A Training Assistant Tool for the Automated Visual Inspection System

---

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Electrical Engineering

---

by
Mohan Karthik Ramaraj
December 2015

Accepted by:
Dr. Adam W. Hoover, Committee Chair
Dr. Richard E. Groff
Dr. Yongqiang Wang

Abstract

This thesis considers the problem of assisting a human user setting up an automated Visual Inspection (VI) system. The VI system uses a stationary camera on an automobile assembly line to inspect cars as they pass by. The inspection process is intended to identify when parts have been missed or incorrect parts have been assembled. The result is reported to a human working on the assembly line who then can take corrective actions. As originally developed, the system requires a setup phase in which the human user places the camera and records a video of at least 30 minutes length to use for training the system. Training includes specifying regions of cars passing by that are to be inspected. After deployment of a number of systems, it was learned that users could benefit from being provided guidance in best practices to delineate training data. It was also learned that users could benefit from simple visual feedback to ascertain whether or not an inspection problem was suitable for a VI system or if the problem was too challenging. This thesis describes a few methods and a new software tool intended to address this need.

Acknowledgement

First and foremost, I would like to express my sincere gratitude to my advisor Dr. Adam Hoover for giving me the opportunity to work on this research. I thank him for his guidance, patience and motivation throughout the research. The research meetings have taught me to look at a problem in different perspectives and produce a solution in the right direction. I thank him for going out of his way to help me with my medical emergencies.

I would like to thank my committee members, Dr. Richard Groff and Dr. Yongqiang Wang for being part of my committee and taking time to review this thesis. I extend my thanks to Dr. Jöerg Schulte from the BMW Manufacturing plant for his valuable time in looking at this work and for acting as the liason for this project.

I also like to thank, Jung Phil Kwon and Ryan Mattfield for their support and guidance. I have learnt many things from them, right from coding, debugging to writing my thesis. I would also like to express my thanks to my other team members Douglas Dawson, Shawn Mathew and Jayadevan Puthumanappilly for helping me in times of need. I could not have asked for a more supportive team.

I thank Ashwinraj Thiagarajan for being a perfect roommate and keeping me in track when I lose momentum in work. I also thank Abha Apte for patiently sitting with me and helping me when I'm stuck with a problem. My time in this university has been the best possible thanks to every one of my friends here.

Finally I would like to express my greatest thanks to my father, mother, brother and Sushma Ramachandran for everything they've done for me. Without their encouragement and undying belief in me, I would not have been able to achieve this degree.

Table of Contents

List of Tables

Table **Page**

List of Figures

Figure **Page**

Figure **Page**

Chapter 1

Introduction

This thesis considers the problem of assisting a human user setting up an automated Visual Inspection (VI) system. The VI system was developed collaboratively between Clemson University and BMW, Spartanburg [3]. During setup, a camera is placed at a point in the assembly line where inspections of cars need to be performed. Subsequently, a video is recorded for 30 minutes or more depending on the difficulty of the inspection. A car in the assembly line takes roughly 90-120 seconds to pass through the view of the camera. It is important to note that the term 'car' refers to any portion of a car during assembly; in some cases it is just a car door or engine or other part being assembled prior to installation in the total car. In a video of 30 minutes, approximately 15-20 cars typically pass the camera and can be used for training the system. In cases where the part to be inspected consists of many different types (classes), it may be necessary to record more than 30 minutes to setup up the VI correctly. After the video has been recorded, the next step is creating training data. A user reviews the video and outlines rectangles covering trigger windows and inspection search windows. The user also outlines trigger templates and inspection templates for each class. This is done by observing the cars and specifying a region which the user thinks would work best for the given inspection problem.

A number of systems have been deployed by setting up and training the VI using this method. However, it was found that the user could not always predict accurately how

VI would work with the trained data. It would be beneficial to the user if guidance was provided during the training stage to extract the best results. Also, predicting the results could help determine whether or not VI would be suitable for a inspection problem. There could be cases where the inspection problem might be too complicated for VI which leads to low accuracy. It is preferable to know how VI would fare with the problem before fully implementing the system in the plant. In this work, a tool is developed to address this problem.

1.1 Visual Inspection (VI) System

A VI system can be operated in three modes: training, video replay or camera mode. When setting up a new inspection problem, the training mode is first initialized to record a video of the cars passing by the camera for training purposes. This mode cannot be used to perform inspections. The video replay mode is used when VI is not connected to the BMW database. This mode works offline on a pre-recorded video to test the performance of the system or for other verification purposes. The camera mode is the default mode for VI which is used for live inspection of cars in the plant. Before using this mode, VI needs to go through the training mode to be set up for operation.

The components of a VI include a computer and camera. In the plant, they are fixed on a stand at a location close to the human inspector. Figure 1.1 shows an example. The VI is intended to assist the human inspector in checking whether a part has been installed or not, or if there are different types of parts, to check if the correct part has been installed. The setup in the plant is such that a camera is placed at the inspection area focusing on the spot where the part to be inspected will pass in the assembly line. Every camera is connected to a single computer. Every computer can run up to two copies of VI, one camera per copy. Results can be displayed locally on the same computer or sent via ethernet to a separate display program running on another computer. Once the VI has been setup and trained, VI is initialized in the camera mode at the assembly line for inspection. An

Figure 1.1: The VI system mounted on fixed stand which includes the computer, display and camera.

example of a part being inspected successfully is shown in Figure 1.2. The figure shows VI inspecting the badge decal installed on the side of the door. This part varies based on the engine model. The camera detects that the door is ready for inspection when it detects the trigger which is shown in the top left box in aqua. The box shows a zoomed view of the trigger. The decal is inspected in two portions. The first portion is shown in the green box on the top left. The image in the box is the captured image from the frame and below the box, the inspection result is displayed. It can be seen that VI successfully classifies it to 'XDRIVE' class with a match score of 0.95. The match score has a range of -1 to 1 with 1 denoting the best match. Similarly the second portion shown in the other green box on the top. Again VI successfully classifies the captured image to '40D' with a match score of 0.77.

BMW allows the customers to customize the parts of the car, hence in some areas of the assembly line, the VI needs to check if a part has been detected and in some other areas, the VI needs to classify which part has been installed. In the case of detection, there will be only two classes: absent and present. The absent class templates are images of a

Figure 1.2: Screenshot of a successful inspection which is shown as output to the human inspector.

missing part while the present class templates are images of the part correctly installed. Figure 1.3 is an example of a detection problem. In the case of classification, there will be two or more classes with each class corresponding to a particular type of part. In this case the VI needs to be connected to the BMW database so that VI has access to information on what parts are assigned to each car. Each car has a vehicle identification number (VIN) which is used to find the options of a particular car. During operation, the image from the live feed is first classified to one of the classes. Then the identified class is compared to the intended class obtained using the VIN. An example of a classification problem is shown in Figure 1.4. In camera mode, the results of inspection can be OK, NOK or MISSED. OK implies that VI detected that the correct part has been installed. NOK implies that VI detected an error which could mean either the part was not installed or the wrong part was installed. MISSED is a result of VI detecting a car passing through but the part installed could not be identified.

In order to show the result of the inspection to the human inspector, VI includes a display program. The result is signalled in 3 ways. First, the inspected image is shown in the display program. Second, a green light is used to signal an OK result and a red light to

(a) Spring coil correctly installed.

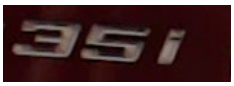
(b) Spring coil not installed.

Figure 1.3: An example of a detection problem where VI detects whether the spring has been installed.

(a) Class 25d.

(b) Class 30d.

(c) Class 40d.



(d) Class 35i.

Figure 1.4: An example of a classification problem.

signal an NOK result. Third, a sound is played through the computer's default speakers. The display program continues to show the inspection results as the car passes through as long the results are OK. In the case of an NOK result, the program creates an error message while simultaneously lighting the red light and the sound file is played. The alerts can be stopped by the human inspector by closing the error message after dealing with the defect.

VI captures 10 frames per second. VI checks the frames for the appearance of a trigger, to find out when a car is ready for inspection. A trigger is any part of the car (or car carrier) that exists for every car passing through that inspection area. When the frame is captured, it is not necessary to analyze the entire frame. There are a few regions in the frame which are the areas to be inspected. During the setup process, the user is expected to have specified the regions for trigger search windows, trigger templates, inspection search windows and inspection templates. The trigger search window is the area in which the trigger is expected to appear when a car is passing by the camera. Inside this trigger search window is the trigger template window in which the trigger shape is detected. The search window is larger than the template window to allow for slight variations in position of the trigger. The trigger search window is continuously inspected for the trigger template when the VI is running. It is important to note that the trigger is required to have a consistent appearance and needs to be in the same frame as the part to be inspected. There may be slight changes in the appearance of the trigger for which a number of trigger templates can be stored. This improves the accuracy and makes sure the trigger is never missed due to slight variations in position or appearance of trigger. When a trigger is detected, VI captures frames of the video until the trigger moves out of the trigger search window. These frames are used to inspect the parts. Similar to the case of triggers, the inspection window is the area where the inspection template is expected to appear and is made to be bigger than the inspection template. The inspection template gives the appearance of the part that is to be inspected. If the entire car to be inspected does not fit in a single frame of the camera, it is necessary to use multiple triggers . Currently VI supports up to 4 triggers per car. Each trigger has its own trigger search window in which different trigger templates are

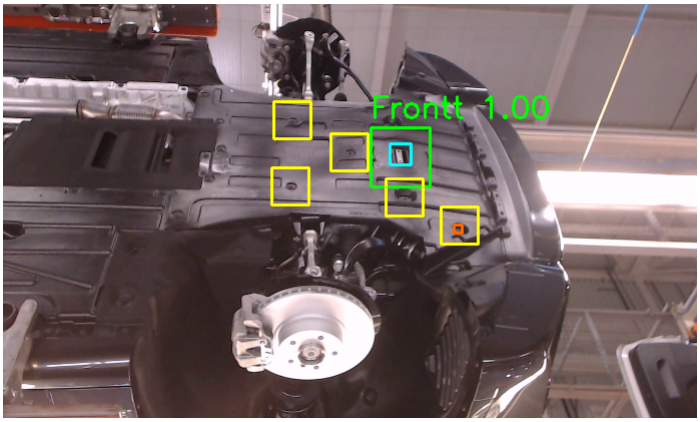


Figure 1.5: Screenshot of under body use case showing the trigger search window in green and trigger template in blue. The yellow boxes are inspection search windows and the red box is the inspection template.

inspected. Similarly, multiple parts can be inspected for each trigger. VI has the capacity to hold 20 inspection windows linked to a trigger. Each inspection window inspects for one particular part. An inspection window can inspect a part from among 20 different appearances. Each of these appearances are classified as different classes. Again, there may be a number of templates in a single class to make the system more robust. To summarize, in each car, VI can support 4 triggers with each trigger supporting 20 inspection windows and each inspection window can hold 20 different classes. Figure 1.5 shows an example of a case where 5 inspection windows are linked to a trigger. Here the under body is being inspected for bolts.

All image operations in VI are done by a variation of template matching. VI continuously cross correlates all stored templates (the training data) against all possible locations with the search windows (first for triggers; after being triggered for inspection areas). Each cross correlation results in a score between -1 and 1. The higher the value, the better the images match. The inspected part is classified by choosing the highest cross correlation score among all the captured images. The part belongs to the class it matches best. Some heuristics help make the process more robust, such as using a voting process over the period of time that the trigger is visible. However, the core operation of VI is template matching.

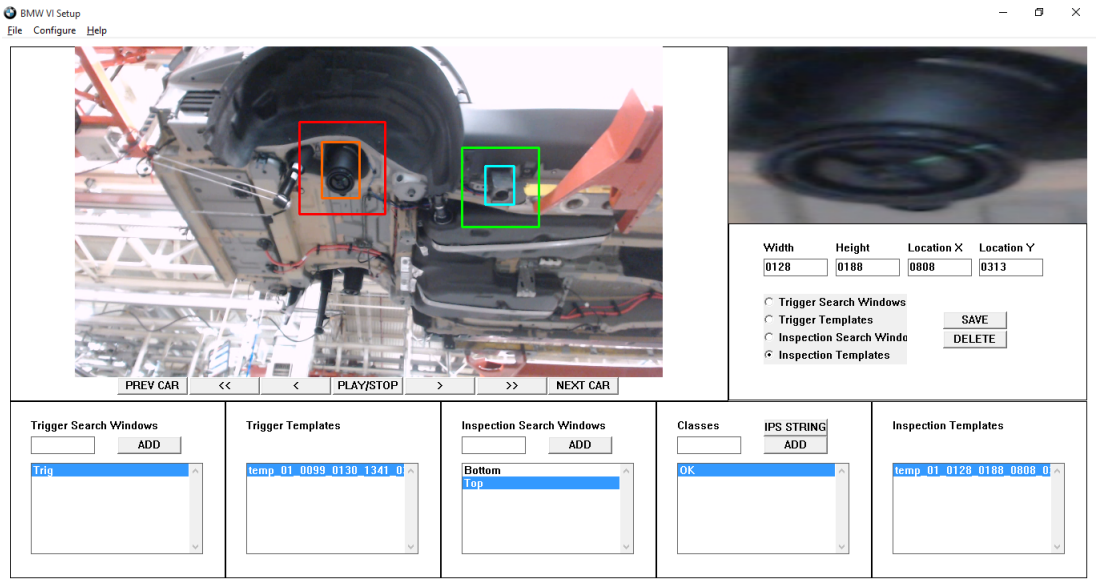
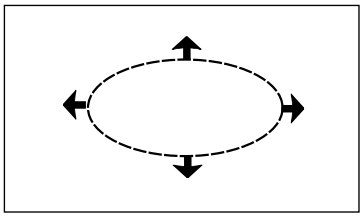


Figure 1.6: Screenshot of the setup tool showing the training of the spring coil use case.

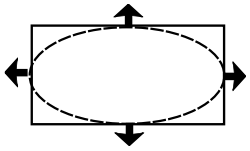
1.2 Problem Definition

This section demonstrates the setup and operation of a VI system, and discusses the difficulty of building training data. The training video is loaded into a setup program. This program is used to define trigger search windows, inspection search windows, trigger templates and inspection templates. Figure 1.6 shows a screenshot of the setup tool.

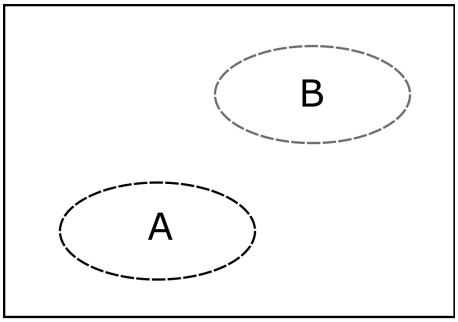
An inspection window must be outlined to be larger than the size of the inspection template. This allows for variations in the position of the part. Figure 1.7a demonstrates an inspection window large enough to account for variation in position of the elliptical part. When the window is made too small, some variations may be missed. This problem is shown in Figure 1.7b where the inspection window is perfectly encompassing the part. In contrast, when it is made too large, there may be other pieces of the car or background that match (or partially match) the area to be inspected. For some parts, the placement variation is quite small, making this challenge trivial. For other parts, the variation can be large, for example when a part is purposefully placed at a different location depending upon the model type of the car. Figure 1.7c shows an inspection window with a part from



(a) Example of an elliptical part inside an inspection window large enough to account for variation in position of part.

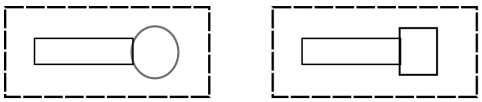


(b) Example of an inspection window too small with no space around the part to account for position variation.



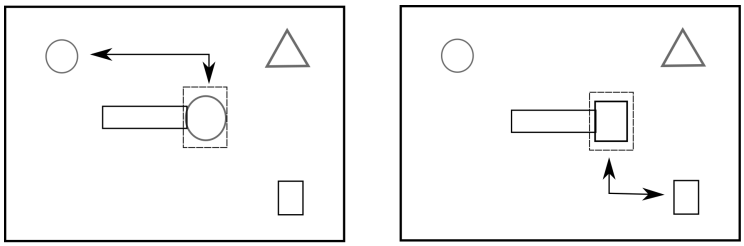
(c) Example of an inspection window too large in cases where 2 classes vary in location of part.

Figure 1.7: Demonstration of problems with different sizes of inspection window.



(a) Example of inspection templates outlined around two parts which includes similar background area.

(b) An inspection template outlined smaller to remove similar background area but no space for variation in position.



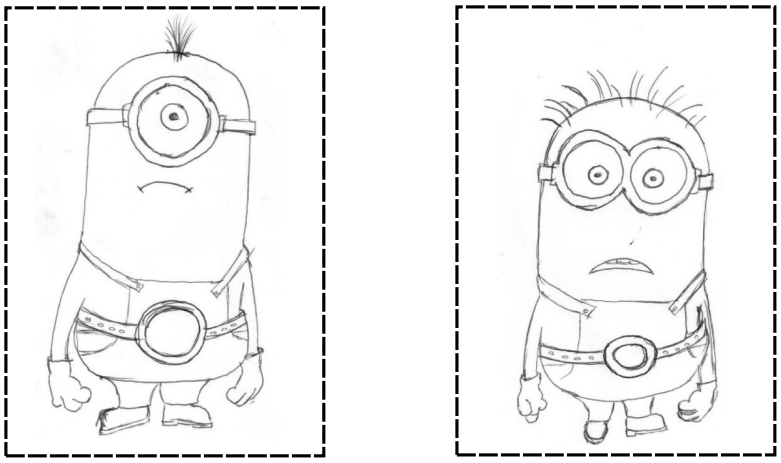
(c) Inspection window with small inspection templates with objects in the background that could match the template.

Figure 1.8: Demonstration of problems with different sizes of inspection templates.

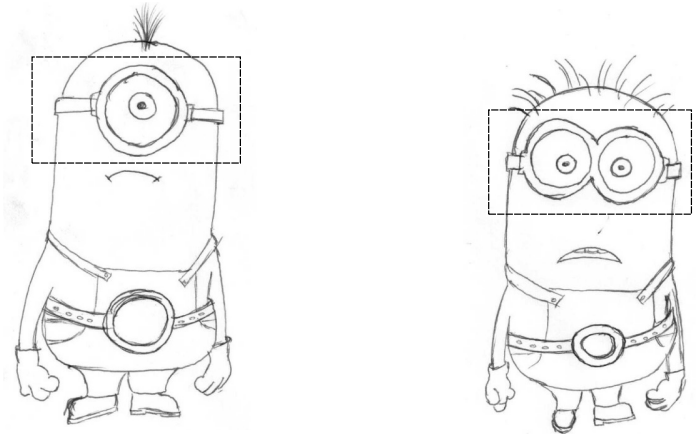
class A appearing on the bottom left and a part from class B appearing on the top right.

Similarly, an inspection template must be outlined around the area to be inspected. All areas are outlined using a rectangle. It is not unusual for the border of the rectangle to encompass background area that appears the same regardless of the part installed. This hinders the inspection process because all cross correlations across these areas yield the same results. Figure 1.8a demonstrates inspection templates encompassing 2 classes with similar background area. In the other direction, the template can be made smaller, encompassing only the area of difference between classes. Figure 1.8b shows the inspection template encompassing only the area of difference which does not account for variation in position. However, this can cause problems when the smaller area also matches or partially matches the appearance of other nearby areas on the car. An example of this problem is shown in Figure 1.8c.

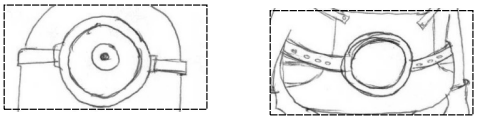
Figure 1.9 shows another case of a problem with size of inspection template. Figure 1.9a shows two classes where the inspection templates are outlined around the minions. But the minions are similar in most areas which does not help in discriminating the two classes.



(a) Example of inspection templates outlined for two classes which are similar in most characteristics.



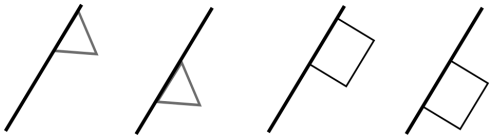
(b) Inspection templates outlined just around the characteristic that differ between the two classes.



(c) Image on the left shows the inspection template and the right image shows a background image of the same class which has a high probability of matching.

Figure 1.9: Another example of problem with different sizes of inspection templates.

(a) The two classes differ well enough that a small inspection template is sufficient to detect the class.



(b) The classes vary only in few areas and in different locations which requires a large inspection templates.

Figure 1.10: Examples of local and global dissimilarity.

Figure 1.9b shows the same classes where the inspection templates are made smaller. Figure 1.9c shows that the extracted smaller template of the first class with a background object in the same class which is similar to the template.

The necessary window and template sizes also depend upon the appearance of the parts being inspected. The greater the difference between the absence and presence of a part, or between two classes of a part, the smaller the templates need to be in order to detect the correct class. An example of this case is shown in Figure 1.10a. However, this may affect the ability of VI to correctly localize the part. Similarly, if there is a small variation in the appearance of a part, then a larger inspection template may be needed. This is demonstrated in Figure 1.10b. Hence it is important to optimize and find the right balance between the size of the inspection window and inspection template while setting up the VI system.

It is important to note that the intended user is a BMW employee who is unfamiliar with computer vision and image processing. The system is intended to be operated without the need to understand cross correlations or how the underlying calculations are made. After observing users setting up a number of systems, it was realized that a training assistant would be of great benefit. The goal of the training assistant is to help the user consider the effects of enlarging or shrinking search windows and templates. This process could also be used to determine if an inspection problem being considered was plausible, or beyond the

capability of the VI. The work in this thesis provides the basis for developing such a tool.

1.3 Related Work

The core problem being addressed by the VI is the ability to distinguish between images of different parts. This is a variation of the image similarity problem. Over the years, there have been many metrics developed to calculate image similarity. Image similarity measures were originally developed for the purpose of image quality assessment [1]. These started with simple pixel-to-pixel comparisons and evolved to more sophisticated measures that are scale and rotation invariant. Digital images are subject to a a wide variety of distortions during acquisition, processing, compression, storage, transmission and reproduction. Metrics were developed to assess the quality of these steps, for example to dynamically monitor and adjust image quality, to optimize streaming based on quality of image, or to rate how well an image restoration algorithm works [4]. For cases where the image content might be rotated, translated or scaled, it is necessary to use an image similarity measure which takes those factors into consideration. A few types of image similarity measure types are discussed in [1] and [5]. Even though the methods are described as a performance metric, they can also be used as similarity measures.

The challenge of image similarity varies depending upon the problem domain. Perhaps the most challenging domain is content-based image retrieval. In this problem, a query image is used to retrieve the most similar images from a large database of images. Classicaly, image retrieval was based on text descriptions attached to each image [6, 7]. This is time consuming as every image in the database would have to be described. Hence researchers started looking into content-based image retrieval which identifies images based on their visual content [8]. Color information provides a number of metrics that can discriminate images [9, 10]. Gross discrimination can be based upon color histograms of entire images. Similarly, texture information can be used to calculate image wide metrics [11]. Gabor wavelets are popular in this domain [12]. Locally, geometric based metrics can be

derived, such as color derivatives [13]. Surveys of many of these methods provide more details [14, 15].

Image similarity plays an important role in recognition problems such as face recognition and fingerprint recognition. Face recognition is the process of identifying the person based on just the face image. A method using a cosine similarity metric takes specific advantage of facial appearances and variations [16]. Another method uses a variation of Gabor coding that uses semantic ratings from human observers of facial expression [17]. These methods tend to be highly specialized to the problem domain.

1.4 Novelty

This work considers a variation of the problem of image similarity. We seek methods to semi-automatically assist a human operator training a system to classify different types of known parts. Typically, we expect there to be 2-6 types of parts. Each type will have variations including color and slight differences in orientation, positioning, texture and/or general appearance. The goal is to assist the human operator with maximizing classification accuracy by adjusting the training data. To our knowledge, the development of this assistance tool is unique.

Chapter 2

Research Design and Methods

2.1 Overview

To start, it is assumed that the user would have gone through the usual steps of setup operation as described in chapter 1. These steps include setting up a camera and recording a video of 30 minutes or longer. As mentioned earlier, a 30 minute video typically provides 15-20 examples which is used to train the system. This training is done in the setup program where the user outlines the trigger search windows, trigger templates, inspection search windows and inspection templates. The methods described in this chapter are intended to augment the interface available during these steps. One tool is designed to guide the user while outlining the templates by assisting with localization and template delineation. Using the tool the user would be able to outline the windows and templates in a manner that best works for the system. The second tool developed visually assists the user in identifying the possible success of the VI system for an inspection problem. It also helps the user identify mislabeled training data which would cause errors in classification. A single image similarity metric may not be well suited for all types of inspection problems, hence the tool gives the user different image similarity metrics as options to select the best metric for the given problem.

2.2 Image Similarity Metrics

This section describes 6 image similarity metrics explored in this work. For some of them, several variations were explored. In preliminary experiments, the first 3 metrics described were implemented and tested on some data, but were found to be too simple for the problem and hence are excluded from the experimental results. The next metric discussed seemed promising at first but it was not appropriate for this problem so it wasn't tested. These experiments led to the development of the last 2 metrics described in this section. In the experiments these metrics were compared against each other in the new software tool to determine which provided the best classification accuracy for different use cases.

2.2.1 Mean Squared Error

The mean squared error (MSE) is defined as,

$$MSE(x, y) = \frac{1}{N} \sum_{i=1}^{N} (x_i - y_i)^2 \tag{2.1}$$

where x and y are two images, each having N number of pixels. x_i and y_i represent the intensity value of the ith pixel. Since MSE is originally used as a quality metric, it is nothing but the error between the original image and distorted image. $x_i - y_i$ is called the error value and represented by e_i. Now it can been seen that the formula is the mean of the error squared. Hence when a quality of an image is being calculated compared to its original image, the MSE would be 0 if the images are exactly alike and MSE would increase as the quality of the test image lowers. There is no upper limit for the MSE. But the MSE does not work perfectly well for some cases of image quality degradation. An example of this problem shown in Figure 2.1. MSE does not perceive the image like that of the human eye [18]. In Figure 2.1, the original Einstein image is distorted by applying different types of noise. Figure 2.1(a) is original image and the distortions applied to the following images in the same order are contrast stretch, impulsive noise contamination, white Gaussian noise

contamination, blurring, JPEG compression, spatial shift, spatial scaling and rotation. It can be seen that the image from Figure 2.1(b)-(g) that have been distorted drastically in different ways seem to having similar MSE values. But the human eye is able to perceive the difference in the types of distortion. In the same case, images Figure 2.1(h)-(j) which have minor loss in quality have large MSE values because of minor geometrical changes. Hence MSE seems to be highly sensitive to factors like translation, scaling and rotating. This would be a major drawback while using MSE as an image similarity measure in template matching due to the fact that the template would not match well even if there are minor changes in the appearance of the object being matched. This can be due to the fact that MSE considers the pixels of the image as a 1-D vector which leads to the fact that MSE does not use the spatial position information.

2.2.2 Peak Signal-to-Noise Ratio

Peak signal-to-noise ratio (PSNR) represents the measure of the peak error. MSE gives the power of the corrupting noise. PSNR can be defined as the ratio of the maximum possible power of a signal to power of corrupting noise. PSNR is primarily used for checking the quality of a compressed image compared to the original image. In contrast to the MSE, higher the PSNR denotes better image quality. The units of PSNR are in decibels (dB). The PSNR can be calculated by using,

$$PSNR = 10.log_{10}\left(\frac{MAX_I^2}{MSE}\right) \tag{2.2}$$

where MAX_I is the maximum possible pixel value of the input image and MSE is the mean squared error calculated by using equation 2.1.

2.2.3 Structural Similarity Index

To overcome the drawback of MSE, the structural similarity index measure (SSIM) was introduced. SSIM used the fact that an image similarity can be better determined

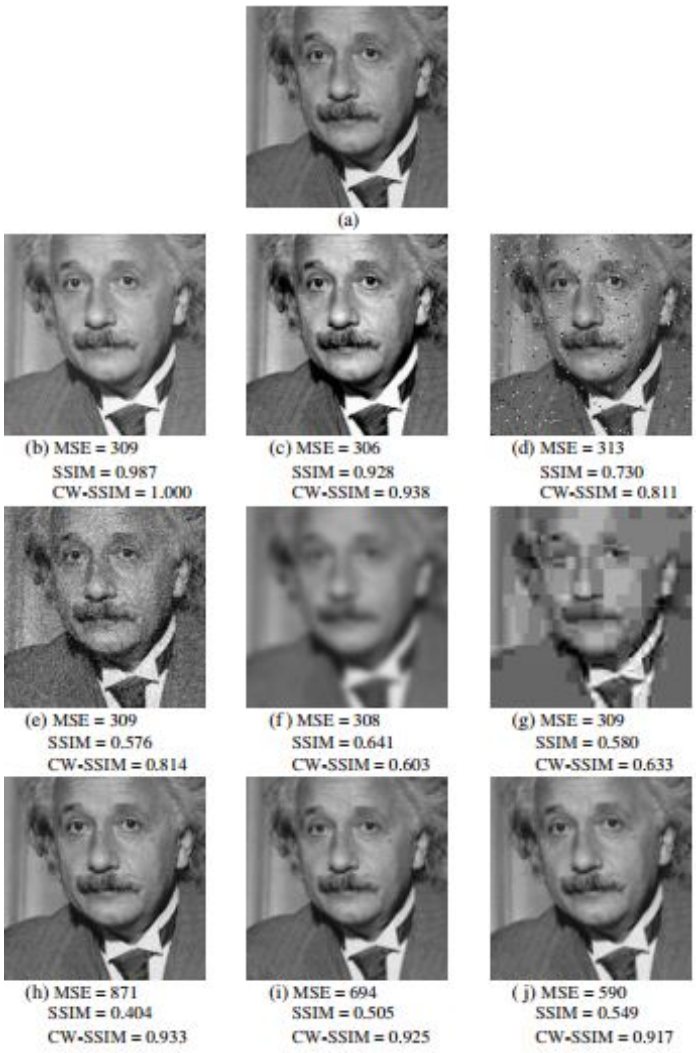


Figure 2.1: Comparison of image similarity measures tested on an "Einstein" image which has been altered with various types of structural and non-structural distortions: (a)Original Image (b)mean luminance shift (c)contrast stretch (d)impulsive noise contamination (e)white Gaussian noise contamination (f)blurring (g)JPEG compression (h)a spatial shift to the left (i)spatial scaling by zooming out (j)counterclockwise rotation. Image borrowed from [1].

by comparing the similarity of structural information derived from the image. Structural changes are distortions that affect the structure of the object such as blurring, jpeg compression or noise. The human visual system is more sensitive to structural changes in the image than non-structural changes such as change in brightness, contrast or spatial shift [18]. So SSIM was developed to be more sensitive to these structural changes and better mimic the human visual system. The SSIM index between two image patches x and y taken from the two images being compared is calculated by,

$$S(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{2.3}$$

where C_1 and C_2 are small constants that are added so that none of the terms reach absolute zero causing instability. μ_x and μ_y are the local means of image patches x and y respectively. σ_x and σ_y are the standard deviations of x and y respectively. σ_{xy} is the cross correlation of x and y after subtracting their means [4]. This gives the SSIM index for the patches x and y. The SSIM is normally applied locally and a SSIM map is computed by using the sliding window approach. The final SSIM index between the image and the template is calculated by summing all the SSIM values in the SSIM map and dividing it by the number of local windows in the image. The SSIM index can take a value between -1 to 1 with 1 denoting the two input images are equal. Figure 2.1 shows the results of SSIM applied to the same images that were used previously with MSE. It can be seen that SSIM index for images with structural changes is lower than that of images with non-structural changes. This demonstrates that SSIM provides better structural similarity than MSE. But SSIM too has a drawback, which is the same as MSE's major drawback. It can be observed from SSIM values of images 2.1(h)-(j) that SSIM is also sensitive to translation, scaling and rotation. This contradicts the underlying goal of SSIM because translation, scaling and rotation are non-structural distortions and yet SSIM is sensitive to these.

2.2.4 Complex Wavelet Structural Similarity Index

The complex wavelet structural similarity index (CW-SSIM) was introduced in [19]. The idea in this method was that when a object is translated, scaled or rotated slightly, the edges of the object move uniformly in the same direction. Hence the method takes the phase information into consideration more than the magnitude information. The local phase measurements in the complex wavelet transform domain gives the information about the movement of edges. If a group of neighbouring wavelet coefficients shift consistently, then it signifies that the edge has moved in that direction. Taking two complex wavelet coefficients, c_x and c_y which are extracted at the same spatial location in the same wavelet subbands of two images x and y, the CW-SSIM index can be calculated by using,

$$\tilde{S}(c_x, c_y) = \frac{2\left|\sum_{i=1}^{M} c_{x,i} c_{y,i}^{*}\right| + K}{\sum_{i=1}^{M} |c_{x,i}|^2 + \sum_{i=1}^{M} |c_{y,i}|^2 + K'} \tag{2.4}$$

where c^* is the complex conjugate of c and K is stabilizing constant which is a small positive value. When there are no structural distortions between the images being compared, then the CW-SSIM index would be 1. The CW-SSIM ranges between 0 to 1, where 1 signifies no difference between the two images. Similar to the SSIM method, this too uses a sliding window approach to compute the CW-SSIM locally first at a subband and then the global CW-SSIM is computed by averaging over all the subbands. From Figure 2.1, it can be seen that the CW-SSIM worked very well. The images with structural distortions had lower scores compared to non-structural distortions (including small translation, rotation and scaling). CW-SSIM eliminates the need for any preprocessing steps that was used to align the objects due to its insensitivity to spatial distortions. This metric is not suitable for our problem because it is tailored to be insensitive to contrast or luminance which would add to the computation cost. But in this work the templates are all under unvarying light conditions so the computations would not be necessary.

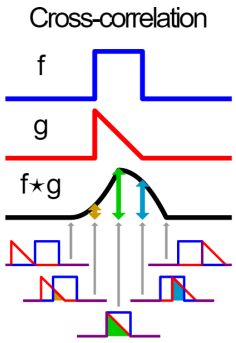


Figure 2.2: Visual representation cross correlation between two signals. Image borrowed from [2].

2.2.5 Normalized Cross Correlation

VI does template matching in real time so it requires the computation to be quick. For an image of size $M \times N$ and template of size $m \times n$, the cross correlation metric was calculated to have a computational complexity of $(m \cdot n \cdot M \cdot N)$ [20] which was tested to determine that the complexity was small enough for VI to handle. Cross-correlation is a measure of similarity between a signal and a template. It can be calculated by,

$$(f * g)(t) = \int_{-\infty}^{\infty} f(x)g(t+x)dx \tag{2.5}$$

where f is the signal and g is the template. The value of cross correlation is the maximum when the signal and template match. A visual representation of cross correlation is shown in Figure 2.2. But in image processing a difficulty arises while using equation 2.5. It is possible that the correlation score between the template and the exact matching area in the image might be lesser than the score between the template and a bright spot in the image.

To overcome the difficulty and for ease of scoring, normalized cross correlation is preferred for use in image processing applications. Hence the cross correlation score will be limited to a value between 0 and 1. The formula for normalized cross correlation is,

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y').I(x+x',y+y'))}{\sqrt{\sum_{x',y'} T(x',y')^2 . \sum_{x',y'} I(x+x',y+y')^2}} \tag{2.6}$$

where I and T stands for image and template respectively. The template is slid over across the image and the cross correlation score when the template is at position I(x,y) is calculated by using equation 2.6. x' and y' represent the pixel coordinates of the template.

2.2.6 Weighted Normalized Cross Correlation

Normalized cross correlation works by calculating the score giving equal importance to all the pixels. This is a problem in cases where the template may match to unwanted artifacts in the image. However, by specifying which pixels in the image are important it is possible to reduce matching to unwanted artifacts. Importance is given to a pixel by giving it more "weight" and de-weighting the less important pixels. Hence a separate image known as the weight image is used to specify the weight of each pixel. Normalized cross correlation calculated using a weight image is called as weighted normalized cross correlation. For two 1-D vectors x and y of length n with weight vector w of the same length,

$$m(x; w) = \frac{\sum_i w_i x_i}{\sum_i w_i} \tag{2.7}$$

$$cov(x, y; w) = \frac{\sum_i w_i (x_i - m(x; w))(y_i - m(y; w))}{\sum_i w_i} \tag{2.8}$$

$$corr(x, y; w) = \frac{cov(x, y; w)}{\sqrt{cov(x, x; w) cov(y, y; w)}} \tag{2.9}$$

Equation 2.7 calculates the weighted mean of a vector x. Equation 2.8 is used to calculate the weighted covariance between the two vectors. Using these two formulas the weighted cross correlation is computed using equation 2.9. This method uses 3 convolutions between template and image sized matrices while normalized cross correlation uses only 2. Hence it is computationally more expensive. The formula stated above for weighted normalized cross correlation is modified to apply for 2-D matrices. The tool developed in this work makes use of 3 variations of this method which is described below.

2.2.6.1 Gaussian Weighted Normalized Cross Correlation

Gaussian weighted cross correlation uses a Gaussian image as the weight image. Gaussian image is nothing but a Gaussian kernel which is the same size as the template. The idea is that the part to be inspected is going to be centered in the template image, hence the pixels in the center needs to weighed more than the pixels in the perimeter. The Gaussian image would have the highest value at the center of the image and reduce gradually moving away from the center. The standard deviation for the kernel is chosen based on the dimensions of the image.

2.2.6.2 Average Difference Weighted Normalized Cross Correlation

In average difference weighted cross correlation, first the average image for each class is computed. Average image of a class is calculated by first taking the first two images in the class and aligning them and averaging the pixel intensities of the overlapping area to create a new image which is the average of the first two images. This new image is then coupled with the third image to create the average of the first three images and this continues till a single average image of all the images in the class is obtained. Once the average image of two classes are obtained, the difference between the two average images is calculated to create a difference image. Unfortunately this method is applicable only for 2 classes cases. The difference image would emphasize the area where the part may be located in the template image.

2.2.6.3 Cumulative Difference Weighted Normalized Cross Correlation

Cumulative difference wighted cross correlation uses a difference image as well. Here the difference between one image and every other image is calculated and the difference images are cumulatively added. By doing this, the area of the object in the template would be highlighted. While average difference image would give a precise shape of the object, this method would result in a image giving importance to the area where the object would occur. This is useful in problems where the object appears in odd angles.

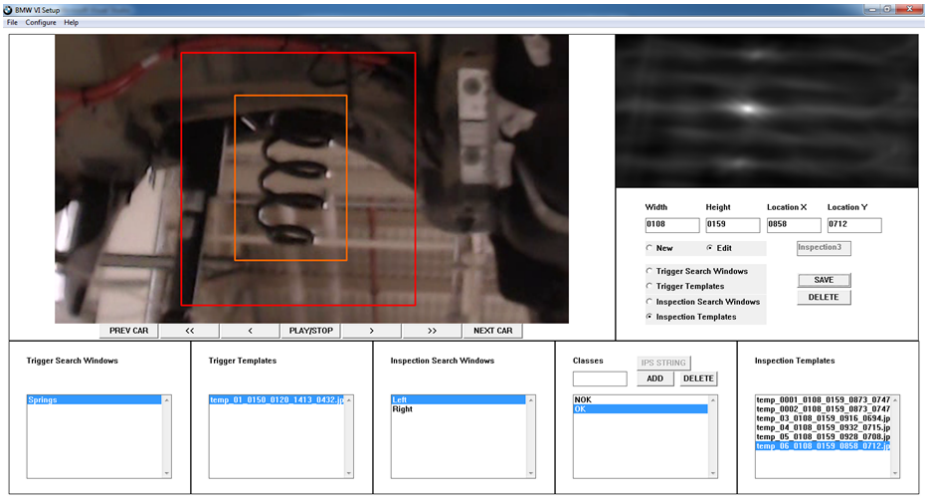


Figure 2.3: An example of a setup for good localization.

2.3 Localization Viewer

The localization viewer assists the user in outlining the trigger search window, trigger template, inspection search window and inspection template. As mentioned earlier, it was found that a tool to guide the user in best practices to delineate training data would be beneficial.

Figure 2.3 show the setup window with the video pane on the left side and the cross correlation window on the top right. The video pane shows the inspection search window and inspection template. Based on the defined inspection window and template, the cross correlation window shows the result of cross correlation of the template across the search window. The point of best match can be identified with a bright dot. In the example we can see that the search window and template have been defined in such a way that the bright dot appears only at one point in the center. This means it is a good localization.

Figure 2.4 shows a case where the small size of the inspection template is resulting in high matching probability across the window. Localization is difficult in this case as there are a number of bright spots seen in the cross correlation window.

In Figure 2.5, the inspection template size is increased compared to the size of the

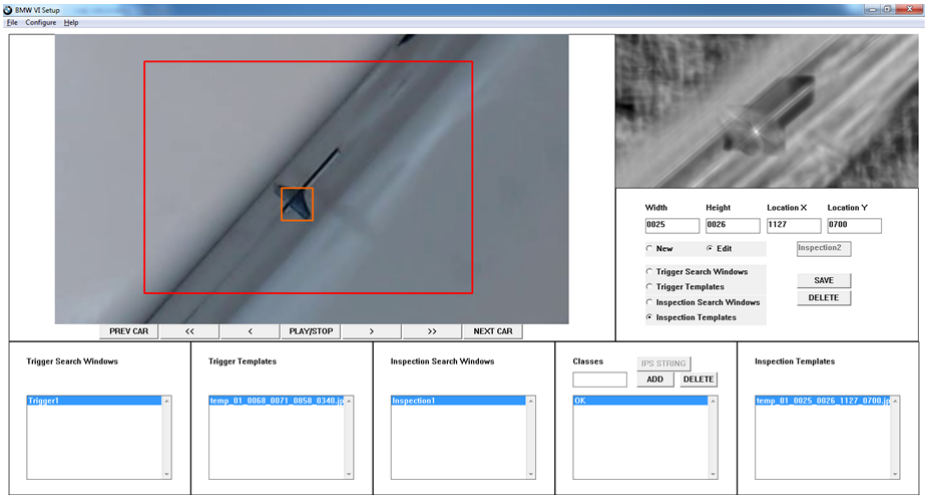


Figure 2.4: An example of bad localization where the template size is too small causing many matches across the search window.

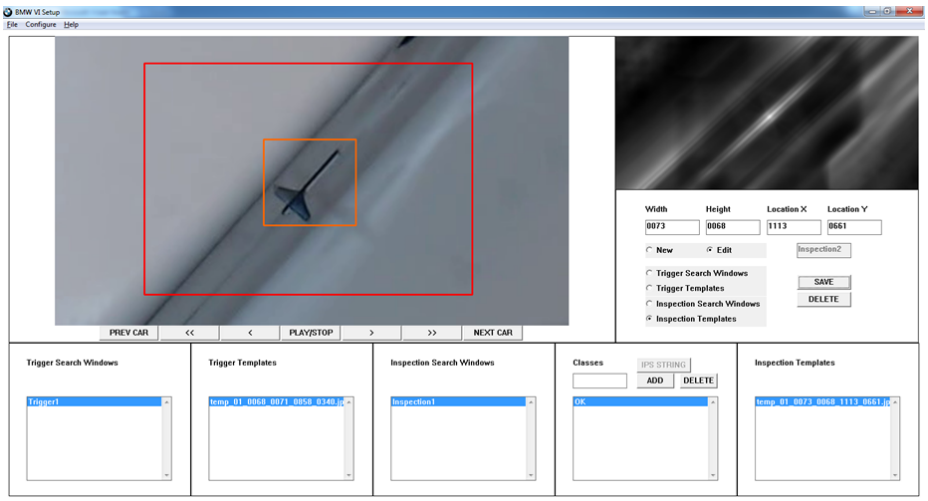


Figure 2.5: Result of changing the size of the template making localization better.

template in Figure 2.4. It can seen in the cross correlation window that the localization has improved and the template matches the best in the center. But there are more than one bright spots meaning that the template still partially matches few more areas.

2.4 Image Similarity Viewer

Image similarity viewer is used to ascertain if a certain inspection problem is suitable for a VI system. The tool looks at the templates that have been stored once the training is done and provides a visual feedback to the user. The visual feedback is in the form of a scatter plot. Each point on the plot represents a template from the inspection problem. Once the system has been trained, an inspection folder is created containing a folder for each class. Each class folder contains one or more templates representing their classes.

The steps to operate this tool are as follows. First, the inspection folder containing the classes are loaded by choosing File $\rightarrow$ Load Inspection. This displays a few template images from each class along with the class name as seen in Figure 2.6. Second, the gradient images are calculated by selecting Compute $\rightarrow$ Gradient. On selection, the template images disappear and in its place their gradient images are displayed. This is shown in Figure 2.7. Using this the user can observe which edges are being highlighted well. As shown in Figure 2.8, under Compute $\rightarrow$ Cross correlation the user has the option to choose from 4 different types of cross correlation: normalized, Gaussian weighted, cumulative difference weighted and average difference weighted. Each option results in plot showing how well the templates would work for the inspection problem. The user can check the result of each method to verify which type of correlation would work best and determine the success of the inspection problem. The final result will look like the screenshot shown in Figure 2.9. The details of each step are discussed below.

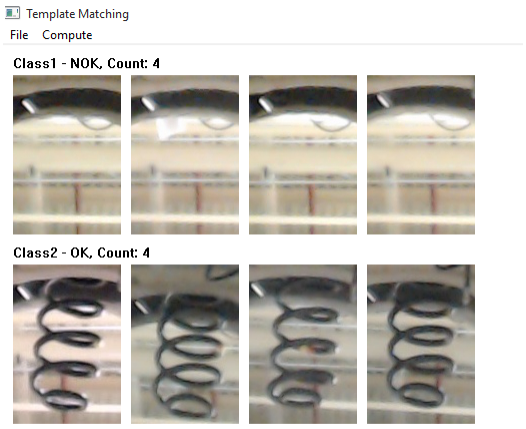


Figure 2.6: Screenshot of the image similarity viewer after loading the inspection folder. A few templates from each class are displayed in the window.

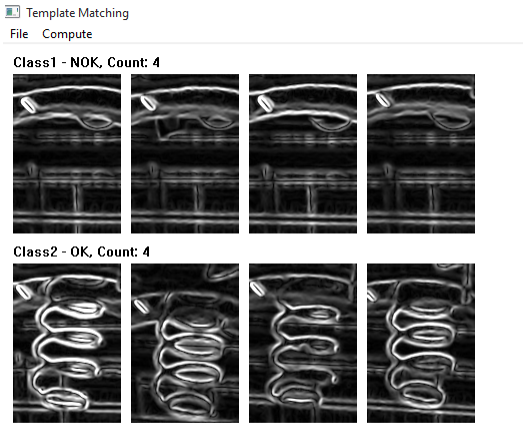


Figure 2.7: Screenshot of the image similarity viewer once the gradient has been computed. A few gradient images of the templates previously loaded are displayed.

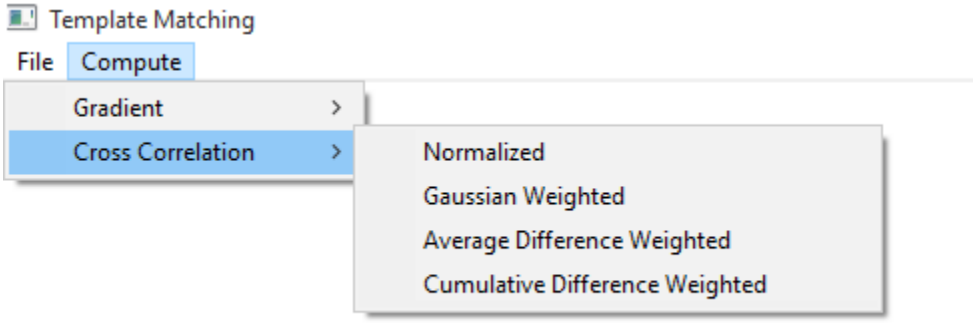


Figure 2.8: The 4 types of image similarity metrics available for the user.

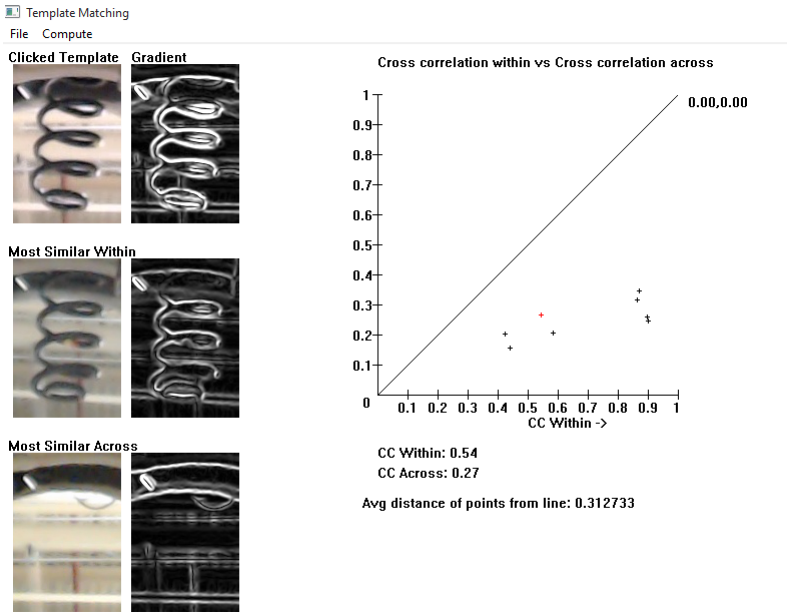


Figure 2.9: Screenshot of the image similarity viewer after the user has selected a metric to test. The point clicked on the plot is highlighted in red and its corresponding template along with the best match within the class and best match across the class are displayed next to the plot.

2.4.1 Gaussian Blur

Every image has undesired noise and most edge detection methods are sensitive to noise. The aim of this step is to reduce the noise by smoothing the image. This helps the edge detection in ignoring the small unnecessary details in the image. Gaussian blur does not alter the geometrical properties of the features and the smoothing effect resembles that of viewing the image through a translucent screen.

The Gaussian function in 1-D is defined as,

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \tag{2.10}$$

The Gaussian blur uses this Gaussian function to calculate the transformation that is applied to each pixel. But since we are dealing with images, we need a 2-D Gaussian function. The 2-D Gaussian function is nothing but the product of two 1-D Gaussian functions. Each 1-D Gaussian function applies to each dimension. The 2-D Gaussian function found in [21] is as follows,

$$G(x, y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.11}$$

where x and y point to the pixel location along the x and y-axis respectively. σ refers to the standard deviation of the distribution. A 2-D Gaussian distribution is shown in Figure 2.10. The standard deviation determines the level of smoothing the image. Higher the value of sigma, more the smoothing but high sigma values require more calculations per pixel. The points greater than 3σ distance from the center of the template does not influence further calculations because of Gaussian smoothing. In the output array, the pixel's value would be the weighted average of the neighboring pixels where the weight of the neighbors decreases as the distance between these pixels and the center pixels increase.

In this paper, a 3x3 Gaussian kernel is used to smooth the image so that the important features are retained. The value of standard deviation is taken as 0.8 which means that points greater than 2.4 pixel distance from the center pixel have no influence. A small

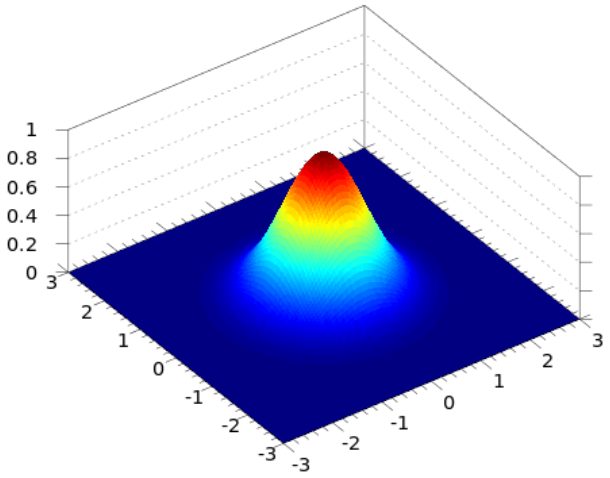


Figure 2.10: 2-D Gaussian Distribution.

standard deviation is chosen so that it does not smooth the edges too much and retains wide edges and smooths the thin edges.

2.4.2 Sobel Gradient

The final step of feature extraction is deriving the edges from the image. After the image has been smoothed, we use the Sobel operator to extract the edges so that the similarity metrics operate on part appearances rather than colors. The Sobel operator or Sobel filter is used to produce an image that contains only the strong edges detected in the original image. The Sobel filter convolves the image with two kernels for the horizontal and vertical direction. It computes an approximation of the gradient of the image intensity function. In this paper, we use two 3x3 kernels to convolve with the image. The two kernels are,

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * I \tag{2.12}$$

(a) Original image. (b) Gradient image.

Figure 2.11: Example of feature extraction.

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * I \tag{2.13}$$

where I is the image to operated. The vertical changes in the image are computed using 2.12 where the 3x3 kernel is convolved with image and the horizontal changes are computed using 2.13 where the transposed 3x3 kernel is convolved with the image. Once we obtain G_x and G_y, we can calculate the magnitude of the gradient using,

$$G = \sqrt{{G_x}^2 + {G_y}^2} \tag{2.14}$$

G is the gradient magnitude which is the image containing the edges.

An example of an image and its gradient image is shown in Figure 2.11. It can be seen in Figure 2.11b that the strong features from Figure 2.11a has been extracted.

2.4.3 Cross Correlation Within

Cross correlation within is when a template in a class is compared with all the templates within the same class. Since all the templates are of the same size and sometimes the object in the template is not centered, we pad the reference image in every comparison with zeros. Here we chose the width and height of the padding to be 20 percent of the

width and height of the image respectively. This way the second image can slide across the reference image and find the best offset where it matches.

The first image is taken and cross correlated with the rest of the templates in the same class and all the cross correlation scores are compared to find the template which matches the first template the best. This is done by comparing all the cross correlation scores and finding the highest score. Similarly every image is cross correlated with every other image within the class and the best matching images are found. Hence we have one cross correlation score for every template in the class. These scores are called the cross correlation within scores.

2.4.4 Cross Correlation Across

Similar to cross correlation within, the reference image is padded with zeros to account for variability in position of object. In this case, an image from one class is taken and cross correlated with the images from every other class to find the best match in another class. But the images are not compared with images in the same class. So after every image is cross correlated and the best match across classes are found, every image will now have one more cross correlation score. These scores are called the cross correlation across scores.

2.4.5 Scatter Plot

Now we have two values for every image: cross correlation within and cross correlation across. The X-axis of the plot is cross correlation within and the Y-Axis of the plot is cross correlation across. Since the correlation scores range from 0 to 1, the X and Y axes will range from 0 to 1. Then a line is drawn from (0,0) to (1,1) splitting the graph into two areas. A sample plot is shown in Figure 2.12.

The cross correlation within and across scores are used to mark the points on the plot. So the number of points on the plot will be equal to the number of templates in all the classes in that inspection area. In an ideal case where there are no bad templates, the cross correlation within scores should be higher than the cross correlation across scores. So

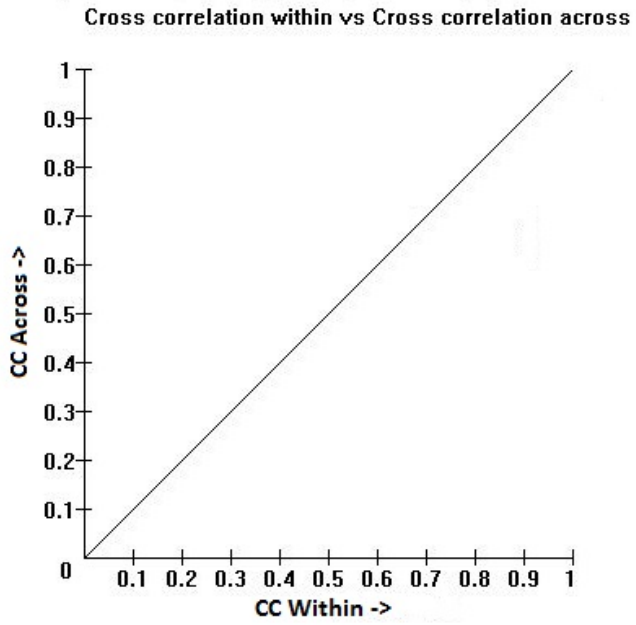


Figure 2.12: Sample plot.

all the points will be towards the bottom right of the plot. But if we have bad templates, what happens is that the cross correlation across matches better than the cross correlation within which is what is causing the error in classification. So images which have a cross correlation across scores equal to or greater than the cross correlation within scores, will have its point on or above the diagonal in the plot. When the point is clicked in the program, the corresponding images are displayed and the bad templates are found. Using this information, the user can observe and find out what is negatively affecting the scores and redraw the templates in a better way.

2.5 Image Similarity Metric Selection

Once the plots have been generated for all the metrics, the user now needs to determine which metric would work best for the problem. As mentioned earlier, ideally a plot having all or most of its points towards the bottom right should work the best. This is because we can find that to achieve a good discrimination between classes, the cross

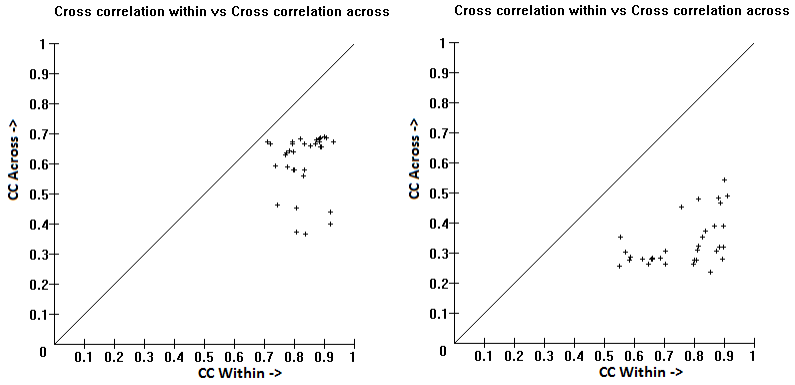


Figure 2.13: Plots of normalized cross correlation on the left and Gaussian weighted normalized cross correlation on the right for the wheel lock use case.

correlation within score should be significantly higher than cross correlation across. So with the plot have within score on the X-axis and across score on the Y-axis, the points should be at the bottom right. Hence the user will be able to approximately tell which metric would be best suited by looking at the plots. An example is shown in Figure 2.13 which are the plots of normalized cross correlation on the left and Gaussian weighted normalized cross correlation on the right for the wheel lock use case. It can be seen that the plot on the right has more points towards the bottom right which signifies that the Gaussian weighted cross correlation would work better than normalized cross correlation in discriminating the two classes.

2.6 Data

Table 2.1 gives the details of the various inspection problems in the BMW Spartanburg plant. The part being inspected is listed along with the number of classes for the part. The number of templates gives the sum of templates from all the classes for a use case. These templates are used to test the image similarity viewer.

The spring coil, wheel lock, under body bolts and safety lock use cases are detection type of inspection. Hence they have only 2 classes which are present or absent. Figures

(a) Present class. (b) Absent class.

Figure 2.14: Example of a template from each class for spring coil use case.

(a) Present class. (b) Absent class.

Figure 2.15: Example of a template from each class for wheel lock use case.

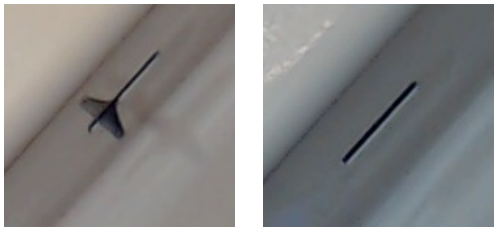
2.14 to 2.17 show a sample template from each class.

The door badge and door emblem cases are classification type. There are more than two classes possible. Both these use cases have a class named tape. Sometimes the workers miss removing the tape on the badge and emblem so VI had to treat this as a separate class to identify it. Otherwise it would result as missed inspection. Figures 2.18 and 2.19 show some sample templates from each class for door badge and door emblem respectively.

Cam0 and cam1 are two different cameras used in the same inspection area where there are two assembly lines. These two cameras point at the two assembly lines and are

(a) Present class. (b) Absent class.

Figure 2.16: Example of a template from each class for under body bolts use case.

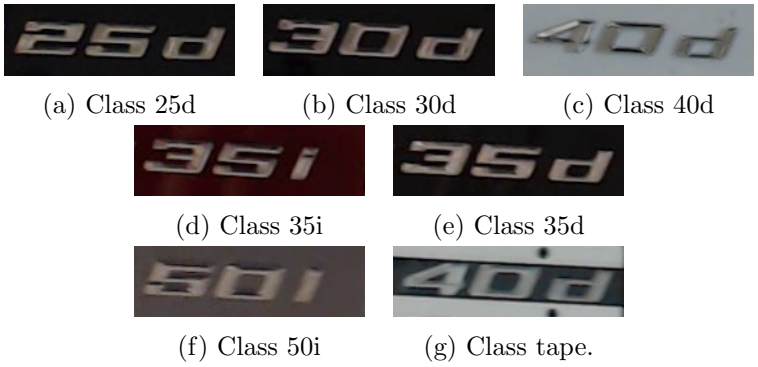


(a) Present class. (b) Absent class.

Figure 2.17: Example of a template from each class for safety lock use case.

(a) X-Drive class. (b) S-Drive class. (c) Tape class.

Figure 2.18: Example of a template from each class for door badge use case.



(a) Class 25d (b) Class 30d (c) Class 40d

(d) Class 35i (e) Class 35d

(f) Class 50i (g) Class tape.

Figure 2.19: Example of a template from each class for door emblem use case.

treated as separate inspection problems.

Usecase	Number of classes	Number of templates
Spring coil	2	8
Wheel lock	2	33
Under body bolts	2	7
Safety lock(cam0)	2	199
Safety lock(cam1)	2	194
Door badge(cam0)	3	15
Door badge(cam1)	3	16
Door emblem(cam0)	7	78
Door emblem(cam1)	7	115

Table 2.1: Details of each use case being used in the BMW Spartanburg plant.

2.7 Evaluation Metric

To evaluate the utility of the tool, the data listed in Table 2.1 will be used to generate the NCC vs WNCC plots. As explained earlier, the plots will be used to select a metric that works best for the data set. The metric which has most of the points towards the bottom right of the plot works best. This is because, in that area the CC within would be much higher than CC across which provides the maximum discrimination between the classes.

In some cases the user may not be able to select a metric by looking at the plots as the points may have moved just slightly or just few points might have moved. To quantify the results, the average perpendicular distance of all points to the line were calculated. The resultant number would clearly tell which metric has points further away from the line. If the point on the plot is at (C_x, C_y), then the corresponding point on line perpendicular to this point (x, y) can be found using,

$$x = \frac{1}{2}(C_x, C_y) \tag{2.15}$$

and y has the same value as x since the equation of the line x=y. Once the point on the

line has been found, the distance between the two points is calculated by,

$$d = \sqrt{(C_x - x)^2 + (C_y - y)^2} \tag{2.16}$$

Now this gives the distance of the point as a positive number even if the point lies on the left side of the line which would give incorrect results when trying to select a metric. So after (x,y) has been found, we check if the point on the plot lies towards the left and if so the the distance is given a negative value. Else it is positive.

Thus with the combination of the plot and the average distance of the points from the line, the user will be able to select the metric best suited for the problem.

Chapter 3

Results

The image similarity metrics described in the last chapter were tested on each of the use cases. The result for each use case is discussed in the following sections.

3.1 Spring Coil

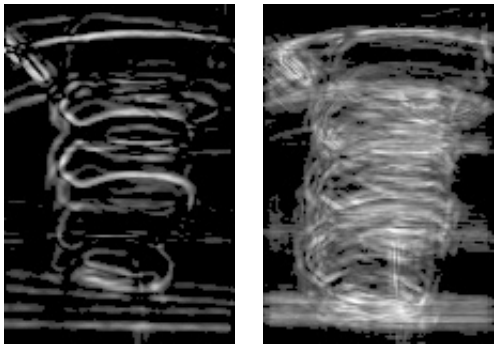
The spring coil use case is a case of detection. A sample image from each class is shown in Figure 2.14. The present class shown in Figure 2.14a shows the spring coil which is the object to be inspected. Figure 2.14b is the template of a missing spring coil so only the background can be seen. These templates were tested with the 4 metrics. The weight images used for average difference weighted normalized cross correlation and cumulative difference weighted normalized cross correlation is shown in Figure 3.1a and Figure 3.1b respectively.

The plot generated for the 4 metrics can be seen in Figure 3.2

Table 3.1: Average perpendicular distance of all the point to the diagonal for spring coil use case.

Normalized CC	Average Diff WNCC	Gaussian WNCC	Cumulative Diff WNCC
0.1242	0.2487	0.3127	0.2692

The quantified results for the metrics are listed in Table 3.1.



(a) Average difference image.

(b) Cumulative difference image.

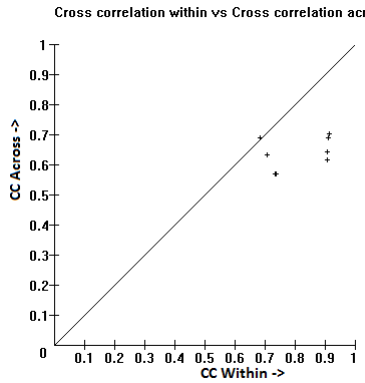
Figure 3.1: Weight images used for the spring coil use case.

From observing the plots, it can be seen that the Gaussian weighted normalized cross correlation has points that lie closest to the bottom right compared to the other metrics. Furthermore from Table 3.1, it can been seen that the Gaussian weighted normalized cross correlation has the highest score. This signifies that the best metric for the spring coil to discriminate the classes would be the Gaussian weighted normalized cross correlation.

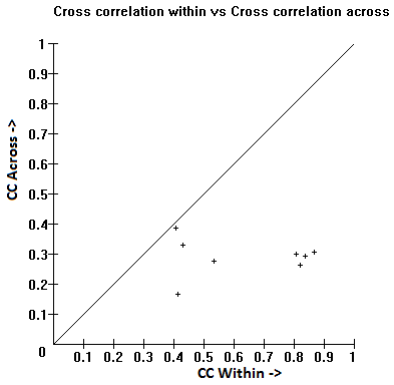
The challenge in the spring coil use case is the fact that the object is very thin so it is difficult to match the edges perfectly when the spring coil varies in appearance. Even though sliding window is being utilised, it cannot match the edges well enough to give good within cross correlation scores for the present class. Even though weight images were used to try and give more weight to edges of the springs, the weight image's strong pixels didn't match properly with the edges in the template thus still resulting in a low within score for the present class.

3.2 Wheel Lock

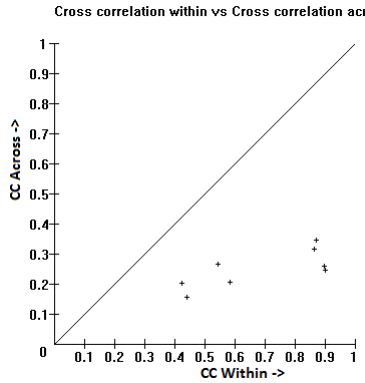
Sample templates from the two classes of wheel lock use case can be seen in Figure 2.15. The metallic object seen in Figure 2.15a is the wheel lock that is being detected. When the wheel lock is missing, the image would like Figure 2.15b. The weight images used for average difference weighted normalized cross correlation and cumulative difference



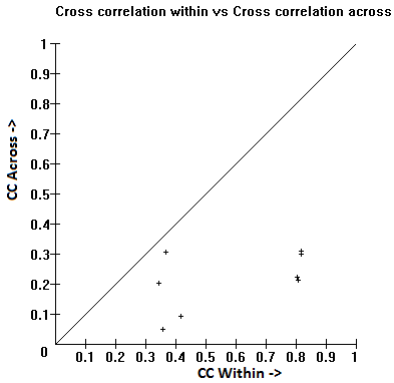
(a) Normalized cross correlation.



(b) Average difference weighted normalized cross correlation.



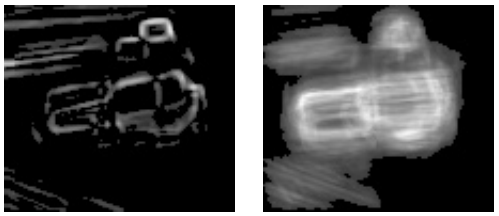
(c) Gaussian weighted normalized cross correlation.



(d) Cumulative difference weighted normalized cross correlation.

Figure 3.2: Plots of CC within vs CC Across for the spring coil use case for the 4 image similarity metrics.

weighted normalized cross correlation is shown in Figure 3.3a and Figure 3.3b respectively.



(a) Average difference image.

(b) Cumulative difference image.

Figure 3.3: Weight images used for the wheel lock use case

The plot generated for the 4 metrics can be seen in Figure 3.4

Table 3.2: Average perpendicular distance of all the point to the diagonal for wheel lock use case.

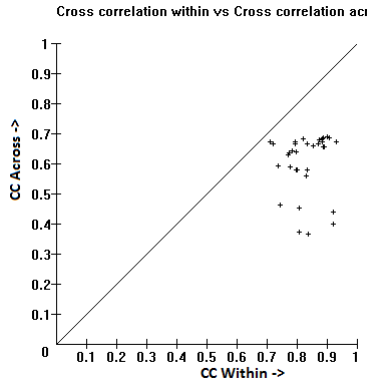
Normalized CC	Average Diff WNCC	Gaussian WNCC	Cumulative Diff WNCC
0.1606	0.2089	0.3052	0.2947

The quantified results for the metrics are listed in Table 3.2.

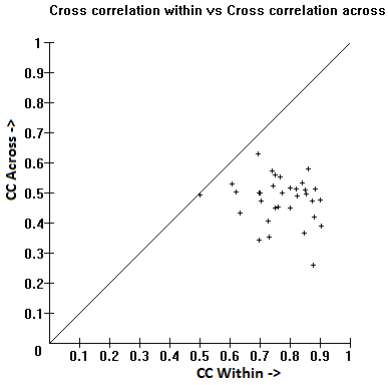
The plots show that the best metric would be either Gaussian weighted normalized cross correlation or cumulative difference normalized cross correlation. Since it is tough to narrow down the best result from the plots, we look at Table 3.2. It can be observed the values are very close between both these metrics but the Gaussian weighted metric is slightly better.

This use case gave results as expected due to the fact that the object being inspected was large and had minimum background information interfering in the similarity calculations. More over the weight images improved the results because it was possible to approximate an area where the object would appear in the template.

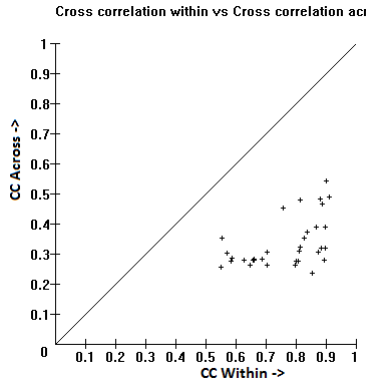
Hence the best metric for this problem would be the Gaussian weighted normalized cross correlation.



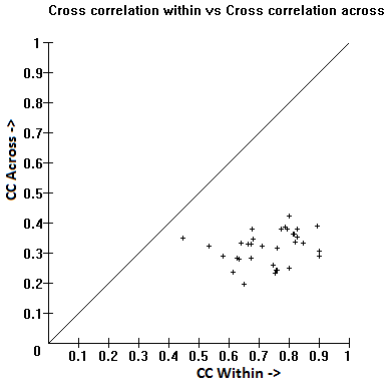
(a) Normalized cross correlation.



(b) Average difference weighted normalized cross correlation.

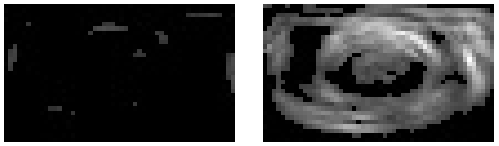


(c) Gaussian weighted normalized cross correlation.



(d) Cumulative difference weighted normalized cross correlation.

Figure 3.4: Plots of CC within vs CC Across for the wheel lock use case for the 4 image similarity metrics.



(a) Average difference image.

(b) Cumulative difference image.

Figure 3.5: Weight images used for the under body use case

3.3 Under Body Bolts

In this use case, the under body of the car is being inspected for missing bolts. The templates shown in 2.16 show the two classes where Figure 2.16a shows a template of an installed bolt and Figure 2.16b shows the template of a missing bolt. The weight images used for average difference weighted normalized cross correlation and cumulative difference weighted normalized cross correlation is shown in Figure 3.5a and Figure 3.5b respectively. The plot generated for the 4 metrics can be seen in Figure 3.6

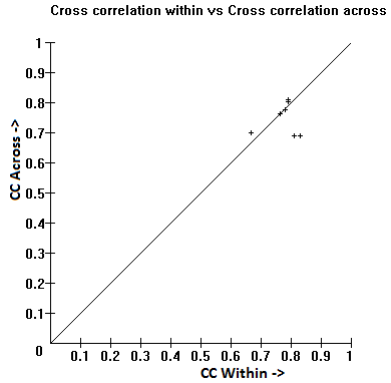
Table 3.3: Average perpendicular distance of all the point to the diagonal for under body use case.

Normalized CC	Average Diff WNCC	Gaussian WNCC	Cumulative Diff WNCC
0.0223	0.0192	0.0287	0.0396

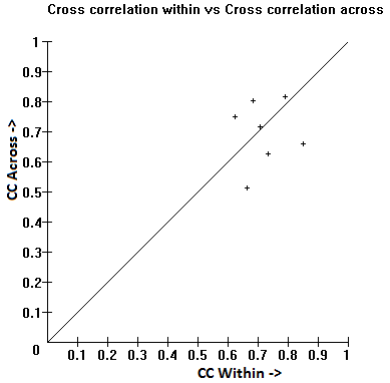
The quantified results for the metrics are listed in Table 3.3.

For this use case, the plots were hard to read as the points were lying on both sides of the line. But based on the quantified data it was found that among the 4 metrics, cumulative difference weighted cross correlation would suit this problem the best.

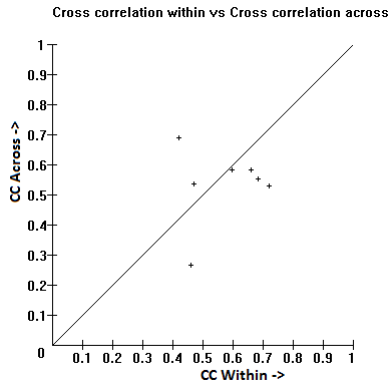
The plots were unreliable due to the fact that the part being inspected is very small and the image quality is making it harder to extract the edges of the object properly. Also the bolts seem to in a color very close to it background which reduces the strength of the edges considerably. Hence the feature extraction stage needs to be improved to identify the object better.



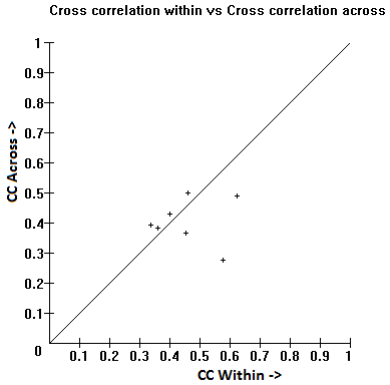
(a) Normalized cross correlation.



(b) Average difference weighted normalized cross correlation.



(c) Gaussian weighted normalized cross correlation.



(d) Cumulative difference weighted normalized cross correlation.

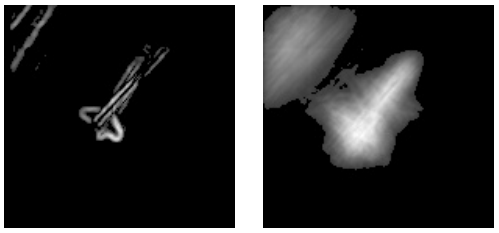
Figure 3.6: Plots of CC within vs CC Across for the under body use case for the 4 image similarity metrics.

3.4 Safety Lock

Safety lock use case has two cameras so these two cameras are treated two different use cases. The templates in Cam0 and Cam1 are similar with the present class having a safety lock as seen in Figure 2.17a and absent class would be missing the safety lock as seen in Figure 2.17b. The difference between Cam0 and Cam1 is the fact that the safety lock appear in different angles.

3.4.1 Cam0

The weight images used for average difference weighted normalized cross correlation and cumulative difference weighted normalized cross correlation is shown in Figure 3.7a and Figure 3.7b respectively.



(a) Average difference image.

(b) Cumulative difference image.

Figure 3.7: Weight images used for the safety lock Cam0 use case

The plot generated for the 4 metrics can be seen in Figure 3.8

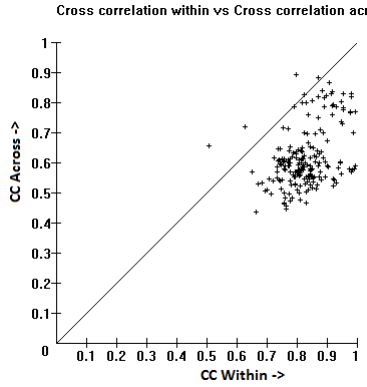
The quantified results for the metrics are listed in Table 3.4.

3.4.2 Cam1

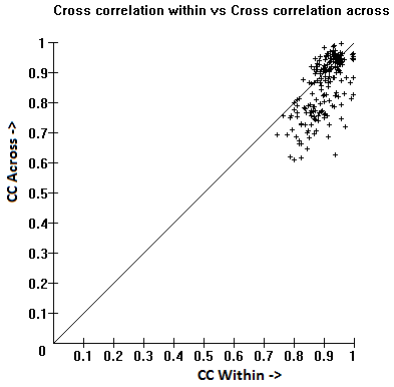
The weight images used for average difference weighted normalized cross correlation and cumulative difference weighted normalized cross correlation is shown in Figure 3.9a and Figure 3.9b respectively.

The plot generated for the 4 metrics can be seen in Figure 3.10

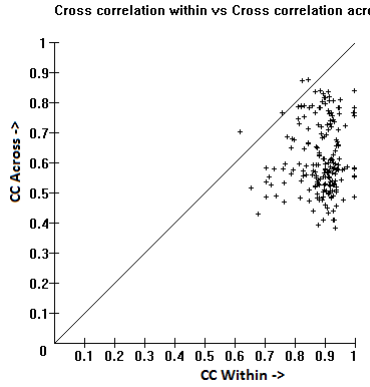
The quantified results for the metrics are listed in Table 3.4.



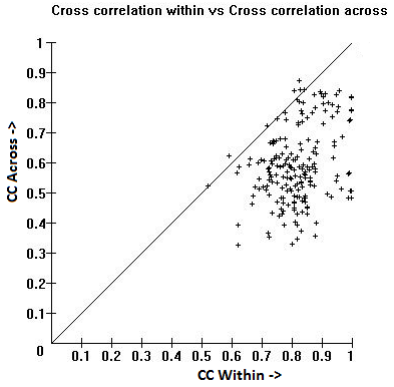
(a) Normalized cross correlation.



(b) Average difference weighted normalized cross correlation.

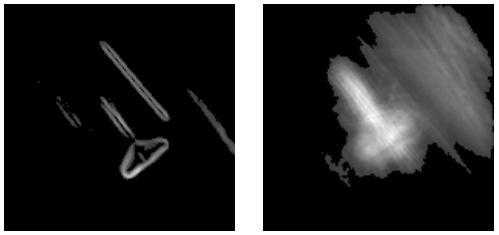


(c) Gaussian weighted normalized cross correlation.



(d) Cumulative difference weighted normalized cross correlation.

Figure 3.8: Plots of CC within vs CC Across for the safety lock cam0 use case for the 4 image similarity metrics.



(a) Average difference image.

(b) Cumulative difference image.

Figure 3.9: Weight images used for the safety lock Cam1 use case.

Table 3.4: Average perpendicular distance of all the point to the diagonal for safety lock use case.

Camera	Normalized CC	Average Diff WNCC	Gaussian WNCC	Cumulative Diff WNCC
Cam0	0.1526	0.0322	0.1936	0.1619
Cam1	0.1457	0.1091	0.2219	0.194

3.5 Door Badge

Similar to the safety lock use case, the door badge also 2 cameras. The 3 classes for this use case are X-Drive, S-Drive and tape. The tape class contains X-Drive and/or S-Drive templates with a tape over the badge. A sample template from each class can be seen in Figure 2.18. Since the average difference weighted cross correlation was meant only for 2 class comparison, this use case cannot be tested with that metric.

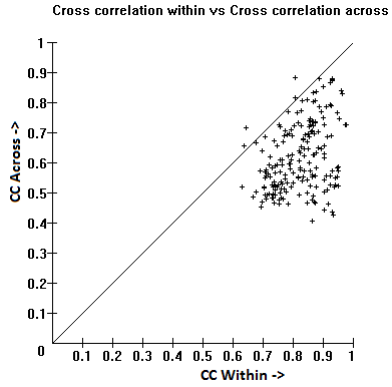
3.5.1 Cam0

The weight image used for cumulative difference weighted normalized cross correlation is shown Figure 3.11.

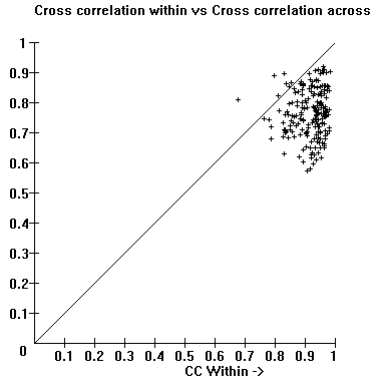
The plot generated for the 3 metrics can be seen in Figure 3.12

Here, the plots were not helpful in determining the best metric so we look at the table to see which would work best. From Table 3.5 it is found that normalized cross correlation would suit the best.

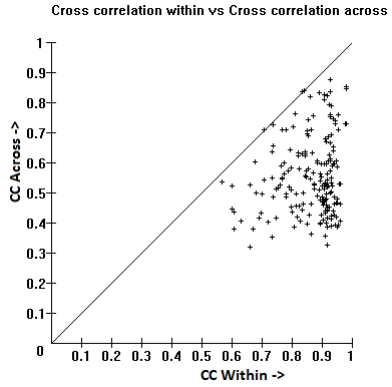
The points in the plot stayed in the proximity of the line due to the fact that the difference between the templates were insignificant. The templates consists of the badge



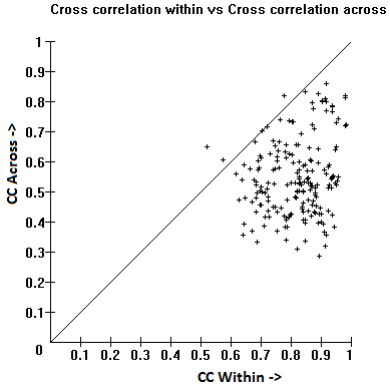
(a) Normalized cross correlation.



(b) Average difference weighted normalized cross correlation.



(c) Gaussian weighted normalized cross correlation.



(d) Cumulative difference weighted normalized cross correlation.

Figure 3.10: Plots of CC within vs CC Across for the safety lock cam1 use case for the 4 image similarity metrics.

Figure 3.11: Cumulative difference image for door badge Cam0 use case.

reading XDrive and SDrive and the only difference between the badges would be small patches of difference between X and S. Otherwise the cross correlation across score turns out to be high because the letter 'Drive' are common in all the classes and contribute a lot to the similarity between the classes. It was found from the gradient images that templates with silver badges on white background were not having strong edges.

3.5.2 Cam1

The weight image used for cumulative difference weighted normalized cross correlation is shown in Figure 3.13.

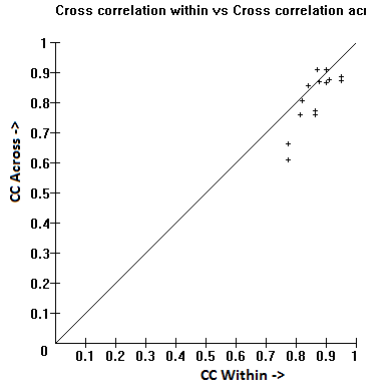
The plot generated for the 3 metrics can be seen in Figure 3.14

Table 3.5: Average perpendicular distance of all the point to the diagonal for door badge use case.

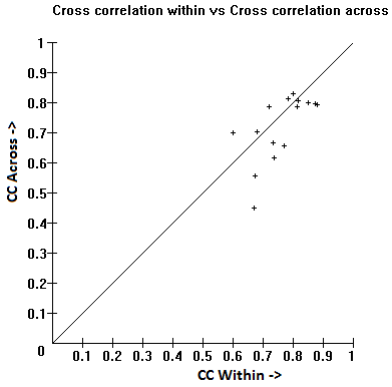
Camera	Normalized CC	Average Diff WNCC	Gaussian WNCC	Cumulative Diff WNCC
Cam0	-0.0098	n.a	-0.0486	-0.0733
Cam1	0.0364	n.a	0.0322	0.0155

The quantified results for the metrics are listed in Table 3.5.

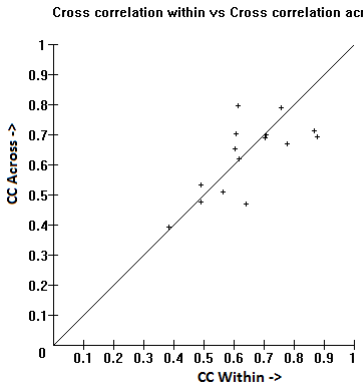
Similar to Cam0, we look at the table to find the best metric to normalized cross correlation again. The values in the table are negative due to the points that lie on the left side of the line in all the plots. Upon further investigation it was found that cross correlation across scores are 1 or close to 1 for few templates because the XDrive templates in tape class match perfectly with the XDrive templates from its class. Hence it was found that the tape class complicates the classification. Other than this, this use case has the same problems as Cam0.



(a) Normalized cross correlation.



(b) Gaussian weighted normalized cross correlation.



(c) Cumulative difference weighted normalized cross correlation.

Figure 3.12: Plots of CC within vs CC Across for the door badge cam0 use case for the 3 image similarity metrics.

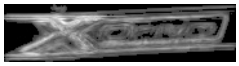


Figure 3.13: Cumulative difference image for door badge Cam1 use case.

3.6 Door Emblem

Door emblem use case consists of 7 classes. A sample template from each class is shown in Figure 2.19. 6 classes are different variations of the part and the 7th class is the tape class which has parts from the 6 classes with a tape over it. Again, this is 7 class use case so the average difference weighted normalized cross correlation does not apply.

3.6.1 Cam0

The weight image used for cumulative difference weighted normalized cross correlation is shown in Figure 3.15.

The plot generated for the 3 metrics can be seen in Figure 3.16

The quantified results for the metrics are listed in Table 3.6.

From the plots it is hard to select a metric because the cluster of points doesn't move away from the line. Although the data from Table 3.6 says that normalized cross correlation works the best.

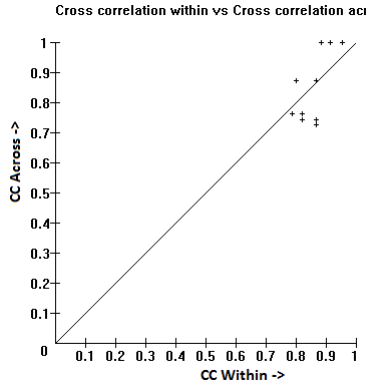
Using the image similarity viewer to check the individual cross correlation score led to the find that the discrimination between the classes is tough due to very small variation among them. The visual difference between '3' and '2' or 'd' and 'i' would be insignificant to provide good across correlation scores.

3.6.2 Cam1

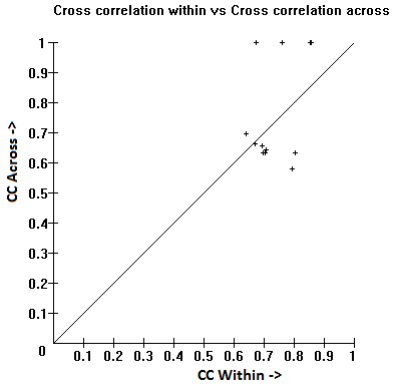
The weight images used for cumulative difference weighted normalized cross correlation is shown in Figure 3.17 respectively.

The plot generated for the 3 metrics can be seen in Figure 3.18

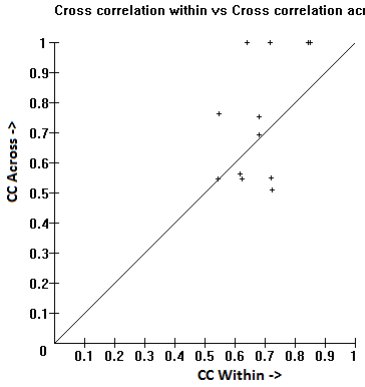
The quantified results for the metric are listed in Table 3.6.



(a) Normalized cross correlation.



(b) Gaussian weighted normalized cross correlation.



(c) Cumulative difference weighted normalized cross correlation.

Figure 3.14: Plots of CC within vs CC Across for the door badge cam1 use case for the 3 image similarity metrics.

Figure 3.15: Cumulative difference image for door emblem Cam0 use case.

Table 3.6: Average perpendicular distance of all the point to the diagonal for door emblem Cam1 use case.

Camera	Normalized CC	Average Diff WNCC	Gaussian WNCC	Cumulative Diff WNCC
Cam0	0.0122	n.a	0.0118	0.0168
Cam1	0.0064	n.a	-0.0041	0.0023

Here again the points in the plot tend to stay around the line and does not move down for any of the metrics. So based on Table 3.6, the best metric would be cumulative difference weighted cross correlation.

The problems faced in this use case are the same as those of Cam0.

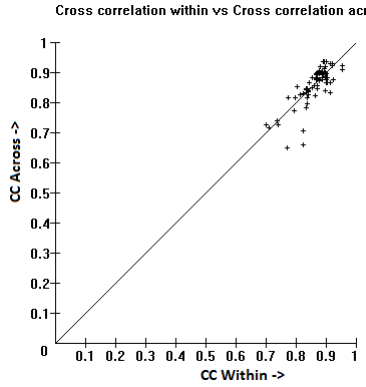
3.7 Summary

Table 3.7 gives the collective data of all the use cases.

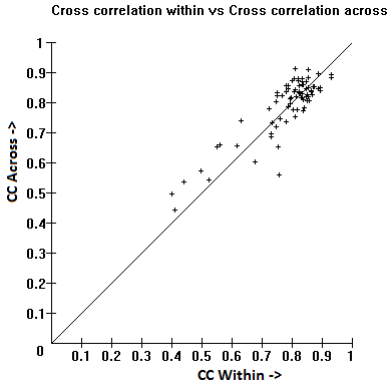
Table 3.7: Summarised results of average perpendicular distance of points to the line for all the use cases. The highest value among the 4 metrics are in bold.

Use case	Normalized	Average Difference	Gaussian	Cumulative Difference
Spring Coil	0.1242	0.2487	**0.3127**	0.2692
Wheel Lock	0.1606	0.2089	**0.3052**	0.2947
Under body bolts	0.0223	0.0192	0.0287	**0.0396**
Safety Lock(cam0)	0.1526	0.0322	**0.1936**	0.1619
Safety Lock(cam1)	0.1457	0.1091	**0.2219**	0.194
Door badge(cam0)	**0.0364**	n.a	0.0322	0.0155
Door badge(cam1)	**-0.0098**	n.a	-0.0486	-0.0733
Door emblem(cam0)	**0.0064**	n.a	-0.041	0.0023
Door emblem(cam1)	0.0122	n.a	0.0118	**0.0168**

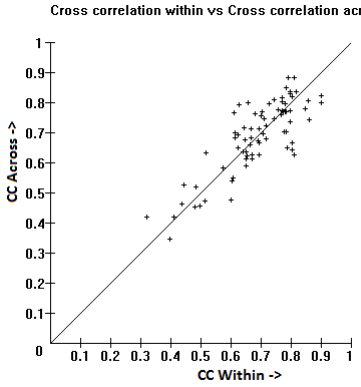
Hence using the image similarity viewer we were able to choose a metric that would suit a problem based on the plot and the quantified data. In cases where the points did not move to bottom right of the plot for any of the metric, the tool could be used to reason out why the templates weren't very effective in discriminating between the classes.



(a) Normalized cross correlation.



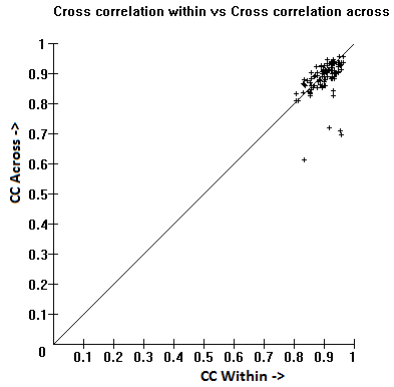
(b) Gaussian weighted normalized cross correlation.



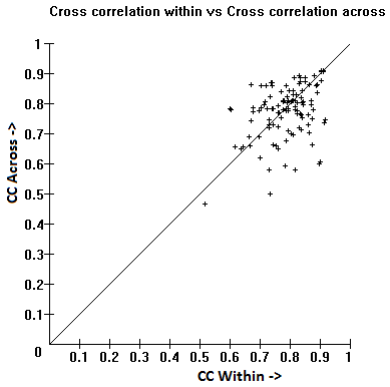
(c) Cumulative difference weighted normalized cross correlation.

Figure 3.16: Plots of CC within vs CC Across for the door badge cam0 use case for the 3 image similarity metrics.

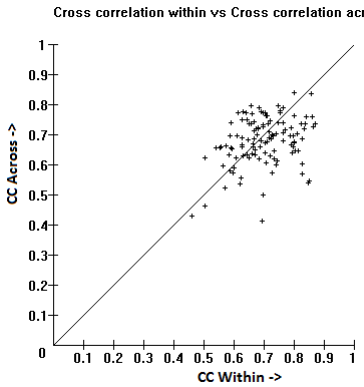
Figure 3.17: Cumulative difference image for door emblem Cam1 use case.



(a) Normalized cross correlation.



(b) Gaussian weighted normalized cross correlation.



(c) Cumulative difference weighted normalized cross correlation.

Figure 3.18: Plots of CC within vs CC Across for the door badge cam1 use case for the 3 image similarity metrics.

Chapter 4

Conclusion and Future Work

In this thesis, the difficulties faced by a user to setup the Visual Inspector and train data was detailed. It was found from previous deployments in the BMW Spartanburg plant that a tool to assist the user to delineate training data during the setup would help the user understand what kind of training data to choose for an inspection problem. To address this need, the Image Localization Viewer tool was developed. The tool shows the user the probability of a inspection template matching across the inspection window while outlining the templates in setup.

The Image Similarity Viewer was developed to provide a visual feedback to the user on the success or failure of an inspection problem based on the training data previously setup. The tool first shows the edge image for the user to understand how the object is being "seen" by the VI. Then the user is given 4 image similarity metrics to choose from to test an inspection problem. Each metric outputs a cross correlation within vs cross correlation across plot with the scatter points corresponding to each image. Depending on the location of the points, the success of the VI is determined. If one metric fails, the user can try another metric from the 3 remaining to check if that would help in the success of the inspection problem. If a training data seems to not suit any of the metrics, the user can select points on the plot to observe each image and its matches within the class and across the class to see why the classification would not work. The tool also gives the average

distance of all the points to the diagonal in the plot to quantify the results. Based on the observation, the user can go back and redraw the templates.

The Image Similarity Viewer was tested on 9 different use cases. It was observed that it was difficult to narrow down a single metric that would work for all the cases as each use case comprised of different parts varying in shape and size. As each case was tested, it was possible to conclude why an inspection problem would not work with a metric by looking at the plots and templates. In cases where none of the metrics worked, the plot again helped in showing the reason for failure. Thus the tool did show promising results in the lab but it would have to deployed for further testing and development.

Future work can be done be to improve the system. As of now the feature extraction stage makes use of just the Gaussian blur and Sobel filter to compute the edge images. Extra preproccessing steps can be added to isolate the object and eliminate background details. The variation in position and angle of an object was a major issue as cross correlation does pixel-to-pixel comparison. Even though the sliding window was implemented to compensate for position variations, the angle variations could not be accounted. There are image similarity metrics developed which makes use of wavelets to detect slight variations in rotation, scaling and translation which would help in this problem.

Bibliography

[1] Z. Wang and A. Bovik, "Modern image quality assessment," *Morgan and Claypool Publishers*, 2006.

[2] Wikipedia, "Cross-correlation — wikipedia, the free encyclopedia," 2015, [Online; accessed 2-December-2015]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Cross-correlation&oldid=689852591

[3] A. Hoover and J. Schulte, "Visual inspector user manual," 2015.

[4] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, 2004.

[5] A. Rehman and et al., "Image classification based on complex wavelet structural similarity," *Signal Processing-Image Processing Communication*, 2012.

[6] S. Chang and A. Hsu, "Image information systems: where do we go from here?" *IEEE transactions on Knowledge and Data Engineering*, vol. 4, no. 5, pp. 431–442, 1992.

[7] H. Tamura and N. Yokoya, "Image database systems: A survey," *Pattern recognition*, vol. 17, no. 1, pp. 29–43, 1984.

[8] V. Gudivada and V. Raghavan, "Content based image retrieval systems," *Computer*, vol. 28, no. 9, pp. 18–22, 1995.

[9] M. Swain and D. Ballard, "Color indexing," *International journal of computer vision*, vol. 7, no. 1, pp. 11–32, 1991.

[10] M. Stricker and M. Orengo, "Similarity of color images," in *IS&T/SPIE's Symposium on Electronic Imaging: Science & Technology*. International Society for Optics and Photonics, 1995, pp. 381–392.

[11] W. Ma and B. Manjunath, "Texture-based pattern retrieval from image databases," *Multimedia Tools and Applications*, vol. 2, no. 1, pp. 35–51, 1996.

[12] N. Kingsbury, "The dual-tree complex wavelet transform: a new efficient tool for image restoration and enhancement," in *Proc. EUSIPCO*, vol. 98, 1998, pp. 319–322.

[13] A. Mojsilovic, J. Kovaćević, J. Hu, R. Safranek, and S. Ganapathy, "Matching and retrieval based on the vocabulary and grammar of color patterns," *Image Processing, IEEE Transactions on*, vol. 9, no. 1, pp. 38–54, 2000.

[14] M. Kokare, B. Chatterji, and P. Biswas, "A survey on current content based image retrieval methods," *IETE Journal of Research*, vol. 48, no. 3-4, pp. 261–271, 2002.

[15] A. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-based image retrieval at the end of the early years," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 12, pp. 1349–1380, 2000.

[16] H. Nguyen and L. Bai, "Cosine similarity metric learning for face verification," in *Computer Vision*. Springer, 2011, pp. 709–720.

[17] M. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba, "Coding facial expressions with gabor wavelets," in *Automatic Face and Gesture Recognition*. IEEE, 1998, pp. 200–205.

[18] Z. Wang and A. Bovik, "Mean squared error: love it or leave it? a new look at signal fidelity measures," *Signal Processing Magazine, IEEE*, vol. 26, no. 1, pp. 98–117, 2009.

[19] Z. Wang and E. Simoncelli, "Translation insensitive image similarity in complex wavelet domain," in *In Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP05). IEEE International Conference on*. Citeseer, 2005.

[20] M. Tsai and C. Lin, "Fast normalized cross correlation for defect detection," *Pattern Recognition Letters*, vol. 24, no. 15, pp. 2625–2631, 2003.

[21] M. Nixon, *Feature extraction & image processing*. Academic Press, 2008.