DESIGN OF ONLINE CLASSIFIER FOR SURFACE DEFECT DETECTION AND CLASSIFICATION OF COLD ROLLED STEEL COIL

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology

in

ELECTRONIC SYSTEMS AND COMMUNICATIONS

by

ATHEEQUR REHAMAN MOHAMAD

Department of Electrical Engineering
National Institute of Technology, Rourkela
2013

DESIGN OF ONLINE CLASSIFIER FOR SURFACE DEFECT DETECTION AND CLASSIFICATION OF COLD ROLLED STEEL COIL

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology

In

ELECTRONIC SYSTEMS AND COMMUNICATION

By

ATHEEQUR REHAMAN MOHAMAD
ROLL NO: 211EE1094

Under the Guidance of

PROF. DIPTI PATRA

Department of Electrical Engineering
National Institute of Technology, Rourkela
2013

National Institute Of Technology
Rourkela

C E R T I F I C A T E

This is to certify that the thesis entitled, "**DESIGN OF ONLINE CLASSIFIER FOR SURFACE DEFECT DETECTION AND CLASSIFICATION OF COLD ROLLED STEEL COIL"** submitted by **Mr. ATHEEQUR REHAMAN MOHAMAD** in partial fulfillment of the requirements for the award of Master of Technology Degree in **ELECTRICAL ENGINEERING** with specialization in **"ELECTRONIC SYSTEMS AND COMMUNICATION"** at National Institute of Technology, Rourkela (Deemed University) is an authentic work carried out by him under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University / Institute for the award of any Degree or Diploma.

Date:

Prof. Dipti Patra

Department of Electrical Engineering

National Institute of Technology
Rourkela

ACKNOWLEDGEMENTS

With the deepest sense of gratitude I express my indebtedness to my honorable , esteemed supervisor Prof. Dipti Patra, Department of Electrical Engineering for her active participation, constant guidance, ablest supervision and moral support during this thesis work. She has opened doors for my bright future.

I cannot find words to express my gratitude to Prof. Sunil Kumar Sarangi Director NIT ROURKELA for his inspiration.

I am very much thankful to our Head of the Department, Prof.A.K.Panda, for providing me with best facilities in the department and his timely suggestions. I humbly acknowledge the creative criticism and helpful suggestions of Prof.P.K.Sahoo, Prof.Susmita.Das, Prof.K.R.Subhashini, Prof Supratim Gupta for providing a solid background for my studies.

I want to especially thank Prof. P.K.Nanda for the enormous amount of help and decision making . I also recognize the support and the help from Prof. Noorul Islam.

During difficult times Mr. Yogananda Patnaik, Mrs. Prajna Parimita Dash, Miss. Smita Pradhan gave me the moral support and guidance .They encouraged me for my passion in implementing ideas and to spend most of the time in IPCV Lab .

I wish to thank my friends Rama Raju Kuram, Krishna Reddy, Suresh, Kamisetty Ashok, Anusha Reddy Vemula and Pranathi , for their great company and support at good and bad times.

I would like to thank my parents, who taught me the value of hard work by their own example. My father is an inspiration to make things perfect. He is my true friend. My mom for her love and affection. I am also very grateful to my loving brother Shafiq, my sweet and caring sister Nazma Afreen.

I would like to thank all those who made my stay in Rourkela an unforgettable and rewarding experience.

Finally, my greatest regards to the Almighty Allah Subhanahu wa ta'ala for bestowing upon me the courage to face the complexities of life and complete this thesis successfully.

ATHEEQUR REHAMAN MOHAMAD

ABSTRACT

The target to be achieved through this project was primarily aimed at detecting the surface defects belonging to different classes in cold rolled steel coils. This was achieved through grabbing the images from the camera, here line scan camera is used which grabs 20 frames per second. Carrying out defect detection on these images and later classifying them.

We present a method to automatically detect and localize defects occurring on the surface. Defect regions are segmented from background images using their distinguishing texture characteristics. This method locates candidate defect regions directly in the DCT (Discrete cosine transform) domain using the intensity variation information encoded in the DCT coefficients. More precisely, defect detection employs DCT analysis of each individual non-overlapping region of the image to determine potentially defective blocks, which are further grown and merged to form a defect region on the image

The measure of detecting the surface edge information of cold-roll steel sheets has been investigated. This relies on the digital image processing toolbox of the mathematical software MATLAB, and the edge detecting experiment of the surface grayscale image has been conducted on the computer using open computer vision 2.0 with visual studio. The method of detecting defects such as black flecks and scratching has been realized in the thesis. The different effect of operators with the disturbance of noise has been compared; the performance of the LOG's edge detection ability varies due to the change of the parameter σ. Conclusions have been made that LOG operator maintains the satisfying performance with the disturbance of noise; smaller the σ is, less satisfying the smoothing ability , which maintains more details, whereas the smoothing ability is better with the loss of more details. Defect detection of cold-roll steel surface based on MATLAB has a quite satisfying performance.

In this thesis a computer vision based, a framework for steel surface defects detection and classification of cold rolled steel strips is implemented. We have designed online classifier for automatic defect detection and classification of defects. In this we measured statistical textural features using gray level co-occurrence matrix presented by Haralick and geometrical features are also calculated. The final decision SVM (Support Vector Machine) handles the problem of classification of the defect types[23].

We also proposed SVM voting strategy for the final decision that handles the problem of multiple outputs of a given input image with a specific defect type. In addition, this approach improves the classification performance. Experimental results demonstrate the effectiveness of the proposed method on steel surface defects detection and classification. In addition, the defect information is encoded in the image. An image viewer application is designed for decoding the defect information.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

1.1. Thesis Objective:

The target to be achieved through this thesis was primarily aimed at detecting the surface defects belonging to different classes of cold rolled steel strips. This was achieved through grabbing the images from the CCD camera. Here line scan camera is used which grabs 20 frames per second. Then carrying out defect detection on these images and later classifying the defect types[23].

Surface defect is one of the most crucial causes which affect the performance of cold-roll steel sheets. The traditional visual detection by human eyes is not an easy way to deal with the massive production of cold rolled steel coils . Besides, the testing results tend to be misled by observers. As a consequence, new detection methods need to be come up with improved online defect detection system. Defects can easily escape from being found out since image processing detection has the advantages of non-contact efficiency, visual intuition and intelligence, home and foreign scholars have deeply researched image detecting methods applying to the steel industry. The idea is that by acquiring images of steel production samples on the production line, the quality of steel productions can be controlled fairly, which takes the place of visual detecting by human.

1.2. Literature Survey:

1.2.1 The Issues in Surface Defect Inspection

At present there are commercially available products which can detect the presence or absence of surface defects at reasonable costs. However, this problem still remains an open research issue due to the difficulties faced during *Real Time Defect Detection*, Identification and *Localization* as well. The major obstacles in this area are mainly due to the computational costs, lack of expert knowledge in the defect feature selection or modeling, availability of proper defect samples etc. The following discuss these issues in further details [4].

A. High Data Throughput:

A typical Cold Rolled Mill (CRM) sheet is 1 to 3 m. Wide with a thickness of 1 to 5 mm. These steel sheets of endless length move continuously on the conveyer belt at a very high speed of about 20 m. Per second. Thus, it is a tough job for the inspection system to acquire and effectively process a high amount of data in a short amount of time.

B. Inter-class Similarity and Intra-class Diversity:

A single class of defect may vary widely in appearance and structure. Moreover, the members of one class of defect may closely resemble to that of the other class. So, the inspection system may easily get confused during the process of defect identification.

C. Large Number of Classes:

A typical defect identification scheme should deal with a large number of defect classes. It is not unusual to deal with a few dozen of defects.

D. Non availability of adequate imperfection imagery:

Another very significant problem encountered during the design and development of the inspection system is the non-availability of adequate imperfect imagery for feature extraction and machine learning[2][3]. The hazardous environment of the steel industry hampers the collection of imagery data. This problem is more acute due to the fact that the process of machine learning requires a very large amount of training data for proper identification of the high number of defect classes.

E. Dynamic Defect Populations:

Little changes and alterations in the manufacturing process may create an entirely new set of defect classes or may add up new types of defects to the existing ones. The inspection system should be intelligent enough to adapt itself with the changing defect population. Thus, the system should have the ability of online learning. The design of the inspection system is determined by the first four factors, which defines the necessary hardware and the algorithms for proper data processing. However, the fifth factor seems to be the most challenging one, which reflects the intelligence of the system in extending its capability with the changing environment.

As positions and shapes of CRM strips vary continuously during producing reflection of lights in different areas of steel strip. It leads to different backgrounds in different images captured by CCD cameras. Moreover the gray levels of background differ everywhere in one image because of uneven illumination and reflection .It is difficult to find out defects from non uniform and ever changing backgrounds due to illumination.

Effective algorithms for defect detection are developed which consists of following procedure:

i. Background Extraction: Backgrounds are found and separated from the original image captured by cameras
ii. Gray level calculation: Gray levels of backgrounds and original images are calculated for better feature extraction.
iii. Threshold decision and effect of σ : one or more threshold values and σ values are decided and applied to differential gray levels to determine whether pixels are in defect areas or background and pixels in defect areas are marked as suspicious pixels.
iv. Region of Interest (ROI): Searching and merging suspicious pixels and the defect area are called as ROI.

1.2.2 Existing Techniques of Defect Identification

The commercially available products approach the issue of surface inspection in a number of ways. These methods are mainly based on the procedures of edge detection [22], profile analysis or multi-resolution image processing. These procedures are briefly discussed in the following paragraphs.

The process of Edge Detection assumes that the perfect steel sheet is smooth enough. The steel surface image filters for removal of noises added up due to the acquisition process. The filtered image is then processed for multidirectional edge detection and proper thresholding to discard noisy edges. The edge map indicates the presence of defects, as a perfect steel sheet is enough smooth not to present edges. The edge map or the segmented image provides the structural features of the defects, which are marked by matching them from a previously stored feature database.

The main problem with this process of defect identification is that several defects may exhibit the same geometry, while the members of the same defect class can be structurally very different and thus they can't be distinguished by merely studying their structural appearance.

1.3 The online surface defect detection key problem:

With the development of cold rolled strips productive scale and enhancement of productive speed, traditional quality detection by human eyes becomes more unsuitable for surface detection of cold rolled strips. The issue of Quality Control is an important aspect of

today's highly competitive Industry. One important way to improve the quality of the end product is to inspect the output of each manufacturing process. However, Manual inspection of end products slows down the entire process as it becomes costly, time consuming and also may impact the effectiveness of human labor due to the hazardous atmosphere of steel industry.

Surface defect is one of the most crucial causes which affect the performance of cold-roll steel sheets. The traditional visual detection by human eyes is not an easy way to deal with massive steel productions. Besides, the testing results tend to be misled by observers. As a consequence, new detection methods need to be come up with and improved automated system to identify the exact defect position. Defects can easily escape from being found out since image processing detection has the advantages of non-contact efficiency, visual intuition and intelligence, home and foreign scholars have deeply researched image detecting methods applying to the steel industry. The idea is that by acquiring images of steel production samples on the production line, the quality of steel productions can be controlled fairly, which takes the place of visual detecting by human.

Currently defects detection methods based on image processing still have disadvantages such as low speed and resolution. So highly efficient and real-time detection is the key problem needs to be settled down during production

1.4 Requirement of Online Surface Quality Inspection System:

When producing cold rolled steel strips, because of high speed, the surface of strips cannot be easily inspected by human eyes, if there are some defects on the surface, it cannot be found by inspector, and all the defects will be sometimes, many impurities would be embedded in it, therefore, the surface condition would become too bad to be recognized. In order to inspect surface quality of hot rolled steel strips, the online surface inspection system must meet such performance indexes as followings:

1. Velocity requirement: it can work at the velocity 5-15m/s;
2. Detection size requirement: It can detect defects of strips with a size of 0.5mmx0. 5mm;
3. Work condition requirement: It can work online under any inner condition;
4. Maximum width requirement : The maximum detection width can meet 2000 mm;

5. Defect position requirement: It can record the exact position of defects;
6. Offline requirement: It can display history process of production, and have many offline functions such as offline query, offline analysis and offline statistics for defect analysis;
7. Detection rate requirement: The detection rate of common defects can meet 90% and about 85% for other defects;
8. Defect type Requirement: It can detect over 10 main defects

1.5. Defect types and their causes:

When producing steel plate and strip, there are always such defect types as scratch, felt horizontal texture, point felt, feather roll imprint, white spot, roll imprint, Edge folding, rust mark orange texture, emulsion mark, edge crack, scale, cracks, air bubble, and so on. The following will analyze the causes of these defects detailed[23].

1.5.1. Scratch

Scratch is a straight imprint on the surface of the steel plate and strip. It is created by the relative movement between two surfaces of steel plate and strip, or between a hard sharp object and one of the surface of the steel strip. Commonly scratch is concave and directional in shape and it is a bit light in black-white image[23].

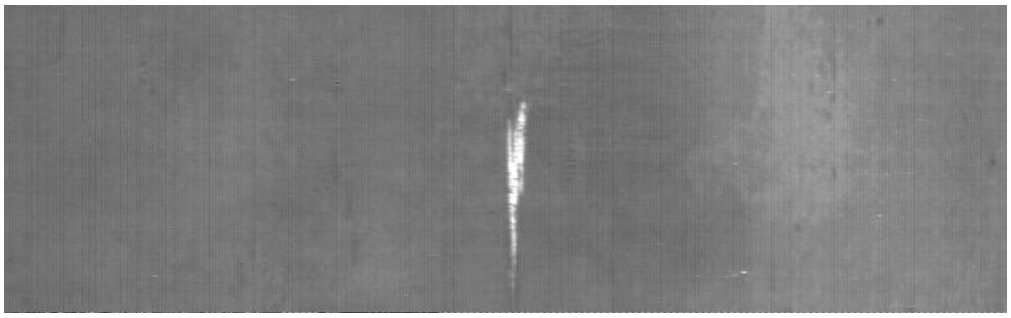


Figure 1.1: Defect type: Scratch in cold rolled steel strip

1.5.2. Fishtail mark:

Fishtail mark is horizontal wave texture along the edge of the steel plate and strip and it is also called edge pucker or edge folding mark. It is mainly caused by the sudden deduced thickness of the edge before temper rolling and extending not enough at the edge when temper rolling, or unfolding not enough in rolling detection this type defect always appears in cold rolled slight sheet[23].

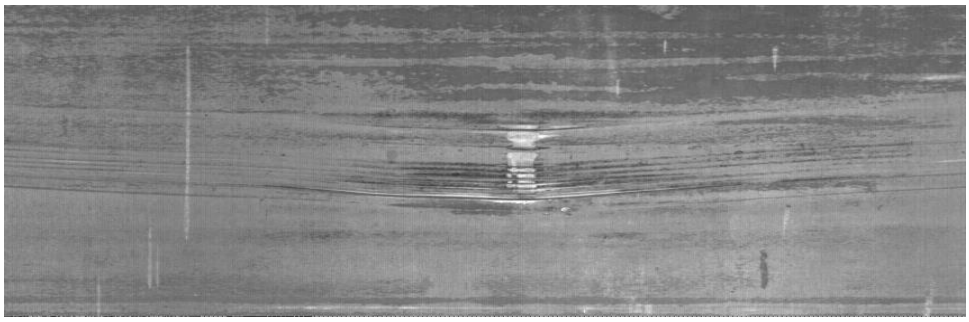


Figure 1.2: Defect type: Fishtail mark

1.5.3. Roll Mark:

Roll mark is a felt phenomenon with a shape of line on the surface of the steel strip. The main cause by bad shape or exceeded curl of the roller[23].

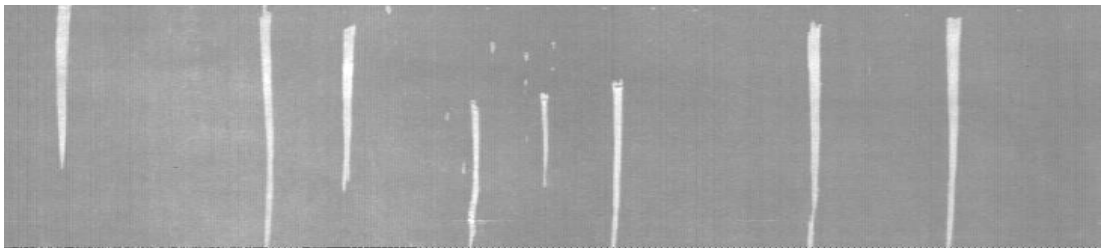


Figure 1.3: Defect type: Roll mark

1.5.4. White spot:

White spot is a white similar circular spot block on the surface of the steel strip its size is variable, but its border is smooth. It is primarily caused by white foam on the surface of plate and strip which has not been disposed immediately; it affects less on surface quality than surface appearance[23].

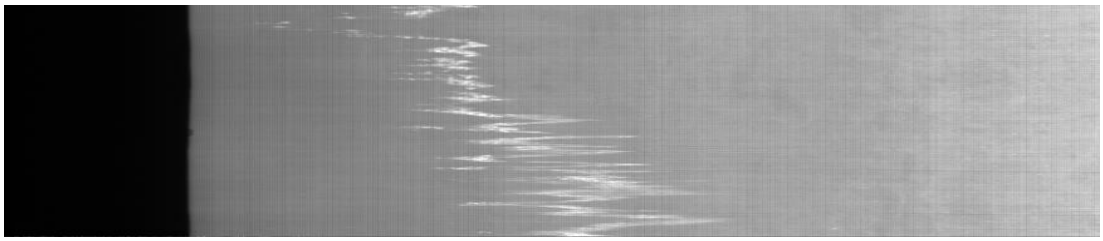


Figure 1.4: Defect type: White Spot

1.5.5. Foot Imprint:

Roll imprint is a periodic indentation on the surface of the steel strip, and appears more or less in a regular form. It is mainly caused by some hard and sharp foreign bodies or impurities on work rollers, and every round the roller rolls over, there will leave a same indentation on the surface of the steel strip.

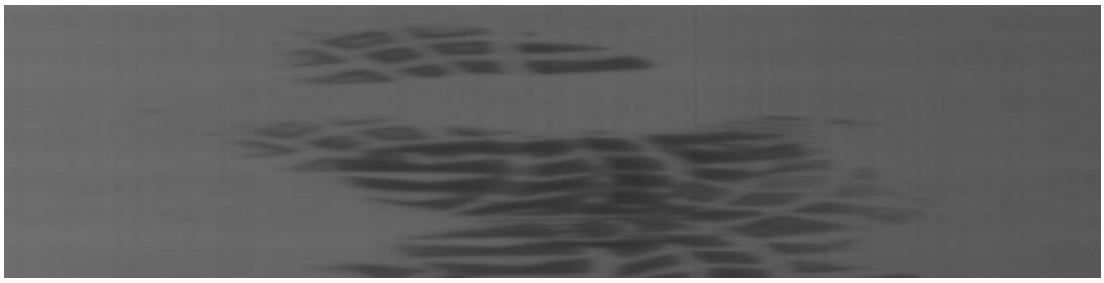


Figure 1.5: Defect type: Foot Imprint

1.5.6. Edge Crack:

Edge folding is a defect with a folding phenomenon on the edge of steel strip. It is caused by untidy curled edge[23].

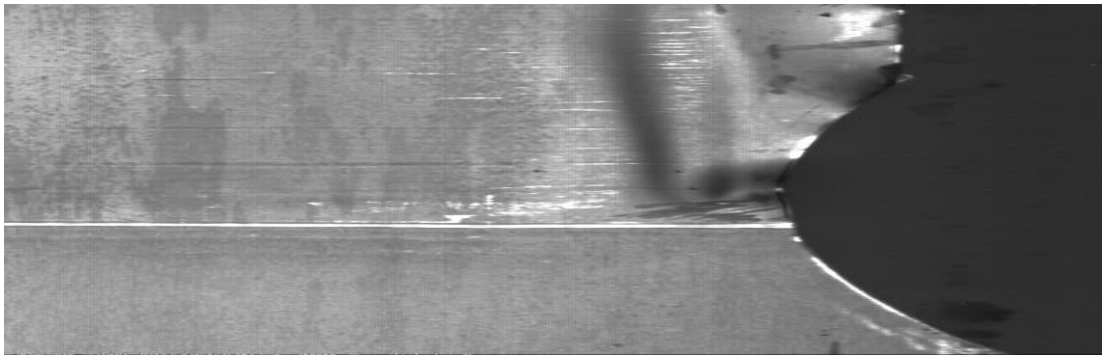


Figure 1.6: Defect type: Edge Crack

1.5.7. Oil Mark:

The oil drop occurs when the cold rolled steel strip contaminated with oily rolled at the temper pass mill. Oil drop is hard to erase[23].

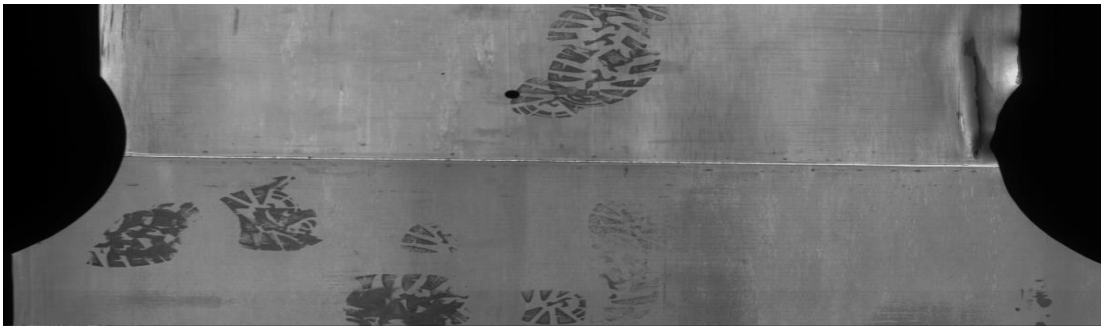


Figure 1.7: Defect type: Oil Mark

1.5.8. Scab:

Scab is a thin layer of corrosion products on the surface of the steel strip. It is caused by high air humidity or small drips on the surface under the circumstance of wide temperature fluctuation. If corrosion is very serious, the touch will be obvious, and it will be an irregular light spot in image[23].

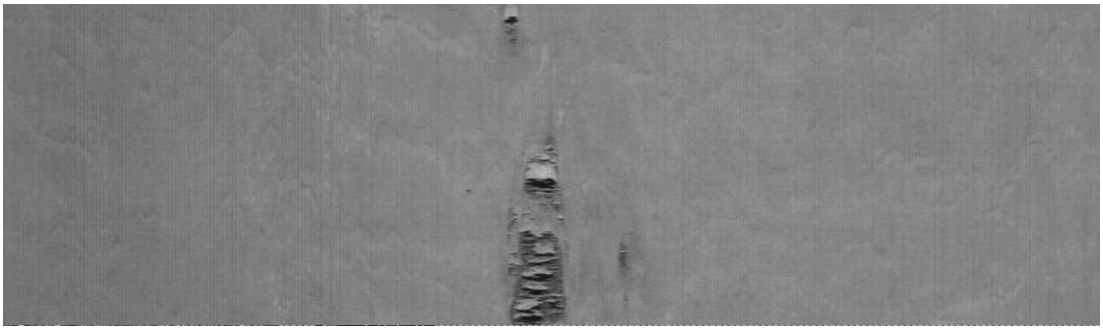


Figure 1.8: Defect type: Scab

Figure 1.9: Defect type: Scab

1.5.9. Scales:

Scales is a block of irregular rough surface like orange skin and is touchable. It is caused mainly by incompletely removed impurities and greasy dirt on work roller during temper rolling[23].

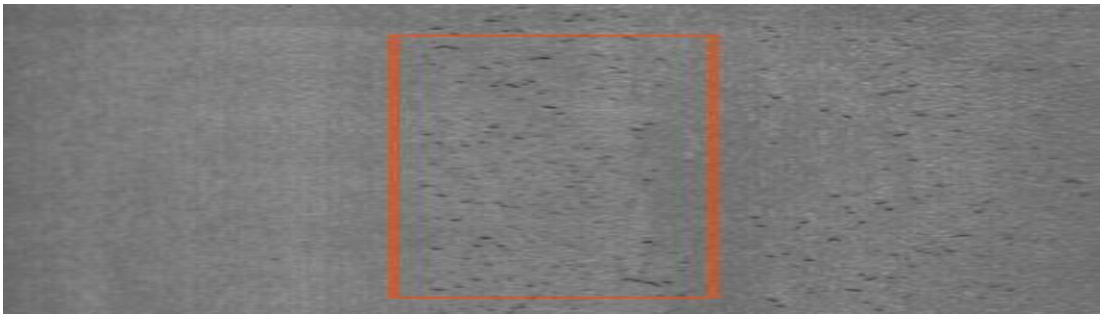


Figure 1.10: Defect type: Scales

1.5.10. Emulsion mark.

Emulsion mark is a block of untouchable spot. It is caused by the incompletely washed oiliness of earlier rolling working procedure, it is hard to be removed from the surface of steel strip, and it generally presents dark in the image[23].

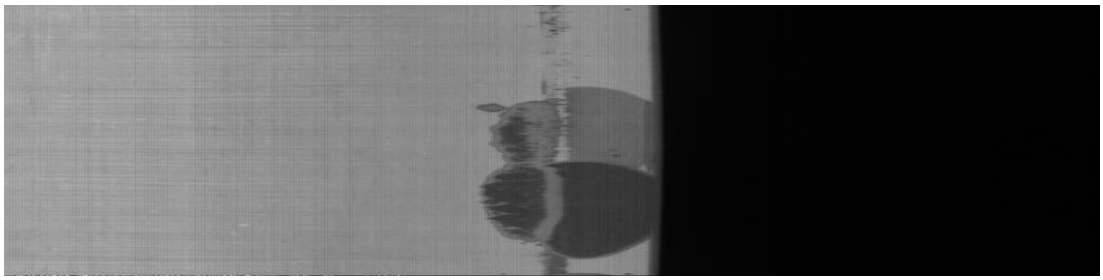


Figure 1.11: Defect type: Emulsion mark

1.5.11. Weld failure mark

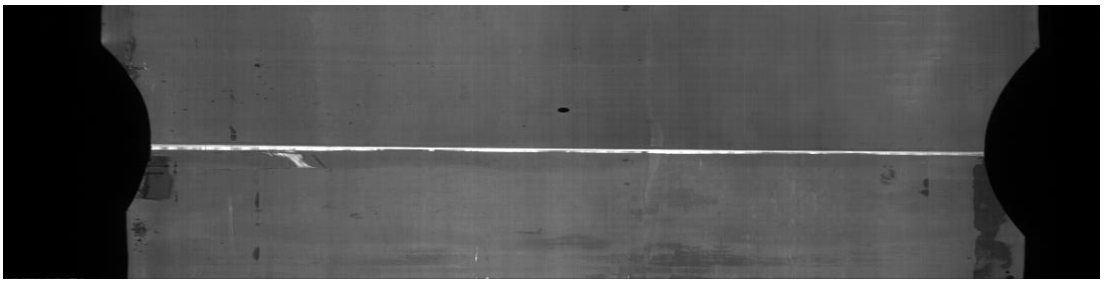


Figure 1.12: Defect type: Weld mark

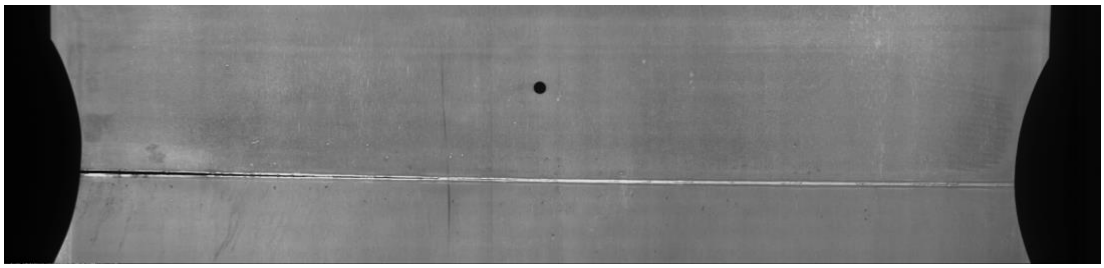


Figure 1.13: Defect type: Weld failure

1.5.12. RIS Mark

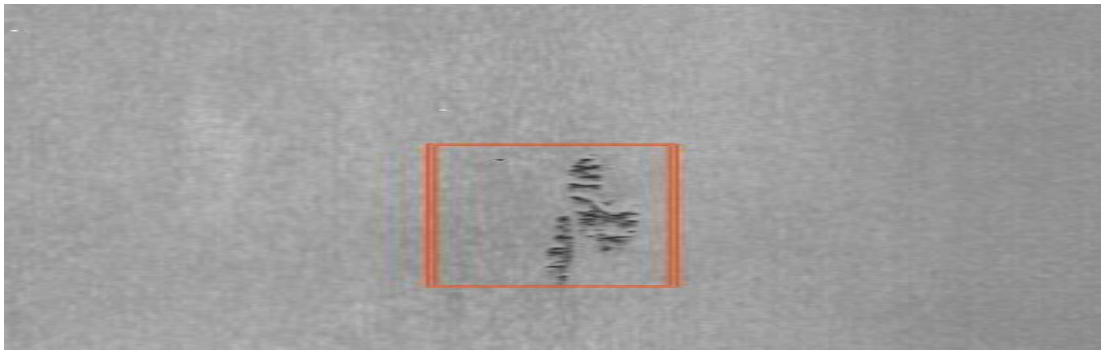


Figure 1.14: Defect type: RIS

1.6. Proposed approach:

Currently defects detection methods based on image processing still have disadvantages such as low speed and resolution. So highly efficient and real-time detection is the key problem needs to be settled down during production

Hence, we proposed a defect detection method of the surface of cold-roll steel sheets in MATLAB and Open Computer Vision with Visual studio. Firstly, Surface image information of cold-roll steel sheets can be acquired by the image capturing equipment (CCD). Secondly, data will be transmitted by communication circuits to controlling PCs for early stage processing. Thirdly, edge detecting operators will be applied to acquire the surface information of cold-rolled steel sheets, in the next step we extracted the textural features of the defective image. These textural features are given to support vector machine (SVM) for classification of defect type[20].

The purpose of localization and identification of surface defects can be realized. With the help of high processing speed of PCs, the efficient and real-time detecting demands can be met with. We focused on the process of various edge detecting and identifying methods of surface images of steel sheets and extracted textural features of the defect image for better classification.

Finally, we designed an online classifier for the process of inspection is also to be automated and inspection results should be called back to the top acquisition service to the manufacturing process for improvement of product quality. However, the Inspection system should be designed to be an efficient composition of human intelligence and experience along with the fastness of a machine. This work deals with the approaches adapted to Design of *Online Classifier for surface defect detection and classification of Cold Rolled Steel Coils.*

General design of online classification system:

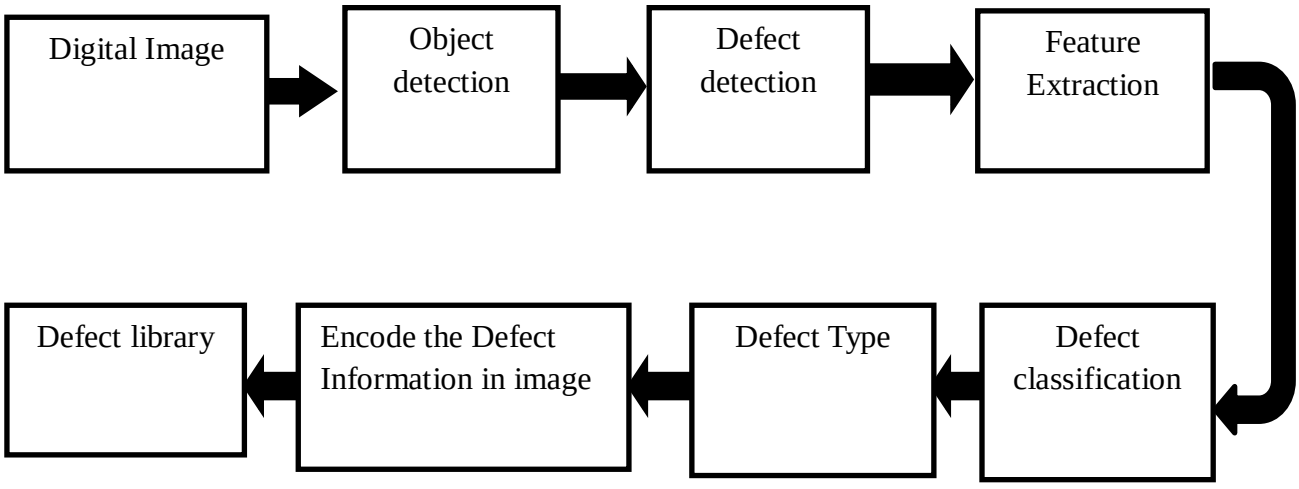


Figure.1.15. Block diagram of Defect detection and Classification

1.7. Proposed Online Classifier Design

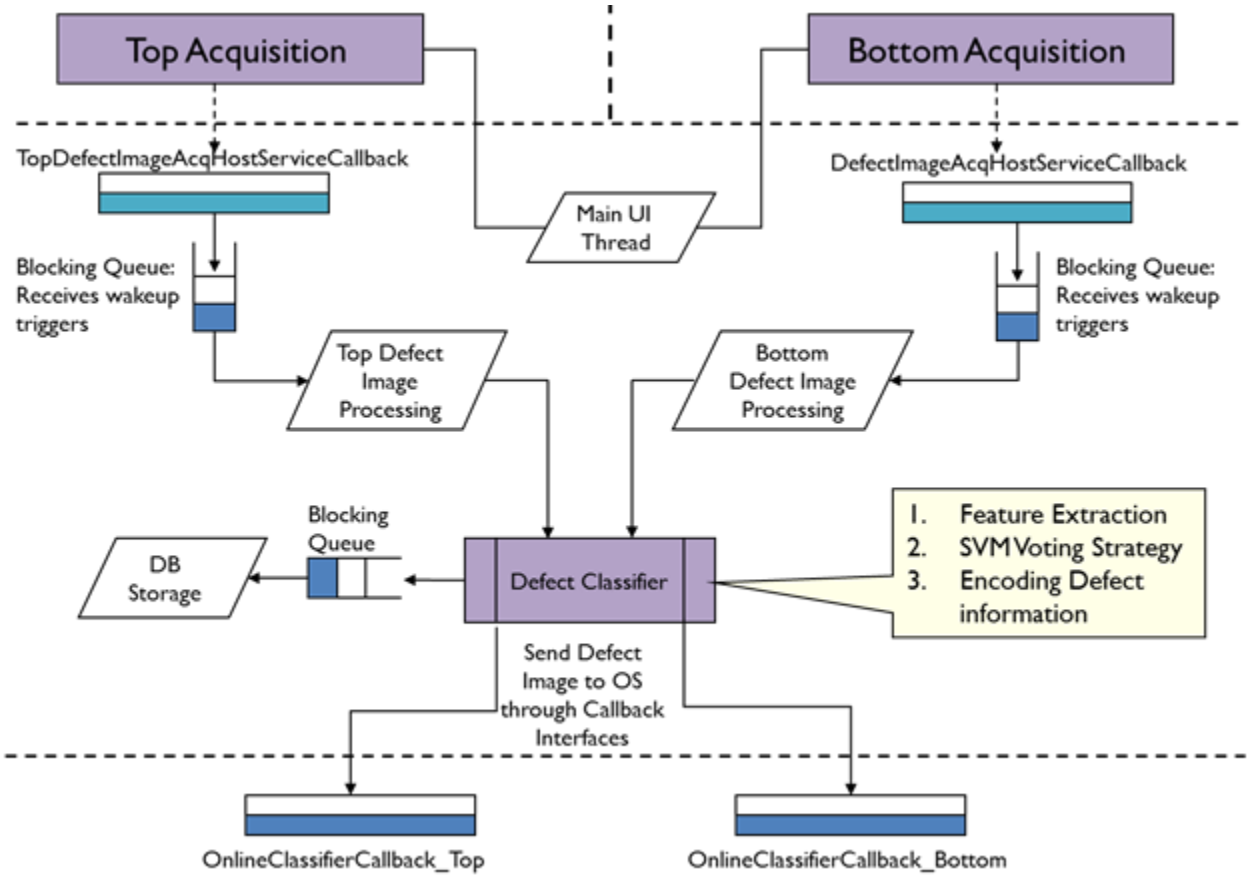


Figure 1.16: Design of Online Classifier for defect detection and classification

1.8. Organization of thesis:

In chapter 1, the real time applications of Defect detection [8], significance of defect detection in various fields like steel industry, paper industry, in heavy production of cold rolled steel coils and various defects and their causes has been discussed. Also the conventional approaches for defect detection and different steps of defect detection and classification in Fig.1.16 and Online Classifier Design for defect detection and classification are also shown in this chapter.

In chapter 2, defect detection using edge detection operators to find the exact position of the defect and identify the defective portion (Region of Interest) is described[22]. The investigation of surface information of steel sheets relies on edge detection technology. The elementary thought of the method is to enhance the display of partial edge information by the process of edge detecting operators in partitioned windows [15]. The important aspects regarding

the importance of δ values for Log operators are also discussed. The automatic defect detection of defect types is implemented using C++ with visual studio and results are shown.

In chapter 3, an overview of texture features, gray-level co-occurrence matrix and feature extraction methods have been discussed. A discussion about texture and statistical methods to extract texture features has been focused, in which co-occurrence matrix and Haralick texture features are used [2]. Geometric textual features are also calculated by using Open computer vision 2.0. Finally a C++ implementation of texture feature extraction based on co-occurrence matrix and Geometric textural features are calculated in experimental results.

In chapter 4, the classification of defect type using support vector machines is discussed. The minimum and maximum of selected features is tabulated for better classification results. A proposed voting strategy method is designed for the offline classification of defect type. SIFT features are extracted which is used to learn classifier for better classification[24].

In chapter 5, we conclude with an analysis of defect information and scope of future work.

Chapter 2

Defect detection of Moving Coil

2.1. Existing Techniques of Defect Identification

Edge detection is a fundamental tool in image processing, machine vision and computer vision, particularly in the areas of feature detection and feature extraction. The purpose of detecting sharp changes in image brightness is to capture important events and changes in properties of the cold roll steel strip-.

The commercially available products approach the issue of surface inspection in a number of ways. These methods are mainly based on the procedures of edge detection, profile analysis or multi-resolution image processing. These procedures are briefly discussed in the following paragraphs.

The steel surface image filters for removal of noises added up due to the acquisition process. The filtered image is then processed for multi-directional edge detection and proper thresholding for discarding noisy edges. The edge map indicates the presence of defects, as a perfect steel sheet is enough smooth not to present edges. The edge map or the segmented image provides the structural features of the defects, which are marked by matching them from a previously stored feature database.

The main problem of the process of defect identification is: several defects may exhibit the same geometry, while the members of the same defect class can be structurally very different and thus they can't be distinguished by merely studying their structural appearance. One of the most basic features of cold-roll steel sheets is the edge. The edge of images contains massive internal information. The investigation of surface information of steel sheets relies on edge detection technology. The elementary thought of the method is to enhance the display of partial edge information from the process of edge detecting operators.

By detecting each pixel and the state of its direct neighborhood of the obtained image, it can be judged whether the pixel is indeed at the boundary of the object or not. The value of the first derivative of digital images generates a step when meeting the mutation in the pixels. As the strip surface may have black flecks, scratching, etc., the features above will be obviously enhanced with the help of binary image generates by edge detection operators.

The Computational Method to realize the function of digital image edge detection is called edge detecting operators. The method of surface defect detection is analyzed according to different operators below.

2.1.1. First Derivative Operator:

The first derivative of digital images has the function of enhancing the change of grayscale values. Hence, derivative values can be regarded as the corresponding output of boundaries. Thresholds can be set to extract the boundaries.

The gradient is a two-dimensional equivalent of the first derivative, by which edge points can be judged. If there is a pixel $f(x, y)$ in the location of (x, y), gradient is defined as below:

$$\nabla f(x,y) = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \end{bmatrix}^T \tag{2.1}$$

G_x, G_y *are* $\boldsymbol{x}$, $\boldsymbol{y}$ direction components of the gradient respectively.

Binary segmentation image can be acquired by the gradient operator:

$$G(x,y) = \begin{Bmatrix} 1, |\nabla f(x,y)\nabla| >> T \\ 0, |\nabla f(x,y)| >> T \end{Bmatrix} \tag{2.2}$$

Where T is the threshold.

Common first derivative operators have the similar principles and performance, Such as Robert, Sobel, Prewitt and Kirsch operators.

2.1.2. Second Derivative Operator:

Gradient operators are vectors, yet second derivative operators are Scalars. Set Laplacian operator as an example, there is a continuous function having a pixel f(x, y) located at (x, y), the value of the Laplacian (second derivative) is defined as follows:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{2.3}$$

Since Laplacian operator is sensitive to noise, the output binary segmentation image is easily contaminated. In order to overcome the above shortcomings, a two dimensional Gaussian low-pass filter can be used to smooth the image first, and then finish the Laplacian operation. The operator of this method is called Laplacian of Gaussian (denoted by LOG) operator.

2.1.3. Canny Operator:

The operators mentioned above are all local-window gradient operators. Because of their sensitiveness to the noise, these operators' performance in reality is not quite satisfying. A new operator called Canny takes three aspects into consideration, which are low false rate, high locating accuracy and suppressing false edge.

The target to be achieved through this project was primarily aimed at detecting the defects belonging to different classes. This was achieved through grabbing the images from the camera, here line scan camera is used which grabs 20 frames per second. Carrying out defect detection on these images and later classifying them.

We have proposed a method to automatically detect and localize defects occurring on the surface. Defect regions are segmented from background images using their distinguishing texture characteristics. This method locates candidate defect regions directly in the DCT domain using the intensity variation information encoded in the DCT coefficients. More precisely, defect detection employs discrete cosine transform (DCT) analysis of each individual non-overlapping region of the image to determine potentially defective blocks, which are further grown and merged to form a defect region on the image.

2.2 Proposed algorithm for defect detection:

1. Receive the image buffer.
2. Change the scale of image. (0-1)
3. Resize the image by a factor – 4.
4. Compute gray level profile.
5. Compute Energy DCT. The result is energy matrix.
    - Break the image into blocks of size BlockSize * BlockSize.
    - Compute the energy of each block.
        a. Compute sum of each block.
        b. Return sum-energy of first element.
6. Compute the average energy of energy matrix.
7. Compute binary image from energy matrix. The result is in energy matrix.
8. Apply morphological closing on energy matrix (Binary image).
9. Change the scale of energy matrix (Binary image); (0-255)
10. Find number of defects using cvFindContours on energy matrix using Open CV 2.0.

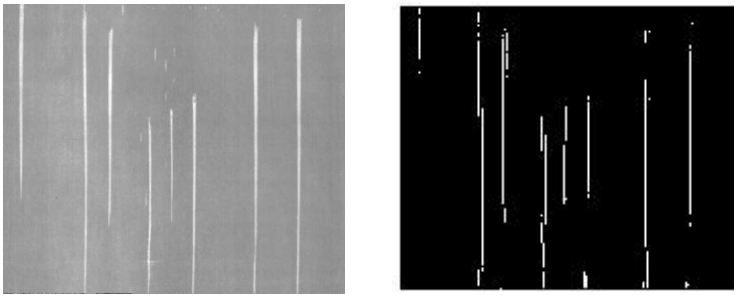
11. Compute edges, width and height of the image.

2.3 Result Analysis of Defect detection:

A. Performance of Operators

In order to compare each operator's edge detecting ability, let's set the surface gray scale image of a certain type of cold-roll steel sheets as an example.

2.3.1 Defect detection of Scratch/Roll Mark:



(a) (b)

Figure 2.1: (a) Original image, (b) Detection using canny operation

2.3.2. Defect detection of Scab:

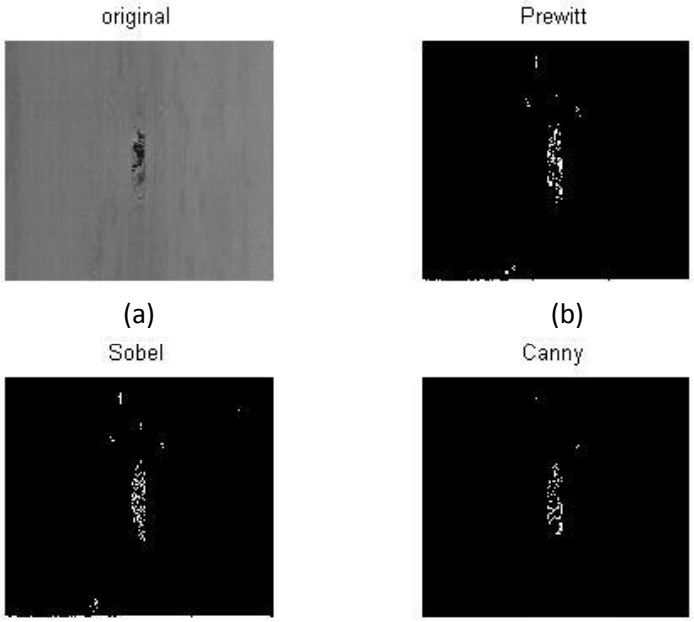


Figure 2.2: (a) Original Image (b) Detection using Prewitt operator (c) Detection using a Sobel operator (d) Detection using Canny operator.

2.3.3. Defect detection of Edge Crack

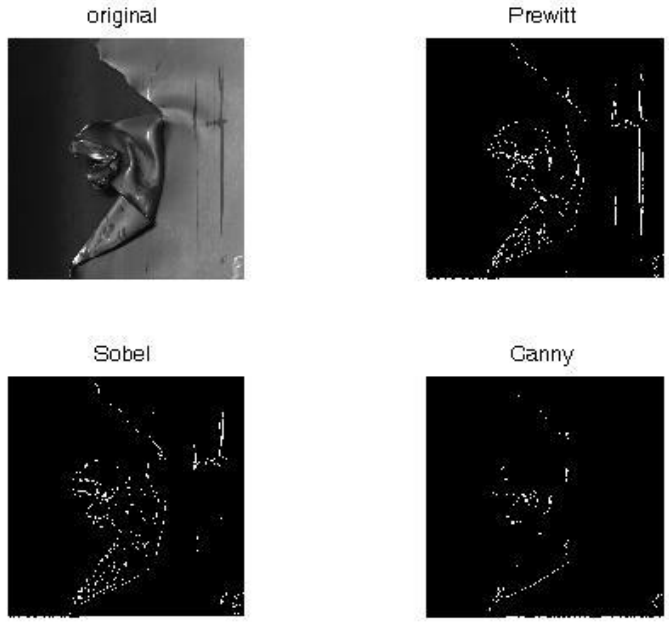


Figure 2.3: (a) original Image (b) detection using Prewitt operator (c)) detection using a Sobel operator (d) detection using the Canny operator.

2.3.4. Defect detection of Weld Fail:

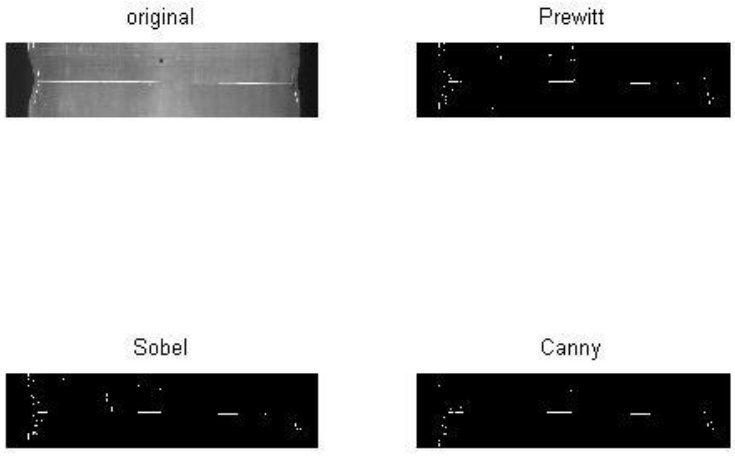


Figure 2.4: (a) original Image (b) detection using Prewitt operator (c) detection using a Sobel operator (d) detection using the Canny operator.

2.3.5.Defect detection of Hole Detection

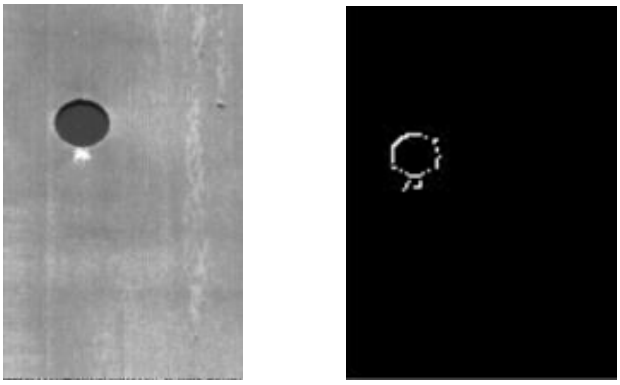


Figure 2.5: (a) original image, (b) detection using canny operation

During the production procedure the surface detects defects of cold-rolled steel sheets, these includes conventional defects and unconventional ones. The former ones are mainly caused by cold rolling processing, with the appearance of abrasions, scratches and rust spots; on the other hand, the latter ones have various kinds of appearance with complex causes. In accordance with the macroscopic appearance, defects can be divided into two parts: the "black line" and the "bright line". Black lines are usually caused by cracks or impurities, and bright lines are generated because the surface of steel sheets is scratched under hot conditions. The surface grayscale image of a defect will be different from surrounding pixels no matter whether the defect is conventional or not, so surface defects can be located easily.

The cold-roll steel sheet on a production line, due to production problems the surface has the appearance of rolled marks, scratches. Operators have little difference in the abilities of defect detecting, which detect merely parts of surface defects. LOG operator detects almost all the defects, and the location is accurate.

2.3.6. Effect of σ to the performance of Log Operator on defect type (Scab):

Figure 2.6: (a) Original image , (b) defect detection with σ=1 (c) defect detection with σ=2 (d) defect detection with σ=3

From the figure 2.6 , when σ=1 defects will have high resolution, but too much detail has a side effect of locating defects; when σ=3, pixels are smoothed too heavily to have a complete edge; when σ=2, almost the whole details and edge have been maintained, which is fit for environment such as factories.

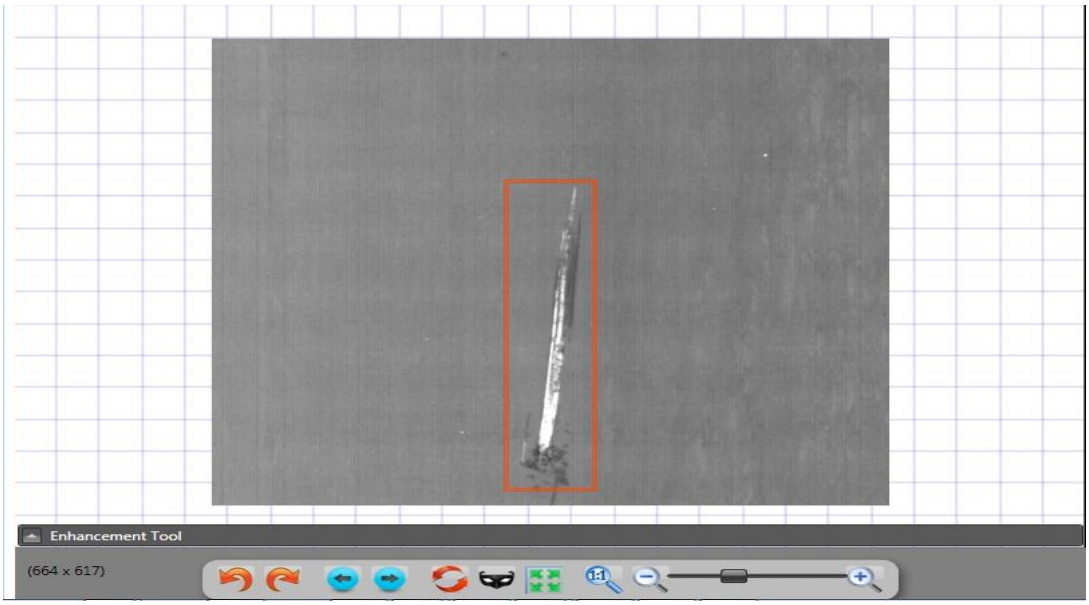


Figure 2.7: Defect detection of scratch mark by Proposed method Using Visual Studio

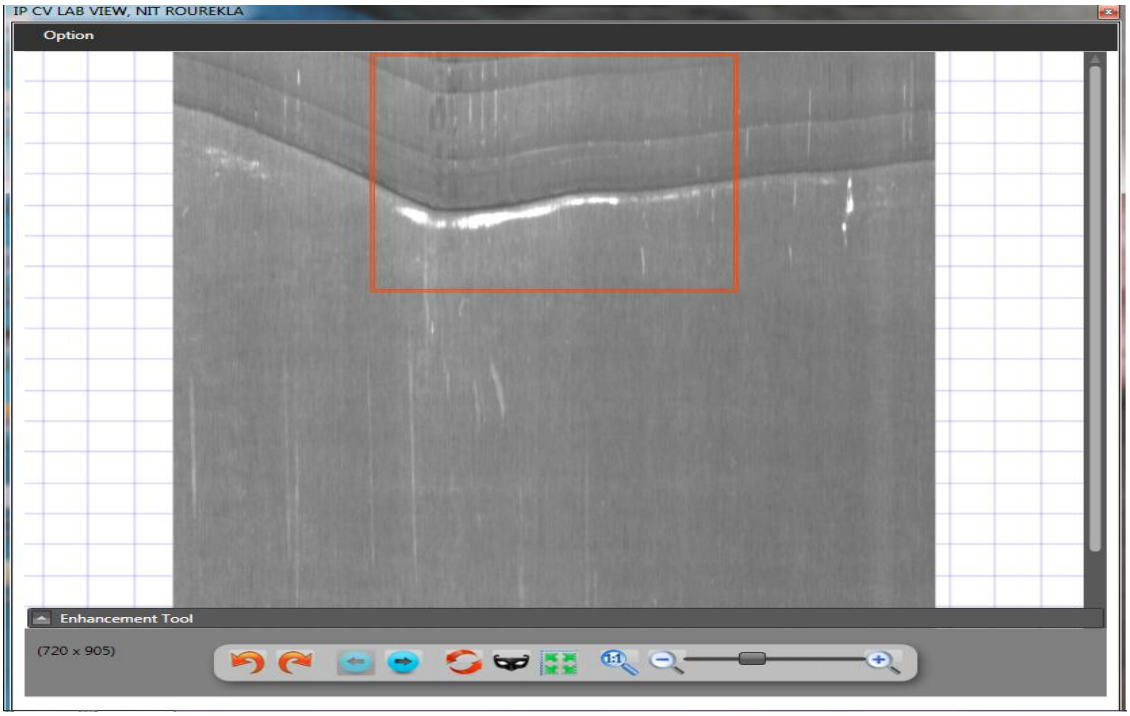


Figure 2.8: Defect detection of Fishtail mark by Proposed method Using visual studio

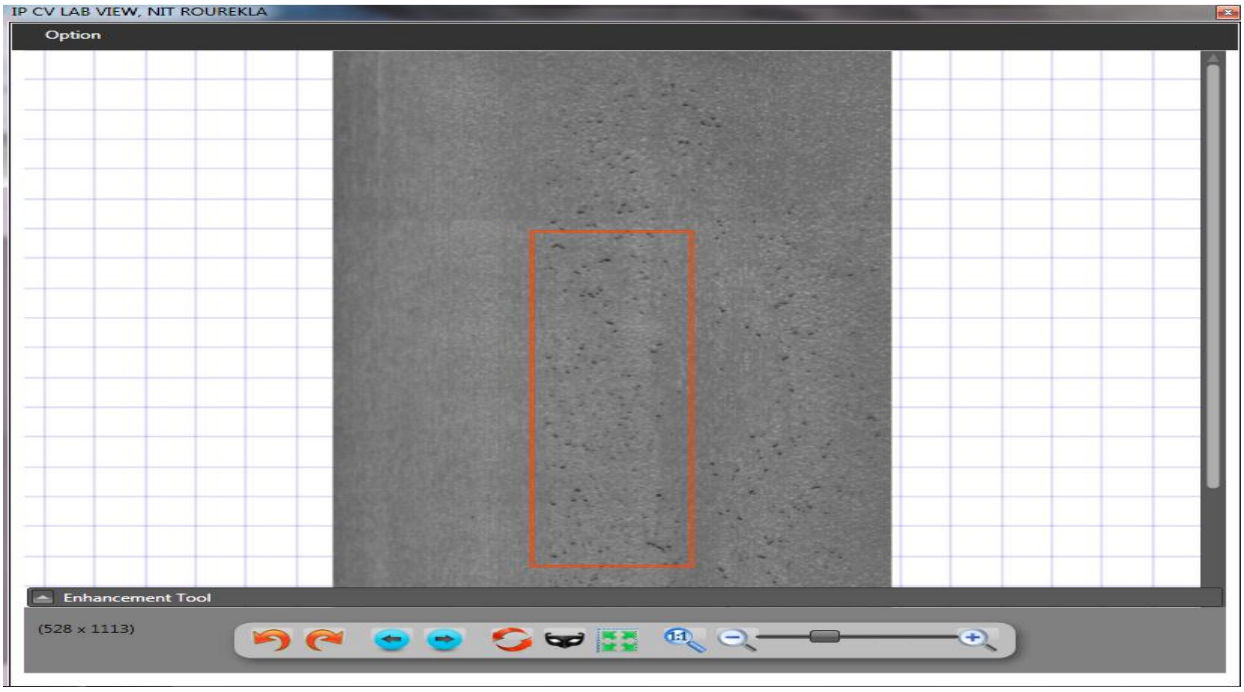


Figure 2.9: Defect detection of sliver mark by Proposed method Using visual studio

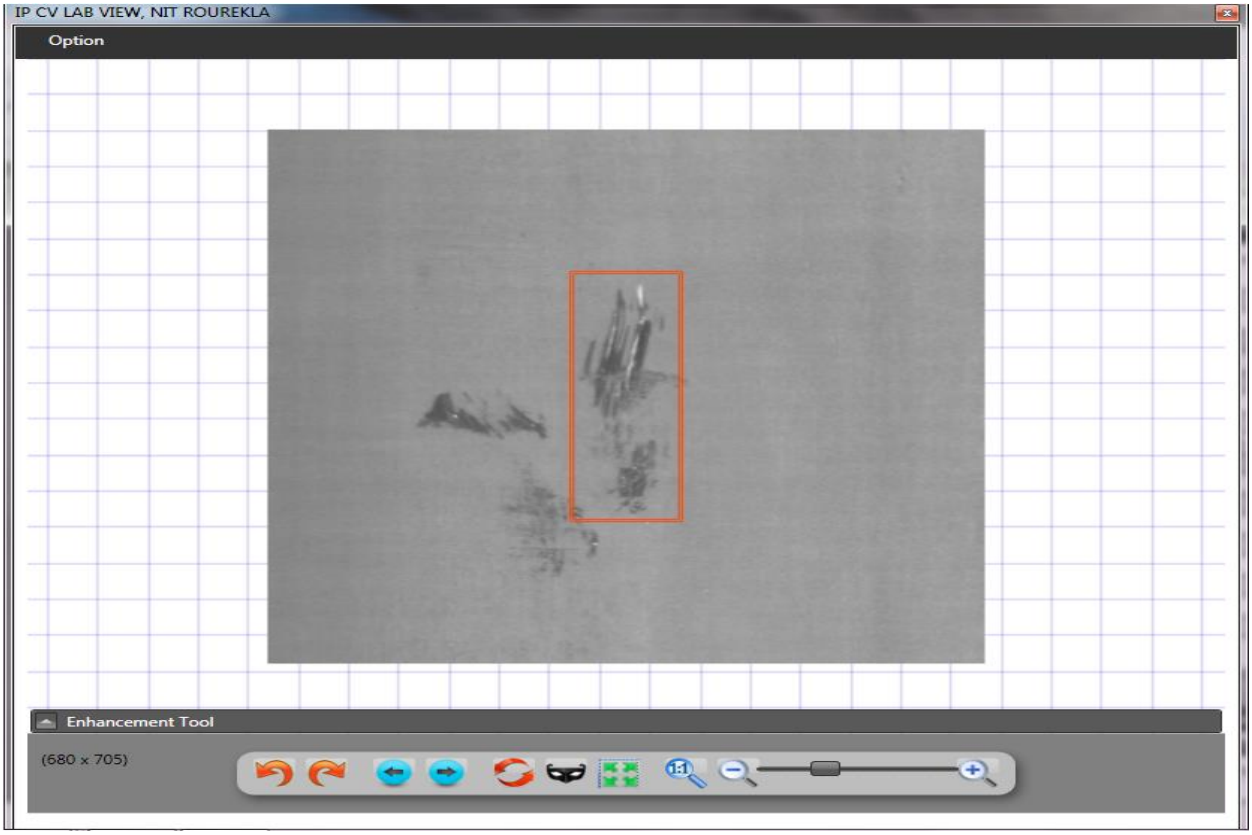


Figure 2.10: Defect detection of roll mark by Proposed method Using visual studio

Figure 2.11: Defect detection of weld mark by Proposed method Using visual studio

2.4. Conclusion:

Defect detection using Prewitt, canny, Sobel, Laplacian of Gaussian operators is performed. The effect of σ values in the performance of Log operator .The proposed algorithm is implemented in C++ with visual studio. The defect detection of defect types is detected. The resulting analysis is shown in the above figures.

Chapter 3

Feature Extraction

3.1 Introduction:

This chapter contains an overview of texture features, gray-level co-occurrence matrix and feature extraction. The following section provides a discussion about texture and statistical methods to extract texture features, in which co-occurrence matrix and Haralick texture features are focused.

Finally, C++ implementation of texture feature extraction based on co-occurrence matrix and experimental results are presented.

Feature extraction: It is the process of acquiring higher level information of an image, such as color, shape, and texture. Features contain the relevant information of an image and will be used in image processing (e.g. Searching, retrieval, storing). Features are divided into different classes based on the kind of properties they describe. Texture feature and geometric features are used in the proposed defect detection algorithm.

3.2 Texture:

Texture is a very general notion that is difficult to describe in words. The texture relates mostly to a specific, spatially repetitive structure of surfaces formed by repeating a particular element or several elements in different relative spatial positions. John R. Smith defines texture as visual patterns with properties of homogeneity that do not result from the presence of only a single color such as clouds and water. Texture features are useful in many applications such as in medical imaging, remote sensing and CBIR. In CBIR, there are many techniques to measure texture similarity, the best-established rely on comparing values of what are known as second-order statistics calculated from query and stored images. Essentially, they calculate the relative brightness of selected pairs of pixels from each image. From these, it is possible to calculate measures of image texture such as the degree of contrast, coarseness, directionality and regularity [9], or periodicity, directionality and randomness. Alternative methods of texture analysis for retrieval include the use of Gabor filters and fractals.

Texture is a conception that is easy to recognize but very difficult to define. This difficulty is demonstrated by the number of different texture definitions attempted by vision researchers, some of them are as follows.

Texture is visual patterns with properties of homogeneity that do not result from the presence of only a single color such as clouds and water. A region in an image has a constant texture if a set of local statistics or other local properties of the picture function are constant, slowly varying, or approximately periodic [3]. An image texture is described by the number and types of its (tonal) primitives and the spatial organization or layout of its (tonal) primitives. A fundamental characteristic of texture: it cannot be analyzed without a frame of reference of tonal primitive being stated or implied. For any smooth gray-tone surface, there exists a scale such that when the surface is examined, it has no texture. Then as resolution increases, it takes on a fine texture and then a coarse texture [2].

The notion of texture appears to depend upon three ingredients: (i) some local order is repeated over a region which is large in comparison to the orders size, (ii) the order consists in the nonrandom arrangement of elementary parts, and (iii) the parts are roughly uniform entities having approximately the same dimensions everywhere within the textured region.

Visual textures refer to the visual impression that textures produce to the human observer, which are related to local spatial variations of simple stimuli like color, orientation and intensity in an image.

3.2.1 Texture Analysis

Major goals of texture research in computer vision are to understand, model and process texture. Four major application domains related to texture analysis are texture classification, texture segmentation, shape from texture, and texture synthesis.

Texture classification: It produces a classification map of the input image where each uniform textured region is identified with the texture class it belongs [8].

Texture segmentation: It makes a partition of an image into a set of disjoint regions based on texture properties, so that each region is homogeneous with respect to certain texture

characteristics. Results of segmentation can be applied to further image processing and analysis, for instance, to object recognition.

3.2.2 Application of Texture

Texture analysis methods have been utilized in a variety of application domains such as automated inspection, medical image processing, document processing, remote sensing and content-based image retrieval. In some of the mature domains (such as remote sensing or CBIR) texture has already played a major role, while in other disciplines (such as surface inspection) new applications of texture are being found.

Remote Sensing

Texture analysis has been extensively used to classify remotely sensed images. Land use classification where homogeneous regions with different types of terrains (such as wheat, bodies of water, urban regions, etc.) need to be identified is an important application. Haralick *et. al.* [2] used gray level co-occurrence features to analyze remotely sensed images. They computed gray level co-occurrence matrices for a distance of one with four directions. They obtained approximately 80% classification accuracy using texture features.

Medical Image Analysis

Image analysis techniques have played an important role in several medical applications. In general, the applications involve the automatic extraction of features from the image which are then used for a variety of classification tasks, such as distinguishing normal tissue from abnormal tissue. Depending upon the particular classification task, the extracted features capture morphological properties, color properties, or certain textural properties of the image. For example, Sutton and Hall discussed the classification of pulmonary disease using texture features.

Tuceryan *et. al.* and Jain *et. al.* Divided the different methods for feature extraction into four main categories, namely: structural, statistical, model based and transform domain [21]. For the implementation of the proposed methods, statistical methods have been used, which are described as follows.

3.2.3 Statistical Method:

Statistical methods represent the texture indirectly according to the non-deterministic properties that manage the distributions and relationships between the gray levels of an image. This technique is one of the first methods in machine vision. By computing local features at each point in the image and deriving a set of statistics from the distributions of the local features, statistical methods can be used to analyze the spatial distribution of gray values. Based on the number of pixels defining the local feature, statistical methods can be classified into first-order (one pixel), the second order (pair of pixels) and higher order (three or more pixels) statistics. The difference between these classes is that the first order statistics estimate properties (e.g. Average and variance) of individual pixel values by waiving the spatial interaction between image pixels, but in the second-order and higher-order statistics estimate properties of two or more pixel values occurring at specific locations relative to each other. The most popular second-order statistical features for texture analysis are derived from the co-occurrence matrix.

3.2.3.1 Gray Level Co-occurrence Matrix & Harlick Texture Features:

In 1973, Haralick *et. al.* Introduced the co-occurrence matrix and his texture features which are the most popular second order statistical features [2]. Haralick proposed two steps for texture feature extraction: the first is computing the co-occurrence matrix and the second step is calculating texture feature base on the co-occurrence matrix. This technique is useful in a wide range of image analysis applications from biomedical to remote sensing techniques.

One of the defining qualities of texture is the spatial distribution of gray values. The use of statistical features is therefore one of the earliest methods proposed in the image processing literature. Haralick suggested the use of co-occurrence matrix or gray level co-occurrence matrix [26]. It considers the relationship between two neighboring pixels, the first pixel is known as a reference and the second is known as a neighbor pixel.

In statistical texture analysis, texture features are computed from the statistical distribution of observed combinations of intensities at specified positions relative to each other in the image. Texture is defined as a function of the spatial variation in pixel intensities. According to the number of intensity points (pixels) in each combination, statistics are classified into first order, second order and higher order statistics.

The Gray Level Co-occurrence Matrix (GLCM) method is a way of extracting second order statistical texture features [1]. The approach has been used in a number of applications.Textural features which can be extracted from each of the gray level co-occurrence matrices. The following equations define these features.Using gray level correlation matrix presented by Haralick textural features are measured.

$p(i,j)(i,j)th$ Entry in a normalized gray level co-occurrence matrix , $= P(i,j)/R$

$P_x(i)$ i^{th} entry in a marginal probability matrix obtained by summing the rows of

$$P(i,j), = \sum_{i=1}^{Ng} P(i,j)$$

N_g Number of distinct gray levels in the quantized image $\sum_i and \sum_j, \sum_{i=1}^{Ng} and \sum_{j=1}^{N_g}$, respectively.

$$p_y(j) = \sum_{i=1}^{N_g} p(i,j)$$

$$p_{x+y}(k) = \sum_{i=1}^{Ng}\sum_{j=1}^{Ng} p(i,j), \qquad k = 2,3,\ldots.,2N_g$$

$$p_{x-y}(k) = \sum_{i=1}^{Ng}\sum_{j=1}^{Ng} P(i,j), \qquad k = 0,1,2\ldots.N_g - 1$$

3.3. Textural Features:

1) Angular Second Moment: The ASM is known as uniformity or energy. It measures the uniformity of an image.

When pixels are very similar, the ASM value will be large

$$f1 = \sum_i \sum_j \{p(i,j)\}^2 \quad (3.1)$$

2) Contrast: Contrast is a measure of intensity or gray-level variability between the reference pixel and its neighbor. In the visual perception of the real world, contrast is determined by the difference in the color and brightness of the object and other objects within the same field of view

$$f_2 = \sum_{n=0}^{N_{p-1}} n^2 \{\sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j)\} \quad (3.2)$$

(3.2)

3) Correlation:

$$f_3 = \frac{\sum_i \sum_j (ij) p(i,j) - \mu_x \mu_y}{\sigma_x \sigma_y} \quad (3.3)$$

Where $\mu_x, \mu_y, \sigma_x, and\ \sigma_y$ are the means and standard deviations of $p_x and\ p_y$

4) Sum of squares: Variance

$$f_4 = \sum_i \sum_j (i-\mu)^2 p(i,j). \quad (3.4)$$

5) Inverse Difference Moment:

$$f_5 = \sum_i \sum_j \frac{1}{1+(i-j)^2} p(i,j) \quad (3.5)$$

6) Sum Averages:

$$f_6 = \sum_{i=2}^{2N_g} i p_{x+y}(i) \quad (3.6)$$

7) Sum Variance:

$$f_7 = \sum_{i=2}^{2N_g} (i - f_8)^2 p_{x+y}(i) \quad (3.7)$$

8) Sum Entropy:

$$f_8 = -\sum_{i=2}^{2N_g} p_{x+y}(i)\log\{p_{x+y}(i)\}. \quad (3.8)$$

9) Entropy:

Entropy is a difficult term to define. The concept comes from thermodynamics, it refers to the quantity of energy that is permanently lost to heat every time a reaction or a physical transformation occurs. Entropy cannot be recovered to do useful work. Because of this, the term can be understood as the amount of irremediable chaos or disorder. The equation of entropy is

$$f_9 = -\sum_i \sum_j p(i,j)\log(p(i,j)) \quad (3.9)$$

10) Different Variance:

$$f_{10} = varience\ of\ p_{x-y} \quad (3.10)$$

11) Different Entropy:

$$f_{11} = -\sum_{i=0}^{N_{g-1}} p_{x-y}(i)\log\{p_{x-y}(i)\}. \quad (3.11)$$

12), 13) Information Measures of correlation:

$$f_{12} = \frac{HXY - HXY1}{\max\{HX,HY\}} \quad (3.12)$$

$$f_{13} = (1 - \exp[-2.0(HXY2 - HXY)])^{1/2} \quad (3.13)$$

$$HXY = -\sum_i \sum_j p(i,j)\log(p(i,j)$$

where HX and HY are entropies of p_x *and* p_y *and*

$$HXY1 = -\sum_i \sum_j p(i,j)\log\{p_x(i)p_y(j)\}$$

$$HXY2 = -\sum_{i}\sum_{j} p_x(i)p_y(\mathrm{j})\log\{p_x(i)p_y(j)\}$$

Hence we obtain a set of four values for each of the preceding 13 measures. These values are given to support vector machine for classification of defect type.

3.4 Geometric features:

Area, Aspect Ratio, Centroid X, Centroid Y, Circularity, Eccentricity, Elongation, Mean, Orientation, Perimeter, Skew 12, Skew 21, Variance X, Variance Y, Weighted Centroid X, Weighted Centroid Y

3.5 Proposed Algorithm for Extraction of Textural & Geometric Features:

The following proposed algorithm is implemented to calculate textural features using Harlick feature extraction method and Geometrical features are also calculated. The major steps of the algorithm are as follows:

Step 1: Read the command line from the users

Step 2: Read the content of the image from .Bmp file

Step 3: Calculate the gray level co-occurrence matrix

Step 4: Calculate Haralick texture features

Step 5: Calculate Geometrical features

Step 6: Save acquired information to a database file

3.6 Results:

The C++ program for experimental implementation of the proposed feature extraction method has been done using open CV with visual basic. The different defected images of cold rolled steel strips and their geometrical and textural features have been studied from the experiment.

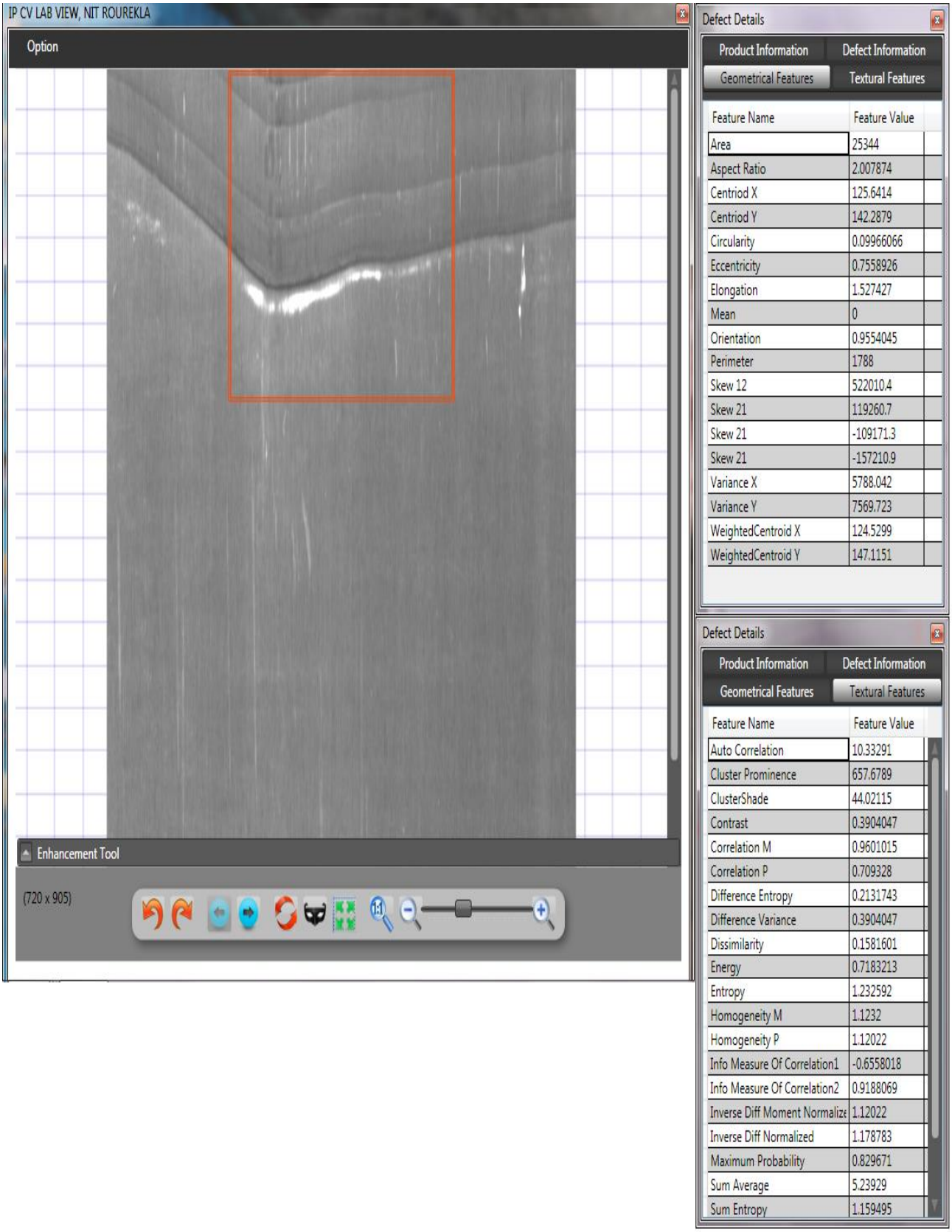


Figure. 3.1: Geometrical & Texture features of Fishtail mark

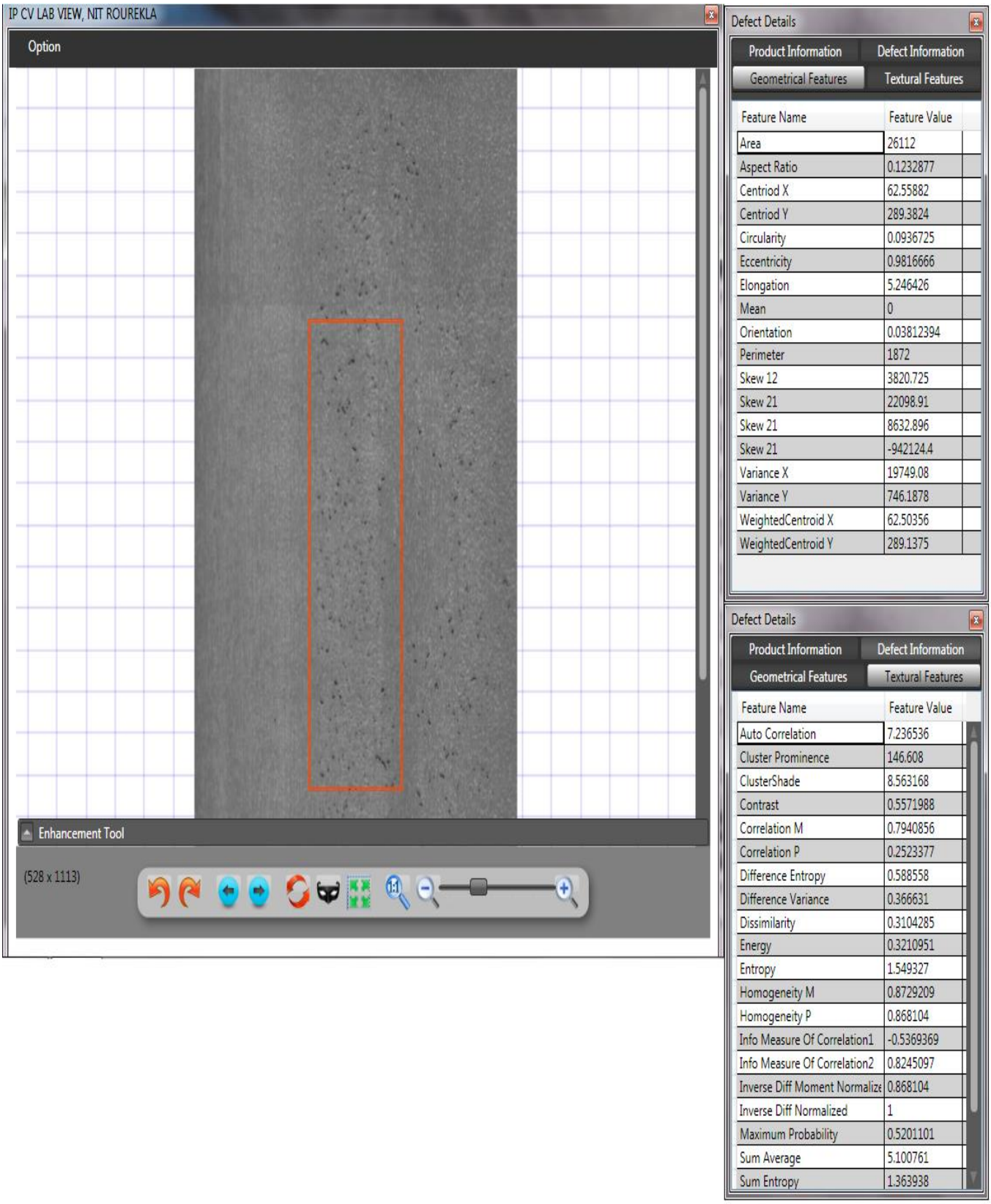


Figure. 3.2: Geometrical & Texture features of Scales mark

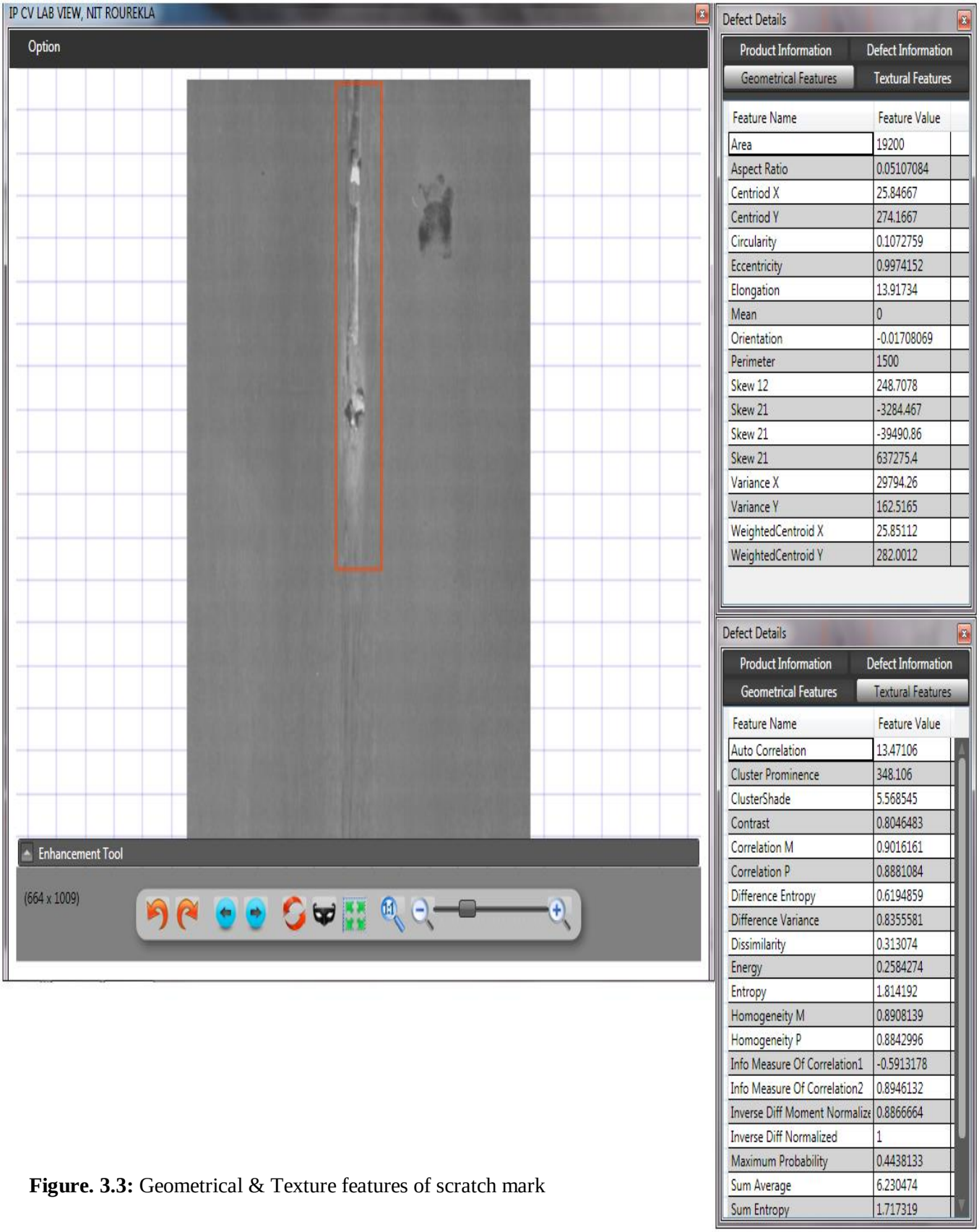


Figure. 3.3: Geometrical & Texture features of scratch mark

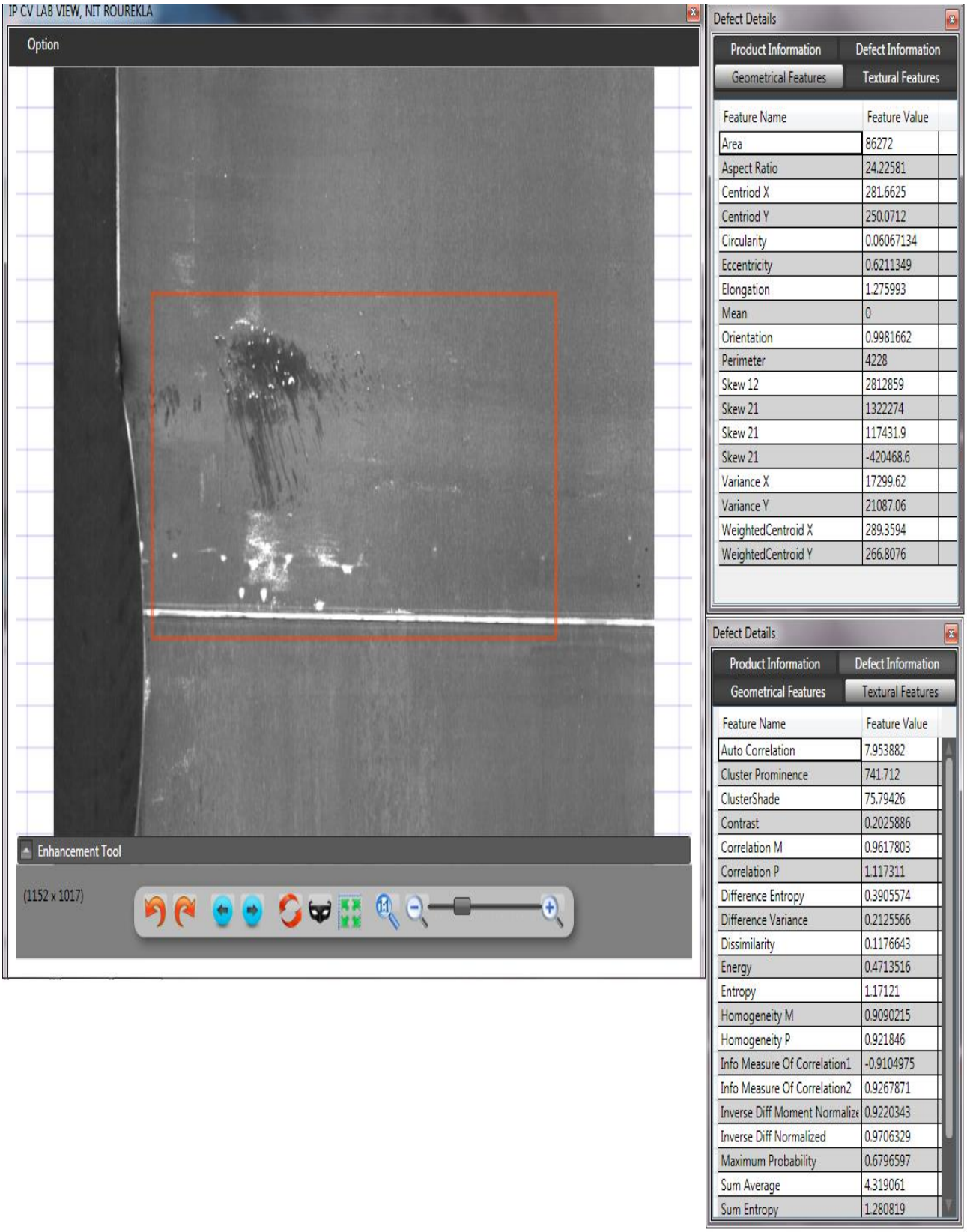


Figure. 3.4: Geometrical & Texture features of edge crack mark

Chapter 4

Defect Classification & Encoding

4.1. Introduction:

In this chapter the geometric and textural features are trained by Support Vector Machine (SVM) technique for data classification. Even though Neural Networks are easier to use, sometimes unsatisfactory results are obtained. A classification task usually involves training and testing data which consist of some data instances. Each instance in the training set contains one target value and several attributes. The goal of SVM is to produce a model which predicts the target value of data instances in the testing set which are given only the attributes. A proposed voting strategy method is introduced for better classification of defect type. The defect information is encoded in the defect image itself.

4.2. Support Vector Machine:

The problem of empirical data modeling is useful to many engineering applications .In empirical data modeling a process of induction is used to build up a model of the system, from which it is hoped to deduce responses of the system that have yet to be observed. Ultimately the quantity and quality of the observations govern the performance of this empirical model. By its observational nature data obtained is finite and sampled; typically this sampling is non-uniform and due to the higher dimensional nature of the problem the data will form only a sparse distribution in the input space. Consequently the problem is nearly always ill posed (Poggio *et al.*, 1985) in the sense of Hadamard. Traditional neural network approaches have suffered difficulties with generalization, producing models that can over fit the data. This is a consequence of the optimization algorithms used for parameter selection and the statistical measures used to select the 'best' model. The foundations of Support Vector Machines (SVM) have been developed by Vapnik *et. al.* (1995)[26]. It is gaining popularity due to many attractive features, and promising empirical performance. The formulation embodies the Structural Risk Minimization (SRM) principle, which has been shown to be superior, (Gunnet *et. al.*, 1997), to traditional Empirical Risk Minimization (ERM) principle, employed by conventional neural networks. SRM minimizes an upper bound on the expected risk, as opposed to ERM that minimizes the error on the training data. It is this difference which equips SVM with a greater

ability to generalize, which is the goal in statistical learning. SVMs were developed to solve the classification problem, but recently they have been extended to the domain of regression problems (Vapnik *et al.*, 1997) [25].

In machine learning, support vector machines (SVMs, also support vector networks[]) are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis. The basic SVM takes a set of input data and predicts, for each given input, which of two possible classes forms the output, making it a non-probabilistic binary linear classifier. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

In addition to performing linear classification, SVMs can efficiently perform non-linear classification using what is called the kernel trick, implicit mapping their inputs into high-dimensional feature spaces.

More formally, a support vector machine constructs a hyper-plane or set of hyper-planes in a high- or infinite-dimensional space, which can be used for defect classification. Intuitively, a good separation is achieved by the hyper-plane that has the largest distance to the nearest training data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.

The minimum and maximum values of the geometric and textural feature are used to form a hyper plane. Classification in SVM is an example of Supervised Learning. Known labels help indicate whether the system is performing in a right way or not. This information points to a desired response, validating the accuracy of the system, or be used to help the system learn to act correctly. A step in SVM classification involves identification as which are intimately connected to the known classes. This is called feature selection or feature extraction. Feature selectors and SVM classification together have a use even when the prediction of unknown samples is not

necessary. They can be used to identify key sets which are involved in whatever processes distinguish the classes [8].

4.3. Selected geometric features:

Geometric features	True/ False
Area	TRUE
Aspect Ratio	TRUE
Perimeter	FALSE
Centriod_X	FALSE
Centriod_Y	FALSE
Circularity	FALSE
Orientation	TRUE
Eccentricity	FALSE
Elongation	FALSE
Mean	FALSE
variance_X	FALSE
variance_Y	FALSE
Skew12	FALSE
skew21	FALSE
skew03	FALSE
skew30	FALSE
Weighted Centroid_X	FALSE
Weighted Centroid_Y	FALSE

Table 4.1: Selected geometric features

4.4Selected Textural Features:

Textural features	True/ False
Auto Correlation	TRUE
Contrast	TRUE
Correlation M	TRUE
Correlation P	TRUE
Cluster Prominence	TRUE
Cluster Shade	TRUE
Dissimilarity	TRUE
Energy	TRUE
Entropy	TRUE
Homogeneity M	TRUE
Homogeneity M	TRUE
Maximum Probability	TRUE
Sum of Squares	TRUE
Sum Average	TRUE
Sum Variance	TRUE
Sum Entropy	TRUE
Difference Variance	TRUE
Difference Entropy	TRUE

Table 4.2. Selected Textural Features

Selected Textural Features=2

Active Textural Feature Count =22

Target Dimension=2

4.5. Transform Vectors:

Transform Vectors	
-0.037696	-0.00047
-0.031996	0.088212
0.254589	-0.259816
0.212197	-0.288506
-0.035832	-0.0578
-0.01209	0.009504
-0.062646	-0.284721
-0.090768	0.054438
0.299965	-0.123042
-0.269639	-0.287105
0.306193	0.21811
-0.053729	-0.141022
-0.065349	-0.166552
-0.369581	-0.349273
0.275387	-0.269271
0.28318	-0.138489
0.251075	-0.371055
0.291804	0.240869
0.214953	-0.288262
0.296444	0.173541
0.11874	-0.069505
0.141525	0.083989
0.007718	-0.113219
-0.037995	-0.118143

Table 4.3. Transform Vectors

4.6. Proposed voting strategy for four classes:

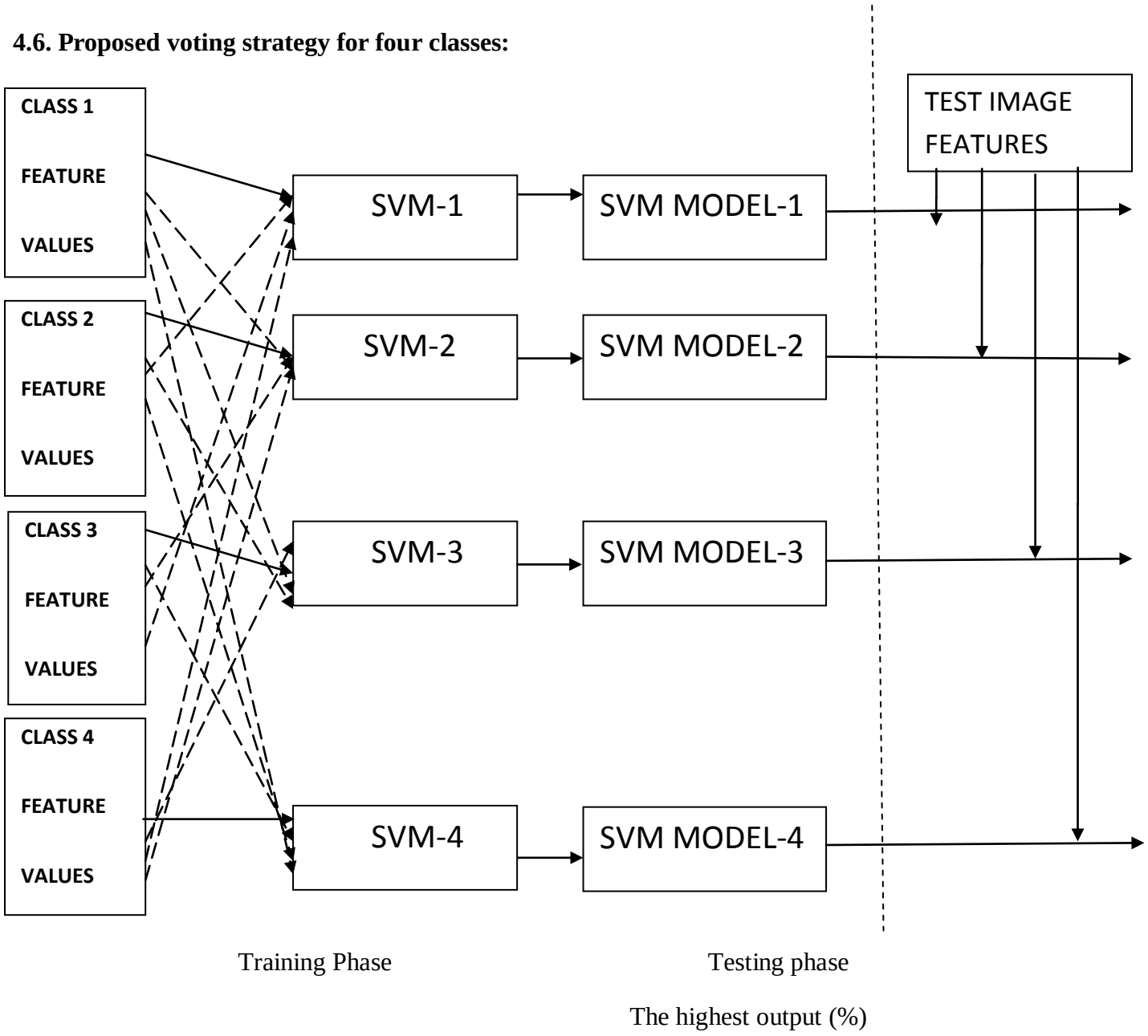


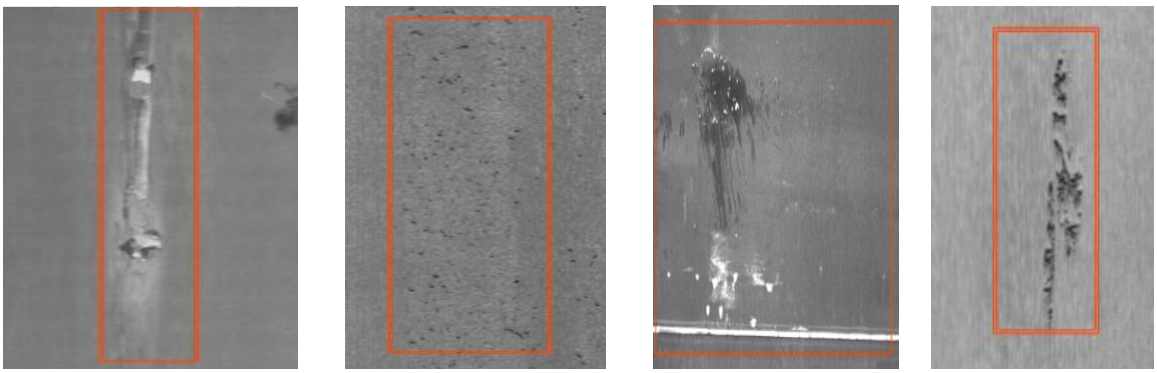
Figure.4.1.Proposed voting strategy for four different classes of defect images

The proposed framework has two steps. The first step is that non rectangular shaped ROI detection and feature extraction using SIFT[24]. The second step is the classification using svm. Classification diagram is shown in the figure, the classification of four defect types, we use 4 swims with one versus all strategy. Each SVM trains the input features vectors with different labels (the solid lines are indicates that the input features are trained with label 1, then dashed lines are indicates that input features are trained with label-1). for giving test image, we can get a

lot of feature points. Then we test each feature points with same label, then the classifier output a label for each feature points. Among the outputs of four SVMs, we vote the highest percent result. By applying the voting strategy we achieved better results.

Experimental results show the classification accuracy in different kernels function with different parameter values for SVM. In this experiment, we divided the data with training-test ratio is 70% and 30% by feature vectors.

The experimental results:



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

Figure.4.2: The defect types a) sliver, (b) RIS,(c)Scratch (d)Scab

4.7. Classification accuracy:

We have shown several examples of the experimental results with different kernels as linear, polynomial, different parameters in the table.

Parameters (kernel)	Class 1	Class 2	Class 3	Class 4
C=1(linear),	92.42	71.63	82.63	73.62
C=1,d=2,g=1/128	93.43	76.32	82.39	71.69
d=2,c=3,g=0.06	95.80	80.29	84.31	71.18

Table: 4.4.CSVM classification accuracy (%) by features

SVM classification accuracy(%) by images

	Class 1	Class 2	Class 3	Class 4
No. of trainig images	8	14	15	6
No. of testing images	4	6	8	15
Total images	12	20	23	21
Testing performance	100%(4)	100%(6)	100%(8)	82%(12)

Table 4.5: SVM voting strategy classification of images:

4.8. Application of SIFT for better classification of defect type:

The scale - invariant feature transform (SIFT) is an algorithm in computer vision to detect and describe local features in images. A rectangular like defects region and thus extracts a single feature vectors on a rectangular region for classification. This approach suffers from insufficient number of feature vectors for training classifiers. For better classification of the defect type include a lot of feature vectors are extracted using SIFT[24]. These features are trained in the SVM Voting strategy method.

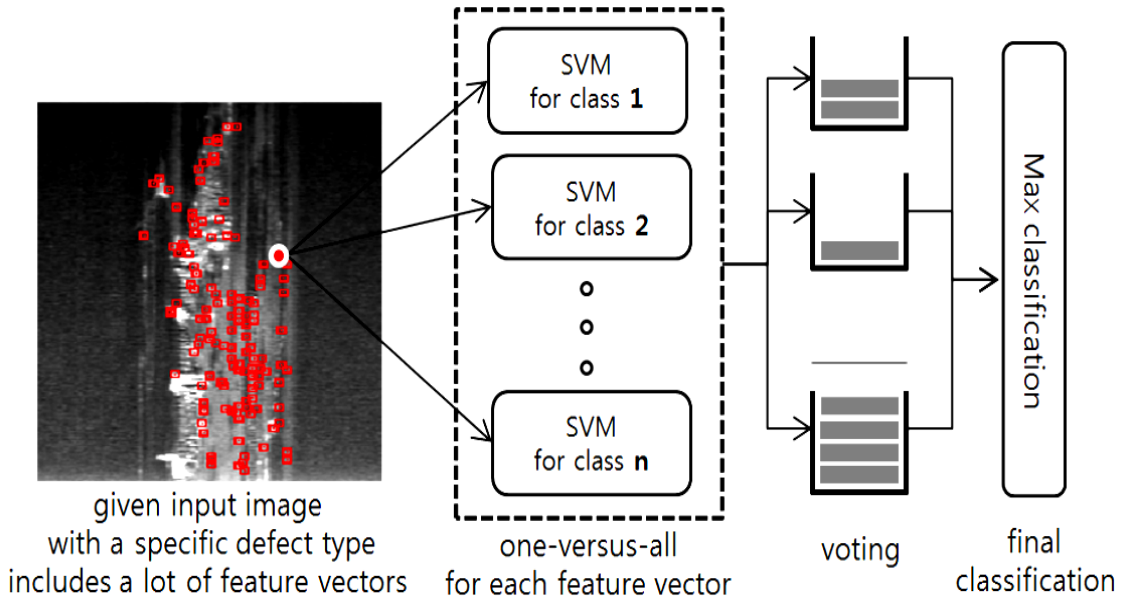


Figure 4.3. A new framework for steel surface defect detection and classification.

4.9. Encoding the defect Information:

The defect information is encoded into the image. The defect info contains textural,geometric features, and severity of the defect type. The morse coding technique is used for encoding.

For decoding purpose an image viewer application is proposed with visual studio and open computer vision.

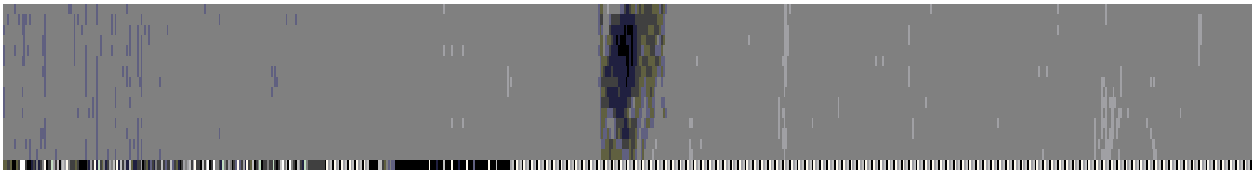


Figure 4.4: Encoding of defect information

Chapter 5

Conclusion and Future work:

5.1. Conclusion

In this dissertation, an image processing based automated surface inspection system to fulfill industry requirements and customer satisfaction for quality is developed, to achieve the automated inspection of cold rolled steel strips which will effectively maintain the production process. As a result of this a surface inspection system of cold rolled steel strip are inspected with online classifier. LOG operator has its advantages over other operators in noisy environment. In comparison to other visual detection, the method of defect detection of cold-roll steel surface based on MATLAB has the satisfying performance in data processing, which meets the high-efficient and real-time defects detection demand. Various textural features are calculated using GLCM presented by Haralick. Textural features and geometrical features are extracted and used for classification of defect types using support vector machine voting strategy. For better classification of defect types SIFT feature descriptor is implemented. With proposed voting strategy it is found that classification accuracy is 100% for weld mark, scratch, scab. For the fishtail mark the classification accuracy is found to be 82%. Defect information is encoded in the defect image itself. The image viewer application is designed for decoding the defect information for analysis of defect types and causes.

5.2. Scope of Future work

An automatic and efficient monitoring system to inspect surface quality using an online classifier is proposed . The complete production chain is possible in the actual standard ,is the monitoring of specific process stages and their visualization on several monitors can predict the defect before the CRM coil enters into the mill This can overcome the breakage of coils.But by designing an automatic alarming system can overcome the damage of cold rolling machine. Reduction of machine malfunction and repair as well as optimization of maintenance rate can be reduced. If the online classifier system is located in HSM coil then the defect can be easily inspected at the cold rolling process.

REFERENCES:

[1]. A. Bouridane M. A. Tahir and F. Kurugollu, "An fpga based coprocessor for glcm and haralick texture features and their application in prostate cancer classification, Analog Integer". *Circuits Signal Process*. pp. no. 2- 205, 2005.

[2]. K. Shanmugam R. M. Haralick and I. H. Dinstein, "Textural features for image classification", *IEEE Transactions on Systems, Man and Cybernetics* 3, 610-621, 1973.

[3]. J. Sklansky, "Image segmentation and feature extraction", *IEEE Transactions on Systems, Man, and Cybernetics*, pp. 237-247, 1978.

[4]. Hon-Son Don, King-Sun Fu, C. R. Liu and Wei-Chung Lin, *"*Metal Surface Inspection Using Image Processing Techniques*", IEEE Transactions of System, Man and Cybernetics*, Vol. SMC-14, No.1 January 1984.

[5]. R.T. Chin and C.A. Harlow, *"*Automated Visual Inspection: A Survey", *IEEE Transactions of Pattern Analysis Machine Intelligence*, Vol. 4, pp. 557-573, November 1982.

[6]. Anil K. Jain, *"*Fundamentals of Digital Image Processing*", Prentice Hall, 1989* .

[7]. S. W. Seol *et al.*, "An automatic detection and tracking system of moving objects using double differential based motion estimation," *Proc. of Int. Tech. Conf. Circ./Syst., Comput. and Comms. (ITC-CSCC2003)*, pp. 260– 263, 2003.

[8]. Lipton, H. Fujiyoshi, and R. Patil, "Moving Target Detection and Classification from Real-Time Video", *In Proceedings of IEEE Workshop on Application of Computer Vision*, 1998.

[9]. Choi Se-Ho, "Development of SDD online signal processing system for cold rolled strip(1), *Research Report, RIST,* 1994.

[10]. J. Serra, "Image analysis and mathematical morphology", *Academic Press, 1982.*

[11]. I. Pitas, N. Sidiropoulos, "Pattern recognition of binary image objects by using morphological shape decomposition", *Computer vision graphics and image processing,* pp. 279-305, Academic Press, 1992.

[12]. Mak, K.L., Peng, P., Lau, H.Y.K., "A Real-time computer vision system for detecting defects in textile fabrics", Industrial Technology, ICIT, *IEEE International Conference* pp. 469-474, 2005.

[13]. SDD Development Team, "SDD Manual", *Research report, Parsytec,* 1997.

[14]. Faustino Obeso, "Intelligent on-line surface inspection on a skin pass mill", *Iron and Steel Engineer,* 1997.

[15]. John C. Badger, "Automated Surface inspection system", *Iron and Steel Engineer,* 1997.

[16]. Kim Kyung Min, "Development of surface inspection algorithm for cold rolling mill", *CASE,* Vol.3, No.2, pp 179-186, 1997.

[17]. Lowe, D.G., "Distinctive image features from scale-invariant key-points." *International journal of Computer Vision 60,* 2004.

[18]. Cord, A., Bach, F., Jeulin, D., "Texture Classification by Statistical Learning from Morphological Image Processing", *Application to Metallic Surfaces. Journal of Microscopy,* Vol. 239, 2010.

[19]. Martins, L.A., Flavio, O.L., Padua, C., Paulo, E., Almeida, M., "Automatic Detection of Surface Defects on Rolled Steel Using Computer Vision and Artificial Neural networks, 2008.

[20]. Wang, X., Wong, B.S., Tan, C. S., "Recognition of Welding Defects in Radiographic Images by Using Support Vector Machine Classifier. *Research Journal of Applied Sciences, Engineering and Technology,* Vol. 2, No. 3, pp. 295-301, 2010.

[21]. M.tuceryan and A.K..Jain, "Texture analysis in the handbook of Pattern Recognition and computer vision ,pp 207-248, 1998.

[22]. John Canny, " A computational approach to edge detection", *IEEE Trans. On pattern analysis and machine intelligence,* vol-8, no. 6, pp 679-698, 1986.

[23]. Guifang Wu, "Multimedia", *Intech,* , pp 205-232, 2010.

[24]. Lowe D. G., "Distinctive Image Features from Scale-Invariant Keypoints", *International Journal of Computer Vision,* vol-60, no.2, pp. 91-110, 2004.

[25]. Vapnik et al**., "**Support Vector Regression Machines **",** *Advances in Neural Information Processing Systems 9,* NIPS 1996, 155–161, MIT Press.

[26]. Vapnik et al., "Support Vector Networks ", Machine Learning , Volume 20, Issue 3, pp 273-297,1995.

Appendix

CPP Implementation of IPCV Defect Detection:

```
// IPCV Defect Detection.cpp : Defines the initialization routines for the DLL.

//

#include "stdafx.h"

#include "DefectDetection.h"

#include "DefectDetectionPrt.h"

#include "omp.h"

#ifdef _DEBUG

#define new DEBUG_NEW

#endif

//

//TO DO: If this DLL is dynamically linked against the MFC DLLs, any functions exported from this DLL will call into MFC must have the AFX_MANAGE_STATE macro added at the beginning of the function.

//

// For example:

//

// extern "C" BOOL PASCAL EXPORT ExportedFunction()

// {

// AFX_MANAGE_STATE(AfxGetStaticModuleState());

// // normal function body here

// }

//

// It is very important that this macro appear in each function, prior to any calls into MFC. This means that it must appear as the first statement within the function, even before any object variable declarations as their constructors may generate calls into the MFC DLL.
```

```
//
// Please see MFC Technical Notes 33 and 58 for additional details.
//
// CDefectDetection App
BEGIN_MESSAGE_MAP(CDefectDetectionEngineApp, CWinApp)
END_MESSAGE_MAP()
// CDefectDetectionEngineApp construction
CDefectDetectionEngineApp::CDefectDetectionEngineApp()
{
// TODO: add construction code here, Place all significant initialization in InitInstance
}
// The one and only C Defect Detection Engine App object C Defect Detection Engine App the App; C Defect Detection Engine App initialization.
BOOL C Defect Detection Engine App::Init Instance()
{
CWinApp::Init Instance();
return TRUE;
}
void DefectDetector(unsigned char *ipBuffer, int inImageWidth, int inImageHeight, int inCompressionFactor, int DDA, int  inBlockSize, float ifAlgorithmSensitivity, int inStrElemColCount, int  inStrElemRowCount, int inStrElemAnchorX, int inStrElemAnchorY, int StrElemShape, int &outDefectCount, byte *outGrayBuffer, float *outEnergyBuffer, EnergyBufferInfo *outEnergyBufferInfo, int inEdgeThreshold, int &outLeftEdge, int &outRightEdge)
{
CvMat *lpSourceMat = NULL;
CvMat *lpSourceScaledMat = NULL;
CvMat *lpRedSourceMat = NULL;
```

```
CvMat *lpEnergyMat = NULL;

float lfAvgEnergy = 0;

lpSourceMat = cvCreateMat(inImageHeight, inImageWidth, CV_8UC1);

lpSourceScaledMat = cvCreateMat(inImageHeight, inImageWidth, CV_32FC1);

lpRedSourceMat = cvCreateMat(inImageHeight / inCompressionFactor, inImageWidth /
inCompressionFactor, CV_32FC1);

lpEnergyMat = cvCreateMat(((inImageHeight / inCompressionFactor) / inBlockSize),
((inImageWidth / inCompressionFactor) / inBlockSize), CV_32FC1);

cvZero(lpEnergyMat);

memcpy(lpSourceMat->data.ptr, ipBuffer, inImageWidth * inImageHeight);

int i;

#pragma omp parallel for private(i) shared(lpSourceMat, pSourceScaledMat)

for(i = 0; i < lpSourceMat->width*lpSourceMat->height; i++)

{

lpSourceScaledMat->data.fl[i] = lpSourceMat->data.ptr[i] * (0.003921f);

}

//cvConvertScale(lpSourceMat, lpSourceScaledMat, 1.0f / 255.0f);

cvResize(lpSourceScaledMat, lpRedSourceMat, CV_INTER_NN);

//TRACE("\nResize : %ld", end - start);

ComputeGrayLevel(lpSourceScaledMat->data.fl, inImageWidth, inImageHeight,
inEdgeThreshold, outLeftEdge, outRightEdge, outGrayBuffer);

//TRACE("\nComputeGrayLevel : %ld", end - start);

if(DDA == 0)

{

}

else

{
```

```
ComputeEnergyDCT(lpRedSourceMat, inBlockSize, outLeftEdge, outRightEdge,
inCompressionFactor, lpEnergyMat);

//TRACE("\nComputeEnergyDCT : %ld", end - start);

}

//lfAvgEnergy = AvgEnergy(lpRedSourceMat, inBlockSize, lpEnergy);

lfAvgEnergy = (float)cvAvg(lpEnergyMat).val[0];

//TRACE("\ncvAvg : %ld", end - start);

ComputeThreshold(lpRedSourceMat, inBlockSize, lfAvgEnergy, ifAlgorithmSensitivity,
lpEnergyMat);

//TRACE("\nComputeThreshold : %ld", end - start);

IplConvKernel *strElement =
cvCreateStructuringElementEx(inStrElemColCount,inStrElemRowCount, inStrElemAnchorX,

inStrElemAnchorY, StrElemShape);

cvMorphologyEx(lpEnergyMat, lpEnergyMat, NULL, strElement, CV_MOP_CLOSE, 1);

cvReleaseStructuringElement(&strElement);

memcpy(outEnergyBuffer, lpEnergyMat->data.fl, sizeof(float) * (lpEnergyMat->width
*lpEnergyMat->height));

outEnergyBufferInfo->leftEdge = (float)outLeftEdge / (inImageWidth / lpEnergyMat->width);

outEnergyBufferInfo->rightEdge = (float)outRightEdge / (inImageWidth / lpEnergyMat>width);

outEnergyBufferInfo->width = lpEnergyMat->width;

outEnergyBufferInfo->height = lpEnergyMat->height;

CvMemStorage *storage = cvCreateMemStorage();

CvSeq *contours;

CvMat* temp = cvCreateMat(lpEnergyMat->height,

lpEnergyMat->width, CV_8UC1);

//cvConvertScale(lpEnergyMat, temp, 255.0f);
```

```
#pragma omp parallel for private(i) shared(lpEnergyMat, temp)
for(i = 0; i < lpEnergyMat->width*lpEnergyMat->height; i++)
{
temp->data.ptr[i] = lpEnergyMat->data.fl[i] * 255.0f;
}
outDefectCount = cvFindContours(temp, storage, &contours, sizeof(CvContour),
CV_RETR_EXTERNAL, CV_CHAIN_APPROX_NONE);
cvReleaseMat(&temp);
cvReleaseMemStorage(&storage);
cvReleaseMat(&lpEnergyMat);
cvReleaseMat(&lpRedSourceMat);
cvReleaseMat(&lpSourceMat);
cvReleaseMat(&lpSourceScaledMat);
}
void ComputeGrayLevel(float *buffer, int width, intheight, int inEdgeThreshold, int &leftEdge,
int &rightEdge,byte *outGrayBuffer)
{
int i, j;
float *lpGrayBuffer = (float*) malloc (sizeof(float) * width);
memset(lpGrayBuffer, 0, sizeof(float)*width);
#pragma omp parallel for private(i,j)
for(i = 0; i < width; i++)
{
for(j = 0; j < height; j+= 200)
{
lpGrayBuffer[i] += buffer[(j * width) + i];
```

```
}
lpGrayBuffer[i] /= (height / 200) + 1;outGrayBuffer[i] = (byte)(lpGrayBuffer[i] * 255.0f);
}
CvMat* dxMat = cvCreateMat(1, width, CV_32FC1);
float max = 0;
for(int i = 0; i < width; i++)
{
if(i == 0)
dxMat->data.fl[i] = lpGrayBuffer[i+1] -lpGrayBuffer[i];
else if(i == width - 1)
dxMat->data.fl[i] = lpGrayBuffer[i] -lpGrayBuffer[i-1];
else
dxMat->data.fl[i] = (lpGrayBuffer[i+1] -lpGrayBuffer[i-1]) / 2.0f;
 if(abs(dxMat->data.fl[i]) > max)
max = abs(dxMat->data.fl[i]);
}
for(int i = 0; i < width; i++)
{
if(abs(dxMat->data.fl[i]) < ((inEdgeThreshold / 100.0) * max))
dxMat->data.fl[i] = 0;
}
for(i = 1; i < width; i++)
{
if(abs(dxMat->data.fl[i]) > 0)
{
leftEdge = i;
```

```
break;
}
}
for(i = width - 1; i >= 0; i--)
{
if(abs(dxMat->data.fl[i]) > 0)
{
rightEdge = i;
break;
}
}
free(lpGrayBuffer);
cvReleaseMat(&dxMat);
}
void ComputeEnergyDCT(CvMat* lpRedSourceMat, int inBlockSize,  int &inLeftEdge, int &inRightEdge, int inCompressionFactor, CvMat* lpEnergyMat)
{
int lnHorBlks = 0, lnVerBlks = 0;
int lnCountHB = 0, lnCountVB = 0;
CvMat *lpDCTBlkMat, *lpBlkMat;
lnHorBlks = (lpRedSourceMat->width) / inBlockSize;
lnVerBlks = (lpRedSourceMat->height) / inBlockSize;
int leftEdgeBlock, rightEdgeBlock;
leftEdgeBlock = (inLeftEdge / inCompressionFactor) / inBlockSize;
rightEdgeBlock = (inRightEdge / inCompressionFactor) / inBlockSize;
int i,j,k;
```

```
#pragma omp parallel for private(lnCountHB, lnCountVB, i, j, k,lpBlkMat,lpDCTBlkMat)
shared(lpEnergyMat) num_threads(10)for(lnCountHB =0; lnCountHB < lnHorBlks;
lnCountHB++)

{

lpBlkMat = cvCreateMat(inBlockSize,inBlockSize,CV_32FC1);

lpDCTBlkMat = cvCreateMat(inBlockSize,inBlockSize,CV_32FC1);

for(lnCountVB =0; lnCountVB < lnVerBlks; lnCountVB++)

{

k = 0;

for(j = lnCountVB*inBlockSize; j < lnCountVB*inBlockSize + inBlockSize; j++)

{

for(i = lnCountHB*inBlockSize; i < lnCountHB*inBlockSize + inBlockSize; i++)

{

lpBlkMat->data.fl[k++] = lpRedSourceMat->data.fl[j*lpRedSourceMat->width + i];

}

}

cvDCT(lpBlkMat, lpDCTBlkMat, CV_DXT_FORWARD);

if(lnCountHB > leftEdgeBlock + 5 && lnCountHB < rightEdgeBlock -5)

lpEnergyMat->data.fl[(lnCountVB*lpEnergyMat->width)+lnCountHB] = ComputeEnergy

(lpDCTBlkMat, inBlockSize);

}

cvReleaseMat(&lpBlkMat);

cvReleaseMat(&lpDCTBlkMat);

}

}
```

```
void ComputeThreshold(CvMat* lpRedSourceMat, int inBlockSize, float lfAvgEnergy, float
ifAlgorithmSensitivity, CvMat* lpEnergyMat)

{

int lnHorBlks = 0, lnVerBlks = 0;

int lnCountHB = 0, lnCountVB = 0;

int k = 0;

lnHorBlks = lpRedSourceMat->width / inBlockSize;

lnVerBlks = lpRedSourceMat->height / inBlockSize;

#pragma omp parallel for private(lnCountVB, lnCountHB)

for(lnCountVB = 0; lnCountVB < lnVerBlks; lnCountVB++)

{

for(lnCountHB = 0; lnCountHB < lnHorBlks; lnCountHB++)

{

if(lpEnergyMat->data.fl[(lnCountVB*lpEnergyMat->width)+lnCountHB] >
(ifAlgorithmSensitivity * lfAvgEnergy))

{

lpEnergyMat->data.fl[(lnCountVB*lpEnergyMat->width)+lnCountHB] = 1.0f;

}

else

{

lpEnergyMat->data.fl[(lnCountVB*lpEnergyMat->width)+lnCountHB] = 0.0f;

}

}

}

}
```

```
float ComputeEnergy(CvMat *ipDCTBlkMat, int inBlockSize)
{
int i, j, k = 0;
float lfTemp, lfDCValue;
float lfSum = 0;
//lfDCValue = fabs(ipDCTBlkMat->data.fl[0]);
lfDCValue = 0;
for(i = 0; i < 3; i++)
{
for(j = 0; j < 3; j++)
{
lfDCValue += fabs(ipDCTBlkMat->data.fl[(j*ipDCTBlkMat->width)+i]);
}
}
for(i = 0; i < inBlockSize; i++)
{
for(j = 0; j < inBlockSize; j++)
{
lfTemp = fabs(ipDCTBlkMat->data.fl[k++]);
lfSum = lfSum + lfTemp;
}
}
return (lfSum - lfDCValue);
}
```

```
void FillDefectInfo(int inImageWidth, int inMinDefectSize, int inMinCropWidth,
int inMinCropHeight, float *inEnergyBuffer, EnergyBufferInfo
*inoutEnergyBufferInfo,DefectBlockInfo *inoutDefectBlockInfo, int &inDefectCount)

{

CvMat *lpEnergyMat = cvCreateMat(inoutEnergyBufferInfo->height, inoutEnergyBufferInfo-
>width, CV_32FC1);

CvMat *tempMat = cvCreateMat(inoutEnergyBufferInfo->height,
inoutEnergyBufferInfo>width, CV_8UC1);

memcpy(lpEnergyMat->data.fl, inEnergyBuffer, sizeof (float)*(inoutEnergyBufferInfo->height
* inoutEnergyBufferInfo->width));

//cvConvertScale(lpEnergyMat, tempMat, 255.0f);

int i;

#pragma omp parallel for private(i) shared(lpEnergyMat, tempMat)

for(i = 0; i < lpEnergyMat->width*lpEnergyMat->height; i++)

{

tempMat->data.ptr[i] = lpEnergyMat->data.fl[i] * 255.0f;

}

int lnEnlargeFactor = inImageWidth / lpEnergyMat->width;

CvMemStorage *storage = cvCreateMemStorage();

CvSeq *contours;

CvRect rect;

cvFindContours(tempMat, storage, &contours, sizeof (CvContour), CV_RETR_EXTERNAL,
CV_CHAIN_APPROX_NONE);

i = 0;

for(; contours != 0; contours = contours->h_next)

{

rect = cvBoundingRect(contours);
```

```
if((rect.width * lnEnlargeFactor * rect.height * lnEnlargeFactor) < inMinDefectSize)

continue;

inoutDefectBlockInfo[i].offsetX = rect.x * lnEnlargeFactor - (int)(inoutEnergyBufferInfo->leftEdge * lnEnlargeFactor);

inoutDefectBlockInfo[i].offsetY = rect.y * lnEnlargeFactor;

inoutDefectBlockInfo[i].height = rect.height *lnEnlargeFactor;

inoutDefectBlockInfo[i].width = rect.width * lnEnlargeFactor;

inoutDefectBlockInfo[i].rightEdge = (int)(inoutEnergyBufferInfo->rightEdge *lnEnlargeFactor);

inoutDefectBlockInfo[i].leftEdge = (int)(inoutEnergyBufferInfo->leftEdge * lnEnlargeFactor);

inoutDefectBlockInfo[i].cropOffsetX = rect.x * lnEnlargeFactor - (int)(inMinCropWidth / 2);

if(inoutDefectBlockInfo[i].cropOffsetX < 0)

inoutDefectBlockInfo[i].cropOffsetX = 0;

inoutDefectBlockInfo[i].cropOffsetY = inoutDefectBlockInfo[i].offsetY - (int)(inMinCropHeight / 2);

if(inoutDefectBlockInfo[i].cropOffsetY < 0)

inoutDefectBlockInfo[i].cropOffsetY = 0;

inoutDefectBlockInfo[i].cropWidth = inoutDefectBlockInfo[i].width + inMinCropWidth;

if(inoutDefectBlockInfo[i].cropWidth + inoutDefectBlockInfo[i].cropOffsetX > inImageWidth)

inoutDefectBlockInfo[i].cropWidth = inImageWidth - inoutDefectBlockInfo[i].cropOffsetX;

inoutDefectBlockInfo[i].cropHeight = inoutDefectBlockInfo[i].height + inMinCropHeight;

if(inoutDefectBlockInfo[i].cropHeight + inoutDefectBlockInfo[i].cropOffsetY > inImageWidth / 2)

inoutDefectBlockInfo[i].cropHeight =

(inImageWidth / 2) - inoutDefectBlockInfo[i].cropOffsetY;

i++;

}

inDefectCount = i;
```

```
cvReleaseMemStorage(&storage);

cvReleaseMat(&lpEnergyMat);

cvReleaseMat(&tempMat);

}

void Crop(unsigned char *ipBuffer, int inImageWidth, int inImageHeight, int inDefectCount,
float *inEnergyBuffer, EnergyBufferInfo *inEnergyBufferInfo, unsigned char *outDefectBuffer,
DefectBlockInfo *inDefectBlockInfo, long inDefectBufferSize)

{

int lnEnlargeFactor = inImageWidth / inEnergyBufferInfo->

width;

CvMat* lpEnergyMat = cvCreateMat(inEnergyBufferInfo->

height, inEnergyBufferInfo->width, CV_32FC1);

CvMat* tempMat = cvCreateMat(inEnergyBufferInfo->height,

inEnergyBufferInfo->width, CV_8UC1);

memcpy(lpEnergyMat->data.fl, inEnergyBuffer, sizeof

(float)*lpEnergyMat->height*lpEnergyMat->width);

// cvConvertScale(lpEnergyMat, tempMat, 255.0f);

int i;

#pragma omp parallel for private(i) shared(lpEnergyMat,

tempMat)

for(i = 0; i < lpEnergyMat->width*lpEnergyMat->height;

i++)

{

tempMat->data.ptr[i] = lpEnergyMat->data.fl[i] * 255.0f;

}

CvMat *lpMaskMat = NULL;
```

```
lpMaskMat = cvCreateMat(inImageHeight, inImageWidth, CV_8UC1);
cvZero(lpMaskMat);
int basex, basey;
int y, x, j;
#pragma omp parallel for private(i,j,basex,basey,y,x)
for(i = 0; i < inEnergyBufferInfo->height; i++)
{
for( j = 0; j < inEnergyBufferInfo->width; j++)
{
if(inEnergyBuffer[(i * inEnergyBufferInfo->
width) + j] != 0)
{
basex = j * lnEnlargeFactor;
basey = i * lnEnlargeFactor;
for(y = 0; y < lnEnlargeFactor; y++)
{
for(x = 0; x < lnEnlargeFactor;
x++)
{
lpMaskMat->data.ptr[(basey +
y) * lpMaskMat->width + (basex + x)] = tempMat->data.ptr[(i *
9
inEnergyBufferInfo->width) + j];
}
}
}
```

```
}
}
CvMat *lpImageMat = cvCreateMat(inImageHeight,
inImageWidth, CV_8UC1);
memcpy(lpImageMat->data.ptr, ipBuffer, inImageHeight *
inImageWidth);
unsigned char *maskBuffer = (unsigned char*) malloc
(inDefectBufferSize);
CvRect rect;
long lnSumSize = 0;
CvMat *lpMat;
for(int i = 0; i < inDefectCount; i++)
{
lpMat = cvCreateMatHeader
(inDefectBlockInfo[i].cropHeight, inDefectBlockInfo
[i].cropWidth, CV_8UC1);
rect.x = inDefectBlockInfo[i].cropOffsetX;
rect.y = inDefectBlockInfo[i].cropOffsetY;
rect.height = inDefectBlockInfo[i].cropHeight;
rect.width = inDefectBlockInfo[i].cropWidth;
cvGetSubRect(lpImageMat, lpMat, rect);
CvMat *lpSubMat;
lpSubMat = cvCloneMat(lpMat);
memcpy(outDefectBuffer + lnSumSize, cvPtr2D
(lpSubMat,0,0), lpSubMat->width*lpSubMat->height);
cvReleaseMat(&lpMat);
```

```
cvReleaseMat(&lpSubMat);
lpMat = cvCreateMatHeader(inDefectBlockInfo
[i].cropHeight, inDefectBlockInfo[i].cropWidth, CV_8UC1);
cvGetSubRect(lpMaskMat, lpMat, rect);
lpSubMat = cvCloneMat(lpMat);
memcpy(maskBuffer + lnSumSize, cvPtr2D
(lpSubMat,0,0), lpSubMat->width*lpSubMat->height);
lnSumSize += lpSubMat->width * lpSubMat->
height;
cvReleaseMat(&lpMat);
cvReleaseMat(&lpSubMat);
}
//Code for Adding Mask
int lnWidth;
#pragma omp parallel for private(lnWidth)
for(lnWidth = 0; lnWidth < inDefectBufferSize; lnWidth++)
{
outDefectBuffer[lnWidth] = outDefectBuffer[lnWidth]& 254;
if(maskBuffer[lnWidth]!=0)
outDefectBuffer[lnWidth] |= 1;
}
//Code ended
cvReleaseMat(&lpImageMat);
cvReleaseMat(&lpMaskMat);
cvReleaseMat(&lpEnergyMat);
cvReleaseMat(&tempMat);
```

```
free(maskBuffer);

}
```

C++ program for extraction of textural and geometrical features.

```
#include "cv.h"
#include "cxcore.h"
#include "highgui.h"
#include "math.h"

#define PI       3.14285


struct TexturalFeatures
{
        float autoCorrelation;
        float contrast;
        float correlationM;
        float correlationP;
        float clusterProminence;
        float clusterShade;
        float dissimilarity;
        float energy;
        float entropy;
        float homogeneityM;
        float homogeneityP;
        float maximumProbability;
        float sumOfSqaures;
        float sumAverage;
        float sumVariance;
        float sumEntropy;
        float differenceVariance;
        float differenceEntropy;
        float informationMeasureOfCorrelation1;
        float informationMeasureOfCorrelation2;
        float inverseDifferenceNormalized;
        float inverseDifferenceMomentNormalized;
};

struct GeometricalFeatures
{
        float area;
        float aspectRatio;
        float perimeter;
        float centriod_X;
        float centriod_Y;
```

```
	float circularity;
	float orientation;
	float eccentricity;
	float elongation;
	float mean;
	float variance_X;
	float variance_Y;
	float skew12;
	float skew21;
	float skew03;
	float skew30;
	float weightedCentroid_X;
	float weightedCentroid_Y;
};



struct DefectBlockInfo
{
	int cropOffsetX;
	int cropOffsetY;
	int cropWidth;
	int cropHeight;
	int offsetX;
	int offsetY;
	int width;
	int height;
	int	leftEdge;
	int	rightEdge;
};

void GetDefectData(unsigned char *inCropData, DefectBlockInfo *inDefectInfo, unsigned char
*outDefectData);

void ComputeGeometricalFeatures(unsigned char *inDefectData, DefectBlockInfo
*inDefectInfo, GeometricalFeatures *outGeometricalFeatures);

void ComputeTexturalFeatures(unsigned char *inDefectData,int indH,int indV,int inResolution,
DefectBlockInfo *inDefectInfo, TexturalFeatures *outTexturalFeatures);

void ComputeGlcm(unsigned char *lpScaledPixels, float **lpGlcmMat, int inResolution, int
inWidth, int inHeight,int indH, int indV);

float ComputeGlcmSum(float **lpGlcmMat, int inResolution);

float ComputeGlcmVariance(float **lpGlcmMat, int inResolution);
```