Entities in Geographical Vector Data. ISPRS International Journal of Geo-Information 7, 208. https://doi.org/10.3390/ijgi7060208

Huang, Z., Leng, J., 2010. Analysis of Hu's moment invariants on image scaling and rotation. Presented at the Proc. of 2nd International Conference on Computer Engineering and Technology (ICCET, pp. V7-476. https://doi.org/10.1109/ICCET.2010.5485542

Hwang, S.-K., Kim, W.-Y., 2006. A novel approach to the fast computation of Zernike moments. Pattern Recognition 39, 2065–2076. https://doi.org/10.1016/j.patcog.2006.03.004

Khotanzad, A., Hong, Y.H., 1990. Invariant image recognition by Zernike moments. IEEE Transactions on Pattern Analysis and Machine Intelligence 12, 489–497. https://doi.org/10.1109/34.55109

Kim, H.-K., Kim, J.-D., Sim, D.-G., Oh, D.-I., 2000. A modified Zernike moment shape descriptor invariant to translation, rotation and scale for similarity-based image retrieval, in: 2000 IEEE International Conference on Multimedia and Expo. ICME2000. Proceedings. Latest Advances in the Fast Changing World of Multimedia (Cat. No.00TH8532). Presented at the 2000 IEEE International Conference on Multimedia and Expo. ICME2000. Proceedings. Latest Advances in the Fast Changing World of Multimedia (Cat. No.00TH8532), pp. 307–310 vol.1. https://doi.org/10.1109/ICME.2000.869602

Ming-Kuei Hu, 1962. Visual pattern recognition by moment invariants. IRE Transactions on Information Theory 8, 179–187. https://doi.org/10.1109/TIT.1962.1057692

Otsu, N., 1979. A Threshold Selection Method from Gray-Level Histograms. IEEE Transactions on Systems, Man, and Cybernetics 9, 62–66. https://doi.org/10.1109/TSMC.1979.4310076

ou, Y., baoping, G., tao, H., xuan, G., 2007. A Real-Time Vision System for Defect Detection in Printed Matter and Its Key Technologies, in: 2007 2nd IEEE Conference on Industrial Electronics and Applications. Presented at the 2007 2nd IEEE Conference on Industrial Electronics and Applications, pp. 2157–2161. https://doi.org/10.1109/ICIEA.2007.4318792

Premaratne, P., Premaratne, M., 2014. Image matching using moment invariants. Neurocomputing 137, 65–70. https://doi.org/10.1016/j.neucom.2013.02.058

Shnain, N., Aljanabi, M., Hussain, Z., Lu, S., 2018. High Order Statistic - Zernike Approach for Image Similarity and Face Recognition. International Journal of Computer Science and Information Security, 16.

Shnain, N., Lu, S., Hussain, Z., 2017. HOS image similarity measure for human face recognition. pp. 1621–1625. https://doi.org/10.1109/CompComm.2017.8322814

Singh, C., Walia, E., Upneja, R., 2013. Accurate calculation of Zernike moments. Information Sciences 233, 255–275. https://doi.org/10.1016/j.ins.2013.01.012

Sklansky, J., 1982. Finding the convex hull of a simple polygon. Pattern Recognition Letters 1, 79–83. https://doi.org/10.1016/0167-8655(82)90016-2

Suzuki, S., Abe, K., 1985. Topological structural analysis of digitized binary images by border following [WWW Document]. undefined. URL /paper/Topological-structural-analysis-of-digitized-binary-Suzuki-Abe/cf021db5e811fd5b67ee3aa4db0a6a0351d276d2 (accessed 2.4.21).

Tu, Z., Yuille, A.L., 2004. Shape Matching and Recognition – Using Generative Models and Informative Features, in: Pajdla, T., Matas, J. (Eds.), Computer Vision - ECCV 2004, Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 195–209. https://doi.org/10.1007/978-3-540-24672-5_16

Wang, W., Jiang, Z., Ping, Z., Wei, C., Xuan, Z., 2014. Color printed image defect detection based on the image feature match. BioTechnology: An Indian Journal 10.

Wang, Z., Bovik, A., Sheikh, H., Simoncelli, E., 2004. Image Quality Assessment: From Error Visibility to Structural Similarity. Image Processing, IEEE Transactions on 13, 600–612. https://doi.org/10.1109/TIP.2003.819861

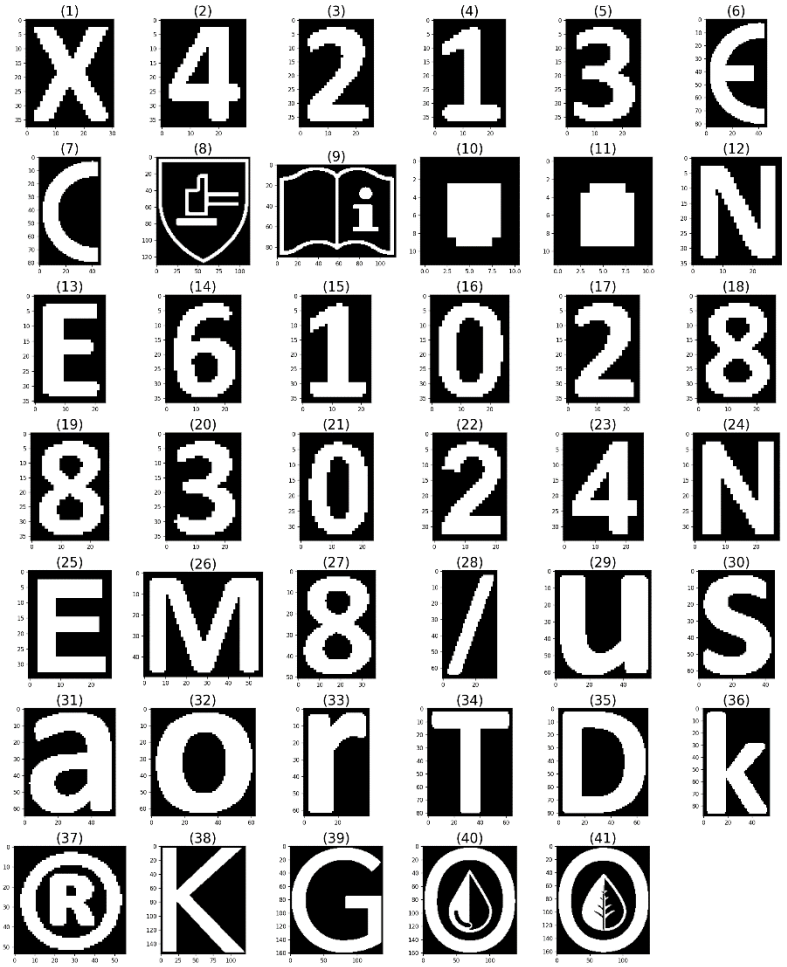
Yangping, W., Shaowei, X., Zhengping, Z., Yue, S., Zhenghai, Z., 2018. Real-time Defect Detection Method for Printed Images Based on Grayscale and Gradient Differences. Journal of Engineering Science and Technology Review 11, 180–188. https://doi.org/10.25103/jestr.111.22

Zhou, M., Wang, G., Wang, J., Hui, C., Yang, W., 2017. Defect detection of printing images on cans based on SSIM and chromatism, in: 2017 3rd IEEE International Conference on Computer and Communications (ICCC). Presented at the 2017 3rd IEEE International Conference on Computer and Communications (ICCC), pp. 2127–2131. https://doi.org/10.1109/CompComm.2017.8322912

APPENDICES

APPENDIX A: Test Results

Template A.1



Object Features

Id	*Area (No. of white pixels)*	*Hull Area*
1	459	799.5
2	438	544
3	416	648.5
4	326	527
5	409	616
6	1582	2565.5
7	1302	2551
8	2927	10989.5
9	2117	8967
10	40	29
11	40	29
12	508	719
13	374	524
14	406	545.5
15	306	452.5
16	423	568
17	368	555.5
18	450	581
19	454	579
20	363	554.5
21	349	447.5
22	329	493.5
23	322	387
24	458	637.5
25	420	596.5
26	1445	2241.5
27	938	1243
28	540	523
29	2008	2877
30	1498	2144.5
31	1986	2642
32	2091	2831
33	1131	1462
34	1807	3032.5
35	2958	4385
36	2323	3497.5
37	1373	1910.5
38	7066	16941.5
39	8231	17847
40	9768	17176.5
41	10361	17353.5

Test Result of Object Visibility

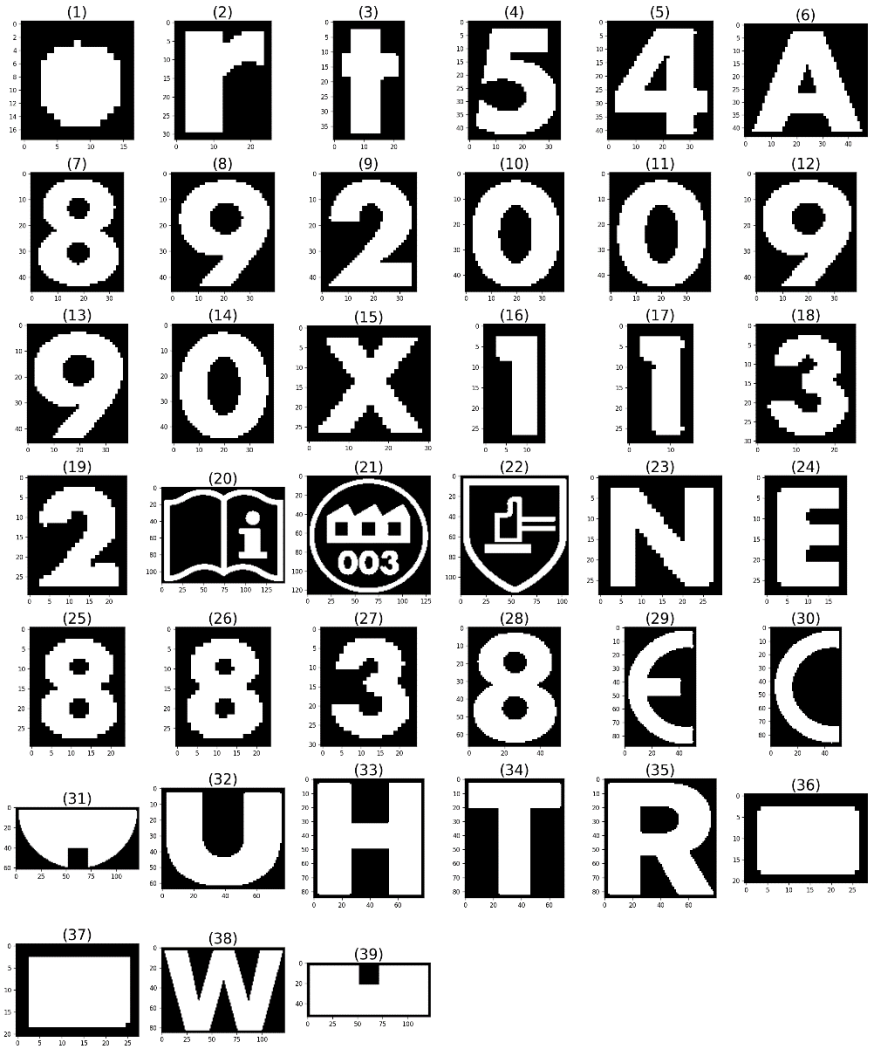
Id	*Size*	*Density*	*Visibility*
1	0.07995	0.574109	0.327029
2	0.0544	0.805147	0.429774
3	0.06485	0.64148	0.353165
4	0.0527	0.618596	0.335648
5	0.0616	0.663961	0.362781
6	0.25655	0.616644	0.436597
7	0.2551	0.510388	0.382744
8	1	0.266345	0.633173
9	0.8967	0.236088	0.566394
10	0.0029	1	0.50145
11	0.0029	1	0.50145
12	0.0719	0.706537	0.389218
13	0.0524	0.71374	0.38307
14	0.05455	0.744271	0.399411
15	0.04525	0.676243	0.360747
16	0.0568	0.744718	0.400759
17	0.05555	0.662466	0.359008
18	0.0581	0.774527	0.416313
19	0.0579	0.784111	0.421005
20	0.05545	0.654644	0.355047
21	0.04475	0.779888	0.412319
22	0.04935	0.666667	0.358008
23	0.0387	0.832041	0.435371
24	0.06375	0.718431	0.391091
25	0.05965	0.704107	0.381879
26	0.22415	0.644658	0.434404
27	0.1243	0.754626	0.439463
28	0.0523	1	0.52615
29	0.2877	0.697949	0.492825
30	0.21445	0.698531	0.456491
31	0.2642	0.751703	0.507952
32	0.2831	0.738608	0.510854
33	0.1462	0.773598	0.459899
34	0.30325	0.595878	0.449564
35	0.4385	0.674572	0.556536
36	0.34975	0.664189	0.506969
37	0.19105	0.71866	0.454855
38	1	0.417082	0.708541
39	1	0.461198	0.730599
40	1	0.568684	0.784342
41	1	0.597055	0.798528

Test Result of Object Significance

Id	*Visibility*	*Domain Importance*	*Significance*	*Expected Error*
1	0.327029	0.2	0.263515	0.465731
2	0.429774	0.2	0.314887	0.409599
3	0.353165	0.2	0.276583	0.450761
4	0.335648	0.2	0.267824	0.460740
5	0.362781	0.2	0.28139	0.445376
6	0.436597	0.4	0.418298	0.316287
7	0.382744	0.4	0.391372	0.338311
8	0.633173	0.5	0.566586	0.218313
9	0.566394	0.5	0.533197	0.237318
10	0.50145	0.7	0.600725	0.200453
11	0.50145	0.7	0.600725	0.200453
12	0.389218	0.7	0.544609	0.230643
13	0.38307	0.7	0.541535	0.232423
14	0.399411	0.7	0.549705	0.227723
15	0.360747	0.7	0.530373	0.239000
16	0.400759	0.7	0.55038	0.227340
17	0.359008	0.7	0.529504	0.239519
18	0.416313	0.7	0.558157	0.222962
19	0.421005	0.7	0.560503	0.221659
20	0.355047	0.7	0.527523	0.240708
21	0.412319	0.7	0.55616	0.224078
22	0.358008	0.7	0.529004	0.239819
23	0.435371	0.7	0.567685	0.217714
24	0.391091	0.7	0.545545	0.230104
25	0.381879	0.7	0.540939	0.232769
26	0.434404	0.9	0.667202	0.169761
27	0.439463	0.9	0.669731	0.168691
28	0.52615	0.9	0.713075	0.151367
29	0.492825	0.6	0.546412	0.229606
30	0.456491	0.6	0.528245	0.240274
31	0.507952	0.6	0.553976	0.225305
32	0.510854	0.6	0.555427	0.224489
33	0.459899	0.6	0.529949	0.239253
34	0.449564	0.6	0.524782	0.242364
35	0.556536	0.6	0.578268	0.212029
36	0.506969	0.6	0.553485	0.225582
37	0.454855	0.2	0.327428	0.396956
38	0.708541	0.3	0.504271	0.255116
39	0.730599	0.3	0.515299	0.248178
40	0.784342	0.3	0.542171	0.232053
41	0.798528	0.3	0.549264	0.227975

Template A.2


WÜRTH
8
EN 388
003
2131X
Art. 0899 400 529


(1)
(2)
(3)
(4)
(5)
(6)
(7)
(8)
(9)
(10)
(11)
(12)
(13)
(14)
(15)
(16)
(17)
(18)
(19)
(20)
(21)
(22)
(23)
(24)
(25)
(26)
(27)
(28)
(29)
(30)
(31)
(32)
(33)
(34)
(35)
(36)
(37)
(38)
(39)

Object Features

Id	*Area (No. of white pixels)*	*Hull Area*
1	127	111.5
2	361	421
3	422	487.5
4	764	975
5	783	947
6	905	1033.5
7	899	1046
8	860	1039.5
9	804	1154
10	938	1134.5
11	933	1141.5
12	851	1036.5
13	854	1034
14	930	1137
15	345	541.5
16	167	169
17	186	198.5
18	312	410
19	312	421
20	4842	14782
21	5475	11737.5
22	3693	9683.5
23	431	551.5
24	297	321
25	347	378.5
26	351	381.5
27	305	404.5
28	2012	2394.5
29	1895	3180
30	1581	3152
31	4986	5261.5
32	2923	3834
33	4243	5766
34	2611	3779.5
35	4394	5612.5
36	380	343
37	367	329.5
38	5766	7698
39	5402	5616

Test Result of Object Visibility

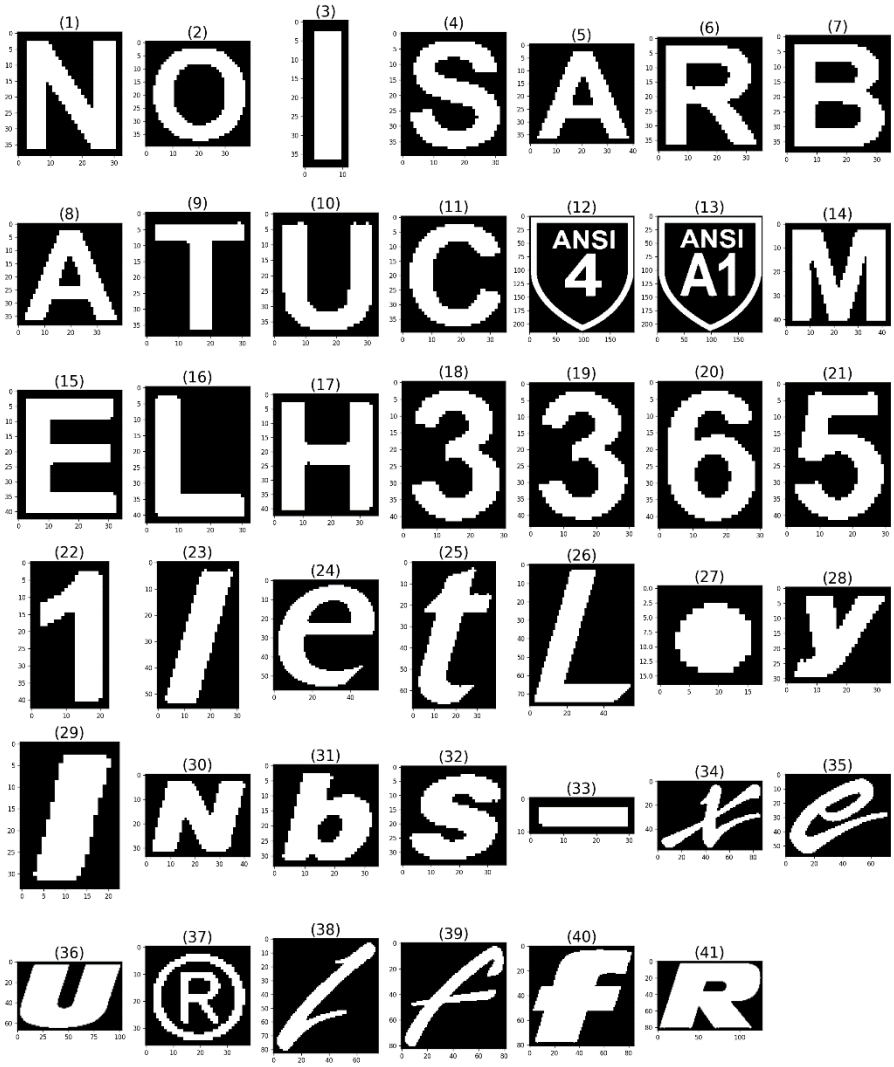
Id	*Size*	*Density*	*Visibility*
1	0.01115	1	0.505575
2	0.0421	0.857482	0.449791
3	0.04875	0.865641	0.457196
4	0.0975	0.78359	0.440545
5	0.0947	0.826822	0.460761
6	0.10335	0.875665	0.489508
7	0.1046	0.859465	0.482032
8	0.10395	0.827321	0.465635
9	0.1154	0.696707	0.406054
10	0.11345	0.826796	0.470123
11	0.11415	0.817346	0.465748
12	0.10365	0.821032	0.462341
13	0.1034	0.825919	0.464659
14	0.1137	0.817942	0.465821
15	0.05415	0.637119	0.345635
16	0.0169	0.988166	0.502533
17	0.01985	0.937028	0.478439
18	0.041	0.760976	0.400988
19	0.0421	0.741093	0.391596
20	1	0.327561	0.66378
21	1	0.466454	0.733227
22	0.96835	0.38137	0.67486
23	0.05515	0.781505	0.418327
24	0.0321	0.925234	0.478667
25	0.03785	0.916777	0.477313
26	0.03815	0.920052	0.479101
27	0.04045	0.754017	0.397234
28	0.23945	0.840259	0.539854
29	0.318	0.595912	0.456956
30	0.3152	0.501586	0.408393
31	0.52615	0.947639	0.736894
32	0.3834	0.762389	0.572895
33	0.5766	0.735865	0.656233
34	0.37795	0.690832	0.534391
35	0.56125	0.782895	0.672073
36	0.0343	1	0.51715
37	0.03295	1	0.516475
38	0.7698	0.749026	0.759413
39	0.5616	0.961895	0.761747

Test Result of Object Significance

Id	*Visibility*	*Domain Importance*	*Significance*	*Expected Error*
1	0.733335	0.500000	0.616667	0.141513
2	0.736894	0.300000	0.518447	0.190006
3	0.761747	0.300000	0.380874	0.287084
4	0.539646	0.900000	0.719823	0.103848
5	0.489321	0.100000	0.294660	0.371821
6	0.449791	0.100000	0.274896	0.394535
7	0.457196	0.100000	0.278598	0.390177
8	0.505600	0.100000	0.302800	0.362852
9	0.759151	0.300000	0.529575	0.183767
10	0.663780	0.500000	0.581890	0.157075
11	0.466106	0.100000	0.283053	0.384997
12	0.482510	0.100000	0.291255	0.375639
13	0.463383	0.100000	0.281691	0.386573
14	0.462341	0.100000	0.281171	0.387177
15	0.572658	0.300000	0.436329	0.243084
16	0.516475	0.300000	0.408238	0.264458
17	0.460761	0.100000	0.280380	0.388096
18	0.517150	0.300000	0.408575	0.264190
19	0.465464	0.100000	0.282732	0.385368
20	0.672073	0.300000	0.486036	0.209408
21	0.469281	0.100000	0.284640	0.383168
22	0.534391	0.300000	0.417196	0.257446
23	0.440193	0.100000	0.270097	0.400256
24	0.384748	0.500000	0.442374	0.238715
25	0.477109	0.500000	0.488555	0.207832
26	0.674557	0.500000	0.587278	0.154557
27	0.408393	0.600000	0.504197	0.198305
28	0.406054	0.100000	0.253027	0.421287
29	0.418327	0.500000	0.459164	0.226989
30	0.478439	0.500000	0.489219	0.207418
31	0.400743	0.500000	0.450372	0.233056
32	0.466423	0.100000	0.283212	0.384814
33	0.398028	0.500000	0.449014	0.234007
34	0.656233	0.300000	0.478116	0.214443
35	0.479101	0.500000	0.489551	0.207212
36	0.502533	0.500000	0.501266	0.200056
37	0.457113	0.600000	0.528557	0.184330
38	0.478053	0.500000	0.489026	0.207538
39	0.345635	0.500000	0.422817	0.253140

Template A.3


Ruf-flex®
by NSI
Lite
15633 HLE M
ANSI A1
ANSI 4
CUT
ABRASION


NOISARB
ATUCANSI 4ANSI A1M
EL3365
1letLy
InbS—e
U®fR

Object Features

Id	*Area (No. of white pixels)*	*Hull Area*
1	631	891
2	625	959.5
3	238	198
4	576	834.5
5	545	712.5
6	647	937.5
7	708	900
8	543	712
9	338	583.5
10	553	899
11	508	873.5
12	11600	33324.5
13	12724	33329.5
14	1083	1406
15	724	1057
16	440	667.5
17	719	1146
18	514	817
19	525	833.5
20	632	822.5
21	571	849
22	374	468.5
23	611	600
24	1317	2054
25	950	1496.5
26	1334	2356
27	123	108
28	420	497
29	315	300.5
30	744	882.5
31	539	610
32	610	730.5
33	161	129.5
34	1301	2722
35	1372	2022.5
36	4293	4874.5
37	444	839.5
38	1155	2009.5
39	1403	2653
40	3520	4140.5
41	6998	8228

Test Result of Object Visibility

Id	*Size*	*Density*	*Visibility*
1	0.089100	0.708193	0.836836
2	0.095950	0.651381	0.690930
3	0.019800	1.000000	0.334459
4	0.083450	0.690234	0.425140
5	0.071250	0.764912	0.351282
6	0.093750	0.690133	0.379490
7	0.090000	0.786667	0.684120
8	0.071200	0.762640	0.318807
9	0.058350	0.579263	0.426535
10	0.089900	0.615128	0.356012
11	0.087350	0.581568	0.400874
12	1.000000	0.348092	0.355860
13	1.000000	0.381764	0.632283
14	0.140600	0.770270	0.530050
15	0.105700	0.684957	0.371000
16	0.066750	0.659176	0.505400
17	0.114600	0.627400	0.392566
18	0.081700	0.629131	0.417353
19	0.083350	0.629874	0.363490
20	0.082250	0.768389	0.506500
21	0.084900	0.672556	0.423347
22	0.046850	0.798292	0.674077
23	0.060000	1.000000	0.438333
24	0.205400	0.641188	0.397256
25	0.149650	0.634815	0.395329
26	0.235600	0.566214	0.391942
27	0.010800	1.000000	0.455791
28	0.049700	0.845070	0.387704
29	0.030050	1.000000	0.418539
30	0.088250	0.843059	0.472258
31	0.061000	0.883607	0.446986
32	0.073050	0.835044	0.385825
33	0.012950	1.000000	0.439425
34	0.272200	0.477957	0.509900
35	0.202250	0.678368	0.465730
36	0.487450	0.880706	0.372969
37	0.083950	0.528886	0.375060
38	0.200950	0.574770	0.454732
39	0.265300	0.528835	0.398647
40	0.414050	0.850139	0.515150
41	0.822800	0.850510	0.306418

Test Result of Object Significance

Id	*Visibility*	*Domain Importance*	*Significance*	*Expected Error*
1	0.836836	0.300000	0.568418	0.163554
2	0.690930	0.500000	0.595465	0.150807
3	0.334459	0.200000	0.267230	0.403714
4	0.425140	0.600000	0.512570	0.193385
5	0.351282	0.200000	0.275641	0.393654
6	0.379490	0.600000	0.489745	0.207091
7	0.684120	0.300000	0.492060	0.205658
8	0.318807	0.200000	0.259403	0.413305
9	0.426535	0.600000	0.513268	0.192981
10	0.356012	0.600000	0.478006	0.214514
11	0.400874	0.900000	0.650437	0.127879
12	0.355860	0.600000	0.477930	0.214563
13	0.632283	0.300000	0.466142	0.222287
14	0.530050	0.900000	0.715025	0.105353
15	0.371000	0.700000	0.535500	0.180530
16	0.505400	0.900000	0.702700	0.109322
17	0.392566	0.900000	0.646283	0.129482
18	0.417353	0.100000	0.258676	0.414207
19	0.363490	0.700000	0.531745	0.182575
20	0.506500	0.300000	0.403250	0.268445
21	0.423347	0.900000	0.661674	0.123640
22	0.674077	0.500000	0.587038	0.154668
23	0.438333	0.100000	0.269167	0.401374
24	0.397256	0.300000	0.348628	0.316243
25	0.395329	0.700000	0.547664	0.174060
26	0.391942	0.100000	0.245971	0.430300
27	0.455791	0.900000	0.677895	0.117767
28	0.387704	0.300000	0.343852	0.320807
29	0.418539	0.100000	0.259270	0.413470
30	0.472258	0.200000	0.336129	0.328326
31	0.446986	0.200000	0.323493	0.341012
32	0.385825	0.100000	0.242912	0.434266
33	0.439425	0.300000	0.369713	0.296859
34	0.509900	0.100000	0.304950	0.360519
35	0.465730	0.200000	0.332865	0.331557
36	0.372969	0.100000	0.236484	0.442722
37	0.375060	0.300000	0.337530	0.326949
38	0.454732	0.200000	0.327366	0.337072
39	0.398647	0.100000	0.249323	0.425994
40	0.515150	0.200000	0.357575	0.307868
41	0.306418	0.300000	0.303209	0.362407

Template A.4


(23)
(20) (22) (19) (21) (18) (17) (16) (15) (14)
MCR SAFETY
(11) (10) (9) (13) (8) (7) (6) (12) (5)
BLACK JACK
(4) (3) (1) (2)
6948


(1)
(2)
(3)
(4)
(5)
(6)
(7)
(8)
(9)
(10)
(11)
(12)
(13)
(14)
(15)
(16)
(17)
(18)
(19)
(20)
(21)
(22)
(23)

Object Features

Id	*Area (No. of white pixels)*	*Hull Area*
1	980	1158
2	1158	1385
3	1130	1350
4	1120	1320
5	2110	3418.5
6	2145	2567
7	1472	2119
8	2108	3418.5
9	2146	2568
10	1389	1967.5
11	2940	3339
12	2421	3392.5
13	2421	3391.5
14	3566	4420
15	4068	5180
16	4730	5438
17	3674	4422
18	4501	4784
19	5121	5639.5
20	6266	7055
21	4555	5150
22	4256	5023.5
23	51223	74570

Test Result of Object Visibility

Id	*Size*	*Density*	*Visibility*
1	0.705500	0.888306	0.796903
2	0.333900	0.880503	0.607202
3	0.196750	0.705972	0.451361
4	0.502350	0.847218	0.674784
5	1.000000	0.687005	0.843503
6	0.256800	0.835670	0.546235
7	0.563950	0.908059	0.736005
8	0.339150	0.713843	0.526497
9	0.515000	0.884466	0.699733
10	0.132000	0.848485	0.490242
11	0.341850	0.616645	0.479247
12	0.135000	0.837037	0.486019
13	0.478400	0.940844	0.709622
14	0.115800	0.846287	0.481043
15	0.138500	0.836101	0.487301
16	0.211900	0.694667	0.453284
17	0.442200	0.830846	0.636523
18	0.256850	0.835507	0.546179
19	0.543800	0.869805	0.706803
20	0.339250	0.713633	0.526442
21	0.518000	0.785328	0.651664
22	0.341850	0.617230	0.479540
23	0.442000	0.806787	0.624394

Test Result of Object Significance

Id	*Visibility*	*Domain Importance*	*Significance*	*Expected Error*
1	0.796903	0.300000	0.548452	0.173650
2	0.607202	0.700000	0.653601	0.126671
3	0.451361	0.700000	0.575681	0.160029
4	0.674784	0.300000	0.487392	0.208558
5	0.843503	0.000000	0.421751	0.253951
6	0.546235	0.700000	0.623117	0.138801
7	0.736005	0.300000	0.518002	0.190259
8	0.526497	0.700000	0.613248	0.142972
9	0.699733	0.300000	0.499867	0.200898
10	0.490242	0.800000	0.645121	0.129935
11	0.479247	0.700000	0.589624	0.153473
12	0.486019	0.800000	0.643009	0.130760
13	0.709622	0.300000	0.504811	0.197939
14	0.481043	0.800000	0.640522	0.131740
15	0.487301	0.800000	0.643650	0.130509
16	0.453284	0.700000	0.576642	0.159568
17	0.636523	0.300000	0.468261	0.220878
18	0.546179	0.700000	0.623089	0.138813
19	0.706803	0.300000	0.503401	0.198778
20	0.526442	0.700000	0.613221	0.142984

21	0.651664	0.300000	0.475832	0.215918
22	0.479540	0.700000	0.589770	0.153406
23	0.624394	0.300000	0.462197	0.224933

APPENDIX B: Source Codes

```
def preprocess(path):
    # Read RGB image
    image = cv2.imread(path)
    # Convert to grayscale image
    image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    if mode == 1:
        type = cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU
    else:
        type = cv2.THRESH_BINARY + cv2.THRESH_OTSU

    # Convert to binary image
    ret, binary = cv2.threshold(image_gray, 0, 255, type)
    return binary
```

```
def _extract_global_features(binary):
    # foreground (white pixels) area of image
    points1 = cv2.findNonZero(binary)

    # Get bounding rectangle
    boundingRect = cv2.boundingRect(binary)
    x, y, w, h = boundingRect
    # Crop-out foreground area
    binary = binary[y - 1:y + h + 1, x - 1:x + w + 1]

    # Get minimum area rectangle
    minAreaRect1 = cv2.minAreaRect(points1)
    # Get angle parameter from minimum area rectangle
    angle = minAreaRect1[2]
    # Calculate convex hull
    hull1 = cv2.convexHull(points1)

    # Calculate convex hull area
    hull_area = cv2.contourArea(hull1)

    if angle < -45:
        angle = -(90 + angle)
    else:
        angle = -angle

    features = dict()
    boundingRect = [str(i) for i in boundingRect]

    features['Bounding_Rect'] = ':'.join(boundingRect)
    features['Angle'] = angle
    features['Hull_Area'] = hull_area

    return binary, features
```

```
def _extract_object_features(image):
    # Extract image contours to segment objects
    contours, hierarchy = cv2.findContours(image, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

    objects = []
    for i in range(len(contours)):
        c = contours[i]

        object = Object()
        object.contour = c

        # Masking-out each object
        mask = np.zeros_like(image)
        cv2.drawContours(mask, contours, i, 255, -1)
        out = np.zeros_like(image)
        out[mask == 255] = image[mask == 255]

        # Crop-out each object
        (y, x) = np.where(mask == 255)
        (topy, topx) = (np.min(y), np.min(x))
        (bottomy, bottomx) = (np.max(y), np.max(x))
        cropped = out[topy - 1:bottomy + 1, topx - 1:bottomx + 1]
        boundingRect = cv2.boundingRect(c)

        object.image = cropped
        object.boundingRect = boundingRect
        objects.append(object)

    # Sort objects from top-x value of object
    objects.sort(key=lambda e: e.boundingRect[0])
    return objects
```

```
def visibility(object:Object, w1=0.5, w2=0.5, upper_bound=10000):
    # No. of white pixels
    wp_area = cv2.countNonZero(object.image)

    # find convex hull & calculate hull area
    hull = cv2.convexHull(object.contour)
    hull_area = cv2.contourArea(hull)

    # calculate density
    density = wp_area / (hull_area + 0.00000001)
    density = min(density, 1)

    # calculate size
    size = min(hull_area / upper_bound, 1)

    # calculate visibility
    visibility = w1 * size + w2 * density
    object.visibility = visibility
    return visibility
```

```
def significance(object:Object, w1=0.5, w2=0.5):
    # calculate significance
    significance = w1*object.visibility + w2*object.domain_importance
    return significance
```

```
def expected_error(object: Object, alpha=0.9, _lambda=-3):
    # calculate maximum expected error for the object
    expected_error = alpha*(math.exp(_lambda * object.significance))
    return expected_error
```

```
def inspect(template:Template, path):
    # start performance monitoring timer
    t1_start = time.perf_counter_ns()

    # do preprocessing & retrieve binary image
    bin_image = _preprocess(path)

    # initialize inspection object
    inspection = Inspection()
    inspection.bin_image = bin_image
    inspection.image = bin_image

    # create error evaluator object
    error_eval = ErrorEvaluation()

    # perform inspection level-1 & obtain result
    result = level1.level_1(template, inspection, error_eval)

    if result:
        # if result is True
        # then perform inspection level-2 & obtain result
        result = level2.level_2(template, inspection, error_eval)

        # if result is True
        # then calculate quality for the sticker
        if result:
            quality = QMI.calc_quality(error_eval, template.objects)

    # stop performance monitoring timer
    t1_stop = time.perf_counter_ns()
```

```
def inspection_level_1(template: Template, inspection: Inspection,
error_eval:ErrorEvaluation):

    binary_2 = inspection.bin_image

    # Find Bounding Rectangle of inspection image
    x, y, w, h = cv2.boundingRect(binary_2)

    # User defined ROI for sticker placement
    roi_x, roi_y, roi_w, roi_h = template.roi

    # Is inside the sticker-placement-region
    if (x<roi_x or y<roi_y or (x+w)>(roi_x+roi_w) or
(y+h)>(roi_y+roi_h)):
        return False
    else:
        # Find Convex Hull and Calculate Hull Area
        points_2 = cv2.findNonZero(binary_2)
        hull_2 = cv2.convexHull(points_2)
        hull_area_2 = cv2.contourArea(hull_2)

        # Calculate Scale Diff factor
        scale_diff_factor = abs(template.hull_area - hull_area_2) /
template.hull_area

        if scale_diff_factor > template.scale_thresh:
            return False
        else:
            # Find Minimum Area Rectangle
            minAreaRect_2 = cv2.minAreaRect(points_2)
            # Get angle from min area rectangle array
            angle = minAreaRect_2[2]

            # Get template image angle
            template_angle = template.angle

            # Calculate Angle Diff
            if abs(template_angle) % 90 == 0:
                a = angle
                k = math.floor(abs(a) / 45)
                theta = (abs(a) ** (1 - k)) * (90 - abs(a)) ** k
            else:
                theta = abs(template_angle - minAreaRect_2[2])

            if theta > template.rotate_thresh:
                return False
            else:

                # Calculate total level-1 error
                inspection.boundingRect = (x, y, w, h)
                inspection.angle = angle
                inspection.hull_area = hull_area_2

                level_1_error = 0.5 * scale_diff_factor + 0.5 * theta
                error_eval.level_1_error = level_1_error
```

```
def inspection_level_2(template: Template, inspection: Inspection,
error_eval: ErrorEvaluation):

    # perform geometric correction
    _geometric_correction(template, inspection)

    # calculate foreign body error
    foreign_error = foreign_body_error.calc_error(template,
inspection)
    # evaluate calculated foreign body error
    r = error_eval.evaluate_foreign_body_error(foreign_error)

    if not r:
        return False

    else:
        objects = template.objects
        binary_2 = inspection.bin_image

        # iterate through template objects
        for i in range(len(objects)):
            object = objects[i]

            x, y, w, h = object.boundingRect
            templ_obj_img = object.image

            # skip the template objects with neglegible size
            if templ_obj_img.shape[0]==0 or
templ_obj_img.shape[1]==0:
                continue

            # crop-out corresponding inspection object's area using
template object coordinates
            # with some padding
            cand_obj_img = binary_2[max(y - 10, 0):min(y + h + 10,
binary_2.shape[0]),
                                    max(x - 10, 0):min(x + w + 10,
binary_2.shape[1])]

            # count white pixels in template object
            template_white_pxls = cv2.countNonZero(templ_obj_img)
            # count white pixels in inspection object
            white_pxls = cv2.countNonZero(cand_obj_img)

            # calculate pixel ratio
            pixel_ratio = white_pxls / template_white_pxls

            error_msg = ''

            # If inspection area is nearly empty
            if pixel_ratio < 0.01:
                return False
            else:
                # If inspection area is nearly filled
                pixel_ratio = white_pxls / (cand_obj_img.shape[0] *
cand_obj_img.shape[1])
                if pixel_ratio > 0.85:
                    return False

                else:
                    # Apply Normed Cross-Correlation
```

```
                    res = cv2.matchTemplate(cand_obj_img,
object.image, cv2.TM_CCORR_NORMED)
                    min_val, max_val, min_loc, max_loc =
cv2.minMaxLoc(res)

                    top_left = max_loc
                    ccorr_sim = max_val

                    # If the objects are highly similar
                    if ccorr_sim > 0.9:
                        # crop-out exact object region
                        cand_obj_img =
cand_obj_img[top_left[1]:top_left[1] + h + 1, top_left[0]:top_left[0]
+ w + 1]

                        # perform XOR operation on template object
image and inspection object image
                        bitwiseXor = cv2.bitwise_xor(object.image,
cand_obj_img)
                        # perform morphological erosion operation
over XOR-ed image with 3x3 kernal
                        bitwiseXor = erosion(bitwiseXor, 3)

                        # count white pixels in result image
                        white_pxls = cv2.countNonZero(bitwiseXor)
                        # calculate pixel ratio
                        pixel_ratio = white_pxls /
template_white_pxls

                        if pixel_ratio < 0.0001:
                            error = 0.0
                            object.error = error
                        else:
                            # if the pixel ratio is high
                            # then move for inspection level-3 &
obtain the error
                            level_3_error = level_3.level_3(object,
cand_obj_img)
                            object.error = level_3_error

                    else:
                        # if result of CCORR operation showing
dissimilarities of two objects
                        # then move for inspection level-3 & obtain
the error
                        level_3_error = level_3.level_3(object,
cand_obj_img)
                        object.error = level_3_error

            # Evaluate the calculated error
            r = error_eval.evaluate_object_error(object)
            if not r:
                return False

        return True
```

```
def _geometric_correction(template: Template, inspection:
Inspection):
    angle_1 = template.angle
    angle_2 = inspection.angle

    if angle_1 < -45:
        angle_1 = -(90 + angle_1)
    else:
        angle_1 = -angle_1

    if angle_2 < -45:
        angle_2 = -(90 + angle_2)
    else:
        angle_2 = -angle_2

    # calculate angle difference
    angle_diff = (angle_1 - angle_2)
    corrected = inspection.bin_image

    # Correct Angle
    if angle_diff != 0 and angle_diff != 90.0:
        (h_, w_) = corrected.shape[:2]
        center = (w_ // 2, h_ // 2)
        M = cv2.getRotationMatrix2D(center, angle_diff, 1.0)
        size = (w_, h_)
        # correct rotation deformation using Bicubic interpolation
        corrected = cv2.warpAffine(corrected, M, size,
flags=cv2.INTER_CUBIC, borderMode=cv2.BORDER_REPLICATE)

    x, y, w, h = cv2.boundingRect(corrected)

    # Crop Sticker Region
    corrected = corrected[y - 1:y + h + 1, x - 1:x + w + 1]

    dim = (template.bin_image.shape[1], template.bin_image.shape[0])
    # Resize to same W x H as template size
    corrected = cv2.resize(corrected, dim,
interpolation=cv2.INTER_AREA)

    return corrected
```

```
def calc_foreign_body_error(template: Template, inspection:
Inspection):
    binary_2 = inspection.bin_image

    # Remove legitimate template objects
    # keep only foreign objects
    background_image = np.copy(binary_2)
    objects = template.objects
    for i in range(len(objects)):
        object = objects[i]
        x, y, w, h = object.boundingRect
        cv2.rectangle(background_image, (max(x - 1, 0), max(y - 1,
0)),
                      (min(x + w + 1, binary_2.shape[1]), min(y + h +
1, binary_2.shape[0])),
                      (0, 0, 0), -1)

    # Remove small variations using morphological erosion with 3x3
kernal
    kernal = np.ones((3, 3), np.uint8)
    background_image = cv2.erode(background_image, kernal)

    # find contours to extract foreign objects
    contours_2, hierarchy_2 = cv2.findContours(background_image,
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    total_error = 0
    max_error = 0
    for i in range(len(contours_2)):
        c = contours_2[i]

        # crop-out each object
        mask = np.zeros_like(background_image)
        cv2.drawContours(mask, contours_2, i, 255, -1)
        out = np.zeros_like(background_image)
        out[mask == 255] = background_image[mask == 255]

        # count white pixels in object
        wp_area = cv2.countNonZero(out)

        # Find convex hull & calc. hull area
        hull = cv2.convexHull(c)
        hull_area = cv2.contourArea(hull)
        if hull_area == 0:
            continue

        # Calculate density
        density = wp_area / hull_area
        density = min(density, 1)

        # Calculate size
        area_thresh = 10000
        size = min(hull_area / area_thresh, 1)

        # Calculate object's error
        error = (0.75 * size + 0.25 * density)

        total_error += error
        max_error = max(max_error, error)
```

```
    # calulate mean error
    mean_error = 0
    if len(contours_2) > 0:
        mean_error = total_error/len(contours_2)

    # calculate final foreign body error
    foreign_body_error = 0.5 * mean_error + 0.5 * max_error
    return foreign_body_error
```

```
def calc_quality(error_eval:ErrorEvaluation, objects):
    total = 0
    defected_object_count = 0
    for object in objects:
        if object.error == 0:
            pass
        else:
            defected_object_count += 1
            total += ((object.error + 2*(object.significance *
object.error)))

    variance = total / defected_object_count
    w1 = 0.6
    w2 = 0.3
    w3 = 0.1

    QMI = 1 - (w1 * variance + w2 * error_eval.foreign_body_error +
w3 * error_eval.level_1_error)
    return QMI
```