UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

DANIEL KATZ BONELLO

CONTRIBUIÇÃO PARA AS TÉCNICAS DE DETECÇÃO DE FALHAS EM PLACAS DE CIRCUITO IMPRESSO UTILIZANDO A TRANSFORMADA RÁPIDA DE WAVELET

CONTRIBUTION TO THE PRINTED CIRCUIT BOARD DEFECT DETECTION TECHNIQUES BASED ON FAST WAVELET TRANSFORM

CAMPINAS
2020

DANIEL KATZ BONELLO

CONTRIBUTION TO THE PRINTED CIRCUIT BOARD DEFECT DETECTION TECHNIQUES BASED ON FAST WAVELET TRANSFORM

CONTRIBUIÇÃO PARA AS TÉCNICAS DE DETECÇÃO DE FALHAS EM PLACAS DE CIRCUITO IMPRESSO UTILIZANDO A TRANSFORMADA RÁPIDA DE WAVELET

Dissertation presented to the Faculty of Engineering Electrical of the University of Campinas in partial fulfillment of the requirements for the degree of Master in the area of Electrical Engineering in the field of Telecommunications and Telematics.

Dissertação apresentada à Faculdade de Engenharia Elétrica da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica, na área de Telecomunicações e Telemática.

Orientador: Prof. Dr. Yuzo Iano

CAMPINAS
2020

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Elizangela Aparecida dos Santos Souza - CRB 8/8098

B641c

Bonello, Daniel Katz, 1991-
Contribution to the printed circuit board defect detection techniques based on fast wavelet transform / Daniel Katz Bonello. – Campinas, SP : [s.n.], 2020.

Orientador: Yuzo Iano.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Localização de falhas (Engenharia). 2. Processamento de imagens. 3. Algoritmos computacionais. 4. Transformada wavelet. 5. Visão por computador. 6. Reconhecimento de padrões (Engenharia). I. Iano, Yuzo, 1950-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Contribuição para as técnicas de detecção de falhas em placas de circuito impresso utilizando a transformada rápida de wavelet
Palavras-chave em inglês:
Fault location (engineering)
Processing image
Computer algorithms
Wavelet transforms
Computer vision
Pattern recognition
Área de concentração: Telecomunicações e Telemática
Titulação: Mestre em Engenharia Elétrica
Banca examinadora:
Yuzo Iano [Orientador]
Silvio Renato Messias de Carvalho
Ricardo Barroso Leite
Data de defesa: 20-01-2020
Programa de Pós-Graduação: Engenharia Elétrica

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0003-2222-9933
- Currículo Lattes do autor: http://lattes.cnpq.br/9538006816758810

COMISSÃO JULGADORA - DISSERTAÇÃO DE MESTRADO

Candidato: Daniel Katz Bonello RA: 208944

Data da Defesa: 20 de janeiro de 2020

Título da Tese: "Contribution to the printed circuit board defect detection techniques based on fast wavelet transform".

A ata de defesa, com as respectivas assinaturas dos membros da Comissão Julgadora, encontra-se no SIGA (Sistema de Fluxo de Dissertação/Tese) e na Secretaria de Pós-Graduação da Faculdade de Engenharia Elétrica e de Computação.

DEDICATION

I dedicate this work firstly to God, by always has give me forces to move forward, and to my family by the unconditional support demonstrated during the development of this work.

ACKNOWLEDGMNETS

I thank God by Always help me reach my objectives.

To my family, by unconditional support demonstrated in all moments during my Master Degree.

To my orientator, Prof. Dr. Yuzo Iano, I am thankful by the given opportunity to realize a Master Degree at Unicamp, by orientation and confidence.

To the research team of Laboratory of Communications (LCV).

To my friends this always be present during my Master Degree.

To all professor and employees of School of Electrical and Computer Engineering (FEEC) this has helped me during the development of this work, always ready to give me help.

To all the people this contributed in a directly or indirectly way for this work would be realized.

ABSTRACT

Various concentrated work has been developed in the area of computer vision applied to detection of failures on printed circuit boards (PCBs), aiming at reducing the possibility of the occurrence of the fabrication defects. In this research, based on PCI's – without mounting reference and test layout models, the objective is to study is the application of an image subtraction technique to the failure detection of those bare printed circuit boards layouts using the Fast Wavelet Transform (FWT) during the image processing. By developing the Discrete Wavelet Transform (DWT) equations, one may compare the efficiency of this image processing technique using linear simulations developed in MATLAB. Significative results were obtained regarding the reduction of the image processing time and image classification efficiency, thus indicating advantages in using this technique in the simulated cases.

Keywords*: Failure prevention. Image processing. Subtraction algorithm. Fast Wavelet Transform. Computer vision. Pattern recognition.*

RESUMO

Muitos trabalhos foram desenvolvidos na área de visão computacional aplicados à detecção de falhas em placas de circuito impresso (PCI's), visando reduzir a possibilidade de ocorrência de defeitos de fabricação. Nesse trabalho, a partir de modelos de layouts de referência e de teste de PCI's – sem componentes, estudou-se a aplicação de uma técnica de subtração de imagem para a detecção de falhas desses layouts de placas de circuito impresso utilizando a Transformada Rápida de Wavelet (FWT) durante o processamento de imagem. Assim, desenvolvendo as equações da Transformada de Wavelet Discreta (DWT), pode-se comparar a eficácia dessa técnica de processamento de imagem utilizando simulações lineares em MATLAB. Foram obtidos resultados significativos na redução do tempo de processamento e eficácia de classificação de imagem, indicando vantagens no uso desse tipo de técnica de processamento de imagem nos casos simulados.

Palavras-chave: Prevenção de falhas. Processamento de imagem. Algoritmo de subtração. Transformada Rápida de Wavelet. Visão computacional. Reconhecimento de padrões.

ILLUSTRATION LIST

TABLE LIST

SUMMARY

1 INTRODUCTION

In this chapter is given the motivation by which beginning the studies presented in this dissertation and comparisons related to the works already done in the computer vision area applied to the detection of failures on printed circuit boards (PCB's). Also are presented the objectives of this work, and the choosed manner to introduce the most relevant concepts to the understanding of developed work to the composition of this dissertation.

1.1 MOTIVATION

According to the (THIBADEAU, 1981), printed circuit boards are rigid or semi-rigid boards on which geometrical patterns are printed in copper or some other conductive material. They function is to replace the wiring and perhaps some of the electrical circuit components in everything from toasters to fighter planes. Printing a wire can be less expensive than fitting a real one and soldering it.

The automated inspection of printed circuit boards (PCB's) serves a purpose which is traditional in computer technology. The purpose is to relieve human inspectors of the tedious and inefficient task of looking for those defects in PCBs which could lead to electrical failure. Automated, computer based, inspection relieves this problem by providing a machine solution.

Even, according to ABNEE (Associação Brasileira da Indústria Elétrica e Eletrônica), the exportations of electro-electronic products have been summary US$ 419.5 million, in the month of December 2019 (http://www.abinee.org.br/abinee/decon/decon10.htm). This not counting the global shipments growth, this motivates the manufacturing investments designed to the PCB's production, also increasing the demand for PCB's inspection processes with reduced inspection time and high-efficiency failures classification to attend the high demand of those products.

Certainly some inspection technique for reduce the bare PCB's image processing time and enhance the failures classification would be profitable, so decreasing the probabilities of a Company ship a defective product to the final consumers, this would not be economically viable.

In this work, the main focus is to develop a bare PCB inspection technique through an image subtraction algorithm using the Fast Wavelet Transform (FWT) viewing the optimization of time reduction and image classifying efficiency of those bare PCB's inspection processes, adopting a theoretical and practical approach, as will be explained in the following chapters of this dissertation.

1.2 LITERATURE REVIEW

Is wished to develop this work with basis in the classical literature of Image Processing area – (ERCAL, 1997), (MOGANTI, 1996) and (TATIBANA, 1997) – as well as materials utilized in courses of image processing techniques – (PHAM AND ALCOCK, 2002) and (GONZALEZ AND WOODS, 2010). An interesting work also in the field of computer vision, although with relation to the project of an PCB failure analyzer can be founded in (OTANI *et al.*, 2012). Books introducing the image processing theory are also utilized – (NIXON, 2008), (GONZALEZ AND WOODS, 1992), (THEODORIDIS AND KOUTROUMBAS, 2009) and (JAIN, 1986).

In despite of the approach utilized in (ERCAL, 1997), this in this point of its articles do not realize a large detailing about the image subtraction technique using wavelets, in this work was decided by include in the image processing mathematical model of the Fast Wavelet Transform (FWT) operation and its consequences in the image processing time and image classifying efficiency, similar to the model proposed by (ORTEGA *et al.*, 2007).

This model – image subtraction technique or image difference technique – also was utilized in the works of, (BORTHAKUR *et al.*, 2015), (PAUL *et al.*, 2016), (IBRAHIM *et al.*, 2011), (SHINDE AND MORADE, 2015) and (KAUSHIK AND ASHRAF, 2012). In those works, a model of image difference technique is utilized in the PCB's image processing, which takes in account the distinction between PCB's without mounting and PCB's with mounting, as well as the effects of image time processing and classifying efficiency between those PCB's templates. Moreover, in those works is utilized the *subtraction algorithm* to the optimal approach of image processing steps:

- *read the PCB image*
- *resize the image to desired size*
- *convert the RGB image to gray scale*
- *thresholding*
- *convert to binary image*
- *XOR operation with template image*
- *extracting the features using regional properties*
- *resultant image where defect detected*

Viewing obtaining a better and faster process which provides both quantitative and qualitative results. The work here presented utilizes this approach initially, and after also extends the calculus to the mathematical model of the Fast Wavelet Transform (FWT), both with the optimization of image time processing and classifying efficiency in each one of the 11 bare PCB's selected layout images to obtain quantitative and qualitative results.

We can still observe the usage of wavelets in the works of (IBRAHIM *et al.*, 2002), (BORTHAKUR *et al.*, 2015), (PAUL *et al.*, 2016), (ORTEGA *et al.*, 2007), (CHOKSI *et al.*, 2014), (MALLAT, 1996), (IBRAHIM AND AL-ATTAS, 2005) and (BAKAR, 1998), utilizing the computation of the wavelet transform in a two-dimensional signal (i.e. image), including with some cases of image time processing of 0.761 s and failure detection efficiency about 75%. Even, in (IBRAHIM AND AL-ATTAS, 2005) were included some modifications of Continuous Wavelet Transform (CWT) from de development of Fourier transform (*mother wavelet* function).

In (IBRAHIM AND AL-ATTAS, 2004), also were used the image subtraction technique model, with a noise filter employed after the threshold step in the image subtraction operation utilizing the parameters of *Noise Free Positive Image* and *Noise Free Negative Image* before

XOR operation. Now in (CHOKSI *et al.*, 2014), a similar model to the (IBRAHIM AND AL-ATTAS, 2005) is employed, however utilizing a quantitative and qualitative method more complete this takes into account as the No. of defects in test image as the No. of defects are found in error image indicating the Efficiency (dB4 Wavelet) in percentage to all simulated cases. A different method of image processing technique, through the application of Template Matching technique for PCB's SMT components image processing was approached in (BHARDWAJ, 2016).

A approach to the problem of digital image processing inspection techniques to PCB's produced in small series is presented in (SZYMANSKI, 2014) through the usage of SIFT algorithm to the development of a processing image architecture proper to PCB's small series production inspection, and in (COSTA, 2014) are utilized Bayesian networks and multiagent system techniques to implement an adaptative PCB fault diagnosis system also to the PCB's small series production. Other work this deserve be noted in the field of computer vision applied to the image processing is (FERREIRA, 2012), utilizing reconfigurable systems.

In (LONDE AND SHIRE, 2014) an image comparison technique utilizing image subtraction is cited as a method for defect detection, although the subtraction algorithm for image processing is not associated with the methods for defect classification. In the approach utilized in (KHUSHWAHA *et al.*, 2015) to found the bare PCB's layout errors, a subtraction method is utilized, as well as a defect correction technique.

A study about the development of an algorithm in image processing which detects flux of defects at PCB re-flow process is introduced in (TEOH ONG *et al.*, 2013), including with an image pattern matching technique and algorithm. An approach of application of an algorithm for bare PCB defect classification classifying six different defects, namely, *missing hole*, *pinhole*, *underetch*, *short-circuit*, *open-circuit*, and *mousebit* is demonstrated in (IBRAHIM *et al.*, 2012) utilizing the image subtraction algorithm to identify the major defects founded in those bare PCB's necessary classify them an quantify as well.

Finally, in (SUHASINI *et al.*, 2015), is utilized an approach of image subtraction technique with *digital PCB's circuit layouts*, similar to those bare PCB's layouts utilized in this work, however applied just only for one image sampling.

1.3 OBJECTIVES

The objective of this work consists in propose an inspection technique of bare PCB's layout digital image processing for defects detection and classification, reducing the bare PCB's layout image processing time and increasing the image classifying efficiency through the implementation of the Fast Wavalet Transform (FWT) in the subtraction algorithm applied to the 11 simulated bare digital PCB's layouts cases: reference and testing samples. Some goals pertinent to this work are delineated, as follow:

- Demonstrate, from applications of image processing concepts, how to find and classify the bare PCB's layout defects with the image subtraction technique;

- Utilizing previously data presented in the articles above mentioned, propose an image subtraction technique utilizing a subtraction algorithm developed in MATLAB to reach and optimize a better performance in the image time processing and defects classifying efficiency from the implementation of the Fast Wavelet Transform (FWT) in this code, still ensuring this its utilization be more necessary and benefic how bigger the bare PCB digital layout complexity;

- Implement the Fast Wavelet Transform (FWT) in two-dimension – from de development of Discrete Wavelet Transform (DWT) equations – and analyze its behavior in the 11 simulated cases, comparing the results with those of were used different mathematical models in the simulations;

- Evaluate the advantages and disadvantages of this technique, proposing better ideas to the future works.

Is believed this work presented here has a differential by present the Fast Wavelet Transform (FWT) as the main model to the optimization of image processing time and classifying efficiency, this will be then utilized in the image subtraction algorithm, with its time and efficiency values varying in function of bare PCB digital layout complexity. In this mode, is expected to reach the objective of image time processing below 0.761 s (the best example founded in the literature, according to ORTEGA *et al.*, 2007) and minimum failure detection efficiency above 75% (best results founded in the literature), reducing the possibility of passing defect PCB's in the image inspection processing.

1.4 ORGANIZATION OF THIS DISSERTATION

In the Chapter 2, is given a shortly introduction to the phenomena involved in the image processing systems, and its computational and mathematic modeling. Now in the Chapter 3, are deducted the main image processing concepts referring to the subtraction model technique taken as basis of study.

In the Chapters 4 and 5 are presented the methodology of image subtraction algorithm technique utilizing the Fast Wavelet Transform (FWT) in function of developing the Discrete Wavelet Transform (DWT) equations, respectively. In the Chapter 6, is demonstrated how to get a model of PCB's failure detection criteria, to the analysis of inclusion of those criterias in the simulation behavior of bare PCB's digital layouts controlled parameters.

Finally, in the Chapters 7 and 8 are presented the computational implementations of image subtraction technique utilizing the Fast Wavelet Transform (FWT) in MATLAB, the simulations results, with a pertinent discussion.

The work is concluded with the Chapter 9, summarizing the results and recommending which would be the possible next steps to future works.

The mathematic deductions of Discrete Wavelet Transform (DWT) and Fast Wavelet Transform (FWT), modeling in the space of time, two-dimensional functions, etc., are given in the Annex of this dissertation.

2 MODELING OF AN IMAGE ACQUISITION SYSTEM

The idea of this chapter is to provide to the reader a general vision about the basics phenomena involved in the image generation process of an image acquisition system. It's important this reader has a basic knowledge about the image acquisition devices, because those are the elements responsible to the PCB's image processing generation. The presentation of elements related to the image acquisition system in this chapter allows this its modeling can be incorporated to the wavelets equations of image subtraction technique. The literature regarding to the image acquisition systems is very vast (TATIBANA, 1997).

Cameras are the basics elements of image capture and generation in a PCB image acquisition system, as well as in the image inspection process. It's justly due to the image inspection algorithm and defects classifying in the subtraction technique this utilized cameras shall be good enough to help the image processing inspection process as whole. By stay connected between PCB's samples and embedded control system, serve as an interface between these entities, generating a high quantity of PCB's templates layouts of reference and testing images in a time interval relatively short. Its application greatly affects the image subtraction algorithm in the most several aspects, for example: image time processing, image readjustment to the desired size, PCB image conversion to gray scale, ect.

Escapes from the scope of this work detailing the exact influence of each PCB image acquisition system device in the image generation process and image subtraction technique. However, allows a small explanation about the basic phenomenology involved.

2.1 QUANTITIES OF INTEREST

The image generation for PCB's is given due a combination of two factors:

- Lighting/illumination in the PCB's conveyor belt;
- Image processing system in the embedded control computer.

Initially in a model of PCB image inspection system can be identified various quantities of interest in the phenomena of image generation process, as in the **Figure 2.1**.

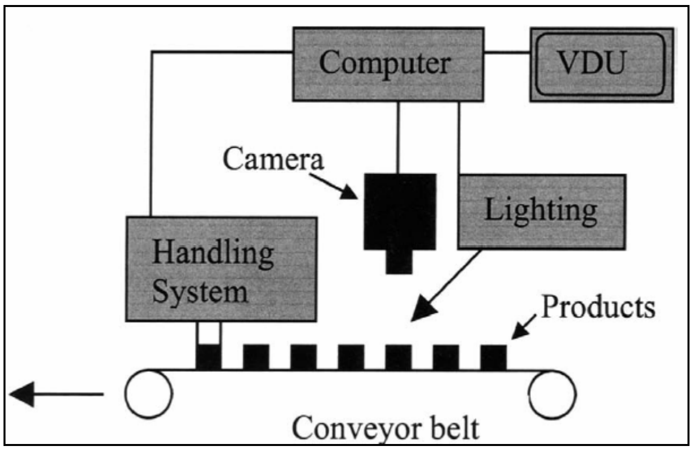


Figure 2.1 – Typical PCB image acquisition system structure

*VDU: Video Device unity

Thus, the PCB image acquisition system can be observed as a subsystem constituted of various inputs and outputs. The input variables are, basically:

- Distance between the PCB plan and sensor plan in the conveyor belt (cm): the variation of the distance between board and camera would cause the PCB's size variations between two different photographical expositions. As the developed subtraction algorithm, as described in the Chapter 3, takes into account a reference image, would be necessary develop an algorithm immune to this kind of variation;

- Rotation between the PCB plan and sensor plan (rad): the fundamental principia of photography establish this is the projection of a tridimensional reality in a sensor two-dimensional, the rotation plan this contains a two-dimensional object and the plan this contains the sensor of a photographic camera produce distortions in the size and in the proportions of photographed object. When a perfect quadrilateral is photographed, for example, this quadrilateral just only appears in the photography as a square in case of him is parallel to sensor in the moment of exposition. Case the square only is inclined in relation to the one axis, he will appear in the photo as a trapeze, which the minor size is the most distant square size from camera, and the major size is the most near square size from camera. Case exists rotation in two axes, the square would become represented in the photography as an irregular quadrilateral. Because this reason, for this can be possible further have conclusions about the trails width and spacing between trails in the photography, it's necessary this don't have rotation between those planes. If there were any rotation, should be necessary to develop an algorithm more complex to transform the image in a mode the distortions would compensate. Therefore, it's important this characteristic vary the minimum possible;

- Rotation between the PCB and the sensor axes (rad): the variation of this characteristic presents the same problem described in the previous characteristic analysis, with the exception of when the distance between none PCB points and camera sensors is modified, the correction of pixels spatial resolution is much more simple;

- PCB position (cm): as in the PCB rotation in relation to anyone axes, if there is difference between the board position inside photography among consecutive pictures, with all another variables maintained constant, even should be taken actions to recognize some board characteristic and to start of her infer the PCB position in the photography. Another way would ask for user indicates to algorithm, through a simple interface, the PCB position. As the position correlation don't necessitates of photography pixels spatial resolution corrections (this don't involve any kind of interpolation algorithm), since be corrected the distortions caused by the camera lens, it would be a simple solution to be implemented, once the spatial resolution of photography is uniform in the whole extension;

- Controlled and uniform illumination (lm): there is the necessity to maintain constant the illumination among the successive photographical expositions. The quality of image subtraction from reference and testing images acquired through a camera depends directly from the board illumination quality.

Also can be listed the outputs:

- Read the PCB image (dpi/ppi): obtain a digital image of the object to be inspect in the computer of image acquisition system;

- Resize the image to desired size (dpi/ppi): to improve the quality of the acquired image, which facilitates later processing.

However, when it's about of bare PCB's layout designs, this represents the object to be inspected and classified, there are more three remaining operations in the image generation process before the application of the image subtraction technique (XOR operation). Become to be of interest only the inputs of board samples illumination (lm), as well as the outputs of image resize (dpi/ppi).

The three remaining operations are, respectively:

- Convert the RGB image to gray scale: RGB image is also known as True color image. A true color image is an image in which each pixel is specified by three values one each for the red, blue, and green components of the pixel scalar. Grayscale image is a monochrome image or one-color image. It contains brightness information only and no color information. Then grayscale data matrix values represent intensities;

- Thresholding: threshold technique is one of the important techniques in image segmentation. This technique can be expressed as the **Eq.(2.1)**:

$$T = T[\mathrm{x}, \mathrm{y}, \mathrm{p}(\mathrm{x}, \mathrm{y}), \mathrm{f}(\mathrm{x}, \mathrm{y})] \tag{2.1}$$

Where *T* is the threshold value *x*, *y* are the coordinates of the threshold value point *p(x,y)*, *f(x,y)* are points the gray level image pixels. Threshold image *g(x,y)* can be define as the **Eq.(2.2)**:

$$g(x,y) = \begin{cases} 1 & if \quad f(x,y) > 1 \\ 0 & if \quad f(x,y) \leq 0 \end{cases} \tag{2.2}$$

- Convert to binary image: image binarization is the process of separation of pixel values into two groups, black as background and white as foreground.

Observing the **Figure 2.2** is verified the block diagram of proposed image acquisition system designed to PCB inspection. In this block can be observed the quantities of interest, as previously described in the specifications of subsystem inputs and outputs quantities.

Under the control of the computer, the PCB delivering components automatically move the given PCBs. Image acquisition and moving control subsystem receives the central computer's control commands, acquires the PCB images real-timely and preprocesses them using MATLAB. The result data will be sent to the central computer. Central computer is the core of the system. On the one hand, it controls the action of the delivering units and the image acquisition units. On the other hand, it needs to receive image data, processes the data, finds out the defect and outputs the detection report (SHINDE AND MORADE, 2015).

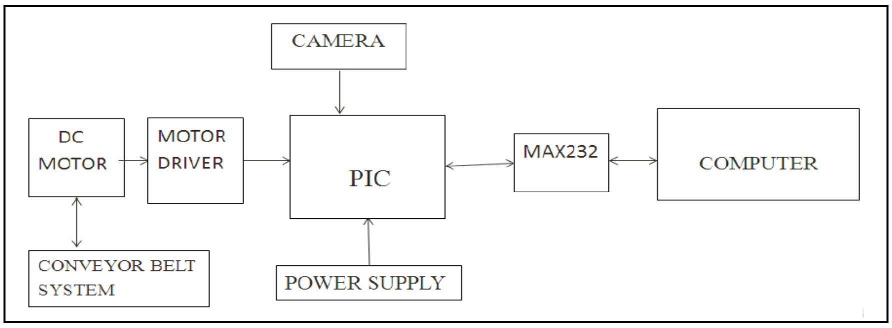


Figure 2.2 – Block diagram of PCB image acquisition system

- MAX232: is an integrated circuit this is a level converter, this transforms the signals of a serial port to adequate signals to the usage in the microprocessed circuits, for example. This circuit is developed using own characteristics, the CI MAX 232 have a structure composed by 16 connection terminals and is specially developed to realize through controllers, the conversion of signals TTL in RS232 and vice-versa.

Now having the a better understanding about the inputs and outputs variables, and the remaining operations present in a PCB image acquisition system, a theoretical modeling of those concepts relations can be started.

2.2 COMPUTATIONAL MODELLING OF AN IMAGE INSPECTION SYSTEM

Automated Visual Inspection (AVI) is the automation of the quality control of manufactured products, normally achieved using a camera connected to a computer. AVI is considered to be a branch of industrial machine vision (BATCHELOR AND WHELAN, 1997). Machine vision requires the integration of many aspects, such as lighting, cameras, handling equipment, human-computer interfaces and working practices and is not simply a matter of designing image processing algorithms. Industrial machine vision contrasts with high-level computer vision, which covers more theoretical aspects of artificial vision, including mimicking human or animal visual capabilities. **Figure 2.3** shows a machine of artificial vision and **Figure 2.4** shows the inspection system block diagram.

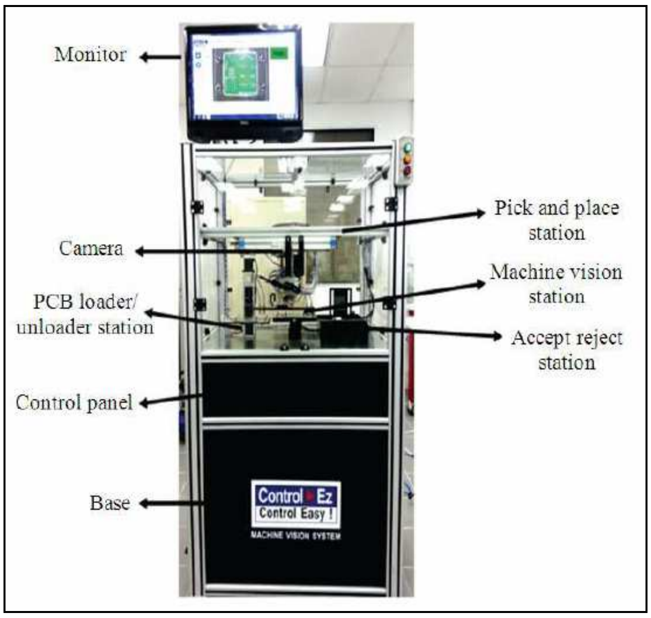


Figure 2.3 – Machine vision configuration set-up

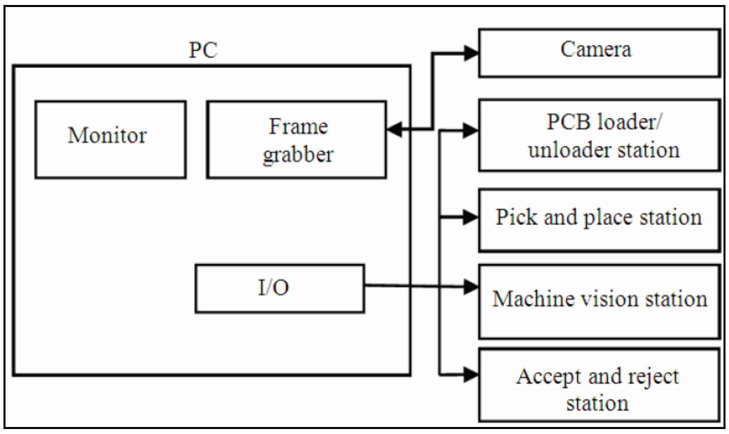


Figure 2.4 – Inspection system block diagram

In modem manufacturing, quality is so important this AVI systems and human inspectors may be used together synergistically to achieve improved quality control (SYLLA, 1993). The machine vision system is used to inspect a large number of products rapidly. The human inspector can then perform slower but more detailed inspection on objects this machine vision system considers to be borderline cases. **Figure 2.5** shows a breakdown of artificial vision.

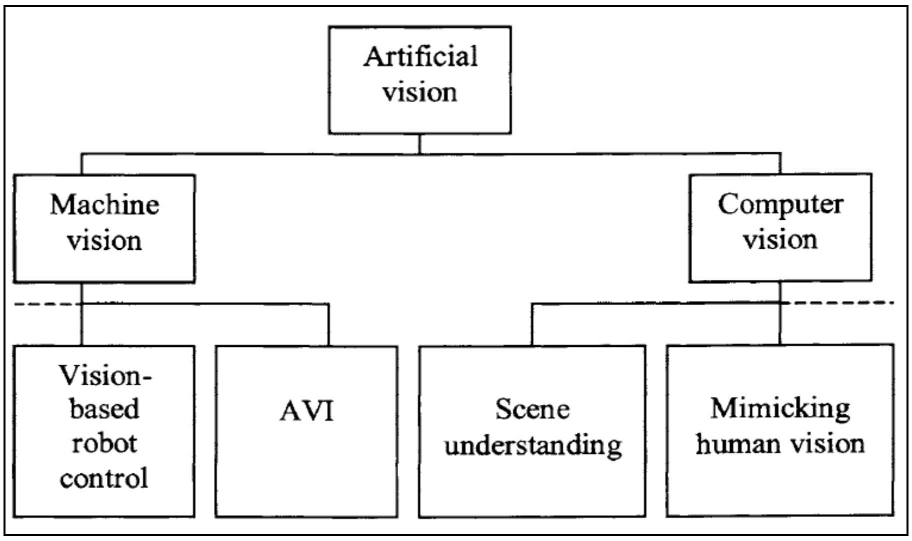


Figure 2.5 – Examples of machine and computer vision

In AVI, many conventional image-processing functions, such as thresholding (described in the previous subtopic), edge detection and morphology, have been employed. However, much recent work has focused on incorporating techniques from the area of artificial intelligence into the process. The next subtopic describes common artificial intelligence mathematic modeling and how they have been used in AVI.

2.3 MATHEMATIC MODELLING OF AN IMAGE INSPECTION SYSTEM

Artificial intelligence (AI) involves the development of computer programs this mimic some form of natural intelligence. Some of the most common AI techniques with industrial applications are *expert systems, fuzzy logic, inductive learning, neural networks, genetic algorithms, simulated annealing* and *Tabu search* (PHAM *et al.*, 1998). These tools have been in existence for many years and have found numerous industrial uses including classification, control, data mining, design, diagnosis, modeling, optimization and prediction.

To the study of PCB's image processing, are distinguish five main approaches:

1. Expert Systems: expert systems are computer programs embodying knowledge about a narrow domain for solving problems related to this domain (PHAM AND PHAM, 1988). An expert system usually comprises two main elements, a *knowledge base* and an *inference mechanism.* In many cases, the knowledge base contains several "IF - THEN" rules but may also contain factual statements, frames, objects, procedures and cases.

2. Fuzzy Logic: the use of fuzzy logic, which reflects the qualitative and inexact nature of human reasoning, can enable expert systems to be more resilient (NGUYEN AND WALKER, 1999). With fuzzy logic, the precise value of a variable is replaced by a linguistic description, the meaning of which is represented by a fuzzy set, and inferencing is carried out based on this representation. Knowledge in an expert system employing fuzzy logic can be expressed as qualitative statements or fuzzy rules.

3. Inductive Learning: the acquisition of domain knowledge for the knowledge base of an expert system is generally a major task. In many cases, it has proved a bottleneck in the process of constructing an expert system. Automatic knowledge acquisition techniques have been developed to address this problem. Inductive learning, in this case the extraction of knowledge in the form of IF-THEN rules (or an equivalent decision tree), is an automatic technique for knowledge acquisition. An inductive learning program usually requires a set of examples as input. Each example is characterized by the values of a number of attributes and the class to which it belongs (PHAM AND ALCOCK, 2002).

4. Neural Networks: There are many classifier architectures which are covered by the term Artificial Neural Networks (ANNs) (PHAM AND LIU, 1999). These networks are models of the brain in this they have a learning ability and a parallel distributed architecture.

   Like inductive learning programs, neural networks can capture domain knowledge from examples. However, they do not archive the acquired knowledge in an explicit form such as rules or decision tress. Neural networks are considered a 'black box” solution since they can yield good answers to problems even though they cannot provide an explanation of why they gave a particular answer. They also have a good generalization capability, as with fuzzy expert systems (PHAM AND ALCOCK, 2002).

5. Genetic Algorithms, Simulated Annealing and Tabu Search: Intelligent optimization techniques, such as genetic algorithms, simulated annealing and Tabu search, look for the answer to a problem amongst a large number of possible solutions (PHAM AND KARABOGA, 2000). A classic example of an optimization problem is the travelling salesman problem. In this problem, a salesman needs to visit a number of cities in the shortest possible distance. As all the cities are known, all possible route combinations are also known. Thus, the problem is one of searching through all the possible routes. When the number of cities becomes large, the number of route combinations increases excessively. Thus, a fast search technique is required to find the optimal route.

Optimization techniques are used to perform fast searching. Common AI optimisation techniques include genetic algorithms, simulated annealing and Tabu search (PHAM AND ALCOCK, 2002).

In this work, the third approach was choosed, due its implementation facility and common usage in the works referred to the image processing techniques.

Bare PCB's digital layouts models are employed to represent the PCB production samples in simulation environments, considering the answers in a linear system. The nomenclature "digital layouts" is given due to the fact of those bare PCB image binarizated samples is a part of image processing step, though its data have origin basically in physical images acquisition in the inspection system.

2.3.1 THE INDUCTIVE LEARNING FORMULA

Several inductive learning algorithms and families of algorithms have been developed. These include ID3, AQ and RULES. The ID3 algorithm, developed by (QUINLAN, 1983), produces a decision tree. At each node of the tree, an attribute is selected and examples are split according to the value this they have for this attribute. The attribute to employ for the split is based on its *entropy* value, as given in **Eq.(2.3)**. In later work, the ID3 algorithm was improved to become C4.5 (QUINLAN, 1993).

$$\text{Entropy(attribute)} = -(P_+ \log_2 P_+) - (P_- \log_2 P_-) \qquad \textbf{(2.3)}$$

Where:

6. P_+: is the proportion of examples this were correctly classified just using the specified attribute;

7. P_-: is the proportion incorrectly classified. For implementation, 0.log(0) is taken to be zero.

(PHAM AND AKSOY, 1993; 1995 a, b) developed the first three algorithms in the RULES (RULe Extraction System) family of programs. These programs were called RULES-1, 2 and 3. Later, the rule forming procedure of RULES-3 was improved by (PHAM AND DIMOV, 1997a) and the new algorithm was called RULES-3 PLUS. Rules are generated and those with the highest 'H measure' are kept. The H measure is calculated as the **Eq.(2.4)**:

$$H = \sqrt{\tfrac{E^c}{E}} \times [2 - 2\sqrt{\tfrac{E_i^c E_i}{E^c E}} - 2\sqrt{(1 - \tfrac{E_i^c}{E^c})(1 - \tfrac{E_i}{E})}] \qquad \textbf{(2.4)}$$

Where:

8. E: is the total number of instances;

9. E^c: is the total number of instances covered by the rule (whether correctly or incorrectly classified);

10. $E^c{}_i$: is the number of instances covered by the rule and belonging to target class i (correctly classified);

11. E_i: is the number of instances in the training set belonging to target class i.

Thus, applying those formulas in the AVI system, an example of decision tree is obtained as shows the **Figure 2.6**:

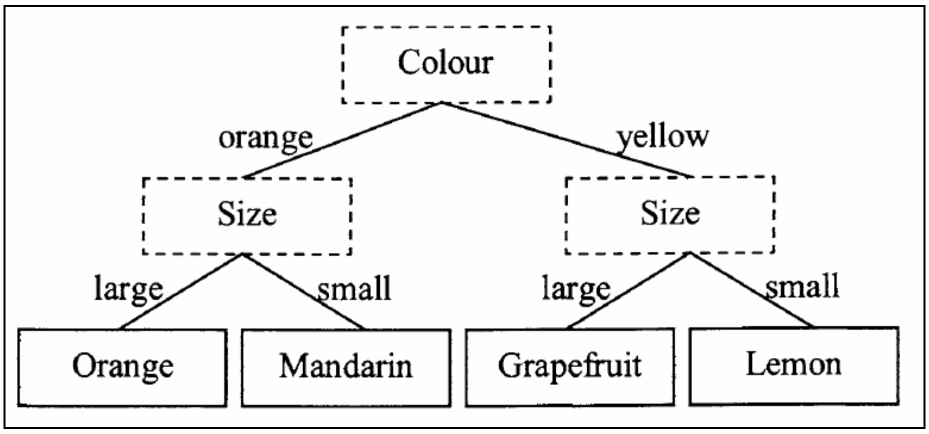


Figure 2.6 – Decision tree for classifying citrus fruit

In the *tree-based* approach to inductive learning, the program builds a decision tree this correctly classifies the training example set. Attributes are selected according to some strategy (for example, to maximize the information gain) to divide the original example set into subsets. The tree represents the knowledge generalized from the specific examples in the set. Figure 1.8 shows a simple example of how a tree may be used to classify citrus fruit. It should be noted this if size were used at the root of the tree instead of color, a different classification performance might be expected (PHAM AND ALCOCK, 2002).

In the *rule-based* approach, the inductive learning program attempts to find groups of attributes uniquely shared by examples in given classes and forms rules with the IF part as combinations of those attributes and the THEN part as the classes. After a new rule is formed, the program removes correctly classified examples from consideration and stops when rules have been formed to classify all examples in the training set (PHAM AND ALCOCK, 2002).

Now having the elements of PCB image processing in an image acquisition system, it allows the study of image subtraction technique in an inspection system, through its linear operations.

3 IMAGE PROCESSING: THE IMAGE SUBTRACTION TECHNIQUE

Further taking the model utilized by (ORTEGA *et al.*, 2007), in this work is still utilized a PCB production sample model (contemplating the quantities of interest of this work and of proper bare PCB design layout employed in this analysis), from which is derivate the PCB's digital image layout (binary image) through elementary image processing steps: *convert the RGB image to gray scale*, *thresholding*, *convert to binary image*. Also is described those image processing steps in detail in the next subtopics as well as the main concepts of XOR operation utilized in this work, unlike what have been done previously in the articles cited in Chapter 1.

In the beginning of this Chapter, are discussed *RBG* and *gray scale* aspects this would be utilized to the image processing subtraction technique applied to a bare PCB production sample in a real case. Those aspects were selected with basis in its applicability, algorithm impact and time & image classifying correlations during simulations in MATLAB.

3.1 THE RGB COLOR MODEL

According to (GONZALEZ AND WOODS, 2010), in the RGB model, each color appears in its primary spectral components of red, green and blue. This model is based on Cartesian coordinates system. The color subspace of interest is the cube shown in **Figure 3.1**, in which

RGB values are at three corners: *cyan*, *magenta*, and *yellow* are at three other corners: *black* is at the origin: and *white* is at the corner farthest from the origin. In this model, the *gray scale* (points of equal RBG value) extends from *black* to *white* along the line joining these two points. The different colors in this model are points on or inside the cube, and are defined by vectors extending from the origin. For convenience, the assumption is this all color values have been normalized so this cube shown in the **Figure 3.1** is the unit cube. This is, all values of *R*, *G* and *B* are assumed to be in the range [0, 1].

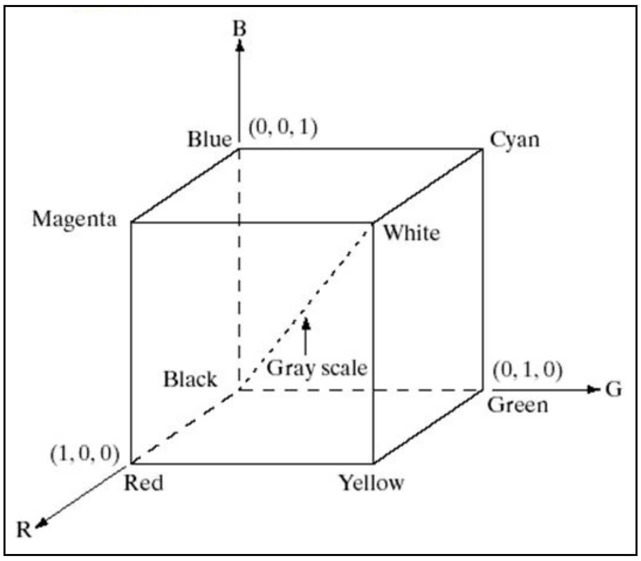


Figure 3.1 – Schematic of the RGB color cube

Images represented in RGB color model consist of three component images, one for each primary color. When fed into an RGB monitor, these three images combine on the phosphor screen to produce a composite color image. The number of bits used to represent each pixel in RGB space is called *pixel depth*. Consider an RGB image in which each of the *red*, *green* and *blue* images in an 8-bit image. Under these conditions each RGB *color* pixel (this is, a triplet of values [*R*, *G*, *B*]) is said to have a depth of 24 bits (3 image planes times the number of bits per plane). The term *full-color* image is used often to denote a 24-bit RGB color image. The total number of colors in a 24-bit RGB image is $(2^8)^3$ = 16.777.216. **Figure 3.2** shows the 24-bit RGB color cube corresponding to the diagram in **Figure 3.1**.

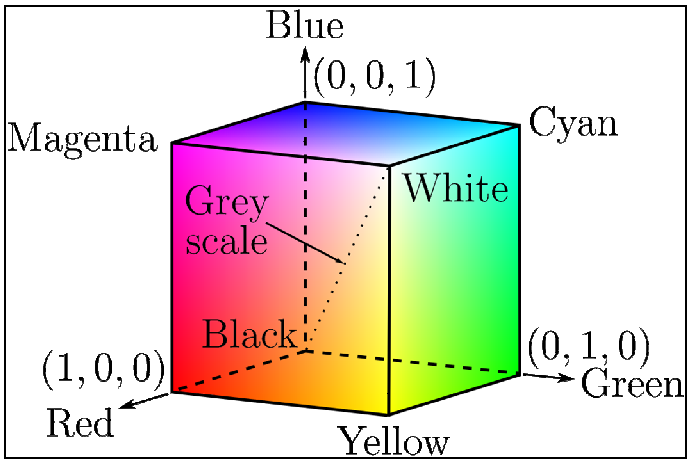


Figure 3.2 – RGB 24-bit cube

The cube shown in **Figure 3.2** is a solid, composed of the $(2^8)^3$ = 16.777.216 colors mentioned in the preceding paragraph. A convenient way to view these colors is to generate color planes (faces or cross selections of the cube). This is accomplished simply by fixing one of the three colors and allowing the other two to vary. For instance, a cross-sectional plane through the center of the cube and parallel to *GB*-plane in **Figure 3.2** and **Figure 3.1** is the plane (127, *G*, *B*) for *G*, *B* = 0, 1, 2,….255. Here we used the actual pixel values rather than the mathematically convenient normalized values in the range [0, 1] because the former values are the ones actually used in a computer to generate colors. **Figure 3.3** shows an image of the cross-sectional plane is viewed simply by feeding the three individual component images into a color monitor. In the component images, 0 represents *black* and 255 represent *white* (note this are gray-scale images). Finally, **Figure 3.4** shows the three hidden surface planes of the cube in **Figure 3.2**, generated in the same manner.

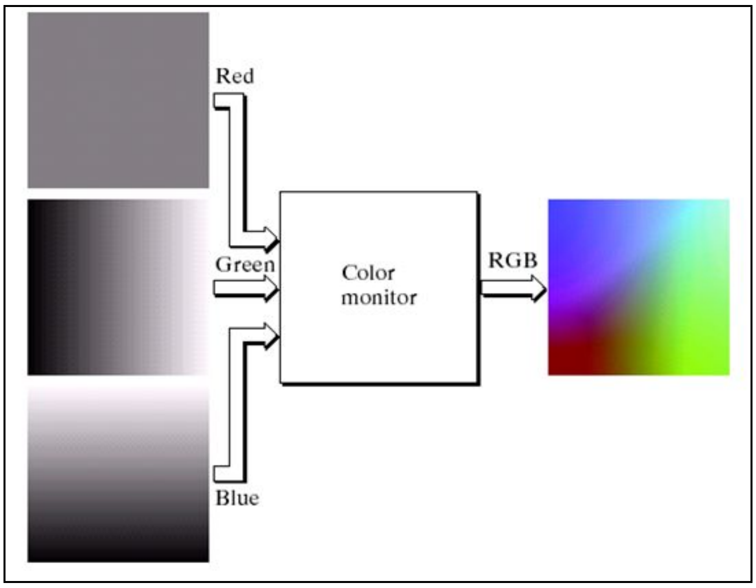


Figure 3.3 – RGB image of cross-sectional plane (127, *G*, *B*)

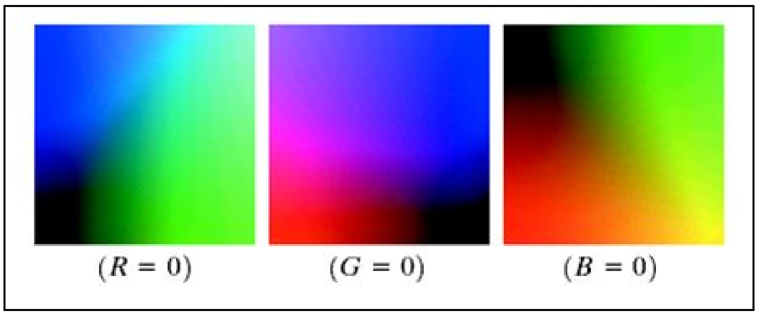


Figure 3.4 – The three hidden surface planes in the color cube

Repeating this process with each filter produces three monochromic images this are the RGB component images of the color scene (in practice, RGB color image sensors usually integrate this process into a single device). Cleary, displaying these three RGB component images in the form shown in **Figure 3.3** would yield an RGB color rendition of the original color scene.

3.1.1 THE THREE HIDDEN SURFACE PLANES IN A BARE PCB PRODUCTION SAMPLE

As described in the previous subtopic, images represented in RGB color model consist of three component images, one for each primary color. Next is demonstrated in the **Figure 3.5** the RGB image of a cross-sectional plane (127, *G*, *B*) for a bare PCB layout production sample captured through a camera in the image inspection system, as well as in the **Figure 3.6** the three hidden surface planes for this sample at (*R* = 0, *G* = 0, *B* = 0).

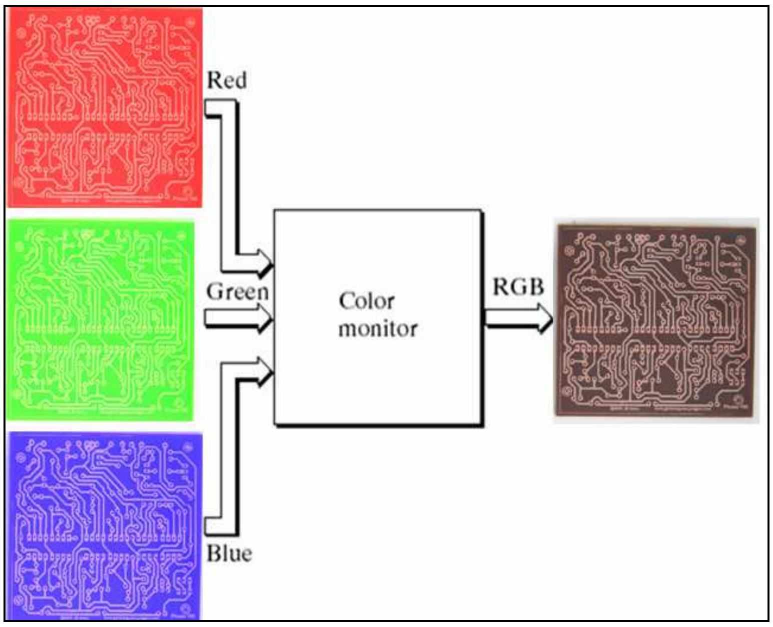


Figure 3.5 – Bare PCB RGB image of cross-sectional plane (127, *G*, *B*)

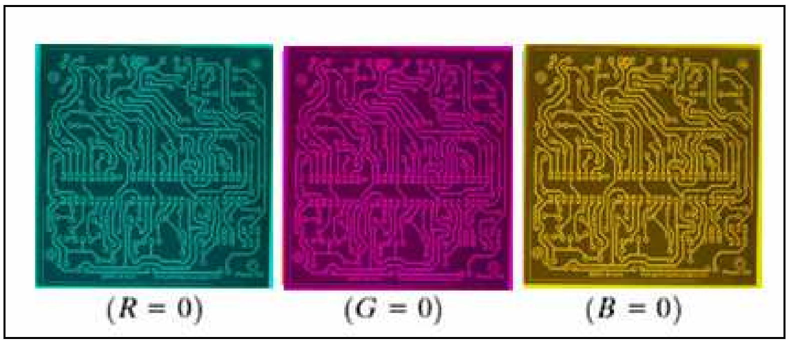


Figure 3.6 – The three hidden surface planes in a bare PCB

3.2 GRAY-SCALE IMAGES

Image formation using sensor and other image acquisition equipment denote the brightness or intensity I of the light of an image as two dimensional continuous function F(x, y) where (x, y) denotes the spatial coordinates when only the brightness of light is considered. Sometimes three-dimensional spatial coordinate are used. Image involving only intensity are called gray scale images (KUMAR AND VERMA, 2010). A representation of a bare PCB production sample in gray scale is illustrated in **Figure 3.7**.

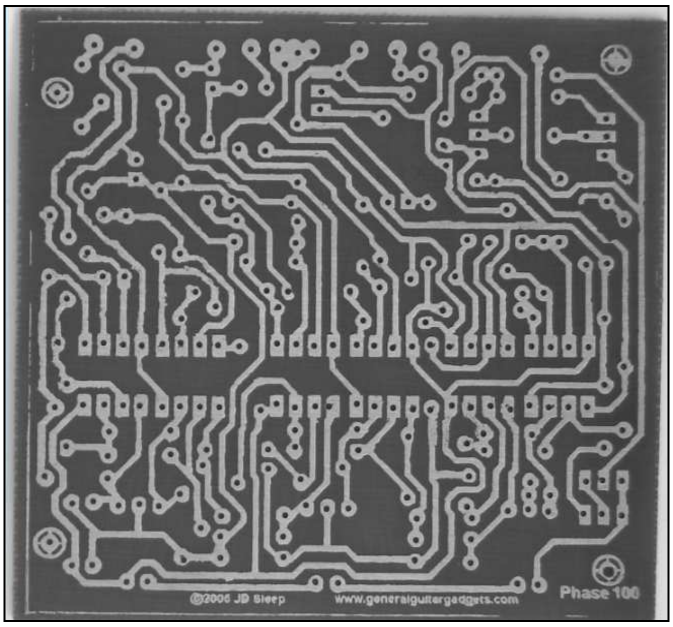


Figure 3.7 –Bare PCB production sample in gray scale

According to (BALA AND BRAUN, 2004) regarding to the image RGB color model to gray-scale conversion, we can describe the following items:

- (1): equal lightness spacing - in this approach, the colors in the image are sorted according to their lightness (L*) values, then equally spaced along the gray axis. The output lightnesses can be distributed either within the range of original lightnesses, or across the full lightness range of the output device. This operation can be described mathematically. Assuming the N image colors are ordered from the darkest (*n* = *1*) to the lightest (*n* = *N*), the lightness of color *n* is given by the **Eq.(3.1)**:

$$L^*_{n,out} = L^*_{\min} + (L^*_{\max} - L^*_{\min})\frac{n-1}{N-1} \tag{3.1}$$

where color *n* = *1* is of lightness $L^* = L^*_{min}$, and color *n* = *N* is of lightness $L^* = L^*_{max}$. The result of this algorithm is shown schematically in Fig. 3 for the same example image containing white, yellow, blue, and black colors. The resulting gray levels G_1, …, G_4 are equally spaced between L*=0 and L*=100 on the gray axis. When using this algorithm in printing applications, the darkest black of the printer is usually higher than L*=0. In these case, we recommend spacing the lightness values along the L* axis from printer black to L*=100;

The **Figure 3.8** is a representation of the presented equal lightness spacing approach.

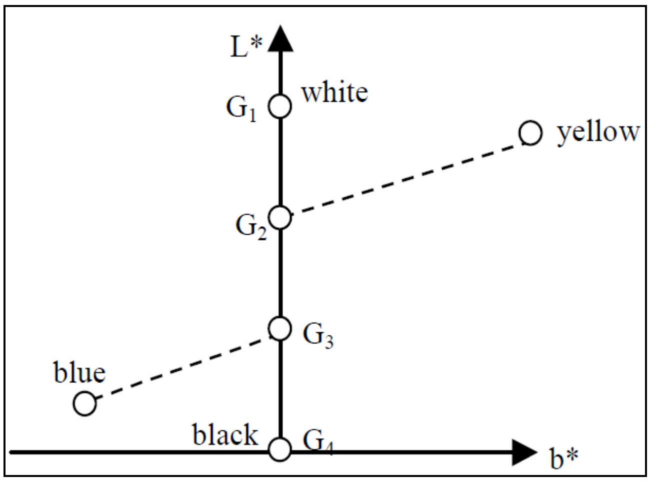


Figure 3.8 –The result of equal lightness spacing technique

- (2): weighted lightness spacing - the idea is to use the total color difference between two colors of adjacent lightness values to weight the resulting lightness spacing of the colors. This is so this if two colors only differ slightly in lightness, chroma, and hue, one does not enhance the lightness difference between these colors. Assuming again the N image colors are sorted from the darkest (*n* = *1*) to the lightest (*n* = *N*), the lightness steps can be weighted by the total color difference between each color and the next lighter or darker color, as shown in **Eq.(3.2)**:

$$L^*_{n,out} = \begin{cases} L^*_{\min} & \text{if} \quad n = 1 \\ L^*_{\min} + (L^*_{\max} - L^*_{\min}) \dfrac{\sum\limits_{i=2}^{n} \Delta E_{i,i-1}}{\sum\limits_{i=2}^{N} \Delta E_{i,i-1}} & \text{if} \quad 2 \le n \le N \end{cases} \tag{3.2}$$

where $\Delta E_{i,i-1}$ is the CIE 1976 ΔE^*_{ab} color difference between patches i and i-1. The numerator in **Eq.(3.2)** shows the summation of all the color differences up to the color of interest and the denominator represents the summation of all the color differences. The result of this algorithm is shown schematically in **Figure 3.9** for the same example colors. Since the ΔE between yellow and blue is substantially greater than the ΔE between other adjacent color pairs, the resulting spacing between these two colors along the gray axis is proportionally increased; likewise for the other pairs.

The **Figure 3.9** is a representation of the presented weighted spacing approach.

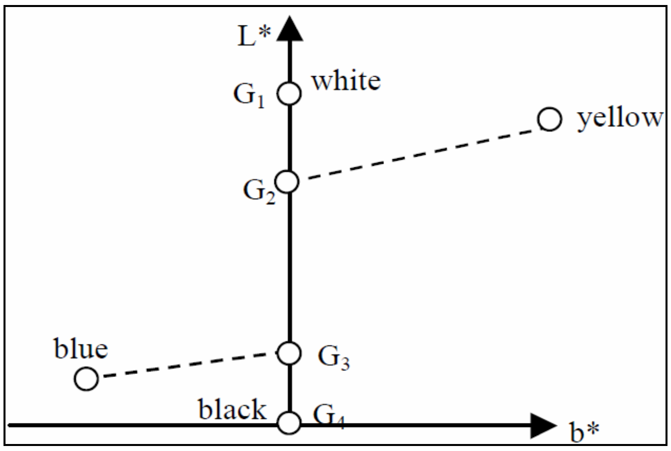


Figure 3.9 –The result of weighted lightness spacing technique

According to (GUR AND ZALEVSKY, 2007), about the grayscale image resolution, similar to one-dimensional time signal, sampling for images is done in the spatial domain, and quantization is done for the brightness values.

In the Sampling process, the domain of images is divided into N rows and M columns. The region of interaction of a row and a Column is known as pixel. The value assigned to each pixel is the average brightness of the regions. The position of each pixel was described by a pair of coordinates (xi, xj) (IRANI AND PELEG, 1991).

Still according to (KUMAR AND VERMA, 2010), the resolution of a digital signal is the number of pixel presented in the number of columns × number of rows. For example, an image with a resolution of 640×480 means this it display 640 pixels on each of the 480 rows. Some other common resolution used is 800×600 and 1024×728, among other.

Resolution is one of most commonly used ways to describe the image quantity of digital camera or other optical equipment. The resolution of a display system or printing equipment is often expressed in number of dots per inch. For example, the resolution of a display system is 72 dots per inch (dpi) or dots per cm.

Having the concepts of RGB color model and gray-scale techniques, is initialized the development of thresholding techniques analyzing the linear equations presented by each mathematic model during the application in an image processing case, such as the PCB image inspection system. Physical justificatives and mathematic formulations able to those thresholding mathematic model are provided in the Chapter 2.

3.3 THRESHOLDING TECHNIQUES

According to (GONZALEZ AND WOODS, 2010), the gray-level image shown in **Figure 3.7** of subtopic 3.2 corresponds to an image, $f(x, y)$, composed of light objects on a dark background, in such a way this object and background pixels have gray levels grouped in two dominant modes. One obvious way to extract the objects from the background is to select a threshold T this separates these modes. Then any point (x, y) for which $f(x, y) > T$ is called *object point*; otherwise, the point is called *background point*. This is the type of threshold introduced in subtopic 2.1 of Chapter 2.

Is initiated the deduction of thresholding equations defining the object of study (bare PCB production sample) as being processed by multiple thresholding techniques , as shows the **Figure 3.10** (SENTHILKUMARAN AND VAITHEGI, 2016).

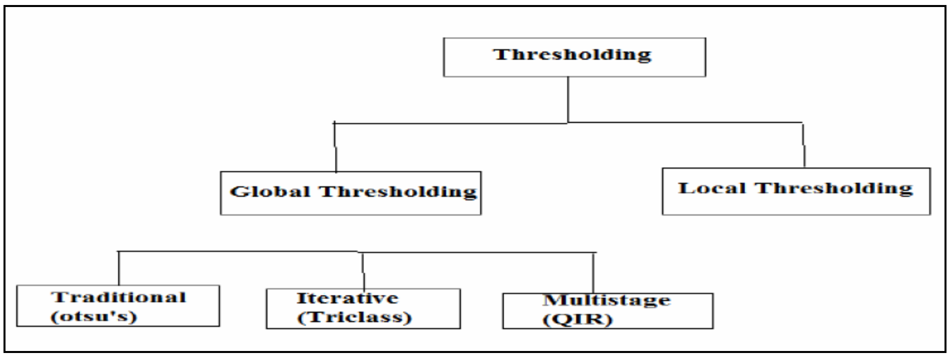


Figure 3.10 –Thresholding techniques

The **Figure 3.11** and **Figure 3.12** shows a slightly more general case of this approach, where three dominant modes characterize the image histogram (for example, two types of light objects on a dark background).

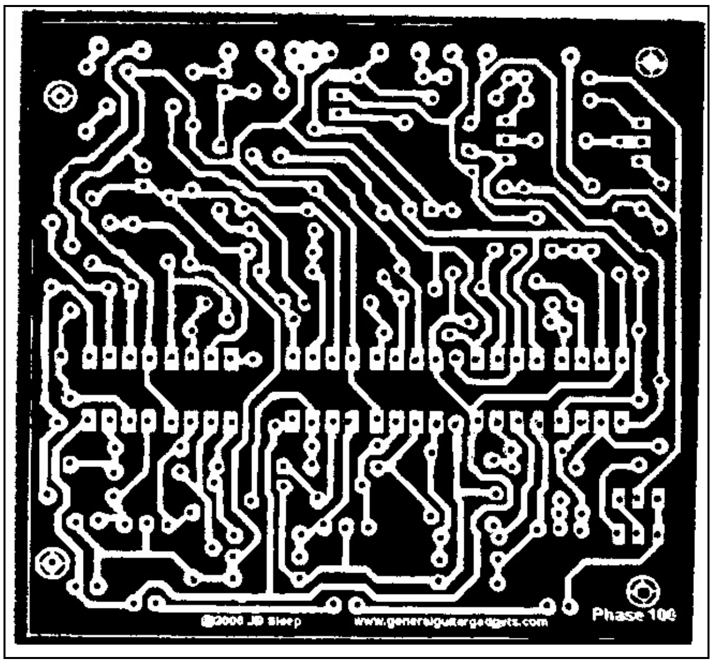


Figure 3.11 –Bare PCB production thresholding at *T =128*

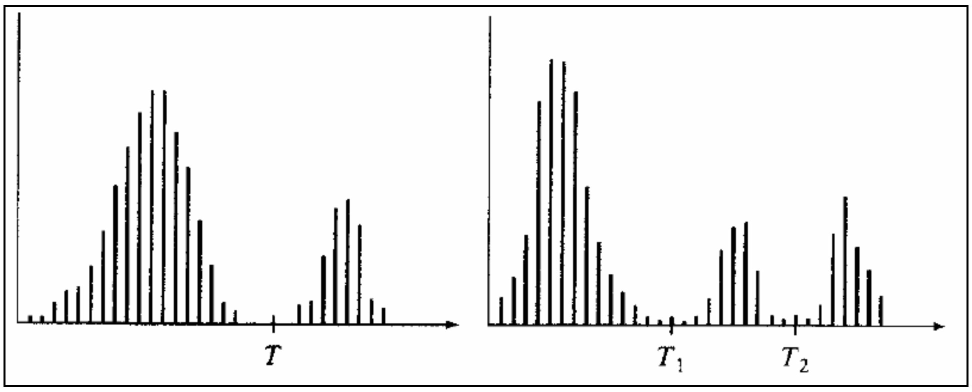


Figure 3.12 – Gray-level histograms: single threshold and multiple thresholds

Here, *multilevel thresholding* classifies a point (x, y) as belonging to one object class if $T_1 < (x, y) \leq T_2$, to other object class if $f(x, y) > T_2$, and to the background if $f(x, y) \leq T_1$. In general, segmentation problems requiring multiple thresholds are best solved using region growing methods, such as those discussed in subtopic 2.1 of Chapter 2.

Based on the preceding discussion in Chapter 2, thresholding may be viewed as an operation this involves tests against a function *T* of the form of **Eq.(3.3)**:

$$T = T[x, y, p(x, y), f(x, y)] \tag{3.3}$$

where *f(x, y)* is the gray level of point (*x, y*) and *p*(*x, y*) denotes some local property of this point – for example, the average gray level of a neighborhood centered on (*x, y*). A thresholded image *g*(*x, y*) is defined as **Eq.(3.4)**:

$$g(x, y) = \begin{cases} 1 & \text{if } f(x, y) > T \\ 0 & \text{if } f(x, y) \leq T. \end{cases} \tag{3.4}$$

Thus, pixels labeled 1 (or any other convenient gray level) correspond to objects, whereas pixels labeled 0 (or any other gray level not assigned objects) correspond to the background.

When *T* depends only on *f(x, y)* (this is, only on gray-level values) the threshold is called *global*. If *T* depends on both *f(x, y)* and *p(x, y)*, the threshold is called *local*. If, in addition, *T* depends on the spatial coordinates *x* and *y*, the threshold is called *dynamic* or *adaptative*.

In the next subtopics the *global* and *local* thresholding techniques will be presented.

3.3.1 GLOBAL THRESHOLDING

Will be defined in this section the various concepts utilized in the thresholding process during the image acquisition in an automated inspection system. Additional aspects will be explained according its necessity. Those concepts have its characteristics given by reference (GONZALEZ AND WOODS, 2010), (SENTHILKUMARAN AND VAITHEGI, 2016), (SANGLE, 2016) and (PRIYA AND NAWAZ, 2017) are presented next:

Global (single) thresholding method is used when there the intensity distribution between the objects of foreground and background are very distinct. When the differences between foreground and background objects are very distinct, a single value of threshold can simply be used to differentiate both objects apart. Thus, in this type of thresholding, the value of threshold T depends solely on the property of the pixel and the gray level value of the image. Some most common used global thresholding methods are *Otsu method*, *entropy based thresholding*, etc. Otsu'salgorithm is a popular global thresholding technique. Moreover, there are many popular thresholding techniques such as *Kittler and Illingworth*, *Kapur*, *Tsai*, *Huang*, *Yen* and another techniques.

With reference to the discussion of this current Chapter, the simplest of all thresholding techniques is to partition the image histogram by using a single global threshold, T, as illustrated in **Figure 3.12**. Segmentation is then accomplished by scanning the image pixel by pixel and labeling each pixel as *object* or *background*, depending on whether the gray level of this pixel is greater or lass than the value of T. As indicated earlier, the success of this method depends entirely on how well the histogram can be partitioned.

The **Figure 3.13** shows a simple image, and **Figure 3.15** shows its histogram. Figure **3.14** shows the result of segmenting of **Figure 3.13** by using threshold T midway between the maximum and minimum gray levels. This threshold achieved a "clean" segmentation by eliminating the shadows and leaving only the objects themselves. The objects of interest in this case are darker than the background, so any pixel with a gray level $\leq T$ was labeled black (0), and any pixel with a gray level $> T$ was labeled white (255). The key objective is merely to generate a binary image, so the black-white relationship could be reversed.

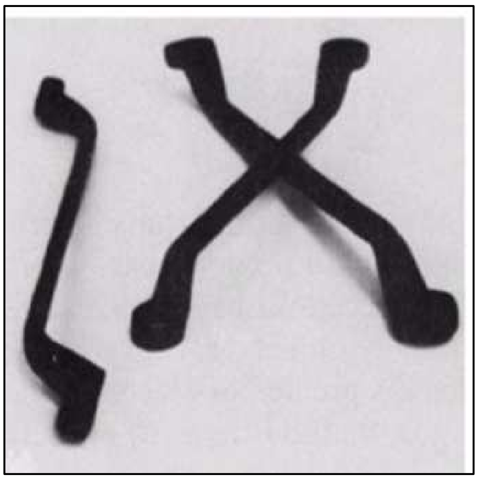


Figure 3.13 – Original image I

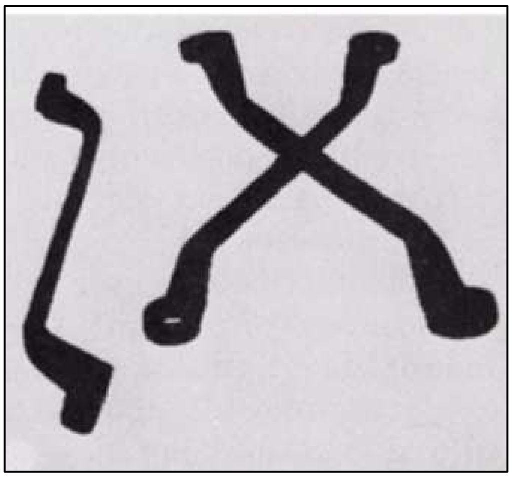


Figure 3.14 – Result of global thresholding with *T* midway

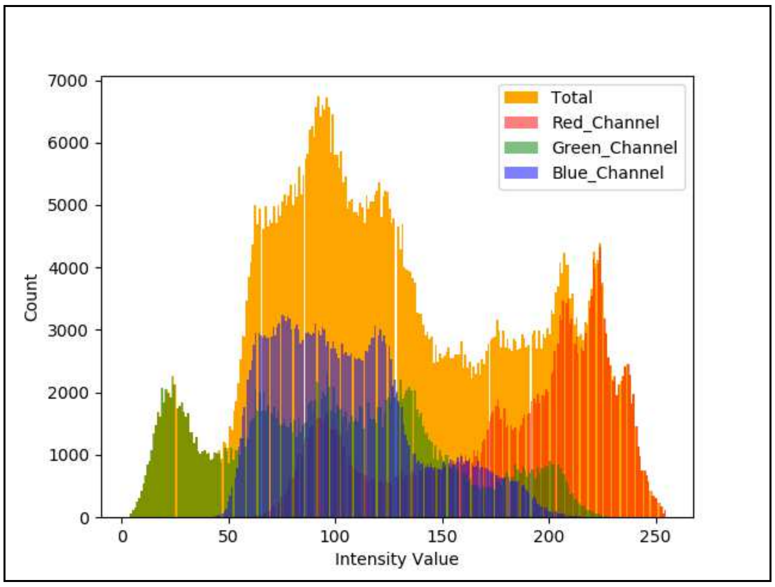


Figure 3.15 – Image histogram I

The type of global thresholding just described can be expected to be successful in highly controlled environments. One of the areas in which this often is possible is in industrial inspection applications, as the case of this current work (bare PCB's production layouts inspection), where control of the illumination usually is feasible.

The threshold in the preceding example was specified by using a heuristic approach, based on visual inspection of the histogram. The following algorithm can be used to obtain T automatically:

1. Select an initial estimate T;
2. Segment the image using T. This will produce two groups of pixels: G_1 consisting of all pixels with gray level values > T and G_2 consisting of pixels with values $\leq T$;
3. Compute the average gray level values μ_1 and μ_2 for the pixels in regions G_1 and G_2;
4. Computer a new threshold value using the **Eq.(3.5)**:

$$T = \frac{1}{2}(\mu_1 + \mu_2). \tag{3.5}$$

5. Repeat the points 2 through 4 until the difference in T in successive iterations is smaller than a predefined parameter T_0.

When there is reason to believe the *background* and *object* occupy comparable areas in the image, a good initial value for T is the average gray level of the image. When objects are small compared to the area occupied by the *background* (or vice versa), then one group of pixels will dominate the histogram and the average gray level is not as good an initial choice. A more appropriate initial value for T in cases such as this is a median value between the maximum and minimum gray levels. The parameter T_0 is used to stop the algorithm after changes become small in terms of this parameter. This is used when speed of interaction is an important issue.

The **Figure 3.16** shows an example of segmentation based on a threshold estimated using the preceding algorithm. **Figure 3.16** is the original image, and **Figure 3.17** is the image histogram. Note the clear valley of the histogram. Application of the iterative algorithm resulted in a value of 125.4 after three iterations starting with the average gray level and $T_0 = 0$. The result obtained using $T = 125$ to segment the original image is shown in **Figure 3.18**. As expected from the clear separation of modes in the histogram, the segmentation between *object* and *background* was very effective.

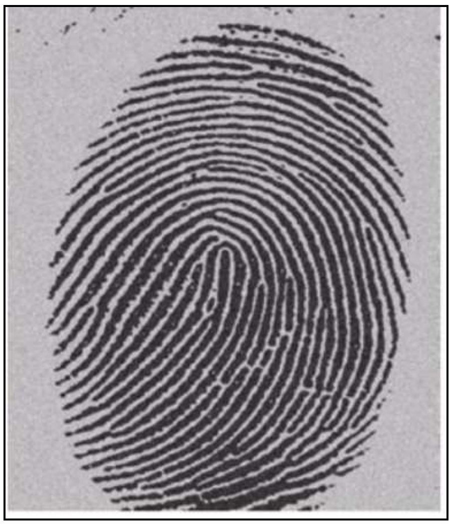


Figure 3.16 – Original image II

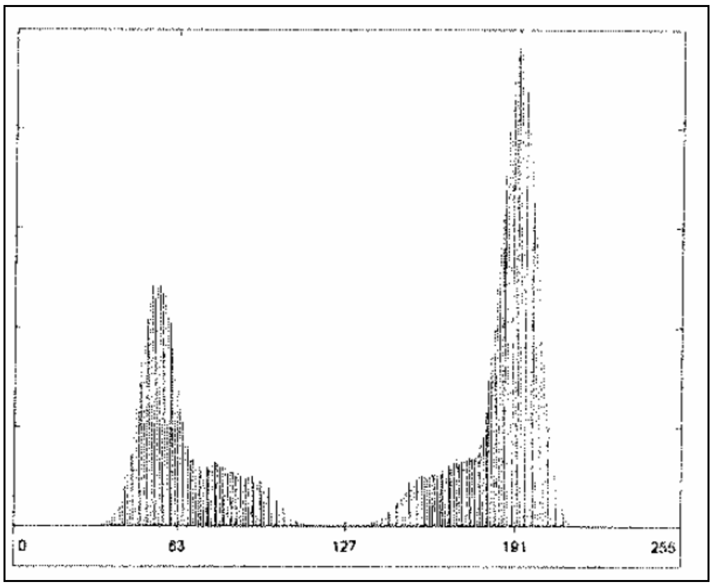


Figure 3.17 – Image histogram II

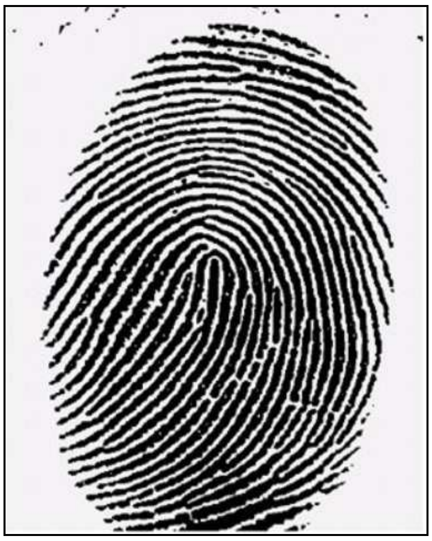


Figure 3.18 – Result of segmentation with the threshold estimated by iteration

Those presented concepts are considered as examples, and will be posteriorly correlated with the other bare PCB's digital layouts production of an inspection system, this will be explained as long as be presented. A schematic representation, based on (SANGLE, 2016), can be verified in the **Figure 3.10**.

Next will be presented the main global thresholding techniques this defines the global thresholding method taking into account the different types of gray scale images. Details of the deductions are presented in the Annex of this dissertation.

3.3.1.1 TRADITIONAL THRESHOLDING (OTSU'S METHOD)

According to (SENTHILKUMARAN AND VAITHEGI, 2016), in image processing, segmentation is often the first step to pre-process images to extract objects of interest for further analysis. Segmentation techniques can be generally categorized into two frameworks, edge-based and region based approaches. As a segmentation technique, Otsu's method is widely used in pattern recognition, document binarization, and computer vision. In many cases Otsu's method is used as a pre-processing technique to segment an image for further processing such as feature analysis and quantification. Otsu's method searches for a threshold this minimizes the intra-class variances of the segmented image and can achieve good results when the histogram of the original image has two distinct peaks, one belongs to the background, and the other belongs to the foreground or the signal. The Otsu's threshold is found by searching across the whole range of the pixel values of the image until the intra-class variances reach their minimum. As it is defined, the threshold determined by Otsu's method is more profoundly determined by the class this has the larger variance, be it the background or the foreground. As such, Otsu's method may create suboptimal results when the histogram of the image has more than two peaks or if one of the classes has a large variance. The **Figure 3.19** shows the Otsu's methods binaries an image to two classes based on *T* by minimizing the within-class variances.

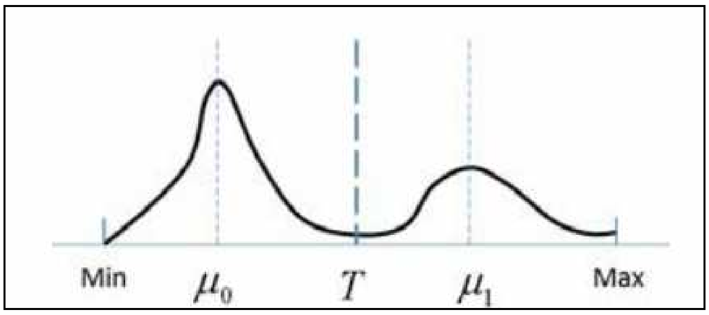


Figure 3.19 – Otsu's method

3.3.1.2 ITERATIVE THRESHOLDING (A NEW ITERATIVE TRICLASS THRESHOLDING TECHNIQUE)

As previously presented and according to (SANGLE, 2016), the *New Iterative Triclass Thresholding* method differs from the Otsu's application in a very important way. It is an iterative method. Firstly, the Iterative method is applied to an image in order to obtain the Otsu's threshold and also the means of the two classes which are in turn separated by the threshold similar to the Otsu's application.

Then, it classifies the image into two classes, which are separated by Otsu's threshold and then separate the image into three classes, which are derived from the respective means of the two classes. Here, the three classes are defined as foreground, which determines the pixel values greater than the larger mean, the background, which determines the pixel values lesser than the smaller mean and the third derived class is called as the to-be-determined region, which determines the values this fall between two class means.

Then finally, a logical union is done, of all the previously determined foreground and background region. This is a parameter free method except the stopping rule for the iterative process and with minimal added computational cost.

As shown in the **Figure 3.20**, the foreground region having pixel values greater than 1 is depicted in yellow color whereas, the background region having pixel values less than 0 is depicted in blue color, and the third region, which is called the TBD region, is depicted in red color. The superscript here, denotes the number of iteration taken by the algorithm.

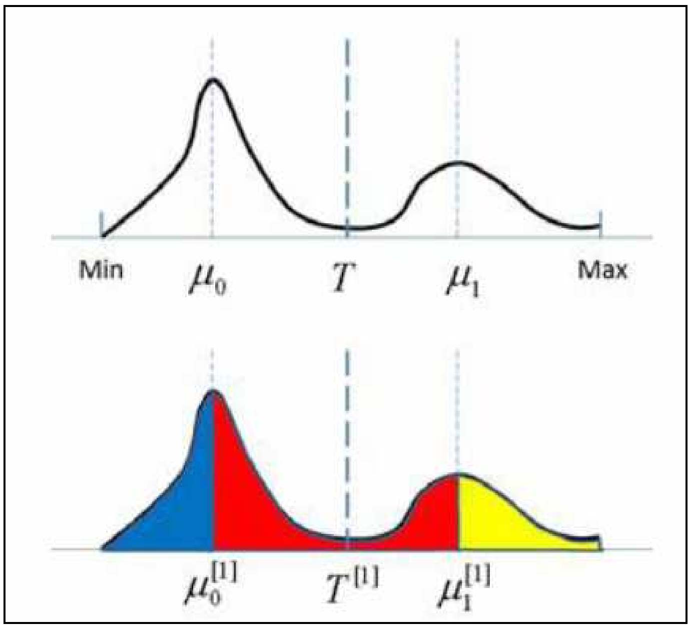


Figure 3.20 – Iterative method

As shows the **Figure 3.21**, we can observe the thresholding result of a bare PCB production sample obtained from iterative method.

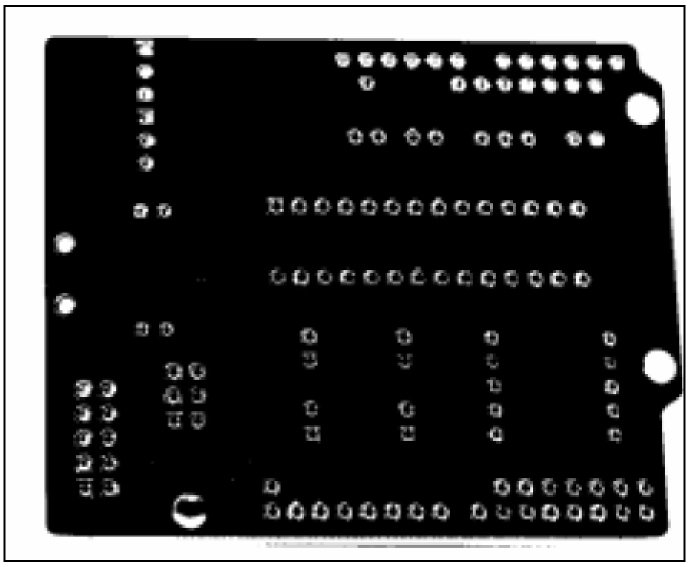


Figure 3.21 – Bare PCB thresholding using iterative method

Next will be presented the ultimate global thresholding technique, before the approach of *Local Thresholding*.

3.3.1.3 MULTISTAGE THRESHOLDING (QUADRATIC RATIO TECHNIQUE FOR HANDWRITTEN CHARACTER)

According to (PRIYA AND NAWAZ, 2017) a new iterative method is based on Otsu's method but differs from the standard application of the method in an important way. At the first iteration, we apply Otsu's method on an image to obtain the Otsu's threshold and the means of two classes separated by the threshold as the standard application does. Then, instead of classifying the image into two classes separated by the Otsu's threshold, our method separates the image into three classes based on the two class means derived. The three classes are defined as the foreground with pixel values are greater than the larger mean, the background with pixel values are less than the smaller mean, and more importantly, a third class we call the "to-be-determined" (TBD) region with pixel values fall between the two class means. Then at the next iteration, the method keeps the previous foreground and background regions unchanged and re-applies Otsu's method on the TBD region (DONGJU AND JIAN, 2009) only to, again, separate it into three classes in the similar manner. When the iteration stops after meeting a preset criterion, the last TBD region is then separated into two classes, foreground and background, instead of three regions. The final foreground is the logical union of all the previously determined foreground regions and the final background is determined similarly. The new method is almost parameter free except for the stopping rule for the iterative process (LEE *et al.*, 1990) and has minimal added computational load.

The QIR technique was found superior in thresholding handwriting images where the following tight requirements need to be met:

6. All the details of the handwriting are to be retained;
7. The papers used may contain strong colored or pattern background;
8. The handwriting may be written by a wide variety of writing media such as a fountain pen, ballpoint pen, or pencil. QIR is a global two stage thresholding technique. The first stage of the algorithm divides an image into three sub images: foreground, background, and a fuzzy sub image where it is hard to determine whether a pixel actually belongs to the foreground or the background (**Figure 3.22**). Two important parameters this separate the sub images are A, which separates the foreground and the fuzzy sub image, and C, which separate the fuzzy and the background sub image. If a pixel's intensity is less than or equal to A, the pixel belongs to the foreground. If a pixel's intensity is greater than or equal to C, the pixel belongs to the background. If a pixel has an intensity value between A and C, it belongs to the fuzzy sub image and more information is needed from the image to decide whether it actually belongs to the foreground or the background.

The first stage of the algorithm divides an image into three sub images: foreground, background, and a fuzzy sub image where it is hard to determine whether a pixel actually belongs to the foreground or the background.

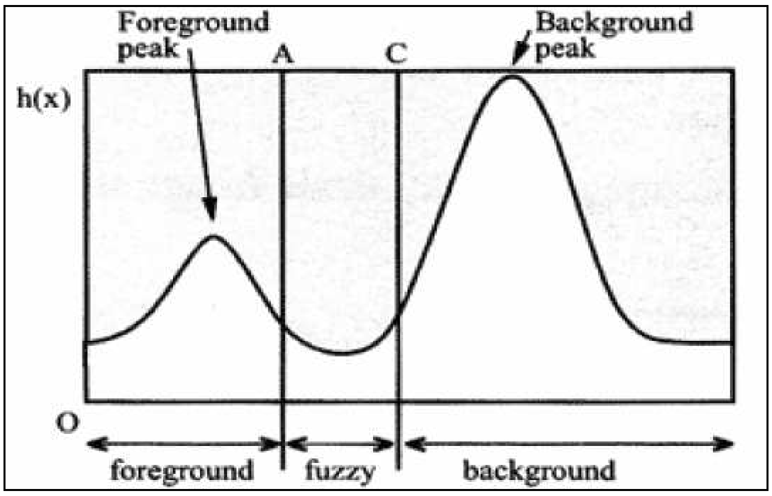


Figure 3.22 – Three sub images of QIR method

Two important parameters this separate the sub images are A, which separates the foreground and the fuzzy sub image, and C, which separate the fuzzy and the background sub image. If a pixel's intensity is less than or equal to A, the pixel belongs to the foreground. If a pixel's intensity is greater than or equal to C, the pixel belongs to the background. If a pixel has an intensity value between A and C, it belongs to the fuzzy sub image and more information is needed from the image to decide whether it actually belongs to the foreground or the background.

The more appropriate model to the study of thresholding process in the image subtraction technique consists in the *Local Thresholding* step – this will be approached in the next subtopic of this Chapter. This model will be utilized in the linear bare PCB's digital layout's during the linear simulations in MATLAB, as well as in the bare PCB production sample model presented in this Chapter.

The concepts presented in the sections of this current subtopic of *Global Thresholding* techniques can be also applied in the bare PCBs digital layout's and in the bare PCB production sample as well. Next, the *Local Thresholding* techniques are presented.

3.3.2 LOCAL THRESHOLDING

Initially, it's observed the **Eq.(3.6)** this corresponds to a local threshold *T* (*x*, *y*) value, this model will be utilized in MATLAB to realize the linear simulations of bare PCB production sample and bare PCB's digital layout's.

$$b\,(x,y) = \begin{cases} 0 & if \quad I(x,y) \leq T(x,y) \\ 1 & otherwise \end{cases} \tag{3.6}$$

Where:

- *b* (*x*,*y*): is the binarized image;
- *I* (*x, y*) ∈ [0,1]: is the intensity of a pixel at location (*x*, *y*) of the image *I*.

According to (SINGH *et al.*, 2011) in local adaptive technique, a threshold is calculated for each pixel, based on some local statistics such as range, variance, or surface-fitting parameters of the neighborhood pixels. It can be approached in different ways such as background subtraction, water flow model, means and standard derivation of pixel values, and local image contrast. Some drawbacks of the local thresholding techniques are region size dependant, individual image characteristics, and time consuming. Therefore, some researchers use a hybrid approach this applies both global and local thresholding methods and some use morphological operators. *Niblack*, and *Sauvola* and *Pietaksinen* use the local variance technique while *Bernsen* uses midrange value within the local block.

3.3.2.1 NIBLACK'S TECHNIQUES

In this method local threshold value T(*x*, *y*) at (*x*, *y*) is calculated within a window of size *w* × *w* was according to the **Eq.(3.7)**:

$$T(x, y) = m(x, y) + k^* \delta(x, y) \tag{3.7}$$

Where:

- m(x, y) and δ (x, y): are the local mean and standard deviation of the pixels inside the local window;
- k: is a bias.

According to (FIRDOUSI AND PARVEEN, 2014) set as k = 0.5.the local mean m(x, y) and standard deviation δ (x, y) adapt the value of the threshold according to the contrast in the local neighborhood of the pixel. The bias k controls the level of adaptation varying the threshold value.

3.3.2.2 SAUVOLA'S TECHNIQUE

In Sauvola's technique, the threshold T(x, y) is computed using the mean m(x, y) and standard deviation δ (x, y) of the pixel intensities in a $w \times w$ window centered around the pixel at (x, y) and express as the **Eq.(3.8)**:

$$T_{Sauvola} = m * \left(1 - k * \left(1 - \frac{s}{R}\right)\right) \tag{3.8}$$

Where:

- R: is the maximum value of the standard (R = 128 for a grayscale document);
- k: is a parameter which takes positive values in the range [0.5] [12].

3.3.2.3 BERNSEN'S TECHNIQUE

This technique, proposed by *Bernsen*, is a local binarization technique, which uses local contrast value to determine local threshold value. The local threshold value for each pixel (x, y) is calculated by the relation indicated in the **Eq.(3.9)**.

$$T(x,y) = \frac{Imax + Imin}{2} \quad \textbf{(3.9)}$$

Where:

- *Imax* and *Imin*: are the maximum and minimum gray level value in a $w \times w$ window centrated at (x, y) respectively (N.KAUR AND R.KAUR, 2011).

But the threshold assignment is based on local contrast value and hence it can be expressed as:

- if *Imax* – *Imin* >L // if the gray scale image is not uniform;
  Then
  T(x, y) = (*Imax* + *Imin*)/2;
  Else
  T(x, y) = GT // else threshold value is calculated by *global threshold* technique.

Where:

- L: is a contrast threshold;
- GT: is a global threshold value.

3.3.2.4 YANOWITZ AND BRUCKSTEIN'S TECHNIQUE

Yanowitz and *Bruckstein* suggested using the grey-level values at high gradient regions as known data to interpolate the threshold surface of image document texture features (LEEDHAM *et al.*, 2010) .The key steps of this method are:

- Smooth the image by average filtering;
- Derive the gradient magnitude;
- Apply a thinning algorithm to find the object boundary points;
- Sample the grey-level in the smoothed image at the boundary points. These are the support points for interpolation in next step;
- Find the threshold surface T(x, y) this is equal to the image values at the support points and satisfies the Laplace equation using South well's successive over relaxation method;
- Using the obtained T(x, y), segment the image;
- Apply a post-processing method to validate the segmented image.

Therewith, were presented the four more distinct models to the realization of *Local Thresholding* technique. As mentioned before, this local technique was employed in this work to thresholding step of image subtraction process, in which the third thresholding technique presented in this subtopic (*Bernsen's Technique*) was used in the bare PCB's images. We can then move forward to the application of image binarization conversion technique, utilized in the bare PCB's images during the image subtraction process.

3.4 IMAGE BINARIZATION

To simplifying purposes, will be presented here the general concepts of bare PCB's binarization process with an example of a bare production PCB sample binarized from the correspondent thresholded image. Then, taking the bare PCB product sample obtained in the *Local Thresholding* process, using the *Bernsen's Technique*, we have the binarizated PCB image in the **Figure 3.23**:

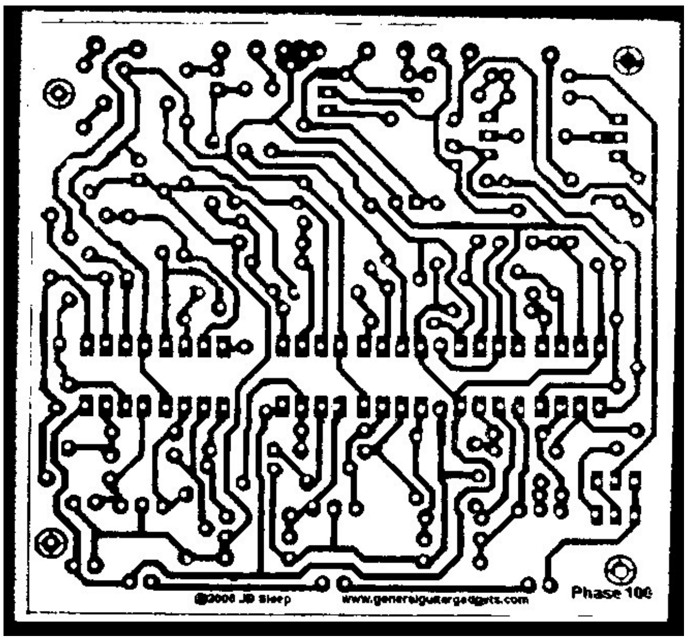


Figure 3.23 – Bare PCB production binarization

Then, while the *Local Thresholding* is associated simply in to convert the bare PCB gray-level image in a black and white image within $T = 128$, the image binarization turn the black pixels, defined as a value equal “1”, of the bare PCB production sample thresholded image into white pixels defined as a value equal “0”. And the the binarization process also turn the white pixels, defined as a value equal “0”, of the bare PCB thresholded image into black pixels defined as a value equal “1”.

Due to this image pixels inversion, exists then the reviewing of bare PCB trials in black and the rest of board in white (without trials). In the case of this work, the possibility of XOR operation application to the bare PCB production sample appears due to this binarization process, in which the PCB trials are highlighted in black.

3.5 THE XOR OPERATION IN THE IMAGE ACQUISITION PROCESS

Is initialized now the modeling step of XOR logic operation in the image subtraction process, considering the previous techniques demonstrated in the presented subtopics of this Chapter, such as: *RGB color model*, *Gray-scale image level*, *Thresholding techniques* and the

Image binarization method. Therefore, based on reference (GONZALEZ AND WOODS, 2010) and (JAYARAMAN *et al*., 2009), the main concepts of XOR operation in the image subtraction process is presented next.

The difference between two images $f(x, y)$ and $h(x, y)$, can be expressed as the **Eq.(3.10)**:

$$g(x, y) = f(x, y) - h(x, y). \tag{3.10}$$

A new image is obtained by computing the difference between all pairs of corresponding pixels from *f* and *h*. Image subtraction using XOR operation is mostly used for change detection C=A-B: maximum value of A-B and zero.

In the XOR operation, if the pixels of Image *A* and Image *B* are complementary to each other then the resultant image pixel is black, otherwise the resultant image pixel is white. The XOR operation of images *A* and *B* is showed in the **Figure 3.24**.

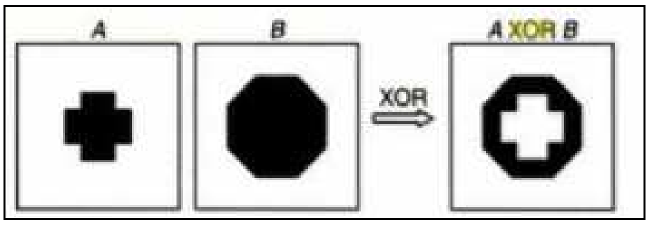


Figure 3.24 – Image XOR operation

The XOR function is only true if just one (and only one) of the input values is true, and false otherwise, XOR stands for *Exclusive OR*. As can be seen in the **TABLE 3.1**, the output values of XNOR are simply the inverse of the corresponding output values of XOR.

TABLE 3.1 – TRUTH-TABLES FOR XOR AND XNOR

XOR			XNOR		
A	B	**Q**	A	B	**Q**
0	0	0	0	0	1
0	1	1	0	1	0
1	0	1	1	0	0
1	1	0	1	1	1

The XOR (and similarly the XNOR) operator typically takes two binary or gray-level images as input, and outputs a third image whose pixel values are just those of the first image, where the XOR operation was applied with the corresponding pixels from the second. A variation of this operator takes a single input image and XORs each pixel with a specified constant value in order to produce the output.

Once with XOR a XNOR concepts explained for image processing, will be possible utilize this logic operation in a PCB production sample to exemplify the final process of image subtraction technique, presented to reach a defective PCB productions sample.

3.5.1 THE DEFECTIVE BARE PCB PRODUCTION SAMPLE

According to (IBRAHIM AND AL-ATTAS, 2004) a variety of defects can affect the copper pattern of PCB. Some of them are identified as functional defects, whereas the others are visual defects. Functional defects seriously cause damage to the PCB, meaning the PCB does not function as needed. Visual defects do not affect the functionality of the PCB in short term. But in long period, the PCB will not perform well since the improper shape of the PCB circuit pattern could contribute to potential defects.

Thus, it is crucial to detect these two types of defects in the inspection phase. **Figure 3.25** shows an image pattern of a binarized bare PCB production sample without defects (*reference image*). **Figure 3.26** shows the same image pattern as in **Figure 3.25** with a variety of defects on it. The printing defects and anomalies this will be looked at, for example, are *breakout*, *short*, *pin hole*, *wrong size hole*, *open circuit*, *conductor too close*, *underetch*, *spurious copper*, *mousebite*, *excessive short*, *missing conductor*, *missing hole*, *spur* and *overetch*. These defects are shown in **TABLE 3.2**.

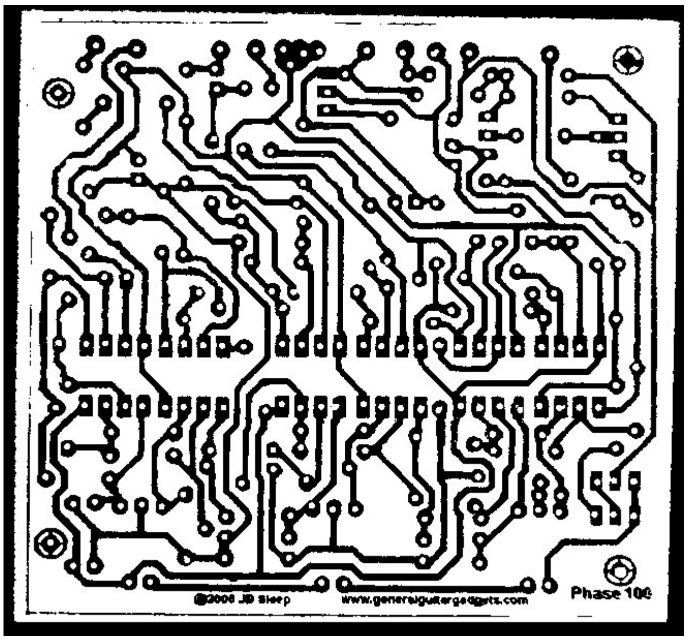


Figure 3.25 – Reference bare PCB production sample

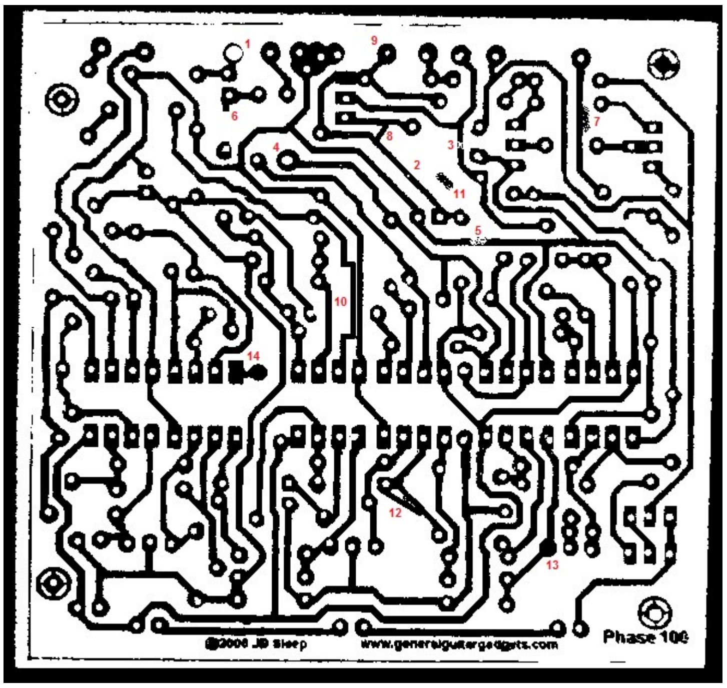


Figure 3.26 – Defective bare PCB production sample

TABLE 3.2 – BARE PCB PRODUCTION SAMPLE DEFECTS

ITEM	**DEFECT**
1	Breakout
2	Pin Hole
3	Open Circuit
4	Underetch
5	Mousebite
6	Missing Conductor
7	Spur
8	Short
9	Wrong Size Hole
10	Conductor Too Close
11	Spurious Copper
12	Excessive Short
13	Missing Hole
14	Overetch

In the next sections, will be developed the items related to the final image subtraction processes (this is, the working principle of XOR operation in the image subtraction technique applied to the PCB inspection systems) and other items also relevant to the formulation of XOR operation in a PCB inspection process.

3.5.2 THE XOR OPERATION IN A PCB INSPECTION MACHINE

According to (YIN, 2014) AOI, automated optical inspection is an automated vision inspection of PCB during the manufacturing process. It is used to scan the inner layers and outer layers of PCB after the processes of etching and stripping. After scanning by AOI machine, the defects which do not meet the manufacturer's requirement will be identified by the machine. In this way, AOI can detect problems early in the production process, so faults would not be passed to next production process and production cost could be saved. Therefore AOI can increase the quality of PCBs and reduce the production cost. The **Figure 3.27** shows the block diagram of AOI machines in manufacturing process and **Figure 3.28** shows the AOI machine.

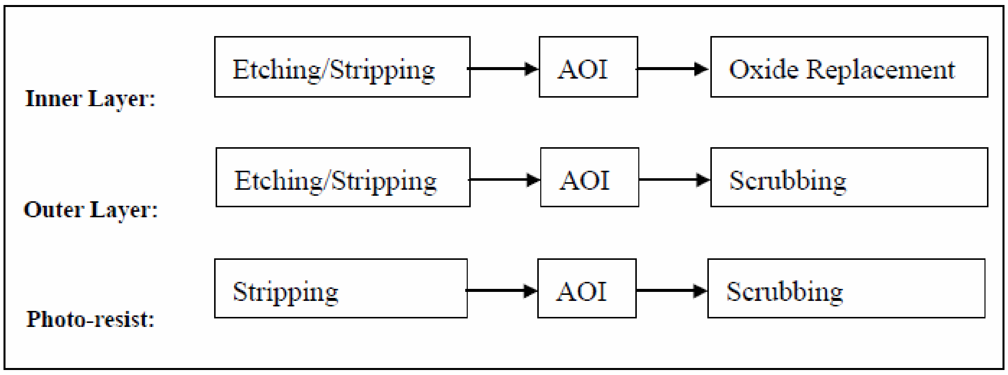


Figure 3.27 – Block diagram of (AOI) machines

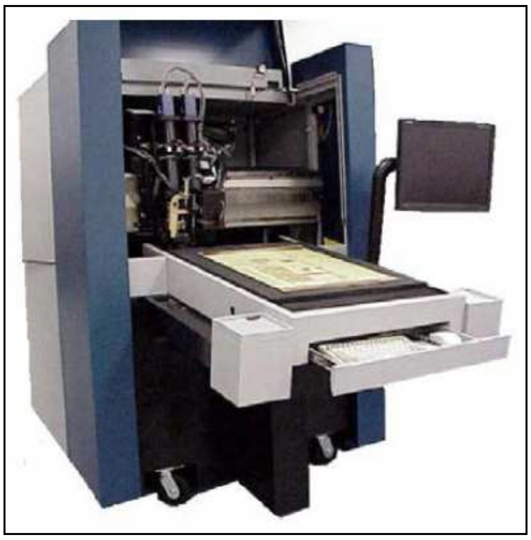


Figure 3.28 – The (AOI) machine

The software is an important part of all (AOI) and (AVI) machine. It provides a graphical interface for the users to operate and perform internal calculation to identify the defects. The process method of the image XOR operation, used in (AOI) machines, affects the process time as some machines just identify the defects by comparing pixel by pixels. In this way, the processing time would be faster. The **Figure 3.29** shows the defects detected pixel by pixel in an (AOI) machine.

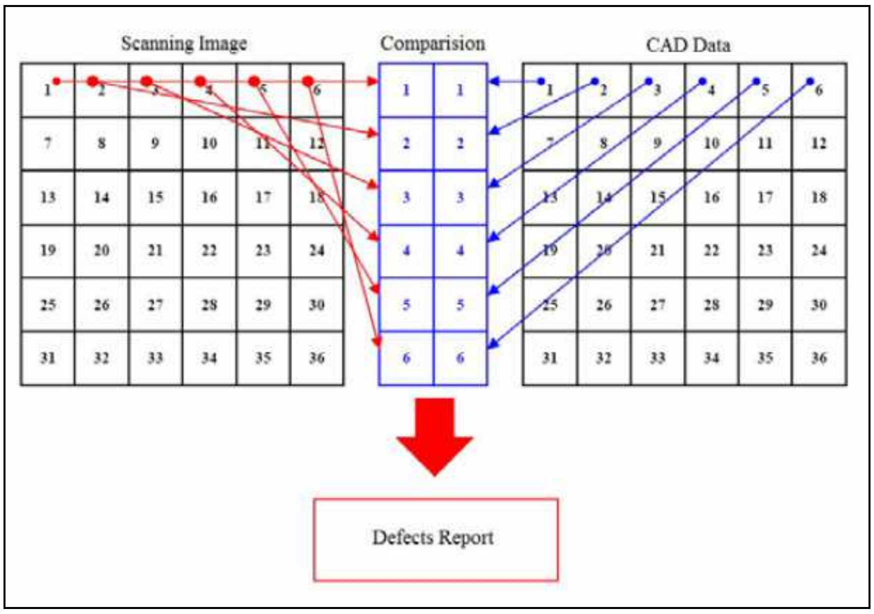


Figure 3.29 – Detect defects by pixel by pixel in an (AOI) machine

In an AOI machine, the defects are found using the mainly a (CAD) data of *Reference Image* during the image subtraction process. For all kind of (AOI) and (AVI) machines, there must be a reference image in order to find out the defects on the boards. It is used to compare with the scanned image to identify the defects.

This reference image can be either a CAD data or a golden board image. A CAD data is a *Computer-aided design* data of PCBs which includes much information such as the circuit layout, solder mask layout, etc. These CAD data are created by CAD software. All regions are classified for a CAD data. The **Figure 3.30** shows an example of CAD data pattern.

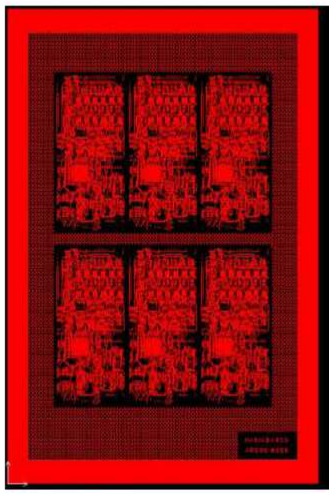


Figure 3.30 – CAD data pattern

Another kind of reference image is the golden board image. A golden board image is an image of a pattern board. To ensure the board is a good board, it is normally checked manually by some magnifying devices first. After this, it will be scanned by AOI/AVI machines and the image captured will be the golden board image. Sometimes, the golden board image is created by scanning a number of boards and using the average image of these boards as the golden board image. The **Figure 3.31** shows the golden board image.

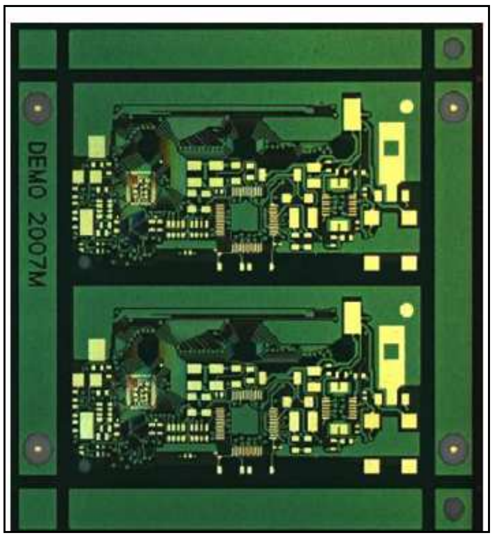


Figure 3.31 – Golden board image

If the reference image is CAD data, the CAD data is classified while the scanned image is not classified. Therefore, the scanned image needed to be classified before comparing with the CAD data. The **Figure 3.32** shows the process diagram block.

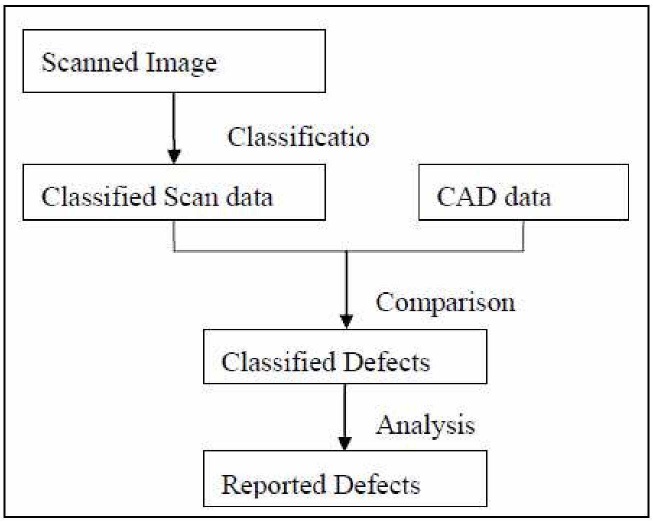


Figure 3.32 – Process diagram block of CAD reference image reporting defect

If the reference image is golden board image, both scanned image and golden board image are not classified. Therefore, the scanned image can be compare directly with the golden board image to get the image difference. The golden board image is classified for analyzing with the image difference as shows the **Figure 3.33**.

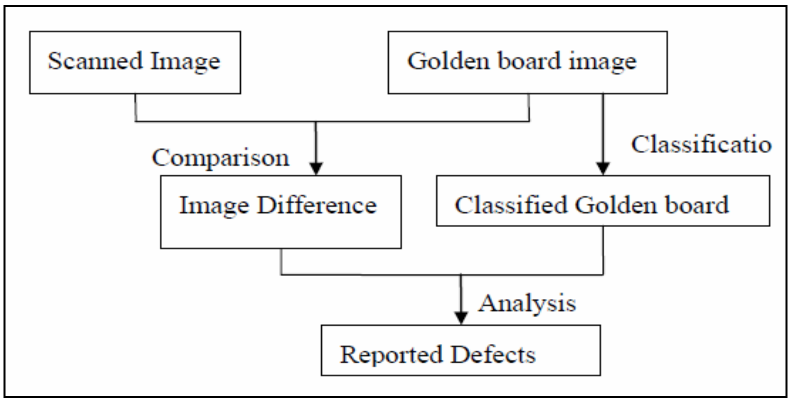


Figure 3.33 – Process diagram block of golden board reference image reporting defect

Then, once defined all the desired aspects of XOR operation used in the AOI machines image difference process, we can aggregate the remaining properties of XOR operation to the previous concepts viewed in the subtopics of this section, and reach a numeric representation about this concepts.

3.5.3 THE MAIN MATHEMATICAL PROPERTIES OF XOR OPERATION USED IN THE IMAGE SUBTRACTION TECHNIQUE PROCESS

According to (LEWIN, 2012) XOR is one of the sixteen possible binary operations on boolean operands. This means this it takes 2 inputs (it's binary) and produces one output (it's an operation), and the inputs and outputs may only take the values of *true* or *false* (it's boolean). In numerical applications, XOR is typically represented by the $\bigoplus$ symbol.

Certain boolean operations are analogous to set operations as shows the **Figure 3.34**: *AND* is analogous to intersection, *OR* is analogous to union, and *XOR* is analogous to set difference.

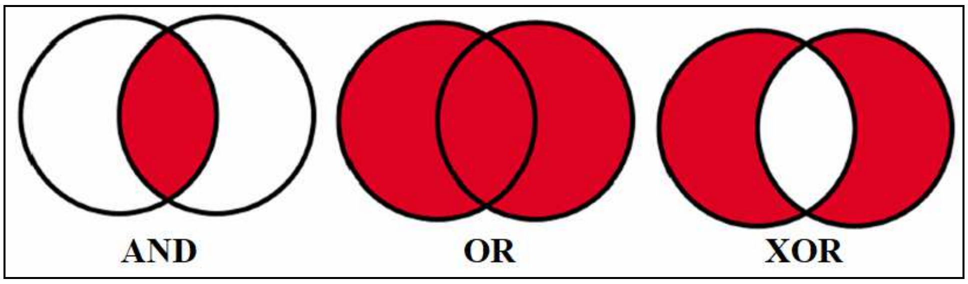


Figure 3.34 – Boolean operations

There are 4 very important properties of XOR this we will be making use of. These are formal mathematical terms but actually the concepts are very simple.

- Commutative as the **Eq.(3.11)**:

$$A \oplus B = B \oplus A \qquad \textbf{(3.11)}$$

This is clear from the definition of XOR: it doesn't matter which way round you order the two inputs.

- Associative as the **Eq.(3.12)**:

$$A \oplus (B \oplus C) = (A \oplus B) \oplus C \qquad \textbf{(3.12)}$$

This means this XOR operations can be chained together and the order doesn't matter.

- Identity element as the **Eq.(3.13)**:

$$A \oplus 0 = A \qquad \textbf{(3.13)}$$

This means this any value XOR'd with zero is left unchanged.

- Self-inverse as the **Eq.(3.14)**:

$$A \oplus A = 0 \qquad \textbf{(3.14)}$$

This means this any value XOR'd with itself gives zero.

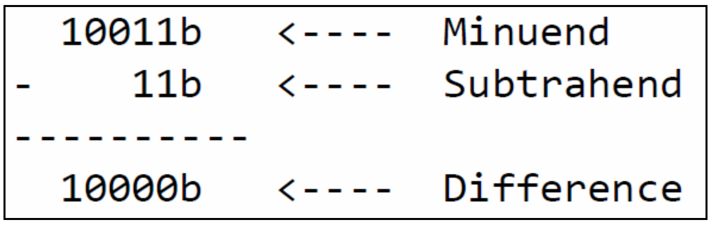
At this point it’s important observe the XOR operation process in an image acquisition model has its mathematical characteristics highly dependent from variables *A* and *B*, being classified in the AOI image processing system as *PCB default image* and *PCB reference image*. In this case, the most used mathematical operation in the PCB image subtraction technique is the *commutative property*.

3.5.4 THE BINARY SUBTRACTION METHOD USED IN XOR OPERATION

We can subtract one number from another in binary to obtain a binary difference. Numbers are subtracted in the same way as in decimal, though borrowing requires some explanation.

3.5.4.1 SUBTRACTION ORDER

Unlike binary addition, order matters in subtraction. The number we are subtracting from (the top number) is called the *minuend*, and the number subtracting (the bottom number) is called the *subtrahend*. The result of subtraction is called the *difference*, as shows the **Figure 3.35**.



```
  10011b   <----  Minuend
-    11b   <----  Subtrahend
----------
  10000b   <----  Difference
```

Figure 3.35 – Binary subtraction

3.5.4.2 BINARY SUBTRACTION RULES

There are four subtraction combinations possible. Learning these combinations makes binary subtraction a simple process as shows the **Figure 3.36**.

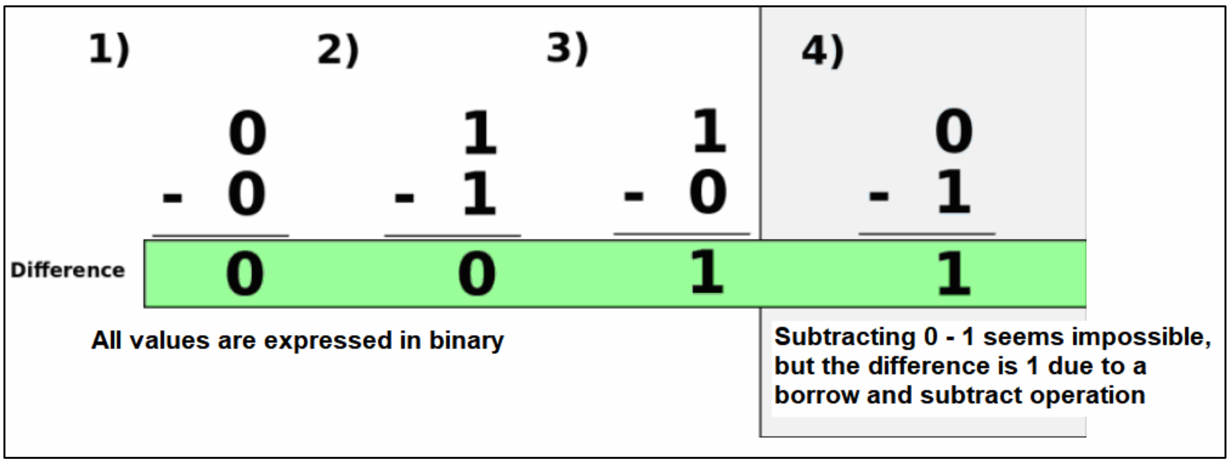
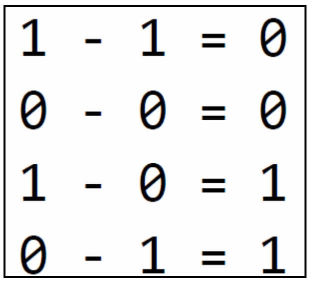

1)
2)
3)
4)
0 - 0
1 - 1
1 - 0
0 - 1
Difference
0
0
1
1
All values are expressed in binary
Subtracting 0 - 1 seems impossible, but the difference is 1 due to a borrow and subtract operation

Figure 3.36 – Four binary subtraction combinations

The binary subtraction rules are indicated in the **Figure 3.37**.



```
1 - 1 = 0
0 - 0 = 0
1 - 0 = 1
0 - 1 = 1
```

Figure 3.37 – Binary subtraction rules

The four possible combinations are the same as exclusive OR (XOR). When we subtract in binary, we are really applying the XOR operation on two bits in the same column to obtain the result for this column.

3.5.5 THE COMPUTATIONAL FUNCTION OF XOR OPERATION

According to (NWABUEZE, 2005) the binary XOR operation is a function in the classical sense, with the following four function calls as shows the **Figure 3.38**:

$$\boxed{\begin{aligned} XOR(True, True) &= False, \\ XOR(True, False) &= True, \\ XOR(False, True) &= True, \\ XOR(False, False) &= False. \end{aligned}}$$

Figure 3.38 – Function calls of XOR operation

This implies this *XOR(x, y)* does not have an inverse, since it is not a one- to-one function. To see this is not one-to-one, observe this there are two instances where *XOR(x, y) = true* and the specific values of *x* and *y* which produced this result of *true* cannot be predicted. However, one can predict the *x* and *y* values had to be different.

Note also this there are two instances where *XOR(x, y) = false*, and the specific values of *x* and *y* which produced this result of *false* cannot be predicted. However, we can conclude this *x* and *y* have to be equal. Although *XOR(x, y)* is not one-to-one, one can restrict the domain by specifying a subset of the function which is one-to-one.

For an inverse to exist, one can, for example, restrict the domain of *XOR(x, y)* to be *x = true*. Other examples of a restriction would be this *x = false*, *y = true*, or *y = false*. Similar to the *OR(x, y)* and *XOR(x, y)* functions, one can derive equivalent conclusion for the *NOR(x, y)*, *NAND(x, y)*, and *AND(x, y)* operations.

In the next subtopics, will be utilized algorithmic analysis to the explanation of XOR operation in the image subtraction process of an image acquisition system through an AOI machine. Then, though all the XOR operation boolean concepts are available in the theory, its practical analysis will be applied in this Chapter, to exemplify a real study case of XOR algorithms.

3.5.6 MATHEMATICAL DEFINITION ABOUT THE IMAGE SUBTRACTION ALGORITHM USING XOR OPERATION

Although be desirable this an image acquisition system can have all the informations about the input variables of a PCB board inspection, in practical cases of industrial applications those input's variables of a PCB to be inspect not always are showed at AOI machine's interfaces. However, it's necessary to know the subtraction boolean values of *reference* and *testing* PCB images in the AOI embedded inspection algorithm. This distinction must be done with basis in the PCB CAD data to be inspected in the AOI machine (see Section 3.5.2).

Mathematically, the image difference detection between *reference* and *testing* PCB's images bit values can be done simply applying a quantization range for ($x_1 \div x_2$) previously selected by the image subtraction algorithm.

In practice, divide two PCB image bit values is equivalent to select only the ranges of interest to the application of XOR operation, discarding others.

According to (L.F PAU, 1989) once two images are registered showing the same scene, two cases may occur: they are either identical or different, in terms of a comparison value. To detect such a difference, if any, one may define the difference image ($x_1 \div x_2$) between two registered images x_1 and x_2 of the same size as indicates the **Eq.(3.15)**:

$$(x_1 \div x_2)(i, j) = x_1(i, j) - x_2(i, j) \tag{3.15}$$

Where the right-rand side is the algebraic gray-level difference. It should be noted the quantization range for ($x_1 \div x_2$) covers ($-D$, $+D$) if D = Max $|di - dj|$ is the largest quantization level difference.

If, however, x_1 and x_2 are both binary images with quantization levels d_0 and d_1, defined by the same thresholds, then a related definition is the exclusive OR of x_1 and x_2 as indicates the **Eq.(3.16)**:

$$XOR\,(x_1, x_2)(i,j) \;=\; \{d[x_1(i,j)]\} \circ XOR \circ \{d[x_2(i,j)]\} \tag{3.16}$$

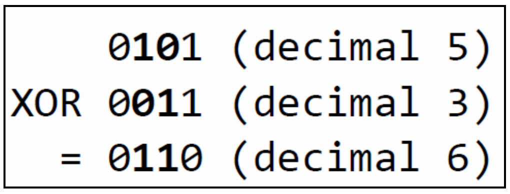
Where the right-rand side is the logical XOR value of the logical quantization levels. One possible dissimilarity measure D is the number of pixels as indicated in the **Eq.(3.17)**:

$$(x_1 \div x_2)(i,j) \;\neq 0 \; or \; XOR\,(x_1, x_2)(i,j) \;=\; d_1 \tag{3.17}$$

Resuming, there are two cases of quantization levels in the mathematical definition of image subtraction algorithm: for the *gray-scale images* quantizations and *binary images* quantizations. The application of each case will depend of the study proposer in a determined point.

3.5.7 THE BITWISE XOR OPERATION USED IN THE IMAGE SUBTRACTION TECHNIQUE PROCESS

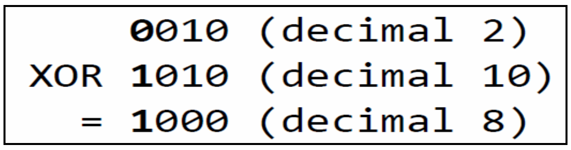
A bitwise XOR takes two bit patterns of equal length and performs the logical exclusive OR operation on each pair of corresponding bits. The result in each position is 1 if only the first bit is 1 or only the second bit is 1, but will be 0 if both are 0 or both are 1. In this we perform the comparison of two bits, being 1 if the two bits are different, and 0 if they are the same. The **Figure 3.39** exemplifies this concept:



```
    0101 (decimal 5)
XOR 0011 (decimal 3)
  = 0110 (decimal 6)
```

Figure 3.39 – Two-bits comparison

The bitwise XOR may be used to invert selected bits in a register (also called *toggle* or *flip*). Any bit may be toggled by XORing it with 1. For example, given the bit pattern 0010 (decimal 2) the second and fourth bits may be toggled by a bitwise XOR with a bit pattern containing 1 in the second and fourth positions as shows the **Figure 3.40**:



```
    0010 (decimal 2)
XOR 1010 (decimal 10)
  = 1000 (decimal 8)
```

Figure 3.40 – Bitwise XOR operation

This technique may be used to manipulate bit patterns representing sets of boolean states.

3.5.7.1 MATHEMATICAL EQUIVALENTS

Assuming $x \geq y$, for the non-negative integers, the bitwise operations can be written as follows the **Eq.(3.18)**:

$$x \,\mathbf{XOR}\, y = \sum_{n=0}^{\lfloor \log_2(x) \rfloor} 2^n \left[\left[\left(\left\lfloor \frac{x}{2^n} \right\rfloor \bmod 2\right) + \left(\left\lfloor \frac{y}{2^n} \right\rfloor \bmod 2\right)\right] \bmod 2\right] = \sum_{n=0}^{\lfloor \log_2(x) \rfloor} 2^n \left[\left(\left\lfloor \frac{x}{2^n} \right\rfloor + \left\lfloor \frac{y}{2^n} \right\rfloor\right) \bmod 2\right] \tag{3.18}$$

3.5.7.2 THE BITWISE XOR OPERATION BETWEEN ELEMENT-WISE INTEGERS OF TWO ARRAYS

- Syntax: w = bitxor(u, v)
- Parameters: u, v, w

If u and v have the same type and intype, this one is the working one. Otherwise:

- if *u* or *v* is decimal-encoded, the working inttype is 0 (real decimal), even if the other operand is int64- or uint64-encoded.

- if u and v are both encoded integers, the working inttype is the widest of both: int8 < uint8 < int16 < uint16 < int32 < uint32 < int64 < uint64.

The result *w* gets the type of the working encoding. And *u* and *v* are processed element-wise:

- if *u* is a single value (scalar) and *v* is a vector, matrix or hypermatrix, *u* is priorly expanded as *u** ones(v) in order to operate u with every *v* component.

- conversely, *v* is priorly expanded as *v** ones(*u*) if it is a single value.

- if neither *u* nor *v* are scalars, they must have the same sizes.

The result *w* gets the sizes of *u* or/and *v* arrays.

3.5.7.3 BITWISE XOR OPERATION ALGORITHM EXAMPLE USING MATLAB

```
bitxor(25, 33)
dec2bin([25 33 56]') // binary representations

--> bitxor(25, 33)
ans =
56.

--> dec2bin([25 33 56]'))
ans =
!011001 !
!100001 !
!111000 !

// Between 2 simple rows with zeros and ones
u = [0 1 0 1];
v = [0 0 1 1];
```

```
bitxor(u, v) // [0 1 1 0] expected

// Encoded integers such as int8 are accepted:
u = int8([0 1 0 1]);
v = int8([0 0 1 1]);
bitxor(u, v)

// Operands of mixed types are accepted.
// The type of the result is decimal if a decimal operand is involved,
// or the widest integer one otherwise:
u = [0 1 0 1];
v = [0 0 1 1];
z = bitxor(u, int64(v)); type(z) // 1 : decimal representation
z = bitxor(uint8(u), int8(v)); typeof(z) // uint8
z = bitxor(uint8(u), int32(v)); typeof(z) // int32


// Usage with 2 matrices
u = [ 1 2 4 8
25 33 25 33 ];
v = [ 2 2+4 4+8 16
33 25 56 56 ];
bitxor(u, v) // [ 3 4 8 24 ; 56 56 33 25 ] expected


// Usage with a distributed scalar:
bitxor([1 2 4 8 9 10 12], 8) // == bitxor([1 2 4 8 9 10 12], [8 8 8 8 8 8 8])
bitxor(4, [1 2 4 8 9 10 12]) // == bitxor([4 4 4 4 4 4 4], [1 2 4 8 9 10 12])


// Examples with big decimal integers:

u = sum(2 .^(600+[0 3 9 20 45])) // ~ 1.46D+194
bitxor(u, 2^630) == u+2^630 // true: XOR sets to 1 the missing bit #630 of u, so adds it
bitxor(u, 2^645) == u-2^645 // true: XOR sets to 0 the existing bit #645 of u, so removes it
bitxor(u, 2^601) == u // false: The bit #601 is 0 in u. XOR changes it.
//
n = fix(log2(u)) // 645 == Index of the heaviest bit of u
bitxor(u, 2^(n-52)) == u // false: The lightest bit of u was at 0 => This changes it
bitxor(u, 2^(n-53)) == u // true: Addressing bits below the lightest one doesn't change u
```

3.5.8 THE DEFECT CLASSIFICATION ALGORITHM USED IN THE IMAGE SUBTRACTION TECHNIQUE PROCESS

According to (IBRAHIM *et al.*, 2011), Image subtraction operation is a process this is primarily a way to discover differences between images. Subtracting one image from another effectively removes from the difference image all features this do not change while highlighting those this do.

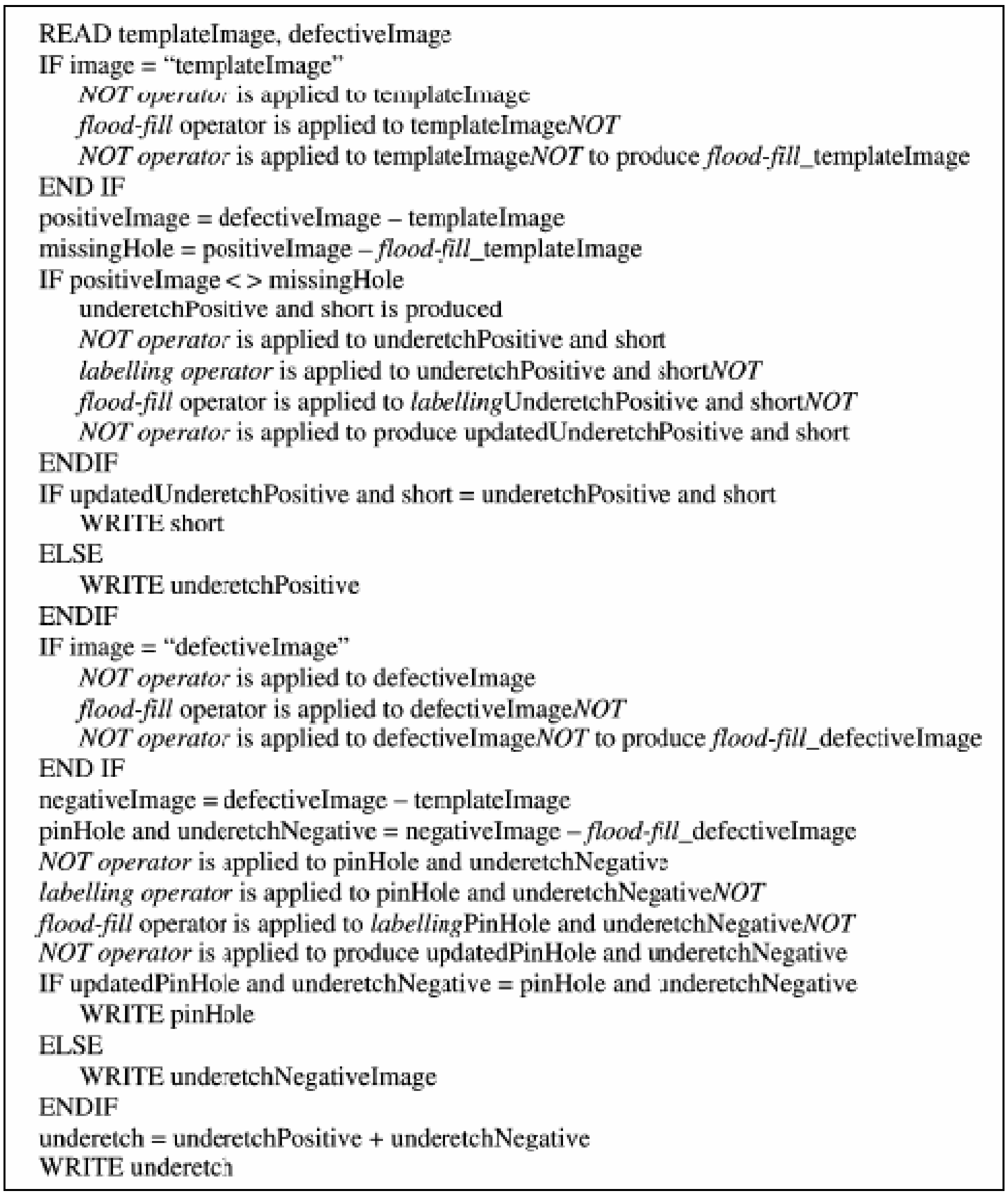
Both images of template image and defective image are compared pixel by pixel. This operation will produce either negative or positive result. Therefore, the outcome of this operation is divided into two; negative image and positive image. Using binary image, “1” represents white pixel and “0” represents black pixel. The **TABLE 3.3** shows the image subtraction rules.

TABLE 3.3 – IMAGE SUBTRACTION RULES

RULE	**RESULT**
If 1-0 = 1	Positive pixel image
If 0-1 = -1	Negative pixel image

According to the **TABLE 3.3**, during the image subtraction of an image, if the rule is “1-0=1” the resultant pixel will appear as “positive” in the resulting image (black pixel, for example), and if the rule is “0-1=-1” the resultant pixel will appear as “negative” in the resulting image (white pixel, for example).

Still according to (IBRAHIM *et al.*, 2011) the proposed defect classification algorithm shown in **Figure 3.41** is developed to detect and classify the defects into five types. Those types of defects are *missing hole*, *open-circuit*, *short-circuit*, *pin hole*, and *underetch*. The algorithm needs two images, namely template image and defective image.



```
READ templateImage, defectiveImage
IF image = "templateImage"
    NOT operator is applied to templateImage
    flood-fill operator is applied to templateImageNOT
    NOT operator is applied to templateImageNOT to produce flood-fill_templateImage
END IF
positiveImage = defectiveImage – templateImage
missingHole = positiveImage – flood-fill_templateImage
IF positiveImage < > missingHole
    underetchPositive and short is produced
    NOT operator is applied to underetchPositive and short
    labelling operator is applied to underetchPositive and shortNOT
    flood-fill operator is applied to labellingUnderetchPositive and shortNOT
    NOT operator is applied to produce updatedUnderetchPositive and short
ENDIF
IF updatedUnderetchPositive and short = underetchPositive and short
    WRITE short
ELSE
    WRITE underetchPositive
ENDIF
IF image = "defectiveImage"
    NOT operator is applied to defectiveImage
    flood-fill operator is applied to defectiveImageNOT
    NOT operator is applied to defectiveImageNOT to produce flood-fill_defectiveImage
END IF
negativeImage = defectiveImage – templateImage
pinHole and underetchNegative = negativeImage – flood-fill_defectiveImage
NOT operator is applied to pinHole and underetchNegative
labelling operator is applied to pinHole and underetchNegativeNOT
flood-fill operator is applied to labellingPinHole and underetchNegativeNOT
NOT operator is applied to produce updatedPinHole and underetchNegative
IF updatedPinHole and underetchNegative = pinHole and underetchNegative
    WRITE pinHole
ELSE
    WRITE underetchNegativeImage
ENDIF
underetch = underetchPositive + underetchNegative
WRITE underetch
```

Figure 3.41 – The defect classification algorithm (IBRAHIM et al., 2011)

This algorithm will be important when the Fast Wavelet Transform (FWT) is applied in the proposed algorithm related in this current work (see Chapter 5). The programming language used in the algorithm of (IBRAHIM *et al.*, 2011) is Pascal.

4 THE FAST WAVELET TRANSFORM (FWT) MATHEMATIC DEVELOPMENT

Having now the logic boolean model of the operation in which is desired to act, it's possible to continue to the development of the necessary mathematic formula for the project of bare PCB production sample time reduction and classifying efficiency using the FWT mathematic component. However, when a real PCB inspection process is analyzed, rarely are available all the necessary input and output variables to the application of the subtraction algorithm. The variable's choose in which is desired to utilize in the subtraction algorithm changes according to the PCB inspection process. For example, it's possible to apply just only one variable from the variables available in the PCB image acquisition system, a case where it's not necessary knows all the states of the PCB inspection process.

However, in this work, was decided to employ a specific type of image acquisition process (through bare PCB scanning), where the printer need the information's of the PCB position in the scanner. With the finality of evaluate this information, is employed usually a system called Video Device Unity (VDU). The theoretical embasement of this Chapter can be founded in (MALLAT, 1996).

4.1 FORMAL DEFINITION OF THE FAST WAVELET TRANSFORM (FWT)

According to (GONZALEZ AND WOODS, 2010), the Fast Wavelet Transform (FWT) is an implementation computationally efficient of the Discrete Wavelet Transform (DWT) this explores an amazing, though favorable, between the coefficients of DWT in adjacent scales. Also called *Mallat Pyramidal Algorithm* (MALLAT, 1989), the FWT is similar to the dual sub-band codification scheme.

Consider the multi-resolution refinement equation:

$$\varphi(x) = \sum_{n} h\varphi(n)\sqrt{2}\,\varphi(2x - n) \tag{4.1}$$

Performing the scale of *x* by 2^j, the translation by *k* and make *m = 2k + n* results in:

$$\varphi(2^j x - k) = \sum_{m} h\varphi(m - 2k)\sqrt{2}\,\varphi\left(2^{j+1} x - m\right) \tag{4.2}$$

Observe the vector of scale *hΦ* can be considered as “weights” utilized to expand $\Phi(2^j - k)$ as a summary of the scale functions, of scale $j + 1$.

Using the **Eq.(4.3)** and **Eq.(4.4)** as the starting point to a similar deduction involving the approximation coefficients of the *wavelets* series expansion (and DWT), we have:

$$c_j(k) = \sum_{m} h\varphi(m - 2k)c_{j+1}(m) \tag{4.3}$$

As the coefficients $c_j(k)$ and $d_j(k)$ of the series *wavelet* expansion became the coefficients $W_\Phi(j,k)$ and $W_\psi(j,k)$ of DWT when $f(x)$ is discrete, we can write the **Eq.(4.4)** and **Eq.(4.5)**:

$$W\psi(j,k) = \sum_{m} h\psi(m - 2k)W\varphi(j + 1, m) \tag{4.4}$$

$$W\varphi(j,k) = \sum_{m} h\varphi(m - 2k)W\varphi(j + 1, m) \tag{4.5}$$

The **Eq.(4.4)** and **Eq.(4.5)** reveal a notable relation between the DWT coefficients of adjacent scales. The **Figure 4.1** resumes the Fast Wavelet Transform (FWT) operations in form of block diagram. Note the diagram is identical to the portion of the codification and decoding system analyze in two sub-bands of the **Figure 4.1**, with $h_0(n) = h_\rho\,(-n)$ and $h_1(n) = h_\psi\,(-n)$. In this way, we can write:

$$W\psi\,(j,k) \;=\; h\psi\,(-n) \bigstar W\varphi\,(j+1,n) \;\mid n \;=\; 2k, k \geq 0 \tag{4.6}$$

$$W\varphi\,(j,k) \;=\; h\varphi\,(-n) \bigstar W\varphi\,(j+1,n) \;\mid n \;=\; 2k, k \geq 0 \tag{4.7}$$

Being the convolutions calculated in the instants $n = 2k$ for $k \geq 0$. Calculate the convolutions to the pair indexes non-negatives are equal to realize the filtering and the sub-sampling for a factor of 2.

The **Eq.(4.6)** and **Eq.(4.7)** define the calculus of the Fast Wavelet Transform (FWT). For a sequence of $M = 2^j$ size, the number of evolved mathematic operations is from the $0(M)$ order. It means, the number of multiplications and addictions is linear in relation to the entrance sequence size – because the number of the evolved multiplications and addictions in the convolutions realized by the FWT analysis bank of the **Figure 4.1** is proportional to the size of convoluted sequences. In this way, the Fast Wavelet Transform (FWT) is compared in a favorable form to the Fast Fourier Transform (FFT) algorithm, this requires some in order of $0(M \log_2 M)$ operations.

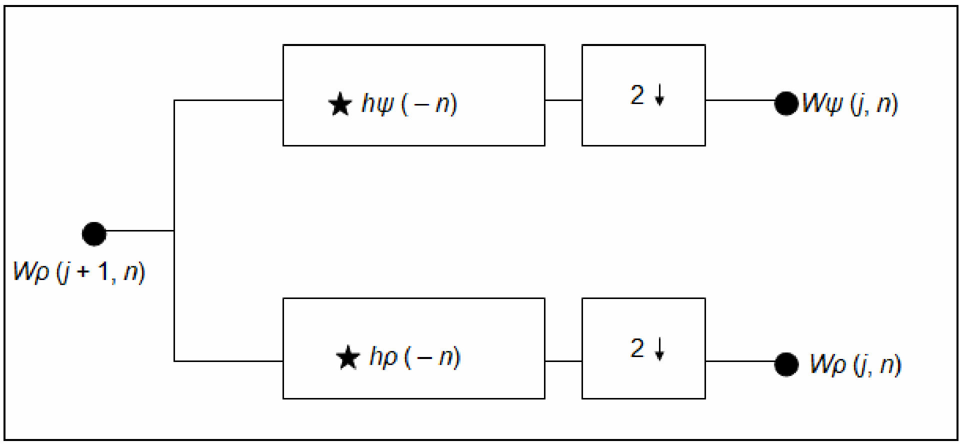


Figure 4.1 – A FWT analysis bank

To conclude the mathematic development of Fast Wavelet Transform (FWT), observe the FWT analysis bank of the **Figure 4.1** can be “repeated” to create multiple-stages structures for the Discrete Wavelet Transform (DWT) coefficients calculus in two or more successive scales. For example, **Figure 4.2** shows a two-stage filter bank for generating the coefficients at the two highest scales of the transform. Note the highest scale coefficients are assumed to be samples of the function itself. This is $W_\rho(J,n) = f$ (n), where J is the highest scale. The first filter bank in **Figure 4.2** splits the original function into a lowpass, approximation component, which corresponds to scaling coefficients $W_\rho(J - 1,n)$, and a highpass, detail component, corresponding to coefficients $W_\psi(J - 1,n)$. This is graphically illustrated in **Figure 4.2**, where scaling space V_J is split into wavelet subspace W_{J-1} and scaling subspace V_{J-1}. The spectrum of the original function is split into two half-band components. The second filter bank of **Figure 4.2** splits the spectrum and subspace V_{J-1}, the lower half-band, into quarter-band subspaces $W_\psi(J - 1,n)$ and $W_\rho(J - 1,n)$, respectively.

The two-stage filter bank of **Figure 4.2** is easily extended to any number of scales. A third filter bank, for example, would operate on the $W_\rho(J - 2,n)$ coefficients, splitting scaling space V_{J-2}, into two-eighth band subspaces W_{J-3} and V_{J-3}. Normally, we choose 2^J samples of $f(x)$ and employ P filter banks to generate P-scale FWT at scales $J - 1, J - 2,\ldots, J - P$.

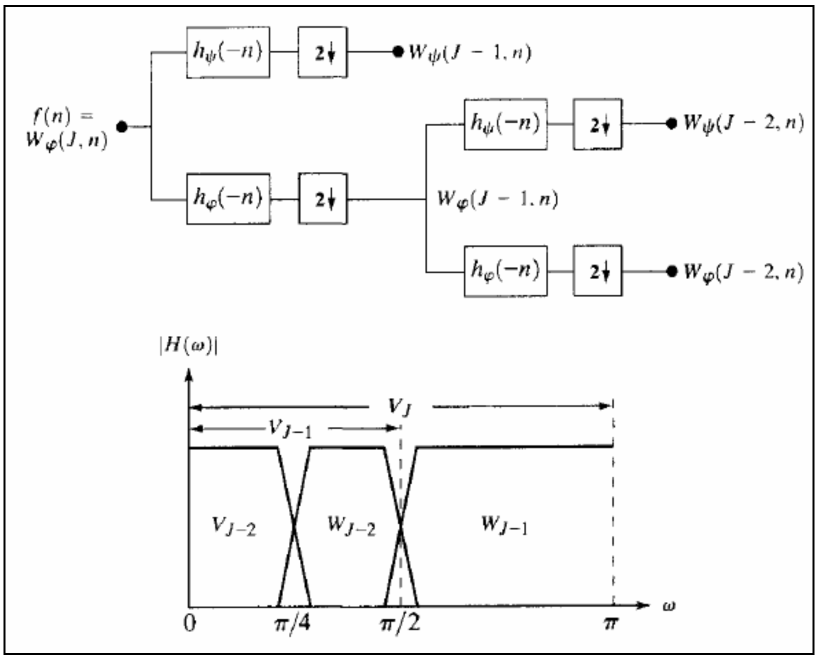


Figure 4.2 – (a) A two-stage or two-scale FWT analysis bank and (b) its frequency splitting

4.2 APPLICATION IN THE SUBTRACTION ALGORITHM

According to (ORTEGA *et al.*, 2007) applying now the concept of the Fast Wavelet Transform (FWT) of mathematic development to the proposed algorithm, we have:

- *coef1(x,y)* Second level approximation Haar wavelet transform of reference image;
- *coef2(x,y)* Second level approximation Haar wavelet transform of test image;
- *coef3(x,y) = coef1(x,y) – coef2(x,y)*;
- *if coef3(x,y) > t* then *coef3(x,y) = 1*, if not, *coef3(x,y) = 0*;
- *Coef4(x,y) = Defecto(x,y)* = fast wavelet transform of *coef4(x,y).*

* *Coef4(x,y)*: This coefficient is the *coef3(x,y)* after the application of an median filter.

* *Defecto(x,y)*: Is the Inverse Wavelet of *Coef4(x,y).*

The **Figure 4.3** illustrate the subtraction algorithm block diagram this represents the XOR logical operation, where a second level Haar wavelet is applied to reference image. The wavelet output approximations *(coef1)* are stored in the memory. A second Haar wavelet is applied to test image and then the output wavelet transform *(coef2)* is subtracted from memory. The resulting matrix *(coef3)* has the information of defects. Noise is eliminated thresholding absolute value of *coef3* and then applying 5 x 5 median filter. The resultant (*Coef4)* is generated and than the Fast Wavelet Transform (FWT) is applied generating the *Defecto(x,y)* class where the Bare PCB's defects are detected and then classified in a reduced time. Methodology convention can be conferred in the Sections 1.2 and 1.3.

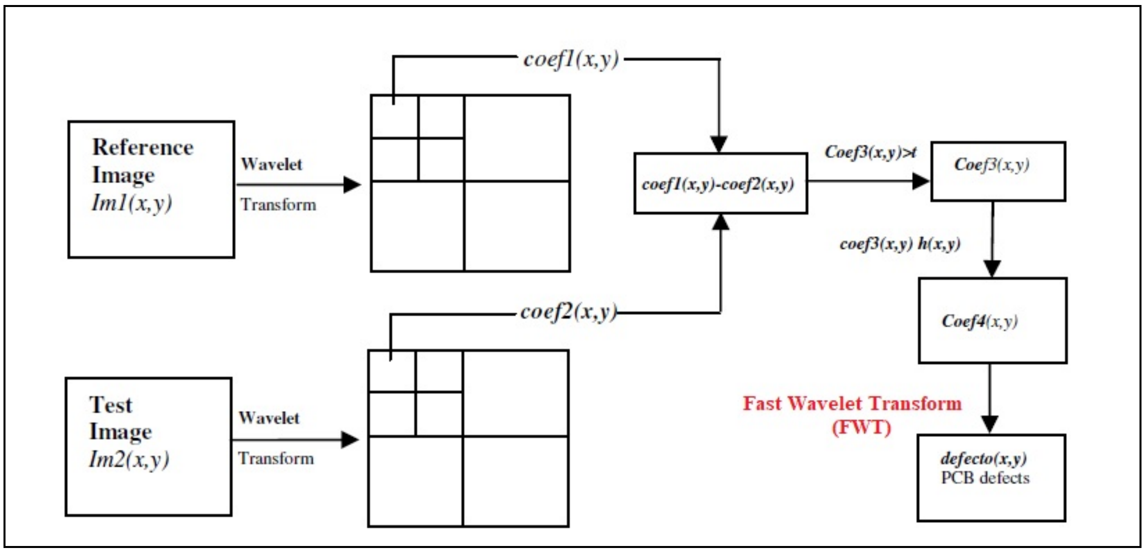


Figure 4.3 – Defect detection block diagram algorithm

The equation of errors in the estimated time to the upsampling and subsequently convolution is given by the **Eq.(4.8)**:

$$a_{k,\ell} = \frac{1}{\sqrt{2}} \sum_{m=-\infty}^{\infty} \left(a_{m,\ell-1} p_{k-2m} + d_{m,\ell-1} q_{k-2m}\right). \tag{4.8}$$

Now is passed to the task of acquire image classification groups from the *Defecto(x,y)* XOR operation. Usually, there is a constant defects classification groups, although, as explained before, the defect detection classification depends of the input variables. Is necessary explain the dependence of each image classification groups in function of distance between the PCB plan and sensor plan in the conveyor belt (cm), and for this propose, is necessary to understand how the system observability varies in function of the distance (cm). In this way, is possible to show the distance choose $\Delta(y1_{sensor} - y0_{PCI})$ in this proposed method will produce image values *Defecto(*x_2*,*

y_2) this generate errors in the resultant image viewing of the XOR operation using the Fast Wavelet Transform (FWT) asymptotically stable to zero.

To the simplicity purposes, the calculus of the observability values and its classification groups to each bare PCB's defect detected was realized using the MATLAB software. Having now this information, is known now this is possible to choose a defect class group from the subtraction algorithm for each desired distance.

4.3 DEFECT CLASS SELECTION FROM THE SUBTRACTION ALGORITHM

Having now the complete information about the observability to all the desired distances, it is possible to locate the dynamic poles referred to the estimated errors. For this purpose, was utilized a parameter in MATLAB software. For each distance, is founded the corresponded set-points m_2, n_2 and the parameter $\Delta(y1_{sensor} - y0_{PCI})$ is constructed in the mode of to locate the poles of m_2, n_2 such those resultant poles (m_x, n_y) can be equal to the twice values of set-points m_2, n_2.

This choice was given to warrant the convergence to the zero of the errors about the estimated states happens with a major distance than the system dynamics, warranting in this way the estimated states can be an estimative in the real time of the sensor states. As demonstrated before, the image inspection process always will have stable poles in the lightness spacing technique (look at Chapter 3), what warranty this located poles will be also stables.

Then, utilizing a discretized distance spaces, it is possible to apply the concept of poles location to found the parameters $\Delta(y1_{sensor} - y0_{PCI})$ for each distance. As would be very difficult to realize a calculus of this parameter for each possible distance in a continuous space, was choose a calculus of components from $\Delta(y1_{sensor} - y0_{PCI})$ for a distance spaces with a gross resolution in the line inspection VDU, to after than interpolate each distance from the inspection process considering a lean resolution of the distances space. For this purpose, was selected an interpolation method in MATLAB.

Once obtained the distances from the image acquisition system, would be possible to have a relation table (*scheduling*) between each value of the actual distance of the image inspection camera and the values of the imagen inspection system of the inspection process. In the praxis, it would be equivalent to allocate a table with 4 different values (input parameters $\Delta(y1_{sensor} - y0_{PCI})$) to each discretized distance. Then, the electronic unity responsible to the realization of the states estimation with basis in the sensors reading should made it taking account also the medium distance of the image inspection system. In an embedded system, the memory limitation would be also have taken in account to the choose of a system in which those relations would be implemented.

Having now a method for the calculus of the estimations to the states of the distance sensors in the image acquisition system, the same can be set with a feedback entrance to the process control of the image acquisition system, ending the grid and generating a signal in the actuator (in case, the conveyor belt) to the enhancement of the inspection velocity performance. Can be passed now to the part of the subtraction algorithm development project using the FWT transform.

5 THE SUBTRACTION ALGORITHM DEVELOPMENT USING THE FWT TRANSFORM

Defines a LS (linear simulation) in MATLAB a simulation where the system dynamics is described through a conjunct of differential linear equations (as in this case the image subtraction algorithm using XOR operation and FWT Transform for time optimization), and the image binarization process (described forward) is defined by a linear equation. One possible solution to this problem is given by the FWT equation (*Fast Wavelet Transform*).

Its objective is to determine a conjunction of filters so this, in the subtraction algorithm, the inspection system reduces the bare PCB difference image processing time defined previously. One of the main advantages of this method is to systematize the manner of process image subtraction time, given to the responsible operator control system specify the coefficients of the time and classifying function using the FWT.

In this work was choose by utilize the control technique using XOR operation due to the frequency which the same is employed in the related works of PCB Image Subtraction Process, beyond to enable an automatic calculus of the time gains in the image subtraction system when considering a smooth adjustment of weighing related to the control efforts.

5.1 FORMAL DEFINITION OF THE FWT SUBTRACTION ALGORITHM

Give a system which general equation is described by **Eq.(5.1)**:

$$g(x, y) = f(x, y) - h(x, y). \tag{5.1}$$

the problem of the FWT transform is based on to found a multi-resolution **g(x)** of image pixels in mode of:

$$g(x, y) = \begin{cases} 1 & \text{if } f(x, y) > T \\ 0 & \text{if } f(x, y) \leq T. \end{cases} \tag{5.2}$$

in mode of the image subtraction XOR operation (function subtraction, **g(x)**) can be filtered and optimized:

$$\varphi(x) = \sum_{n} h\varphi(n)\sqrt{2}\,\varphi(2x - n) \tag{5.3}$$

where the coefficients ***n*** and ***x*** are the instants and the image pixel, respectively.

Note this coefficient ***n*** represents a weight attributed to the FWT image time status (its variation in turning *ms* or *s*, by have linear form), and ***x*** the binary pixel status varying between 0 or 1 by have binary form giving the result **φ(x)** of the resultant filtered image (x,y).

Still note this, assuming the multi-resolution law **g(x)** given by the **Eq.(5.2)**, is there the equation who describes the dynamic of states for a finite number of non zero coefficients, also called mother wavelet function represented by the **Eq.(5.4)**.

$$\psi(x) = \sum n \, (-1)^{n} \, C_{1-n} \, \phi \, (2x-n)$$

(5.4)

so the system will be represented, with a higher resolution, case the values of the bottom signal (Haar wavelet function) **ψ (x)** and the top signal (Haar scaling function) **φ(x)** have both projections onto V_0 and V_1 spanned by two resolutions of the Haar basis.

The resolution of the problem of the Haar basis can be found in (GONZALEZ AND WOODS, 2010), and at this point just only its solution will be presented. The values of each pixel of the resultant image subtraction XOR operation can be founded through the solution of the Fourier Transform associated:

$$m_0(\xi) = \frac{1}{\sqrt{2}} \sum_{0}^{2N-1} h_k \, e^{-1k\xi}$$

$$m_1(\xi) = \frac{1}{\sqrt{2}} \sum_{0}^{2N-1} g_k \, e^{-1k\xi} \; .$$

(5.5)

The **Eq.(5.5)** can be easily solved through computational techniques commonly implemented in softwares as the MATLAB, used to make the linear Haar simulations implemented in the FWT transform in the Chapter 7. Still, from the optimized FWT equation, we can show this, if the coefficients ***n*** and ***x*** existis and is positive-defined, the bottom signal **ψ (x)** and the top signal **φ(x)** will be stable. In this way, the values for each pixel of resultant image can be expressed by the FTW multi-resolution analysis:

$$\phi(x) = \sqrt{2} \sum_n h_n \phi(2x-n) \quad and$$

$$\psi(x) = \sqrt{2} \sum_n g_n \phi(2x-n).$$

(5.6)

The resolution of XOR operation bare PCB defect detection using the FWT transform then consists in the following steps:

1. Defines de general equation of the system – Equation (5.1);
2. Defines the function subtraction, **g(x) –** Equation (5.3);
3. Solve the Z-Transform associated – Equation (5.5);
4. Found the FWT multi-resolution equation – Equation (5.6).

Beyond this, similary to the case of the Fast Wavelet Transform Mathematic Development (as viewed in the Chapter 4), we must also analyze the concept of controllability for this system.
The concept of controllability of states consists in, given any initial state, will be possible to start of application of an entrance elevate this state for another state desired, in a finite time interval. One more time, it must be noticed this for each inspection velocity, we have a state with different dynamic characteristics. So, we must to analyze the Inductive Learning Formula, described by:

$$H = \sqrt{\tfrac{E^c}{E}} \times [2 - 2\sqrt{\tfrac{E_i^c E_i}{E^c E}} - 2\sqrt{(1 - \tfrac{E_i^c}{E^c})(1 - \tfrac{E_i}{E})}]$$

(5.7)

have posted *E*. Also in the case of controllability, is necessary to calculate the parameter *H* for each velocity value of the inspection system.

5.2 APPLICATION OF THE FWT SUBTRACTION ALGORITHM IN THE INSPECTION MODEL

Applying now the concept of controllability now in the inspection system, we must to analyze the Inductive Learning Formula *H* for a conjunct of discrete velocities. And we have obtained the following algorithm, which is responsible to the XOR operation in the image subtraction between two PCB images (default image and testing image):

```
function varargout = PreGui(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @PreGui_OpeningFcn, ...
                   'gui_OutputFcn',  @PreGui_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

handles.output = hObject;

function pushbutton3_Callback(hObject, eventdata, handles)

F = getframe(handles.axes3);
Image = frame2im(F);
filter = {'*.jpg;*.png'};
[file, path] = uiputfile(filter);
if file > 0
  filename = fullfile(path,file);
  imwrite(Image, filename);
end
```

5.3 PATTERNS SELECTION FROM THE FWT ALGORITHM

Once this algorithm is constructed to optimize time inspection and classifying efficiency, is indicated this parameters of time and classification depends of the inspection velocity. Still defined some parameters as constant, some equations of the inspection system is part of the **Eq.(5.5)**, and carry on the information of velocity dependence. Is proceeded in this same way as explained in the Chapter 4 to the calculus of gains for a small conjunct of different velocities and interpolation of those values.

As well as in the case of the subtraction algorithm, in the praxis the FWT transform have access to a relation table between the applicable gain to each estimated state and the actual velocity of the inspection system. In this way, the electronic unity of control can utilize 4 distinct gains (matrix velocities) to feedback the system with basis in the longitudinal velocity actual of the inspection system, generating a control signal adequate.

Finalized this step, we have then a manner to estimate the state of the inspection system (as viewed in the Chapter 4) and a sensate choose of how the states are related to the control signal (viewed in this Chapter). We can now proceed to the process of Bare PCB’s Failure Detection Criteria for simulation, measuring the output signals, actuating over the inspection system.

6 BARE PCB’S FAILURE DETECTION CRITERIA FOR SIMULATION

In the Chapter 6, is formulated the patterns of the PCB circuit masks utilized for the subtraction algorithm process, as well as for the pre-algorithm binarization process, which is the part of the FWT algorithm used in this research. Independently of the utilized control strategy, is used the output average (and possibly the entrances of the process) to define which is the better control signal to be utilized in the inspection process in mode of to acquire an optimized image angle for the subtraction algorithm, viewing the better performance in the inspection system.

For this purpose, is defined the mask of PCB-I circuit and the mask of PCB-II circuit utilized in the binarization process of the FWT algorithm of this thesis in the previous-processing (pre-processing algorithm). This is necessary for this simulations can be done in mode of reach the PCB's reading of input and output variables.

6.1 MASK OF PCB-I CIRCUIT

Following is represented the mask of PCB-I circuit utilized in the binarization process of the FTW pre-processing algorithm. This mask was done utilizing a FR4 material and then the coming corrosion in perchlorate. The next images shows the masks after corrosion, ready to be utilized in the binarization algorithm:

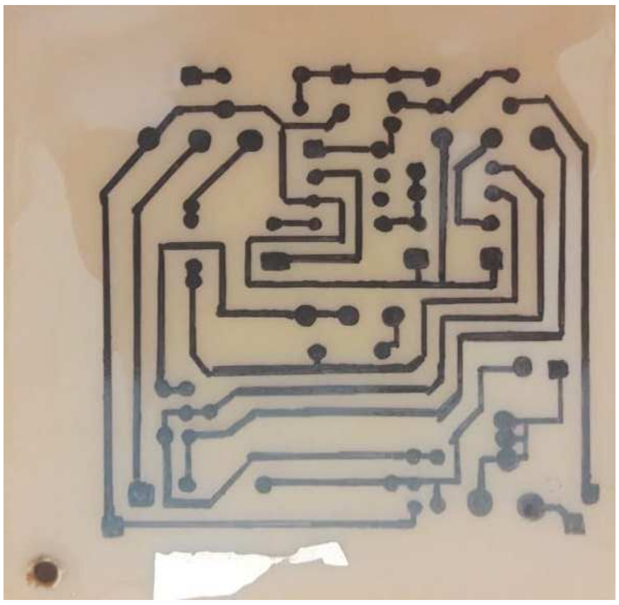


Figure 6.1 – Mask of PCB-I after corrosion

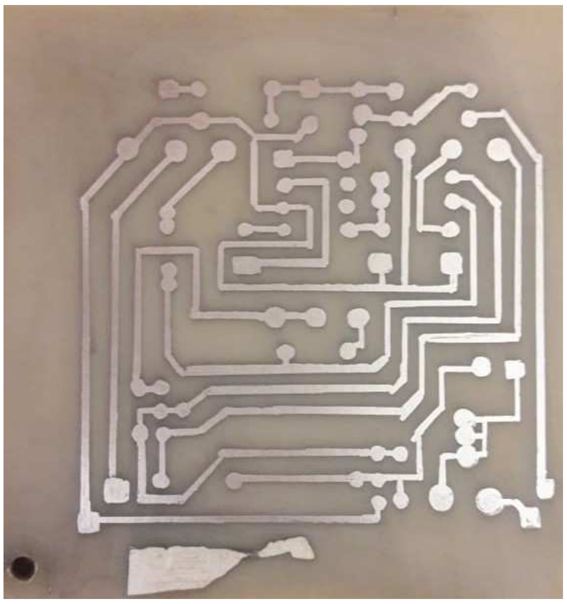


Figure 6.2 – Mask of PCB-I ready to input

The next step is to put into the FWT pre-processing algorithm the **Figure 6.2** and then realize the linear simulations in the software developed in MATLAB. For this simulations, was utilized the parameters of input variables such as *distance between PCB plan and sensor belt (cm)*, *rotation between PCB and sensor axes (cm)* and *PCB position (cm)*.

6.2 APPLICATION OF PCB-I IN THE FWT ALGORITHM

Considering this realist case, we can work with the hypothesis this there are just only three variables passible to measurement (as shown previously in the Chapter 2), contained in the begin of this currently Chapter. In this case, it is possible to feed the algorithm system just only with one proportional factor of those measurements, configuring a particular case of image inspection input algorithm as shown in the **Figure 6.3**.

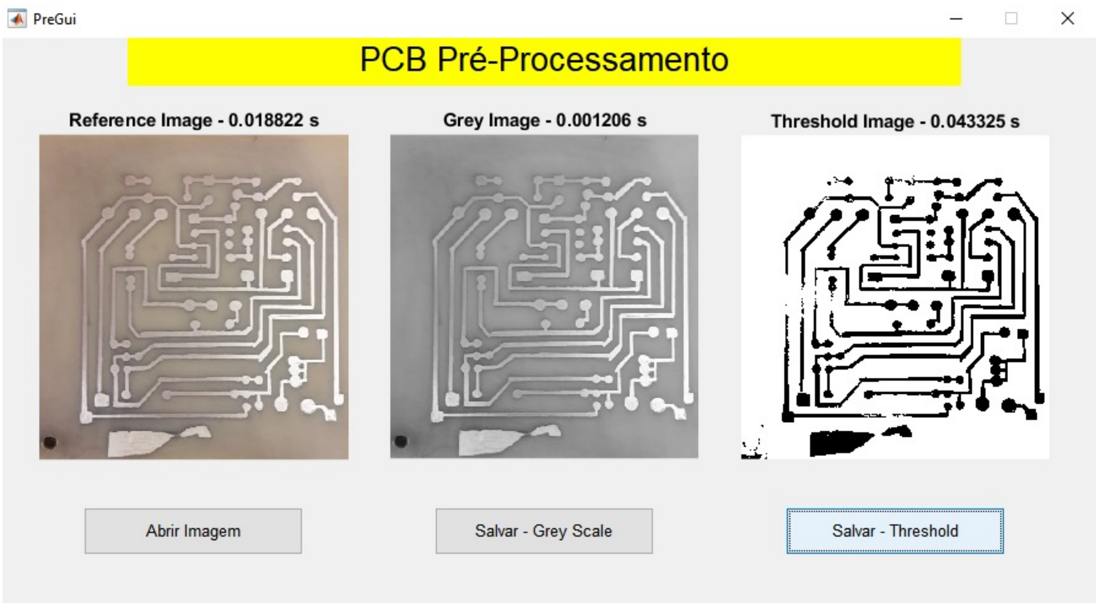


Figure 6.3 – Mask of PCB-I ready to be scanned in the FWT algorithm

Is observed the inclusion of a resultant image of the mask of PCB-I circuit in gray-scale and another image applied a thresholding of T=126, being the scanned image of the FWT pre-processing image algorithm. This case is realist because take into account this in practice hardly we have all the informations in respect of the state of the input image.

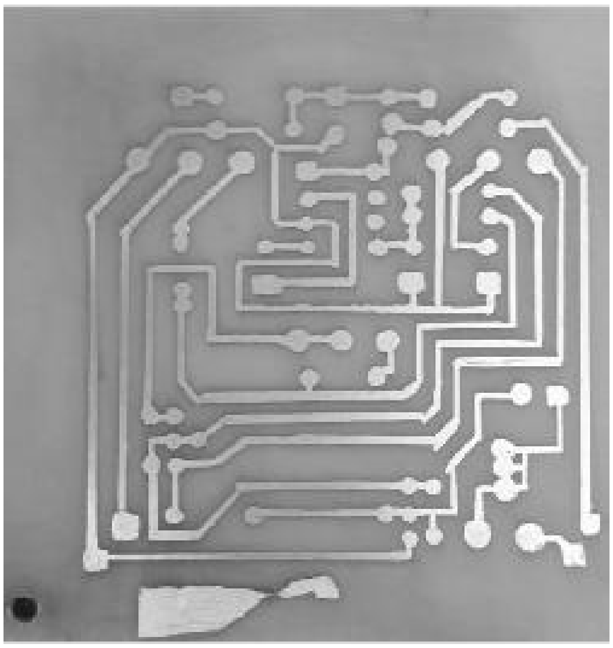


Figure 6.4 – Gray-scale mask of PCB-I

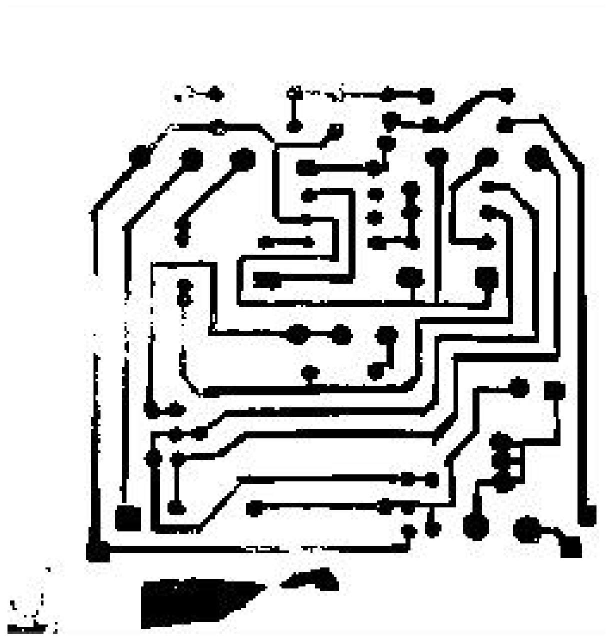


Figure 6.5 – Scanned image for PCB-I mask

After reference PCB-I was runned in the FWT pre-processing algorithm, we have obtained two main imagens in the inspection system: the **Figure 6.4** this represents the gray-scale mask for the next thresholding process utilizing the *global threshold* for T=126.

As the failure detection criteria for simulation, the classification algorithm can detect:

TABLE 6.1 – PCB-I MASK DEFECTS DETECTION

ITEM	**DEFECT**
1	Breakout
2	Pin Hole
3	Open Circuit

6.3 MASK OF PCB-II CIRCUIT

Following is represented the mask of PCB-II circuit utilized in the binarization process of the FTW pre-processing algorithm. This mask was done utilizing a FR4 material and then the coming corrosion in perchlorate. The next images shows the masks after corrosion, ready to be utilized in the binarization algorithm:

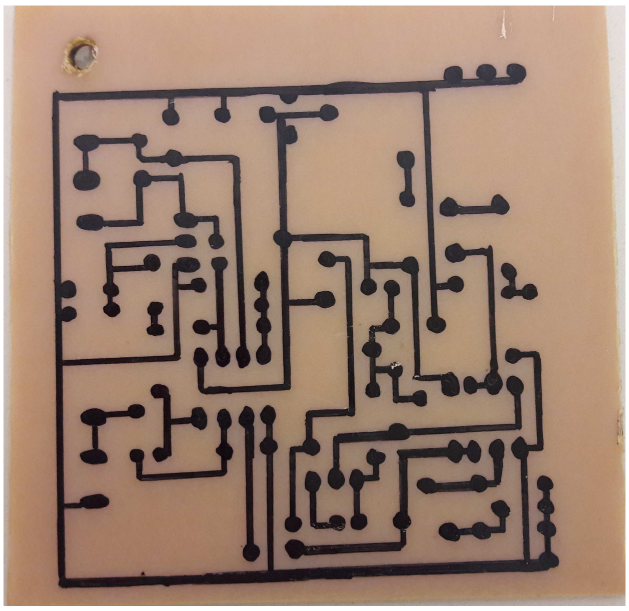


Figure 6.6 – Mask of PCB-II after corrosion

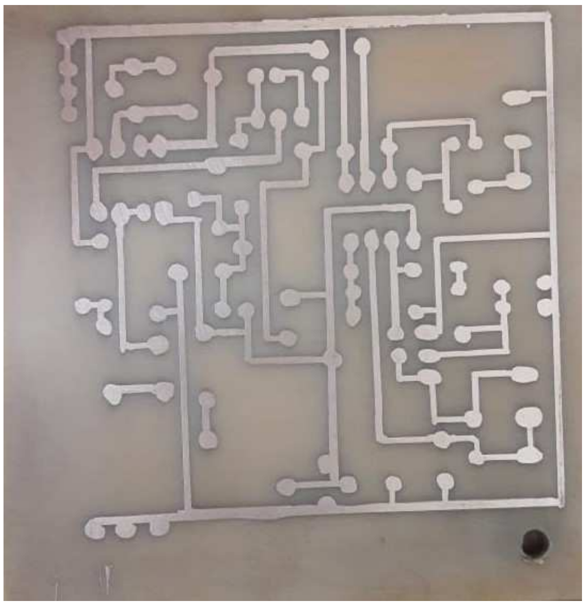


Figure 6.7 – Mask of PCB-II ready to input

The next step is to put into the FWT pre-processing algorithm the **Figure 6.7** and then realize the linear simulations in the software developed in MATLAB. For this simulations, was utilized the parameters of input variables such as *distance between PCB plan and sensor belt (cm)*, *rotation between PCB and sensor axes (cm)* and *PCB position (cm)*.

6.4 APPLICATION OF PCB-II IN THE FWT ALGORITHM

Considering this realist case, we can work with the hypothesis this there are just only three variables passible to measurement (as shown previously in the Chapter 2), contained in the begin of this currently Chapter. In this case, it is possible to feed the algorithm system just only with one proportional factor of those measurements, configuring a particular case of image inspection input algorithm as shown in the **Figure 6.8**.

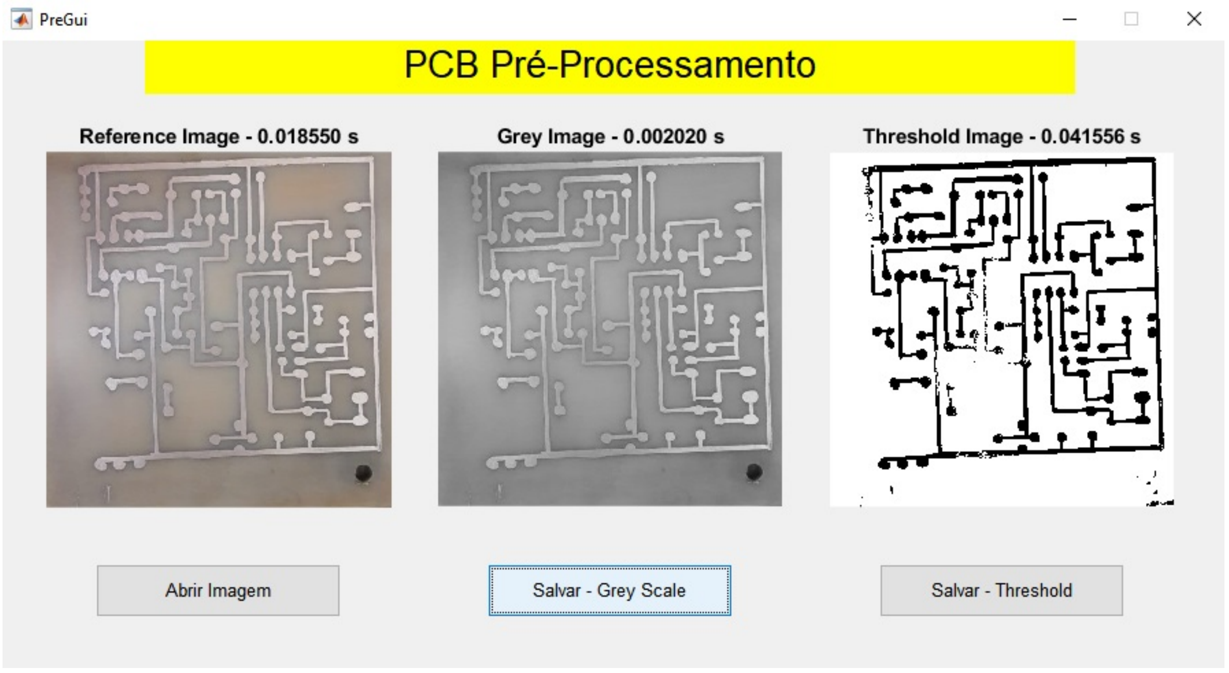


Figure 6.8 – Mask of PCB-II ready to be scanned in the FWT algorithm

Is observed the inclusion of a resultant image of the mask of PCB-I circuit in gray-scale and another image applied a thresholding of T=126, being the scanned image of the FWT pre-processing image algorithm. This case is realist because take into account this in practice hardly we have all the informations in respect of the state of the input image.

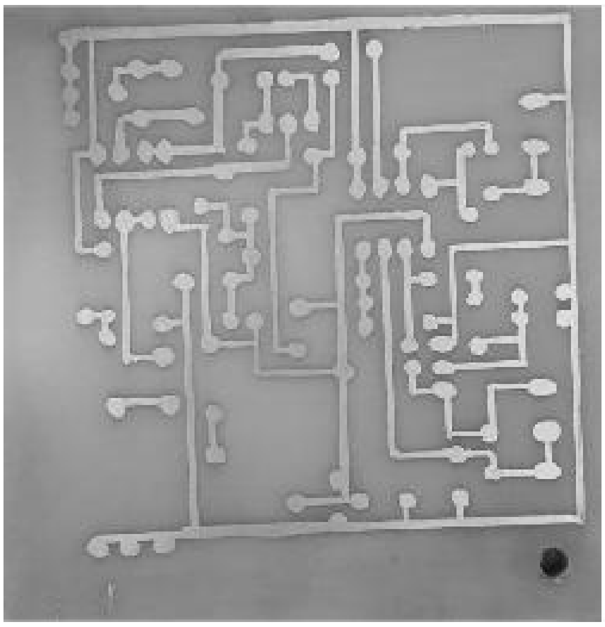
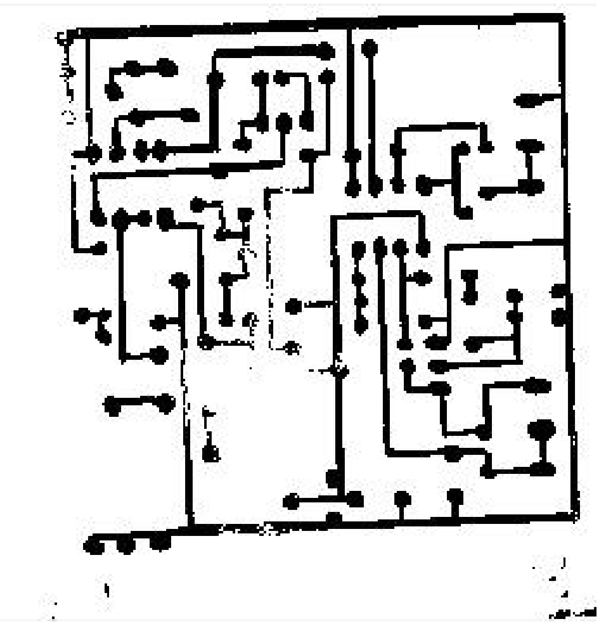



Figure 6.9 – Gray-scale mask of PCB-II **Figure 6.10 – Scanned image for PCB-II mask**

After reference PCB-II was runned in the FWT pre-processing algorithm, we have obtained two main imagens in the inspection system: the **Figure 6.9** this represents the gray-scale mask for the next thresholding process utilizing the *global threshold* for T=126.

As the failure detection criteria for simulation, the classification algorithm can detect:

TABLE 6.2 – PCB-II MASK DEFECTS DETECTION

ITEM	**DEFECT**
1	Breakout
2	Pin Hole
3	Open Circuit

7 DESCRIPTION OF THE SIMULATIONS

Once finalized the system modelling, we keep forward to the computational implementation of those models. We can proceed of two manners to realize the simulations related to the process of interest:

- Linear simulations: utilizing the proper Matlab, through the agroupment of the state matrix related to the inspection system and the command *decomposition*, this simulate the answers of the continuous or discrete linear systems to the arbitrary entrances. This kind of simulation will utilize the model of Global Thresholding (Subsection 3.3.1);

- Non-linear simulations: to realize those simulations, we can use Simulink, because the presence of various sources of non-linearity. In this case, was preferred to do not utilize Simulink to the optimization of the modularity of the system simulations, studying the behavior of the image inspection system (reference image, testing image, etc.) through a FWT separated code. For this kind of simulation is utilized the model of the FWT subtraction algorithm in the inspection system model (Subsection 5.2).

The linear simulations have implementation in the state spaces, as described previously. The implementation in Simulink bring more complications, by dealing with non-linarites. In this case, we must work modelling each part of the system separately, for after a while to analyze the performance of the system in front of the interest entrances.

7.1 UTILIZATION OF WAVELETS IN THE LINEAR SIMULATIONS

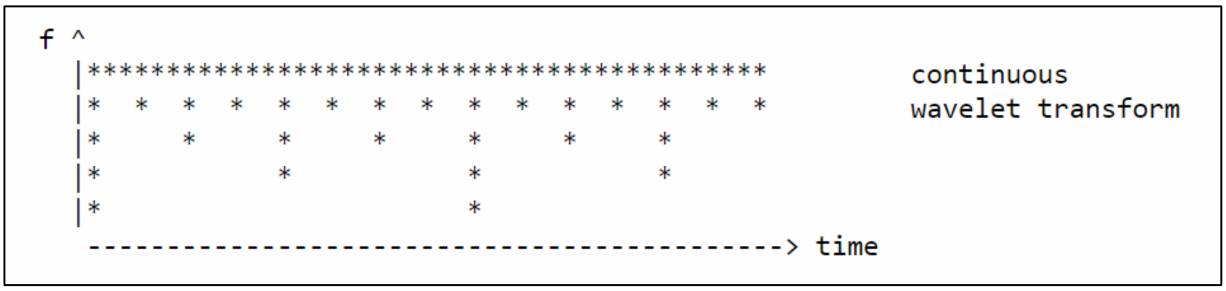
According to the (POLIKAR, 1996) the Wavelet Transform provides time-frequency representation. Often times a particular spectral component occurring at any instant can be of particular interest. In these cases it may be very beneficial to know the time intervals these particular spectral components occur.

Wavelet transform is capable of providing the time and frequency information simultaneously, hence giving a time-frequency representation of the signal.

The frequency and time information of a signal at some certain point in time-frequency plane cannot be known. In other words: we cannot know the *what spectral component* exists at any given time *instant*. The best we can do is to investigate what *spectral component* exist at any given *interval* of time. This is a problem of resolution.

Higher frequencies are better resolved in time, and low frequencies are better resolved in frequency. This means, a certain high frequency component can be located better in time (with less relative error) than a low frequency component. On the contrary, a low frequency component can be located better in frequency compared to high frequency component.

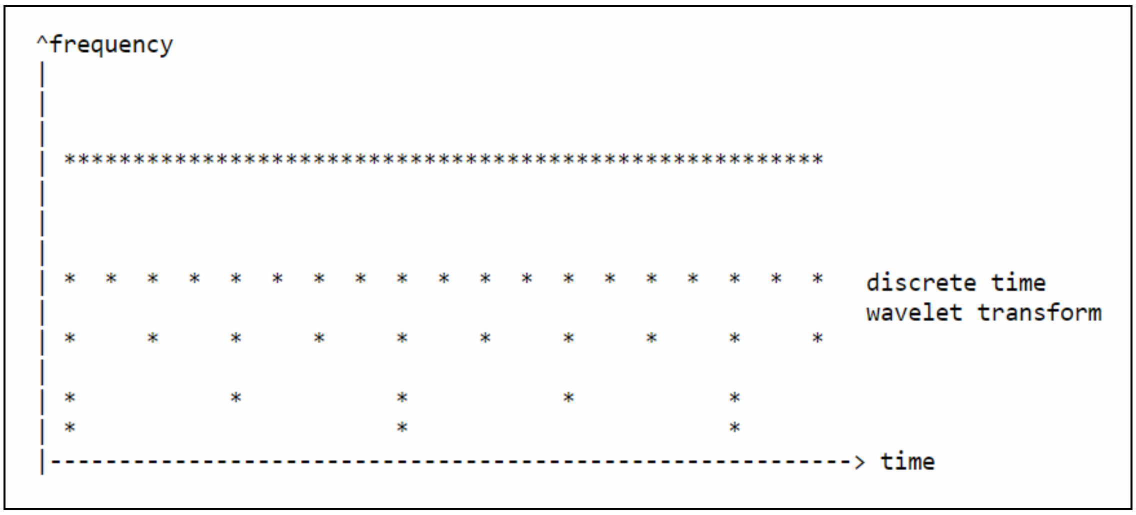
Take a look at the **Figure 7.1**, this represents the Continuous Wavelet Transform:



```
f ^
  |*******************************************          continuous
  |*  *  *  *  *  *  *  *  *  *  *  *  *  *  *          wavelet transform
  |*     *     *     *     *     *     *
  |*           *           *           *
  |*                       *
   --------------------------------------------> time
```

Figure 7.1 – Continuous Wavelet Transform

In discrete time case, the time resolution of the signal works the same as above, but now, the frequency information has different resolutions at every stage too. Note this, lower frequencies are better resolved in frequency, where higher frequencies are not. Note how the spacing between subsequent frequency components increase as frequency increases. The **Figure 7.2** shows the Discrete Time Wavalet Transform:



```
^frequency
|
|
|
| *******************************************************
|
|
|
| *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *   discrete time
|                                                           wavelet transform
| *     *     *     *     *     *     *     *     *     *
|
| *           *           *           *           *
| *                       *                       *
|----------------------------------------------------------> time
```

Figure 7.2 – Discrete Wavelet Transform

7.2 FAST WAVELET TRANSFORM (FWT) ALGORITHM

In 1998, Mallat produced a fast wavelet decomposition and reconstruction algorithm [Mal189]. The Mallat algorithm for discrete wavelet transform (DWT) is, in fact, a classical scheme in the signal processing community, known as a two-channel subband coder using conjugate quadrature filters or quadrature mirror filters (QMFs).

- The decomposition algorithm starts with signal *s*, next calculates the coordinates of A_1 and D_1, and then those of A_2 and D_2, and so son.
- The reconstruction algorithm called the inverse discrete wavelet transform (IDWT) starts from the coordinates A_J and D_J, next calculates the coordinates A_{J-1}, and then using the coordinates of A_{J-1} and D_{J-1} calculates those A_{J-2}, and so on.

7.3 FILTERS USED TO CALCULATE THE DWT AND IDWT

For an orthogonal wavelet, in the multiresolution framework, we start with the scaling function φ and the wavelet function ψ. One of the fundamental relations is the twin-scale relation **Eq.(7.1)** (dilation equation or refinement equation):

$$\frac{1}{2}\phi\left(\frac{x}{2}\right)=\sum_{n\in Z} w_n\phi(x-n) \tag{7.1}$$

All the filters used in DWT and IDWT are intimately related to the sequence $(Wn)_{n\varepsilon Z}$

Clearly if φ is compactly supported, the sequence (Wn) is finite and can be viewed as a filter. The filter W, which is called the scaling filter (nonnoremalized), is

- Finite Impulse Response (FIR)
- Of lenght 2*N*
- Of sum 1

- Of norm 1/√2
- A low-pass filter

For example, for the db3 scaling filter,

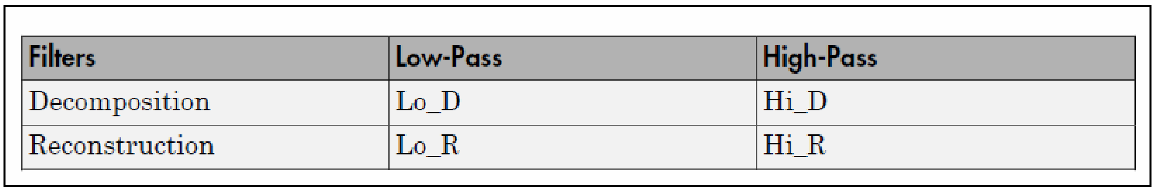
```
load db3
db3
        db3 =
        0.2352        0.5706        0.3252        -0.0955        -0.0604        0.0249




sum (db3)
        ans =
                1.0000

norm(db3)
        ans =
                0.7071
```

From filter *W*, we define four FIR filters, of lenght 2*N* and of norm 1, organized as follows the **Figure 7.3**.



Filters	Low-Pass	High-Pass
Decomposition	Lo_D	Hi_D
Reconstruction	Lo_R	Hi_R

Figure 7.3 – FIR filters

The four filters are computed using the following scheme as shows the **Figure 7.4**.

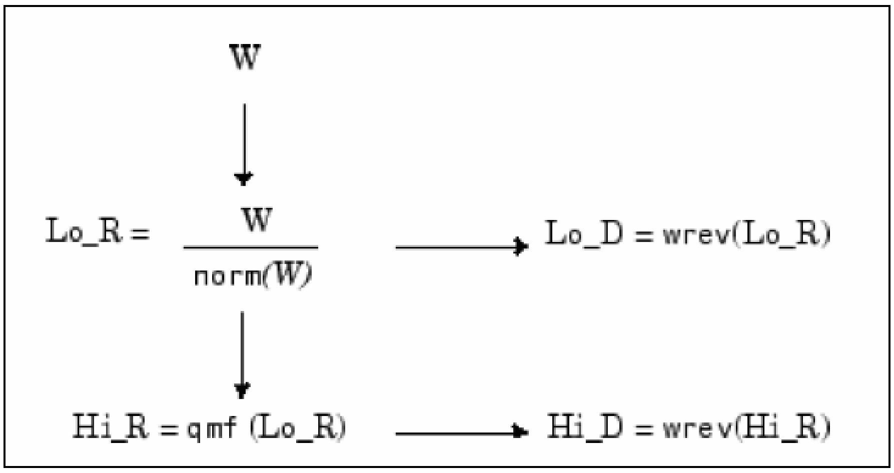


Figure 7.4 – Filters computation

where *qmf* is such this Hi_R and Lo_R are quadrature mirror filters (i.e., Hi_R(k) = $(-1)^k$ Lo_R($2N + 1 - k$)) for $k = 1, 2, ..., 2N$.

Note this *wrev* flips the filter coefficients. So Hi_D and Lo_D are also quadrature mirror filters. The computation of these filters is performed using *orthfilt*. Next, we illustrate these properties with db6 wavelet.

Load the Daubechies' extremal phase scaling filter and plot the coefficients.

```
load db6;
subplot(421); stem(db6,'markerfacecolor',[0 0 1]);
title('Original scaling filter');
```

Use *orthfilt* to return the analysis (decomposition) and synethsis (reconstruction) filters.

Obtain the discrete Fourier transforms (DFT) of the lowpass and highpass analysis filters. Plot the modulus of the DFT

```
LoDFT = fft(Lo_D,64);
HiDFT = fft(Hi_D,64);
freq = -pi+(2*pi)/64:(2*pi)/64:pi;
subplot(427); plot(freq,fftshift(abs(LoDFT)));
```

```
set(gca,'xlim',[-pi,pi]); xlabel('Radians/sample');
title('DFT Modulus - Lowpass Filter')
subplot(428); plot(freq,fftshift(abs(HiDFT)));
set(gca,'xlim',[-pi,pi]); xlabel('Radians/sample');
title('Highpass Filter');
```

These test is represented in the **Figure 7.5**, where the graphics represents the *decomposition low-pass filter*, *decomposition high-pass filter*, *reconstruction low-pass filter*, *reconstruction high-pass filter*, *DFT Modulus – Lowpass Filter* and *Highpass Filter*.

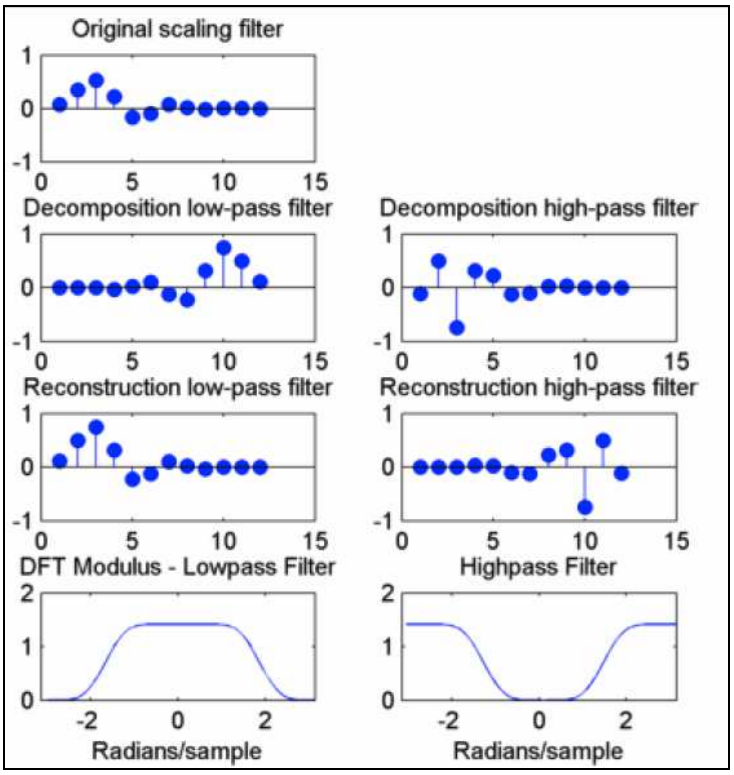


Figure 7.5 – Modulus of the DFT

7.4 ALGORITHMS USED TO CALCULATE THE FWT TRANSFORM

Given a signal *s* of lenght *N*, the DWT consists of $\log_2 N$ stages at most. Starting from *s*, the first step produces two sets of coefficients: approximation coefficients cA_1, and detail coefficients cD_1. These vectors are obtained by convolving *s* with the low-pass filter Lo_D for approximation, and with the high-pass filter Hi_D for detail, followed by dyatic decimation.

More precisely, the first step is represented by the **Figure 7.6** bellow:

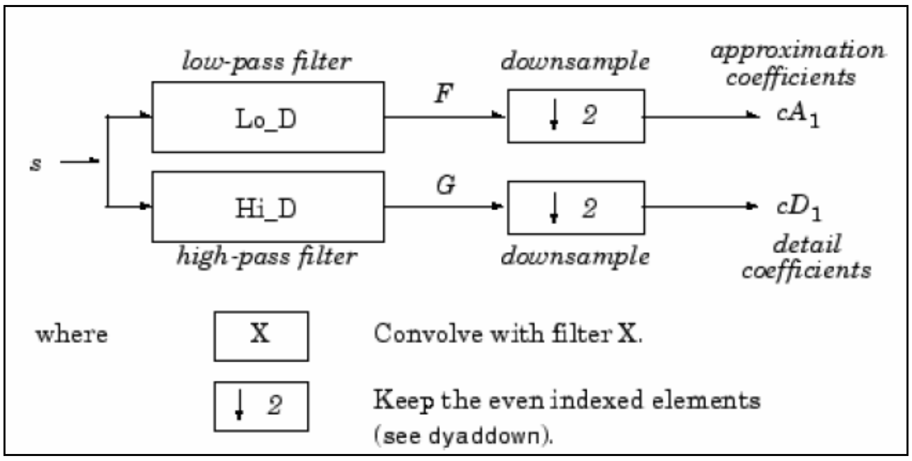


Figure 7.6 – First step of the DWT calculation

The length of each filter is equal to 2*L*. The result of convolving a length *N* signal with a length 2*L* filter is N+2*L*-1. Therefore, the signals *F* and *G* are of length N + 2*L* – 1. After downsampling by 2, the coefficient vectors cA_1 and cD_1 are of length

$$\left\lfloor \frac{N-1}{2} + L \right\rfloor .$$

The next step splits the approximation coefficients cA_1 in two parts using the same scheme, replacing *s* by cA_1 and producing cA_2 and cD_2, and so on. The **Figure 7.7** shows the *decomposition* step of One-Dimensional DWT:

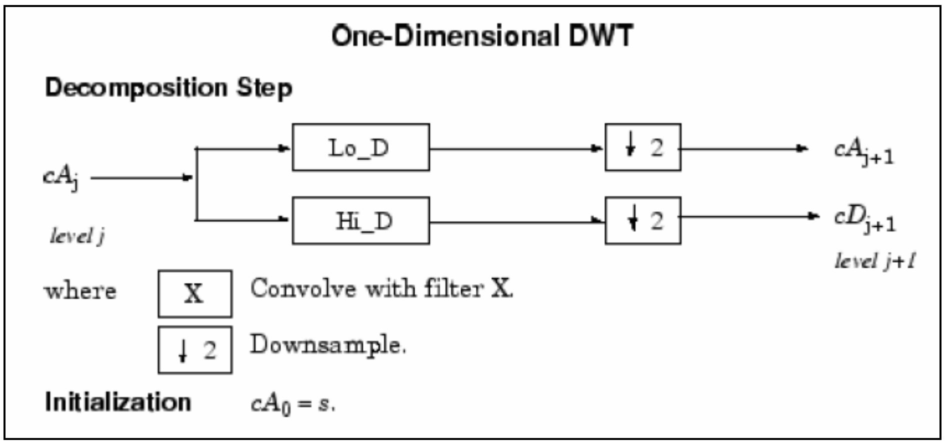


Figure 7.7 – Decomposition step of One-Dimensional DWT

So the wavelet decomposition of the signal *s* analyzed at level *j* has the following structure: [cA_j, $cD_j, \ldots, cD_1$].

This structure contains for J = 3 the terminal nodes of the following tree as the **Figure 7.8** represents:

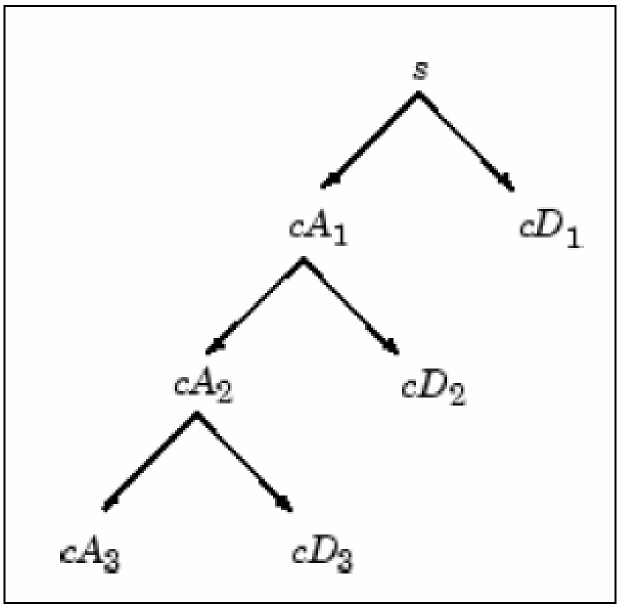


Figure 7.8 – Tree of terminal nodes

- Conversely, starting from cA_j and cD_j, the IDWT reconstructs cA_{j-1}, inverting the decomposition step by inserting zeros and convolving the results with the reconstruction filters as indicates the **Figure 7.9**:

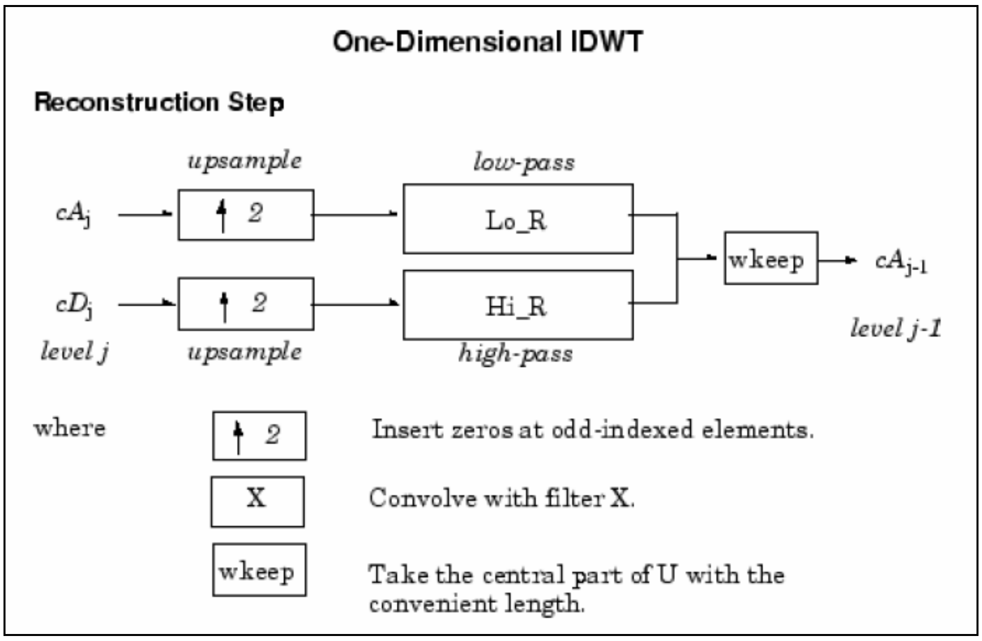


Figure 7.9 – One-Dimensional IDWT

- For images, a similar algorithm is possible for two-dimensional wavelets and scaling functions obtained from one-dimensional wavelets by tensorial product.

- This kind of two-dimensional DWT leads to a decomposition of approximation coefficients at level j in four components: the approximation at level j + 1 and the details in three orientations (horizontal, vertical, and diagonal).

- The following charts describe the basic decomposition and reconstruction steps for images.

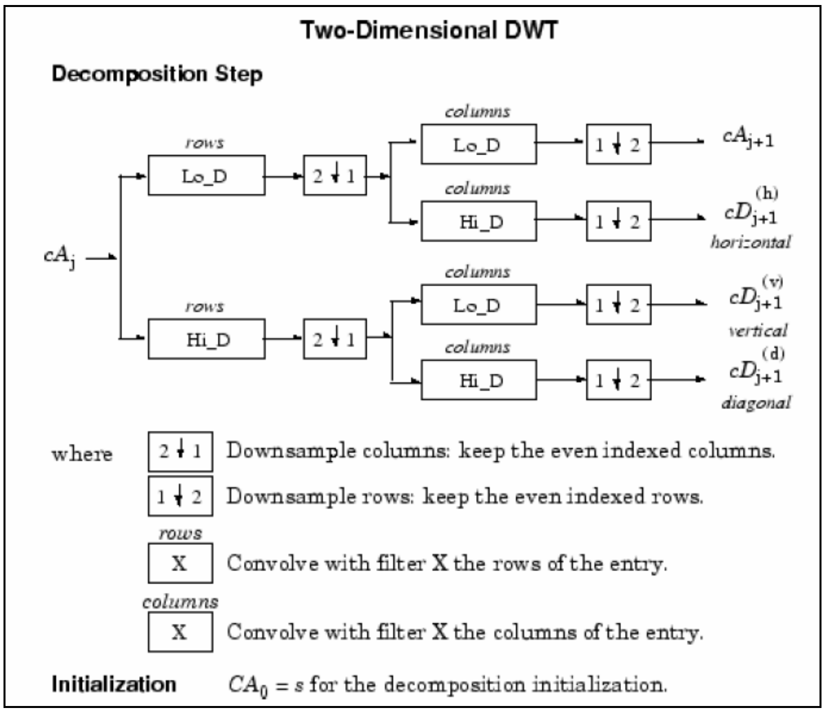

Two-Dimensional DWT
Decomposition Step
rows
Lo_D
2 ↓ 1
columns
Lo_D
1 ↓ 2
cA_{j+1}
columns
Hi_D
1 ↓ 2
cD^{(h)}_{j+1}
horizontal
cA_j
rows
Hi_D
2 ↓ 1
columns
Lo_D
1 ↓ 2
cD^{(v)}_{j+1}
vertical
columns
Hi_D
1 ↓ 2
cD^{(d)}_{j+1}
diagonal
where
2 ↓ 1 Downsample columns: keep the even indexed columns.
1 ↓ 2 Downsample rows: keep the even indexed rows.
rows X Convolve with filter X the rows of the entry.
columns X Convolve with filter X the columns of the entry.
Initialization CA_0 = s for the decomposition initialization.

Figure 7.10 – Two-Dimensional DWT

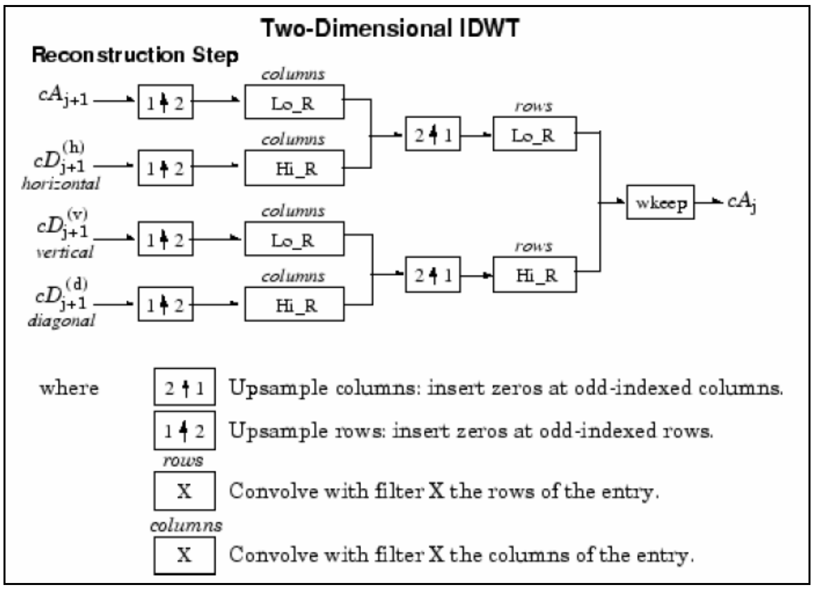

Two-Dimensional IDWT
Reconstruction Step
cA_{j+1}
1 ↑ 2
columns
Lo_R
cD^{(h)}_{j+1}
horizontal
1 ↑ 2
columns
Hi_R
2 ↑ 1
rows
Lo_R
wkeep
cA_j
cD^{(v)}_{j+1}
vertical
1 ↑ 2
columns
Lo_R
cD^{(d)}_{j+1}
diagonal
1 ↑ 2
columns
Hi_R
2 ↑ 1
rows
Hi_R
where
2 ↑ 1 Upsample columns: insert zeros at odd-indexed columns.
1 ↑ 2 Upsample rows: insert zeros at odd-indexed rows.
rows X Convolve with filter X the rows of the entry.
columns X Convolve with filter X the columns of the entry.

Figure 7.11 – Two-Dimensional IDWT

So, for $J = 2$, the two-dimensional wavelet tree has the following form as indicates the **Figure 7.12**:

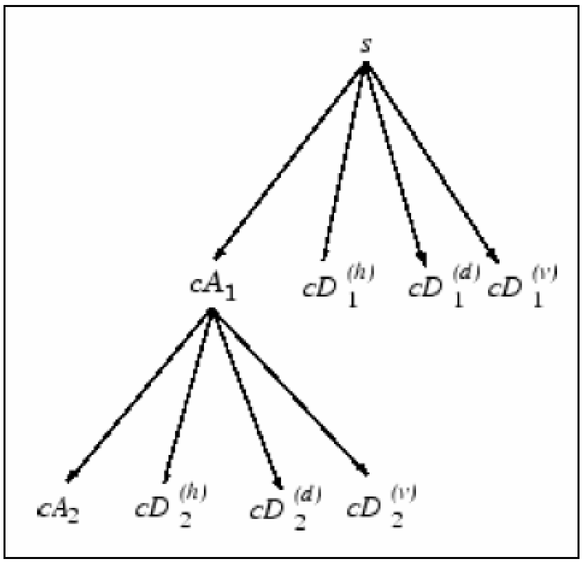


Figure 7.12 – Two-Dimensional wavelet tree for J = 2

Finally, let us mention this, for biorthogonal wavelets, the same algorithms hold but the decomposition filters on one hand and the reconstruction filters on the other hand are obtained from two distinct scaling functions associated with two multiresolution analyses in duality.

In this case, the filters for decomposition and reconstruction are, in general, of different odd lengths. This situation occurs, for example, for "splines" biorthogonal wavelets used in the toolbox. By zero-padding, the four filters can be extended in such a way this they will have the same even length.

8 RESULTS

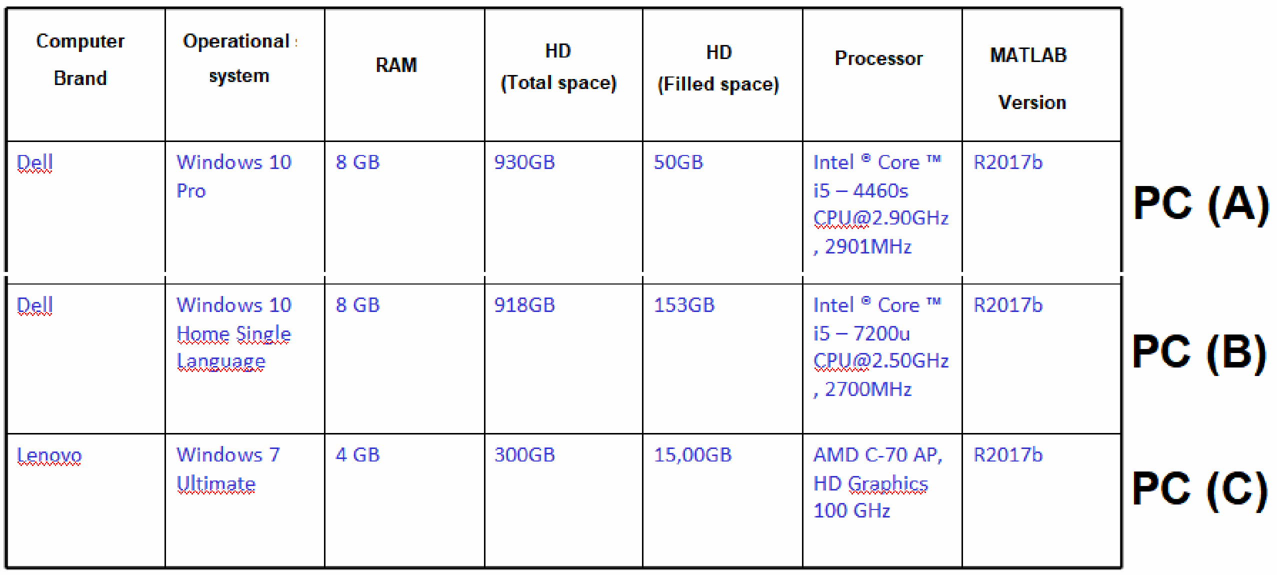
This Chapter presents numerical data of a specific PCB inspection method utilized to the simulation, and summarizes the results of times and image inspection classification about the 11 simulated bare PCB's production samples. Is done as well an analysis about the time, classifying and compression ratio before and after implementation of the Fast Wavelet Transform Algorithm.

8.1 RESULTS BEFORE IMPLEMENTATION OF FWT ALGORITHM

Was realized a study in three different computers, with different configurations, to the calculus of image time processing in two models of bare printed circuit boards (PCB – without defects) in relation to the sample with three models of printed circuit boards (PCB – with defects) to be compared in the image subtraction algorithm without the implementation of the FWT algorithm.

- The computer configurations are given in the following table:

TABLE 8.1 – COMPUTER "A, B, C" CONFIGURATION



Computer Brand	Operational system	RAM	HD (Total space)	HD (Filled space)	Processor	MATLAB Version	
Dell	Windows 10 Pro	8 GB	930GB	50GB	Intel ® Core ™ i5 – 4460s CPU@2.90GHz , 2901MHz	R2017b	PC (A)
Dell	Windows 10 Home Single Language	8 GB	918GB	153GB	Intel ® Core ™ i5 – 7200u CPU@2.50GHz , 2700MHz	R2017b	PC (B)
Lenovo	Windows 7 Ultimate	4 GB	300GB	15,00GB	AMD C-70 AP, HD Graphics 100 GHz	R2017b	PC (C)

- The images of PCB's samples without defects are the following:

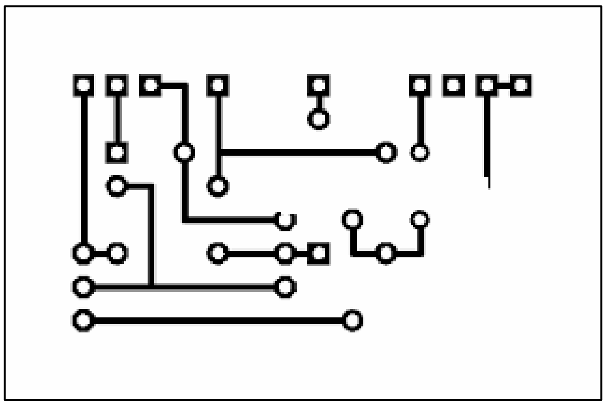


Figure 8.1 – Sample 1

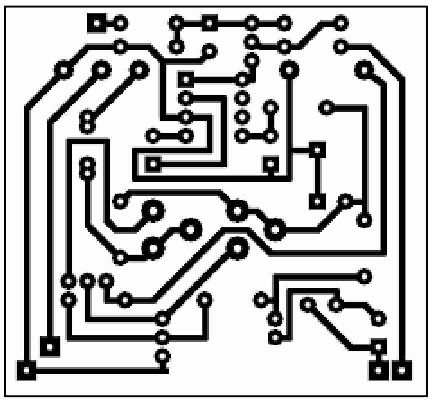


Figure 8.2 – Sample 2

- The PCB's sample images with defects to the sample 1 are the following:

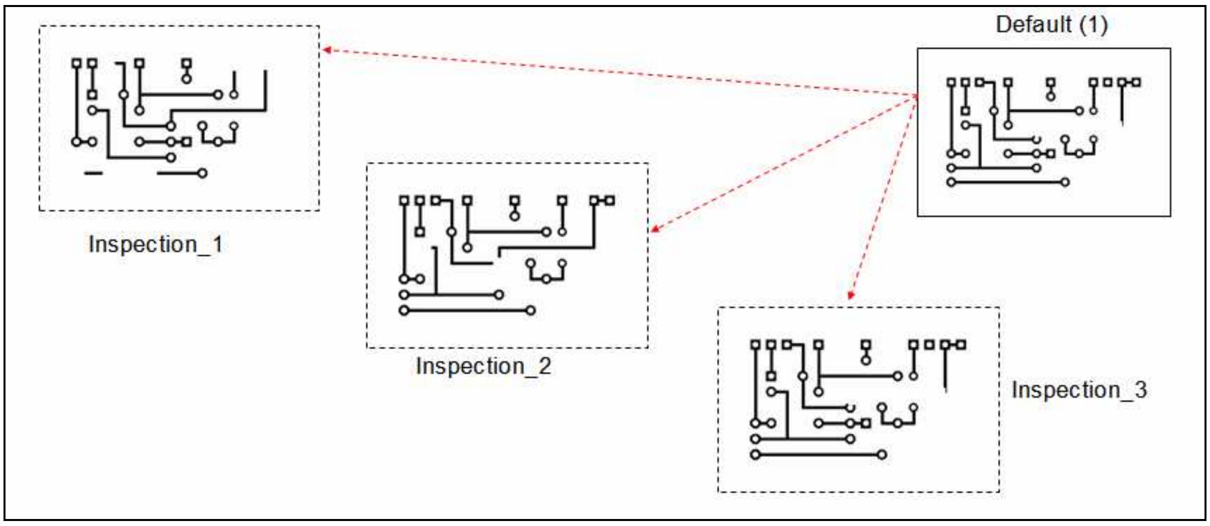


Figure 8.3 – PCB's defected images to the sample 1

- The PCB's sample images with defects to the sample 2 are the following:

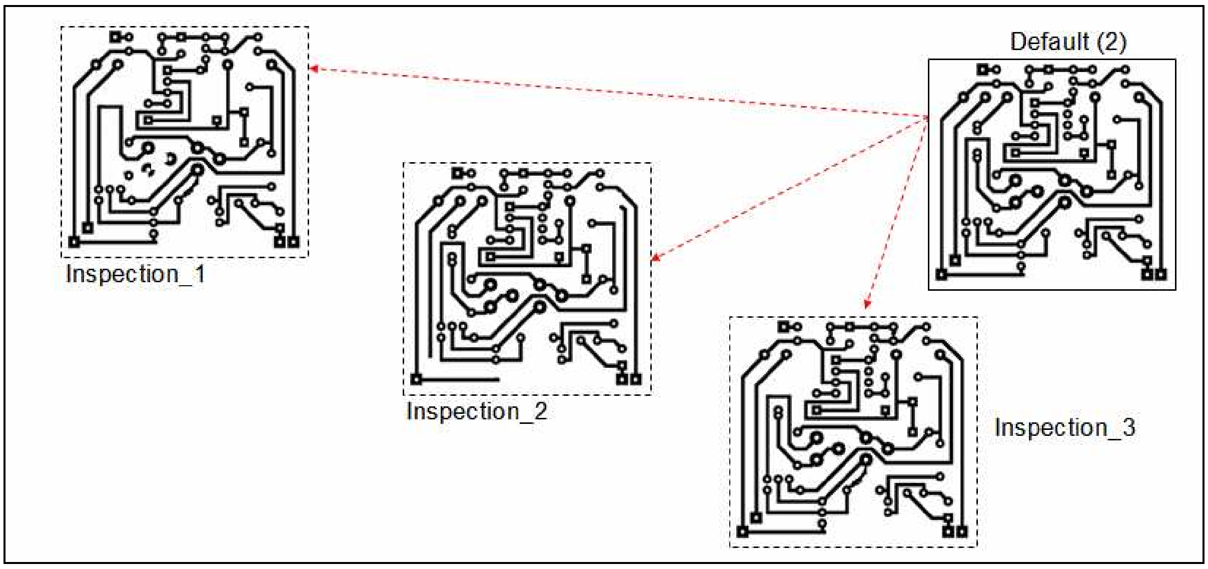


Figure 8.4 – PCB's defected images to the sample 2

- The results in relation to the image time processing to the sample 1, during the inspection 1, 2 and 3 are the graphics bellow:

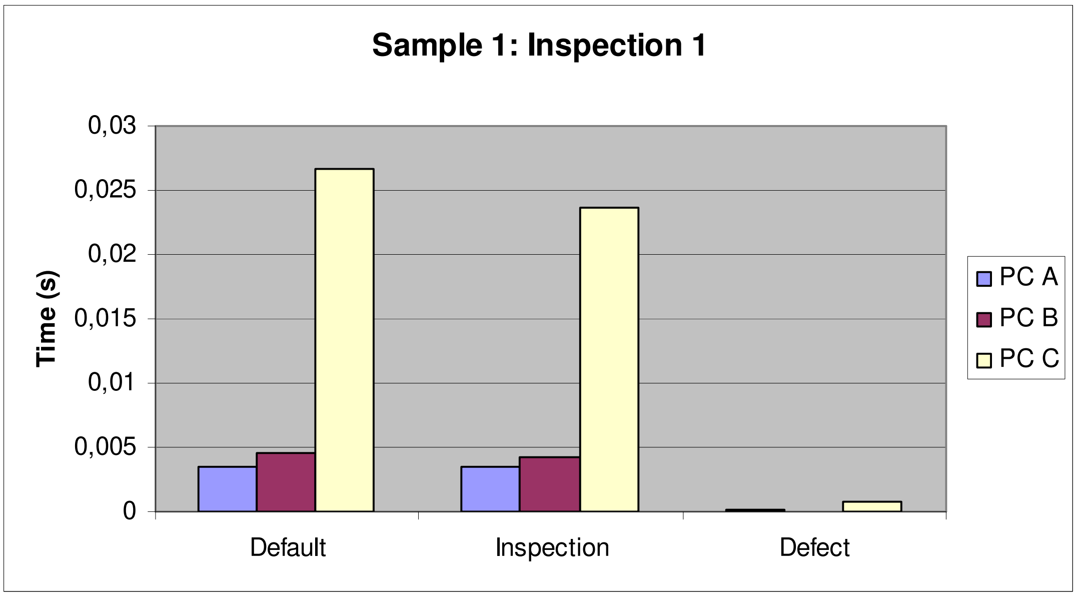


Figure 8.5 – Graph sample 1, inspection 1

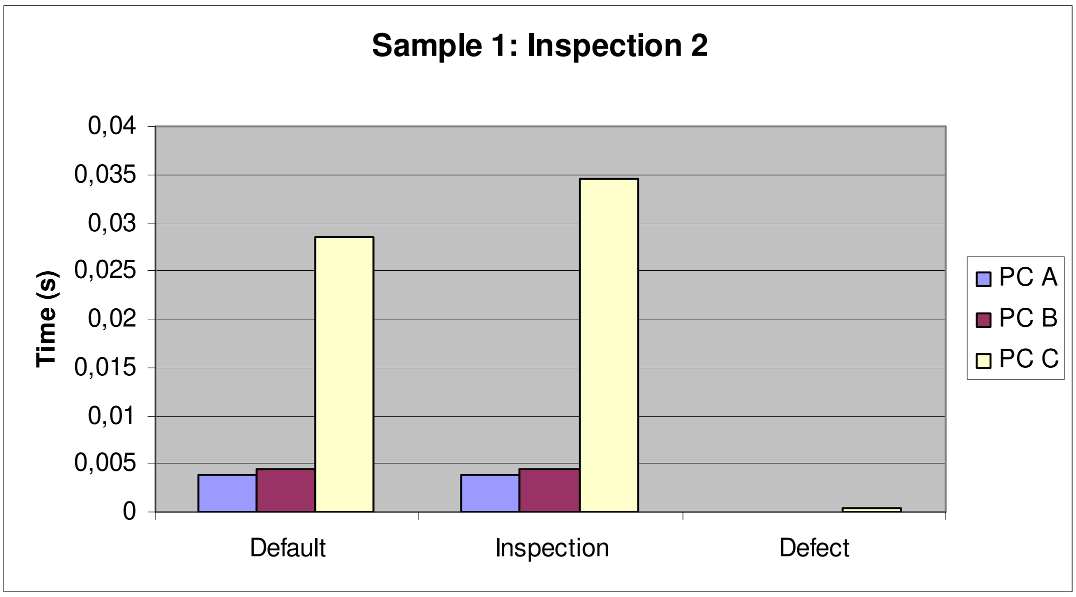


Figure 8.6 – Graph sample 1, inspection 2

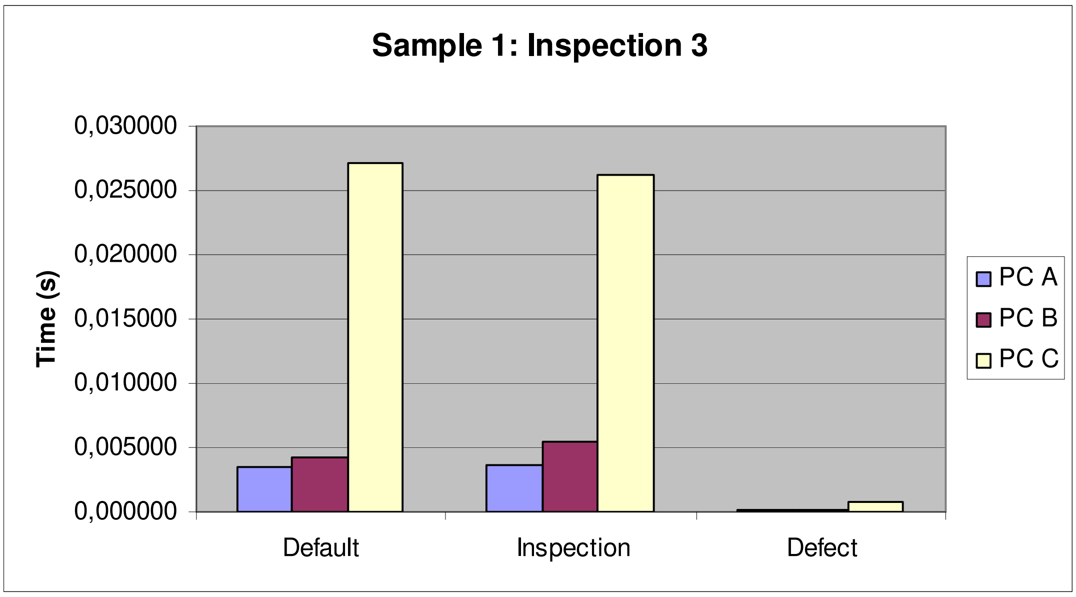


Figure 8.7 – Graph sample 1, inspection 3

- The results in relation to the image time processing to the sample 2, during the inspection 1, 2 and 3 are the graphics bellow:

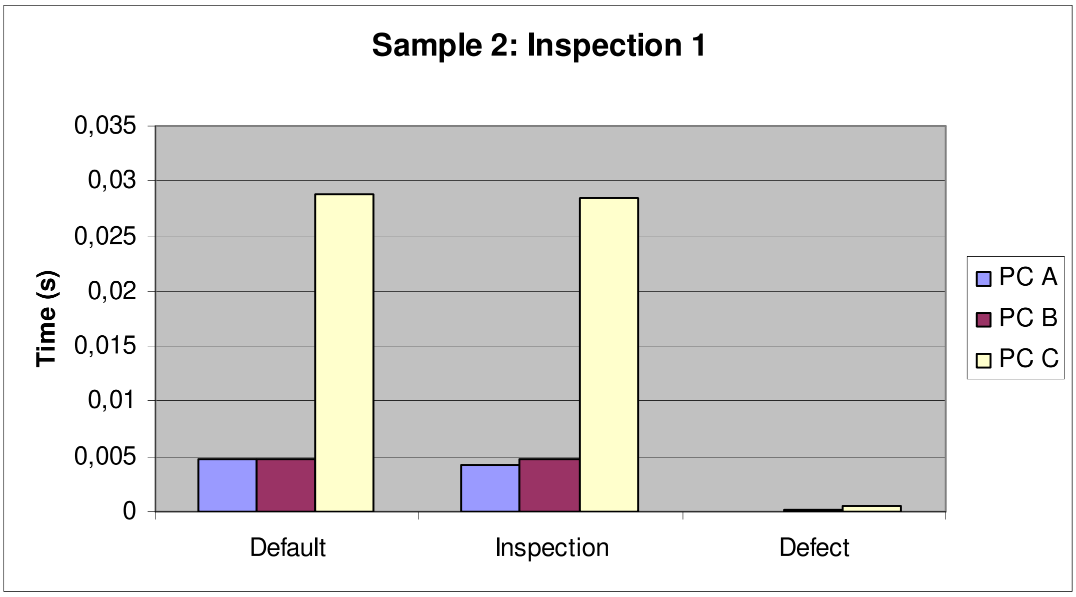


Figure 8.8 – Graph sample 2, inspection 1

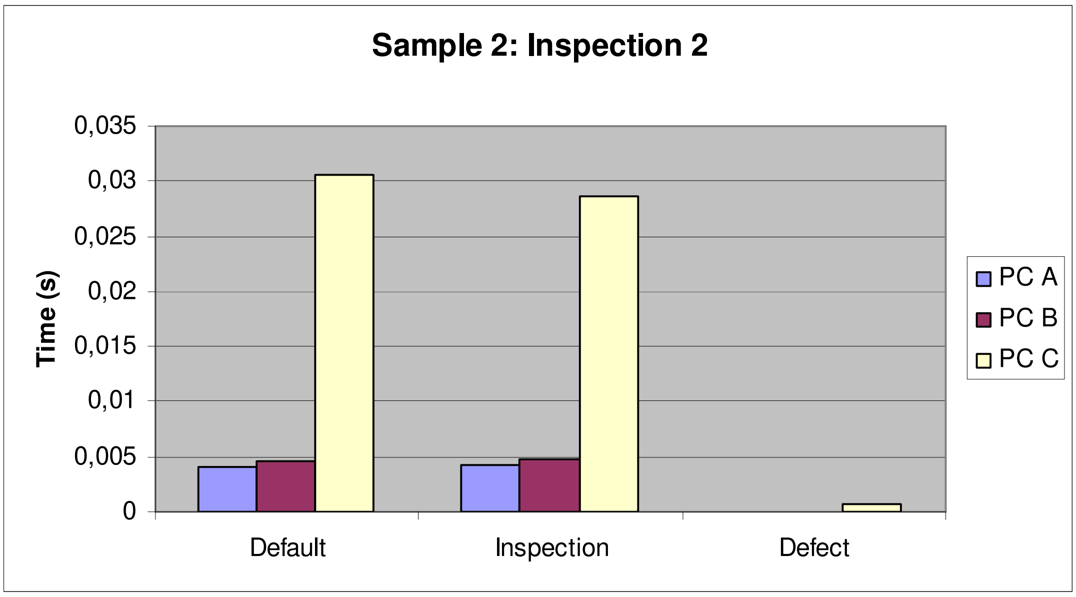


Figure 8.9 – Graph sample 2, inspection 2

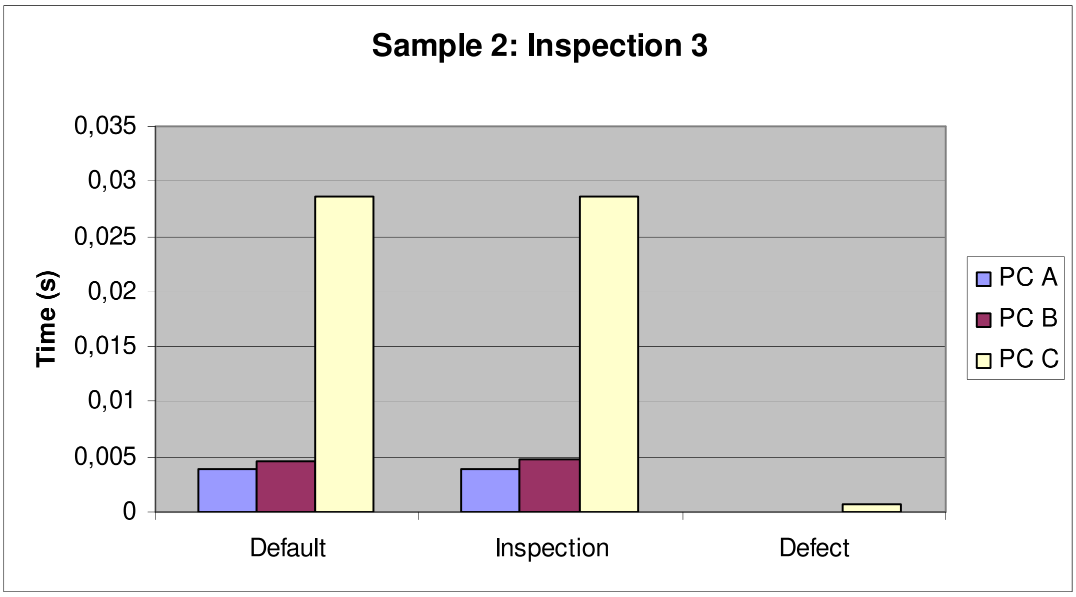


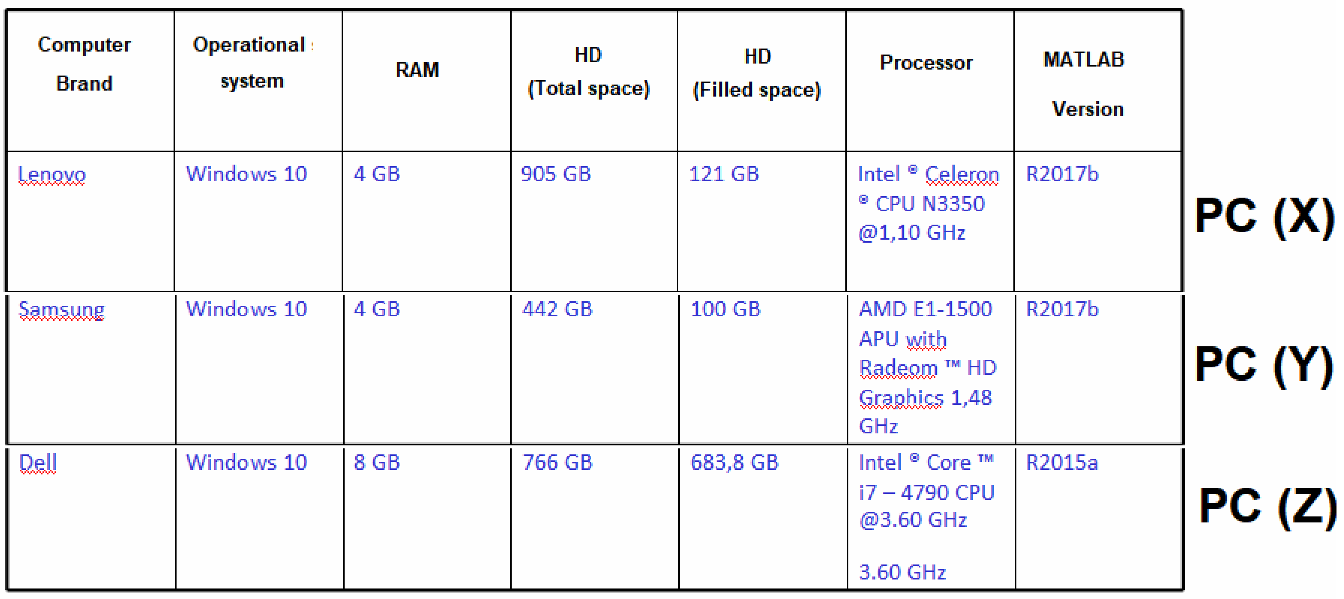
Figure 8.10 – Graph sample 2, inspection 3

8.2 RESULTS BEFORE IMPLEMENTATION OF FWT ALGORITHM: FOR SCANNED ALGORITHM PCB IMAGES

Was realized a study in three different computers, called PC "X", PC "Y" and PC "Z", with different configurations, to the calculus of image time processing in two models of scanned bare printed circuit boards (PCB – without defects) in relation to the sample with three models of scanned printed circuit boards (PCB – with defects) to be compared in the image subtraction algorithm without the implementation of the FWT algorithm.

- The computer configurations of PC "X", PC "Y" and PC "Z" are given in the following table:

TABLE 8.2 – COMPUTER "X, Y, Z" CONFIGURATIONS



Computer Brand	Operational system	RAM	HD (Total space)	HD (Filled space)	Processor	MATLAB Version	
Lenovo	Windows 10	4 GB	905 GB	121 GB	Intel ® Celeron ® CPU N3350 @1,10 GHz	R2017b	PC (X)
Samsung	Windows 10	4 GB	442 GB	100 GB	AMD E1-1500 APU with Radeom ™ HD Graphics 1,48 GHz	R2017b	PC (Y)
Dell	Windows 10	8 GB	766 GB	683,8 GB	Intel ® Core ™ i7 – 4790 CPU @3.60 GHz 3.60 GHz	R2015a	PC (Z)

- The images of PCB's samples without defects are the following:

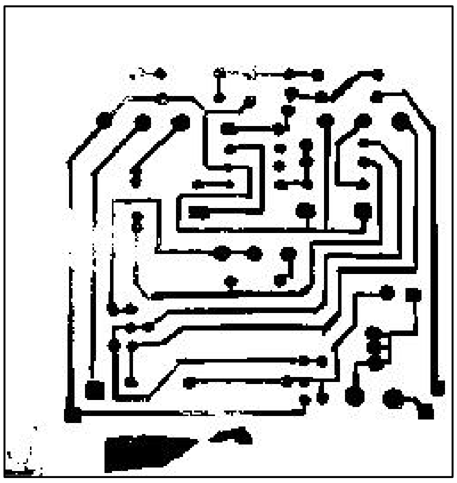


Figure 8.11 – Sample 1

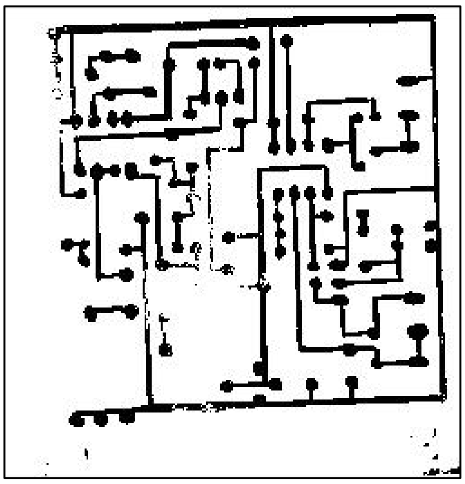


Figure 8.12 – Sample 2

- The scanned PCB's sample images with defects to the sample 1 are the following:

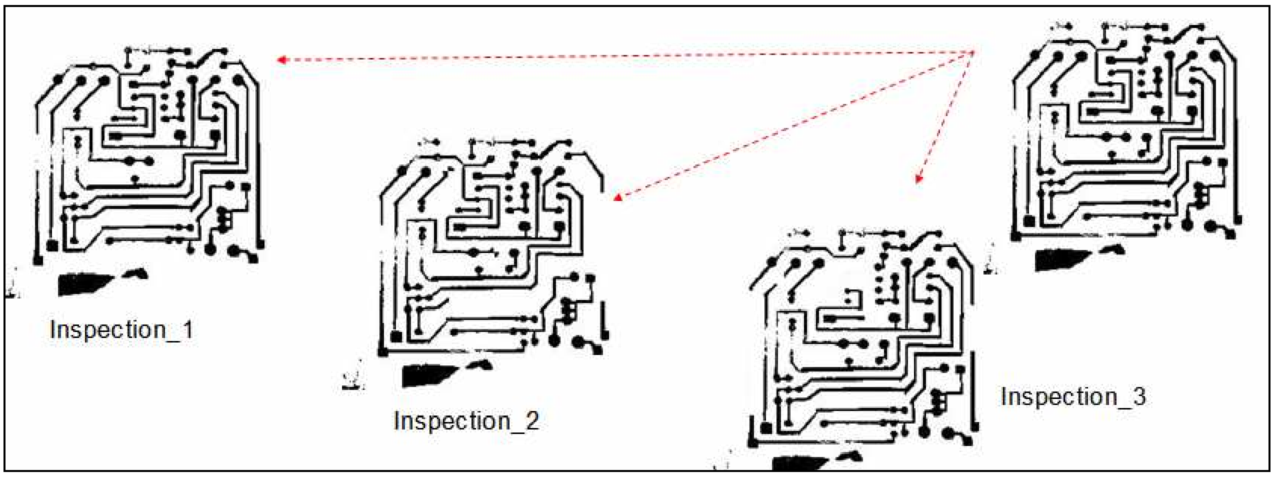


Figure 8.13 – PCB's defected images to the sample 1

- The scanned PCB's sample images with defects to the sample 2 are the following:

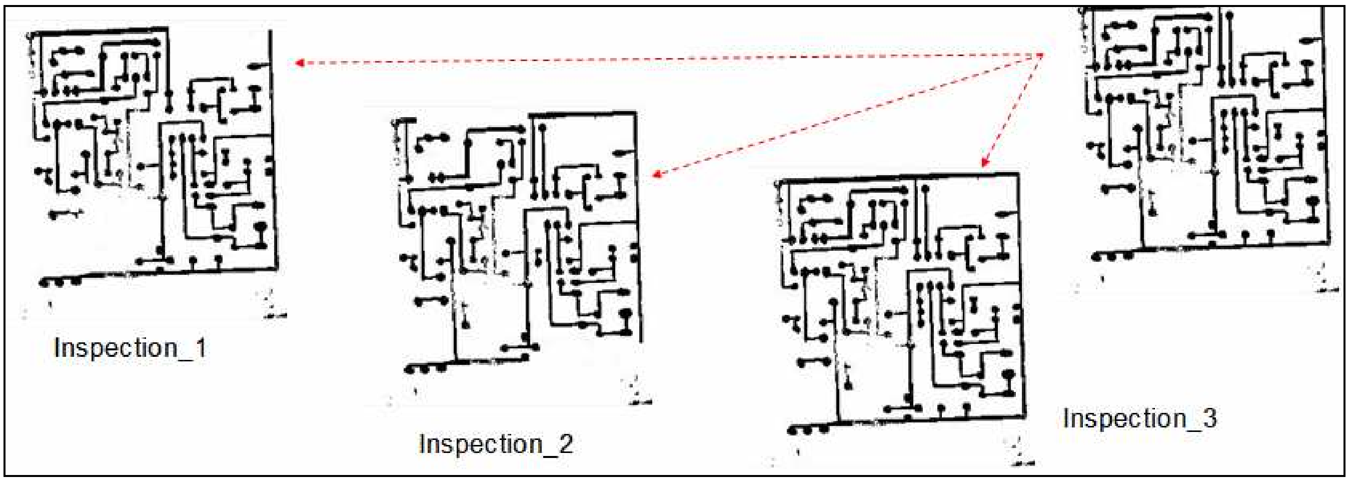


Figure 8.14 – PCB's defected images to the sample 2

- The results in relation to the scanned image time processing to the sample 1, during the inspection 1, 2 and 3 are the graphics bellow:

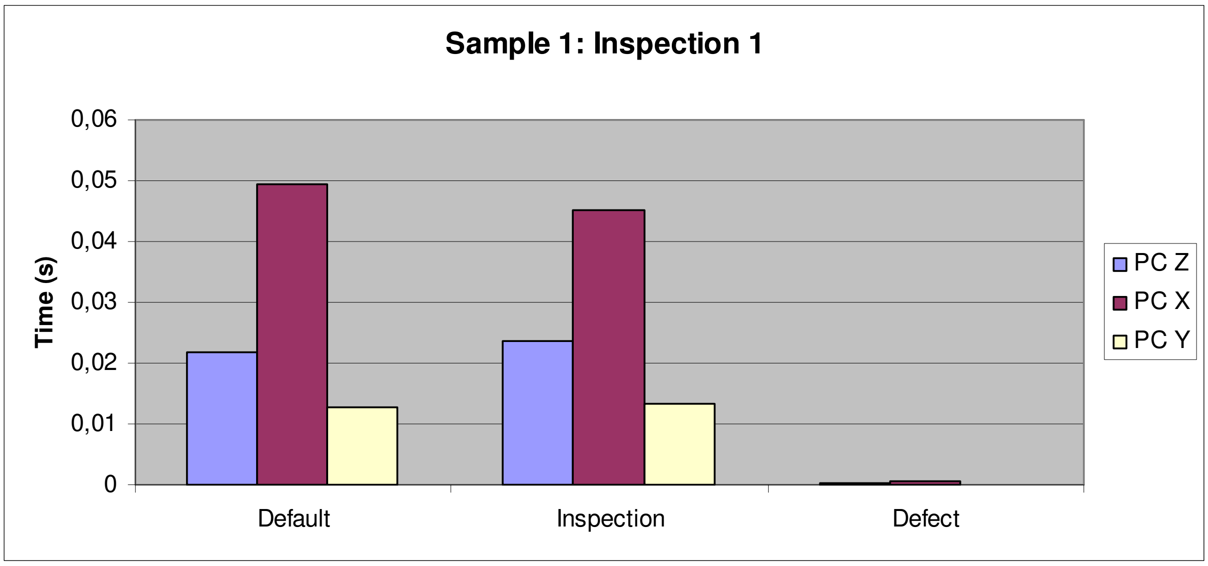


Figure 8.15 – Graph sample 1, inspection 1

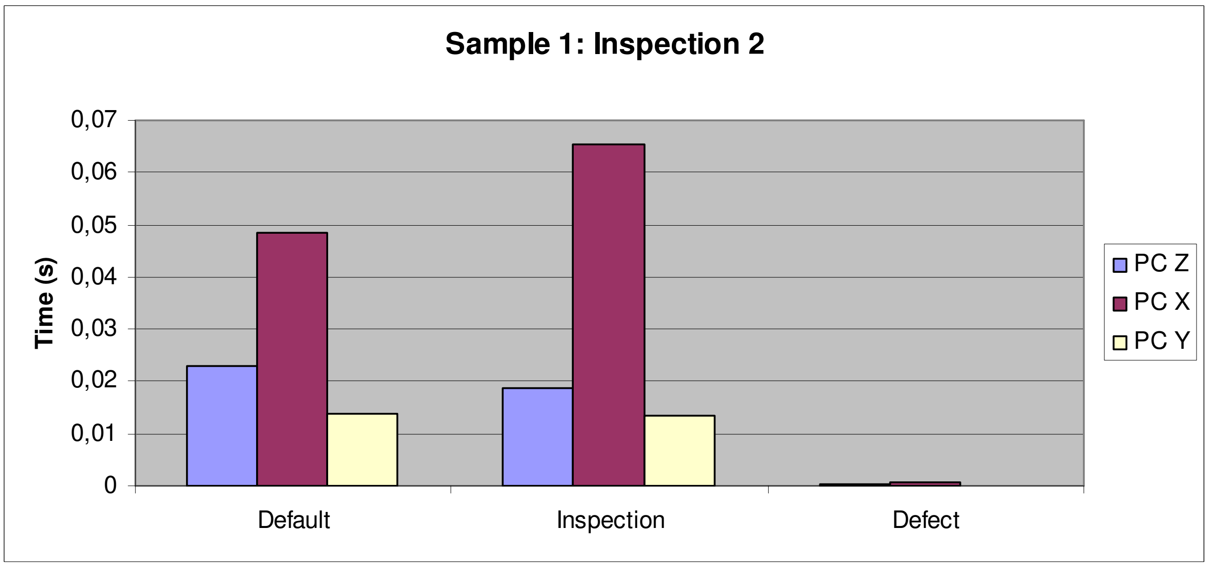


Figure 8.16 – Graph sample 1, inspection 2

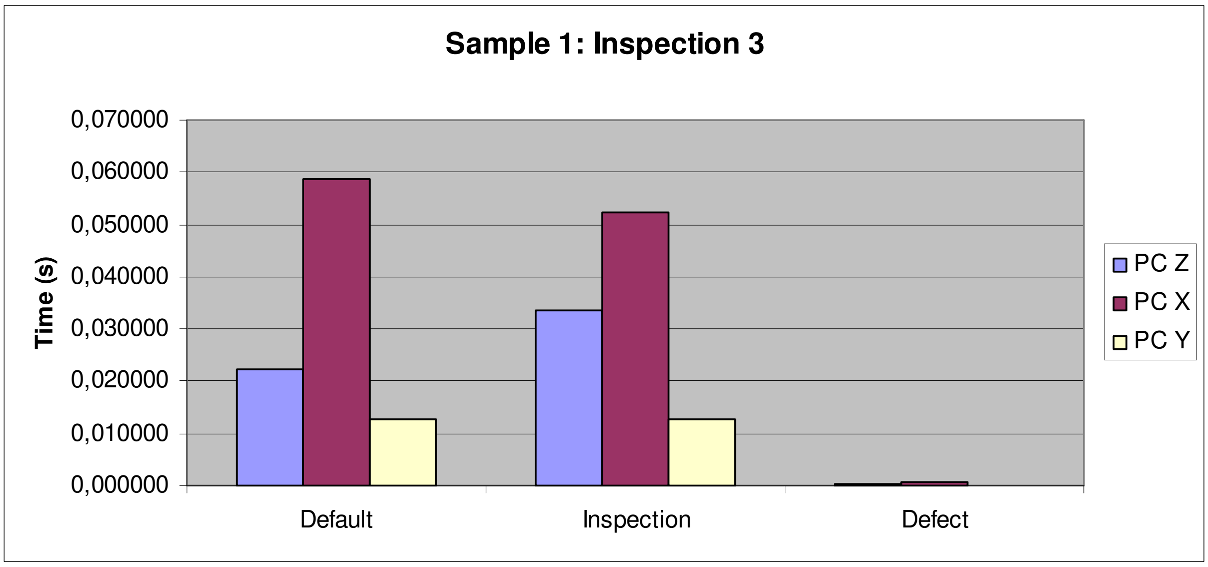


Figure 8.17 – Graph sample 1, inspection 3

- The results in relation to the scanned image time processing to the sample 2, during the inspection 1, 2 and 3 are the graphics bellow:

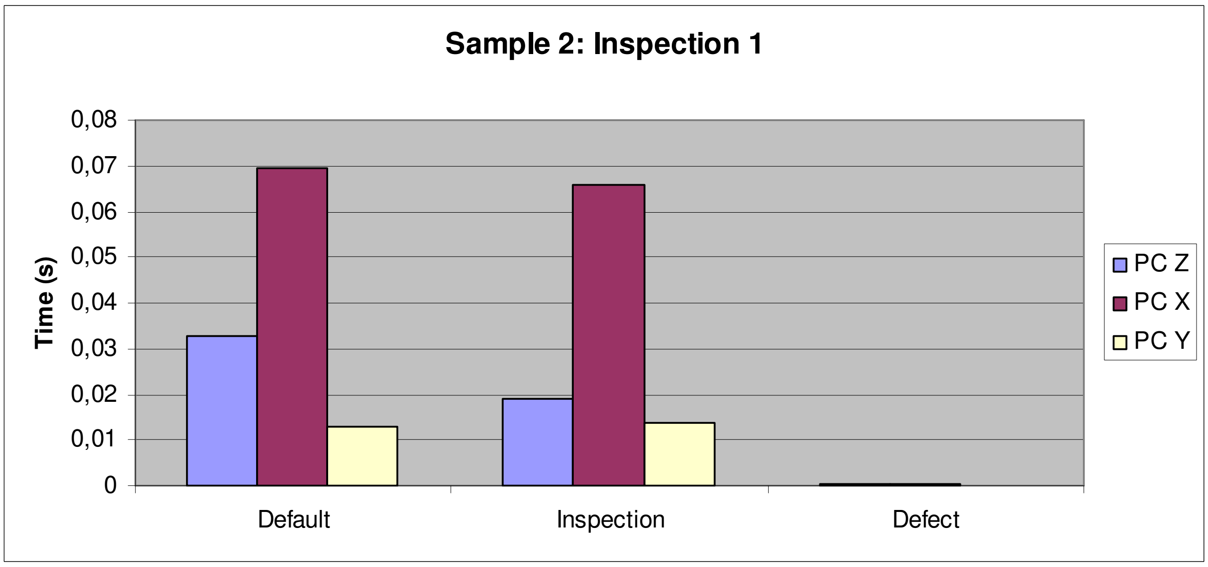


Figure 8.18 – Graph sample 2, inspection 1

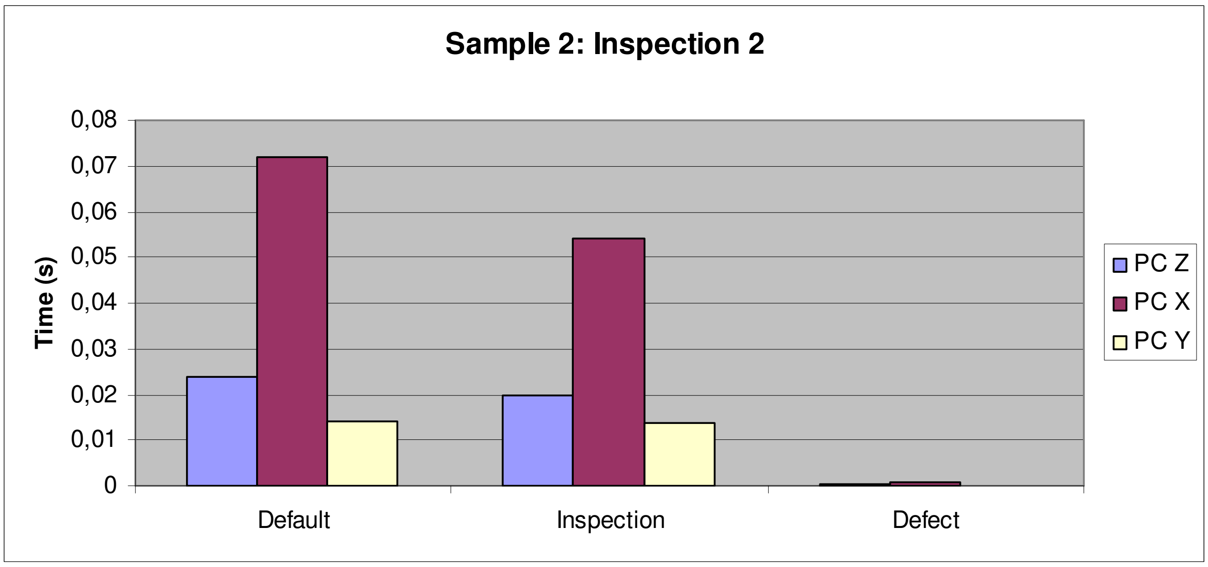


Figure 8.19 – Graph sample 2, inspection 2

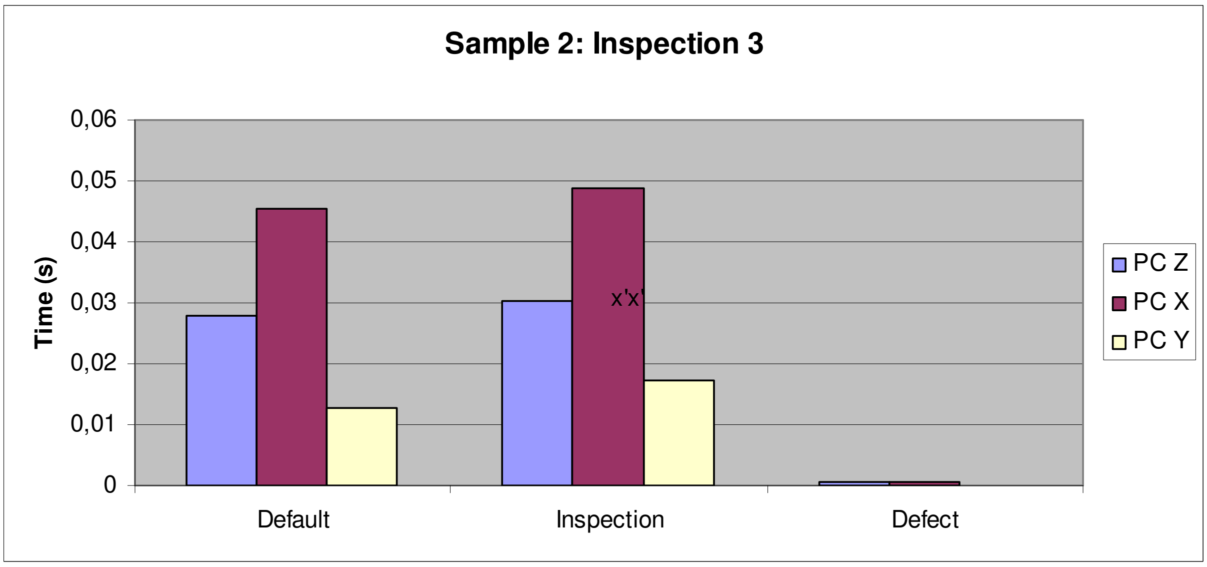


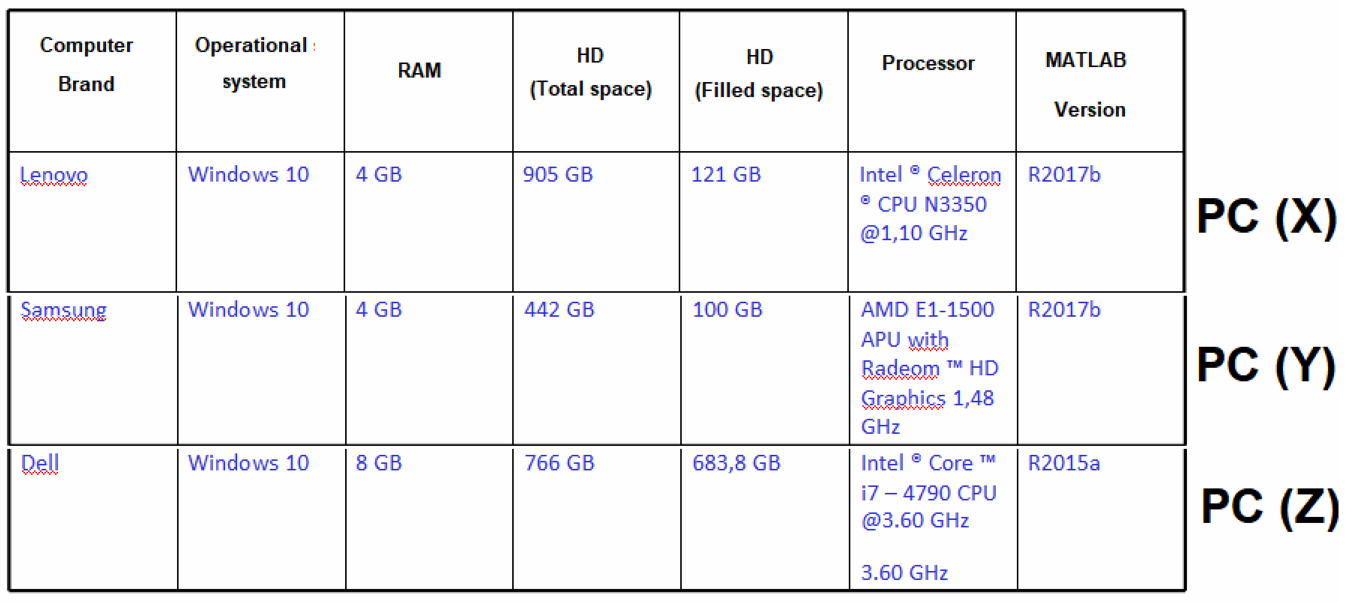
Figure 8.20 – Graph sample 2, inspection 3

8.3 RESULTS AFTER IMPLEMENTATION OF FWT ALGORITHM

Was realized a study in three different computers, with different configurations, to the calculus of image time processing in two models of bare printed circuit boards (PCB – without defects) in relation to the sample with three models of printed circuit boards (PCB – with defects) to be compared in the image subtraction algorithm with the implementation of the FWT algorithm.

- The computer configurations are given in the following table:

TABLE 8.3 – COMPUTER "X, Y, Z" CONFIGURATIONS



Computer Brand	Operational system	RAM	HD (Total space)	HD (Filled space)	Processor	MATLAB Version	
Lenovo	Windows 10	4 GB	905 GB	121 GB	Intel ® Celeron ® CPU N3350 @1,10 GHz	R2017b	PC (X)
Samsung	Windows 10	4 GB	442 GB	100 GB	AMD E1-1500 APU with Radeom ™ HD Graphics 1,48 GHz	R2017b	PC (Y)
Dell	Windows 10	8 GB	766 GB	683,8 GB	Intel ® Core ™ i7 – 4790 CPU @3.60 GHz 3.60 GHz	R2015a	PC (Z)

- The images of PCB's samples without defects are the following:

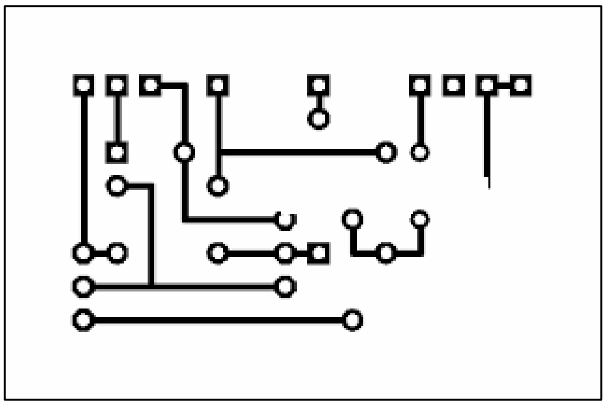


Figure 8.21 – Sample 1

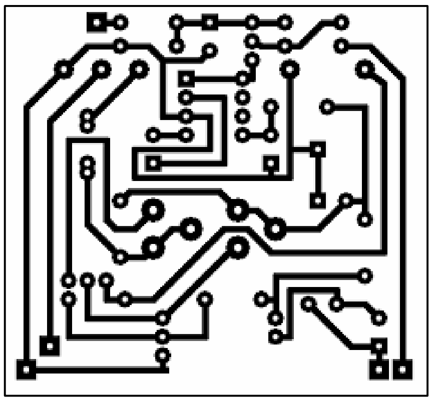


Figure 8.22 – Sample 2

- The PCB's sample images with defects to the sample 1 are the following:

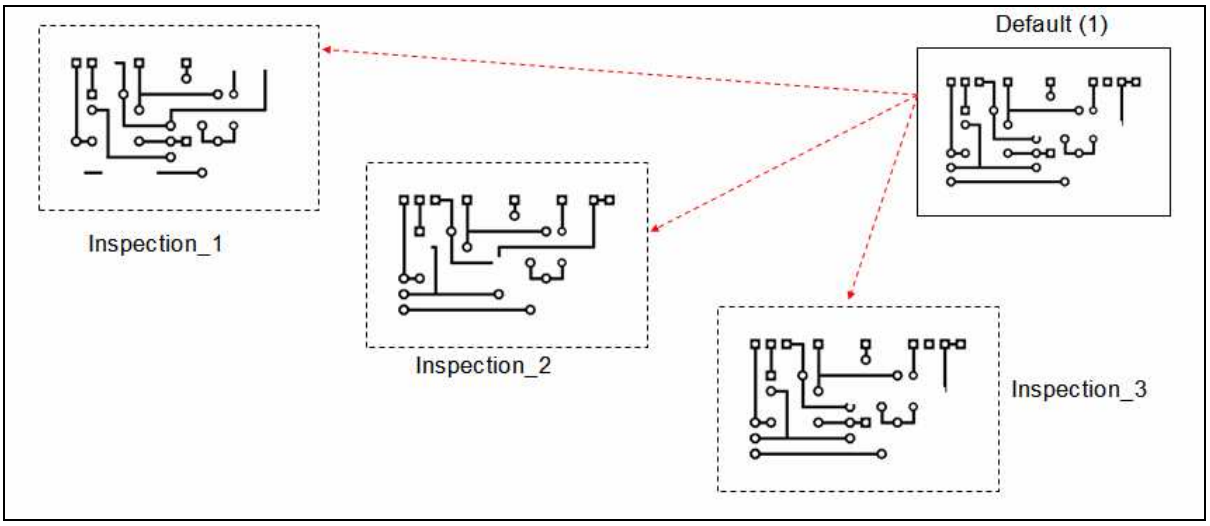


Figure 8.23 – PCB's defected images to the sample 1

- The PCB's sample images with defects to the sample 2 are the following:

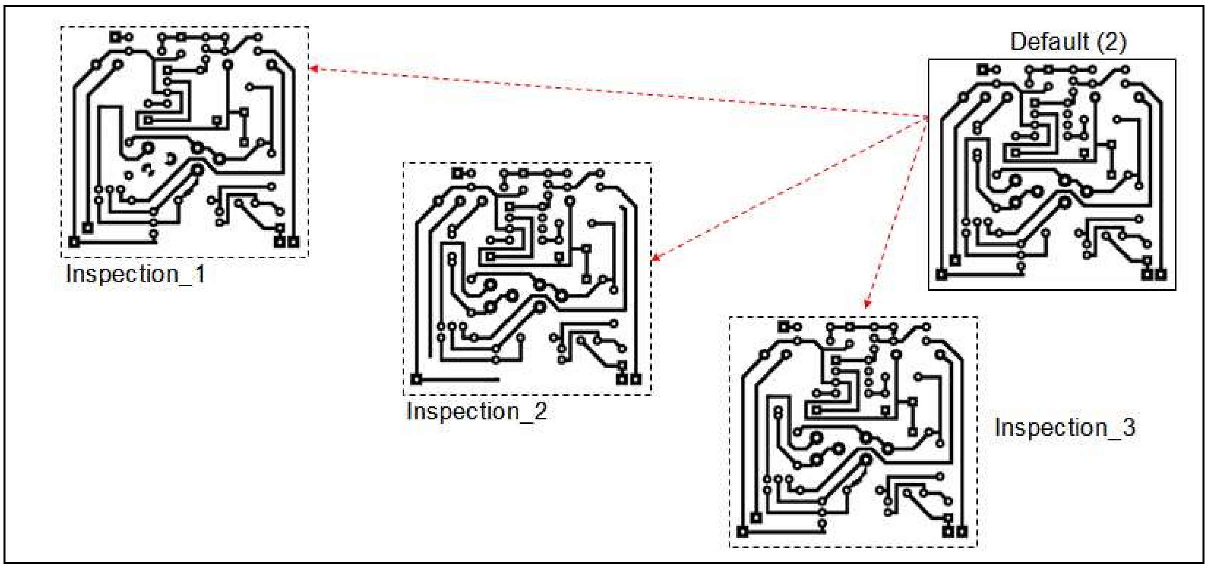


Figure 8.24 – PCB's defected images to the sample 2

- The results in relation to the image time processing to the sample 1, during the inspection 1, 2 and 3 are the graphics bellow:

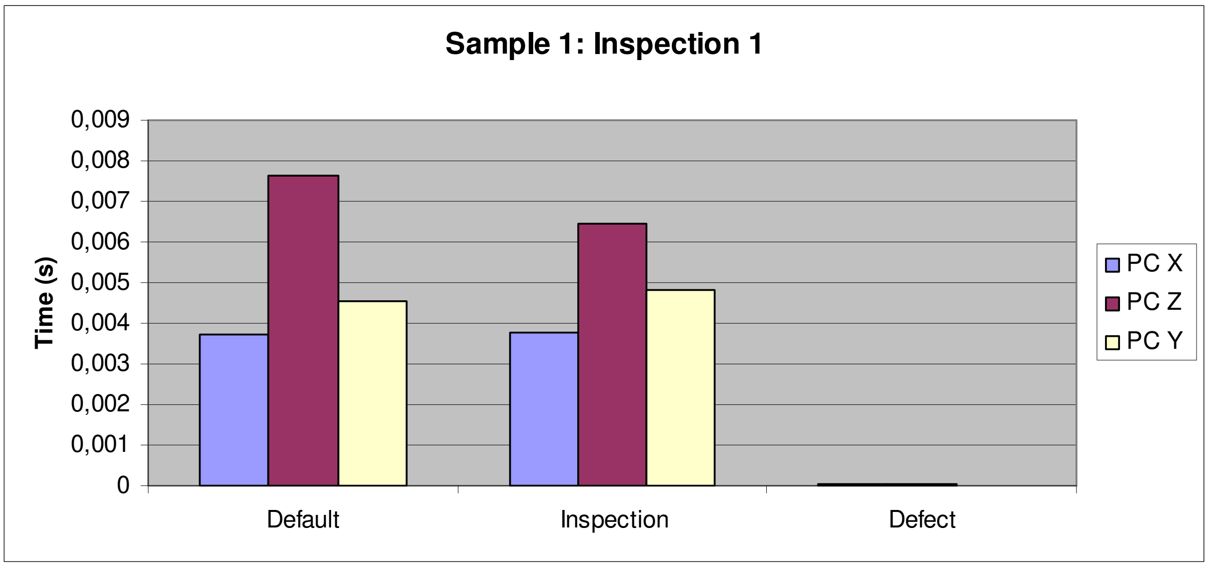


Figure 8.25 – Graph sample 1, inspection 1

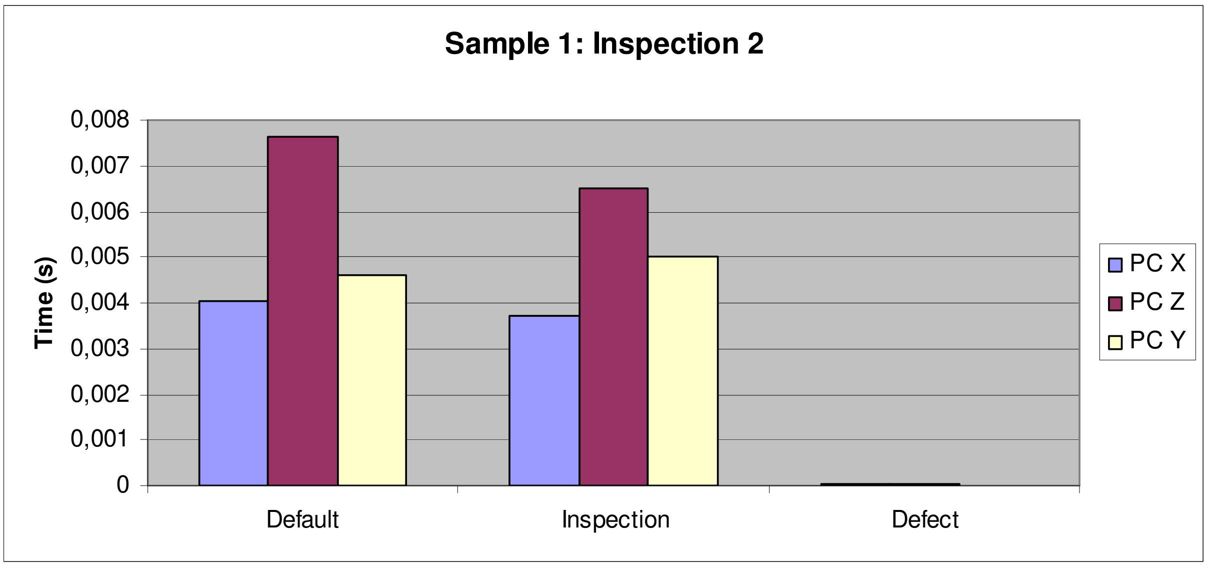


Figure 8.26 – Graph sample 1, inspection 2

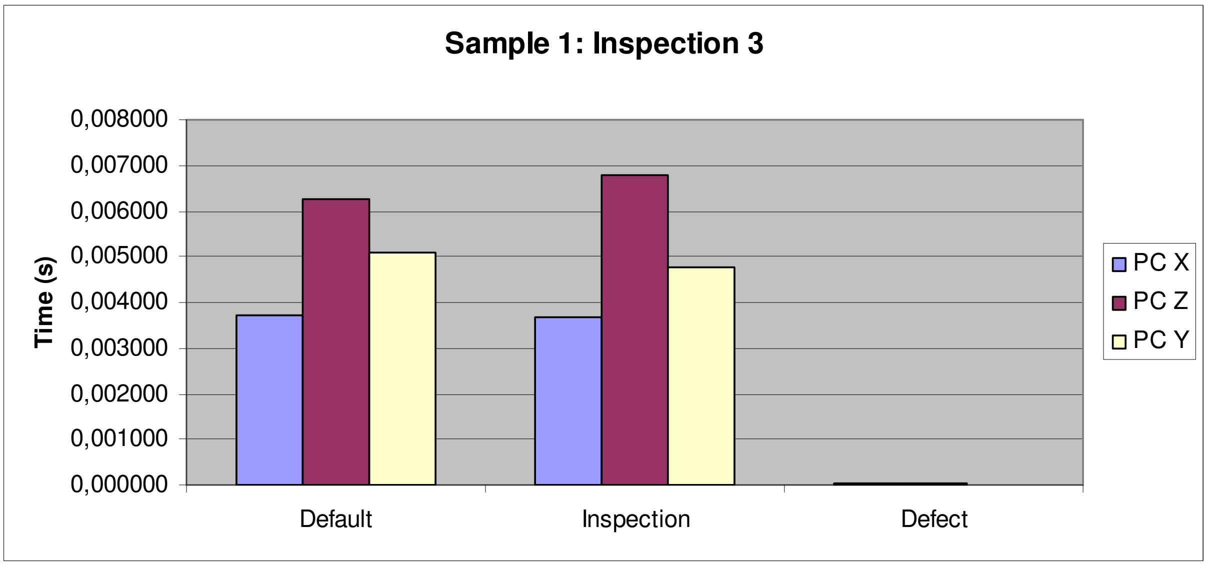


Figure 8.27 – Graph sample 1, inspection 3

- The results in relation to the image time processing to the sample 2, during the inspection 1, 2 and 3 are the graphics bellow:

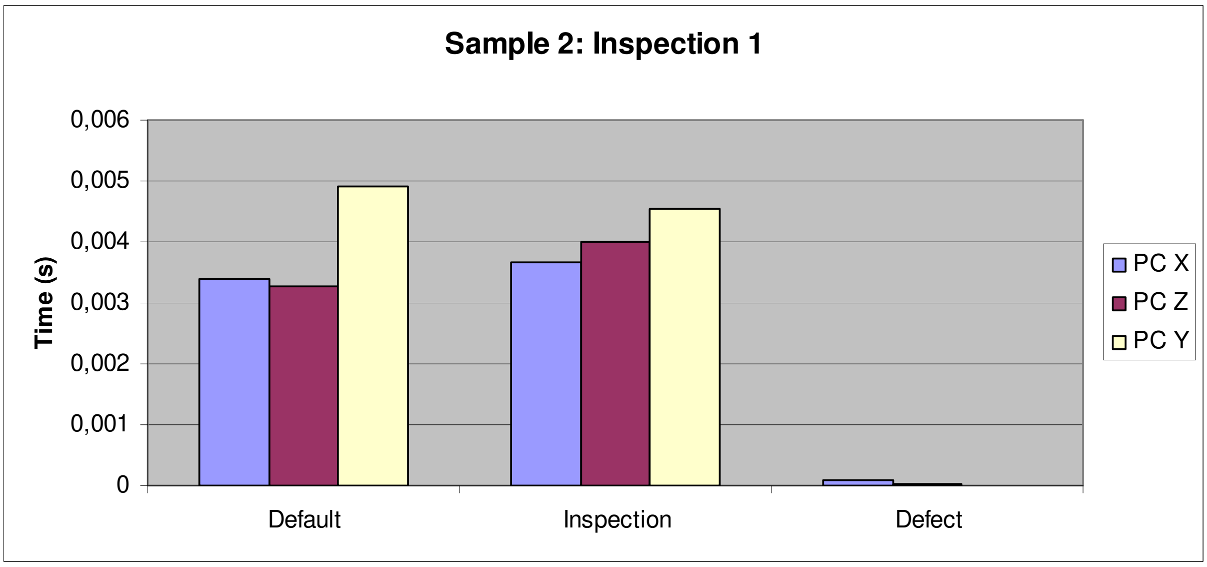


Figure 8.28 – Graph sample 2, inspection 1

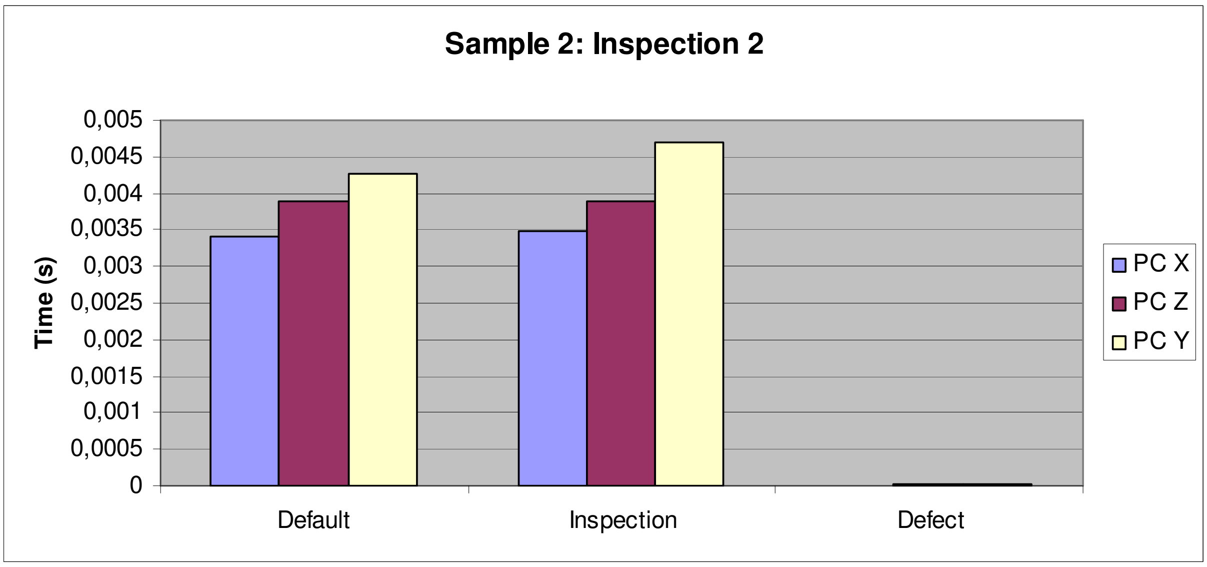


Figure 8.29 – Graph sample 2, inspection 2

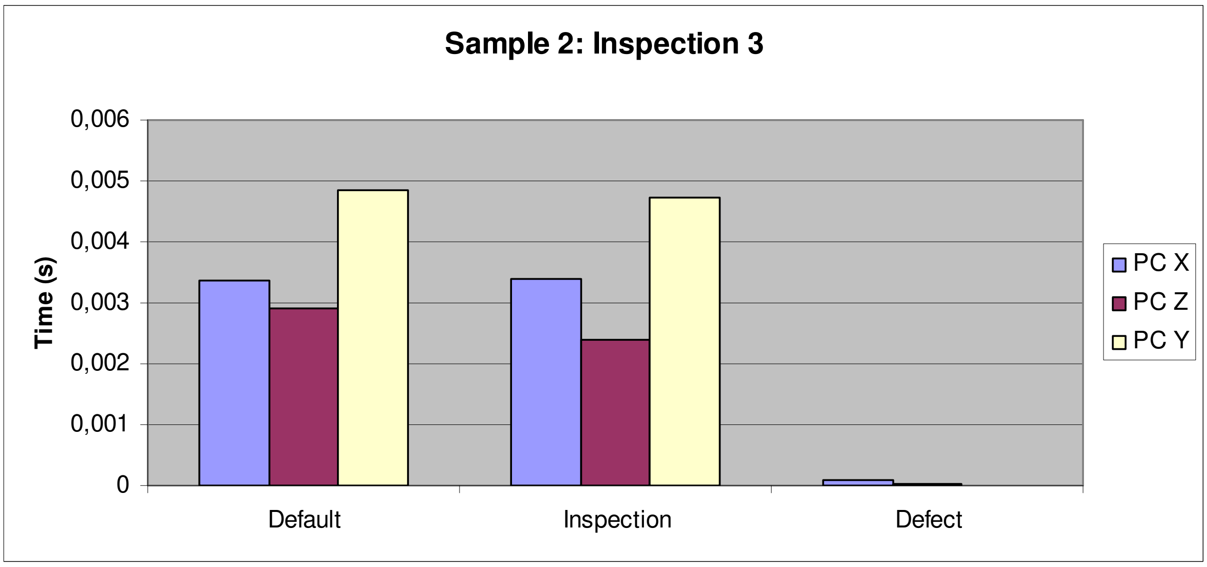


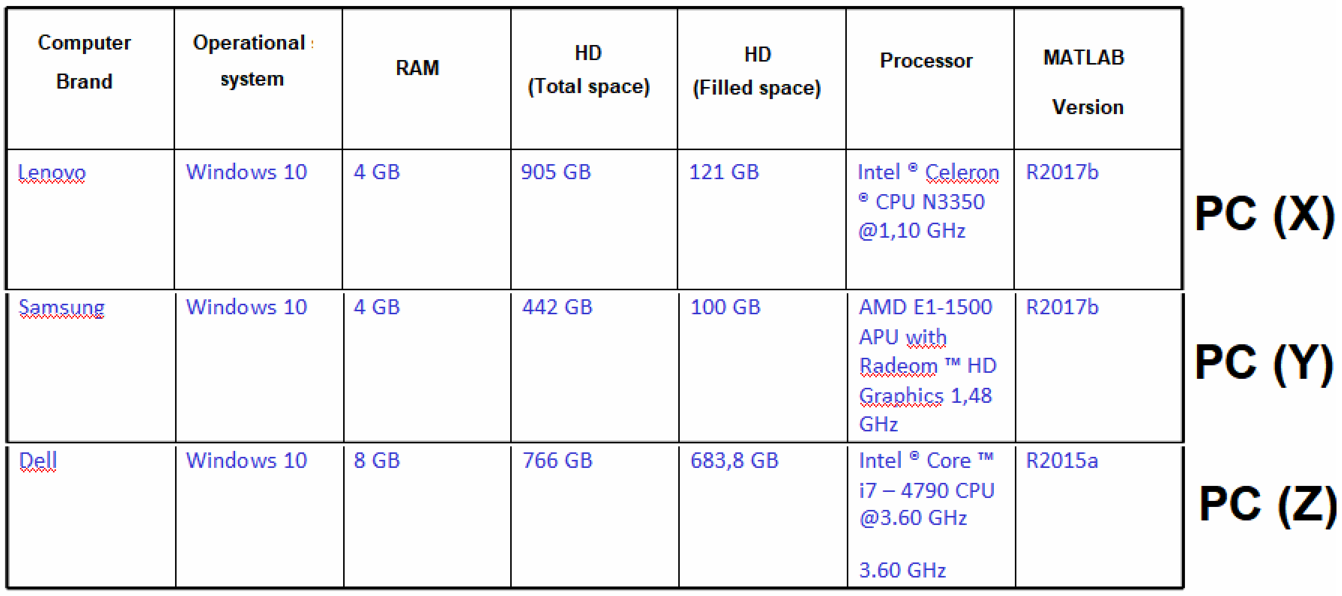
Figure 8.30 – Graph sample 2, inspection 3

8.4 RESULTS AFTER IMPLEMENTATION OF FWT ALGORITHM: FOR SCANNED ALGORITHM PCB IMAGES

Was realized a study in three different computers, called PC “X”, PC “Y” and PC “Z”, with different configurations, to the calculus of image time processing in two models of scanned bare printed circuit boards (PCB – without defects) in relation to the sample with three models of scanned printed circuit boards (PCB – with defects) to be compared in the image subtraction algorithm with the implementation of the FWT algorithm.

- The computer configurations of PC “X”, PC “Y” and PC “Z” are given in the following table:

TABLE 8.4 – COMPUTER "X, Y, Z" CONFIGURATIONS



Computer Brand	Operational system	RAM	HD (Total space)	HD (Filled space)	Processor	MATLAB Version	
Lenovo	Windows 10	4 GB	905 GB	121 GB	Intel ® Celeron ® CPU N3350 @1,10 GHz	R2017b	PC (X)
Samsung	Windows 10	4 GB	442 GB	100 GB	AMD E1-1500 APU with Radeom ™ HD Graphics 1,48 GHz	R2017b	PC (Y)
Dell	Windows 10	8 GB	766 GB	683,8 GB	Intel ® Core ™ i7 – 4790 CPU @3.60 GHz 3.60 GHz	R2015a	PC (Z)

- The images of PCB's samples without defects are the following:

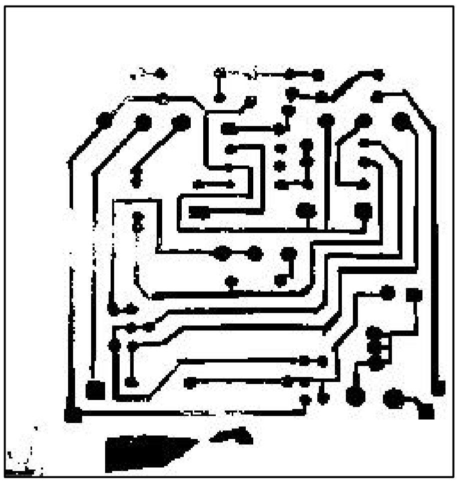


Figure 8.31 – Sample 1

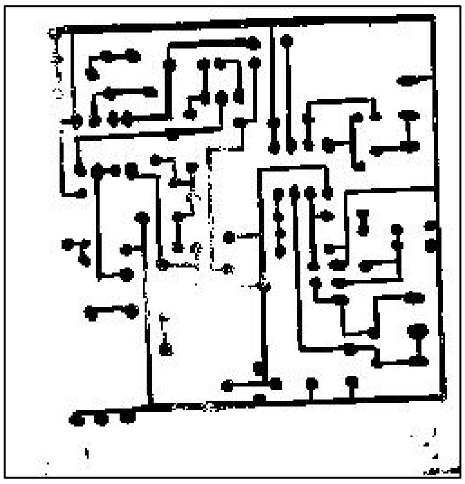


Figure 8.32 – Sample 2

- The scanned PCB's sample images with defects to the sample 1 are the following:

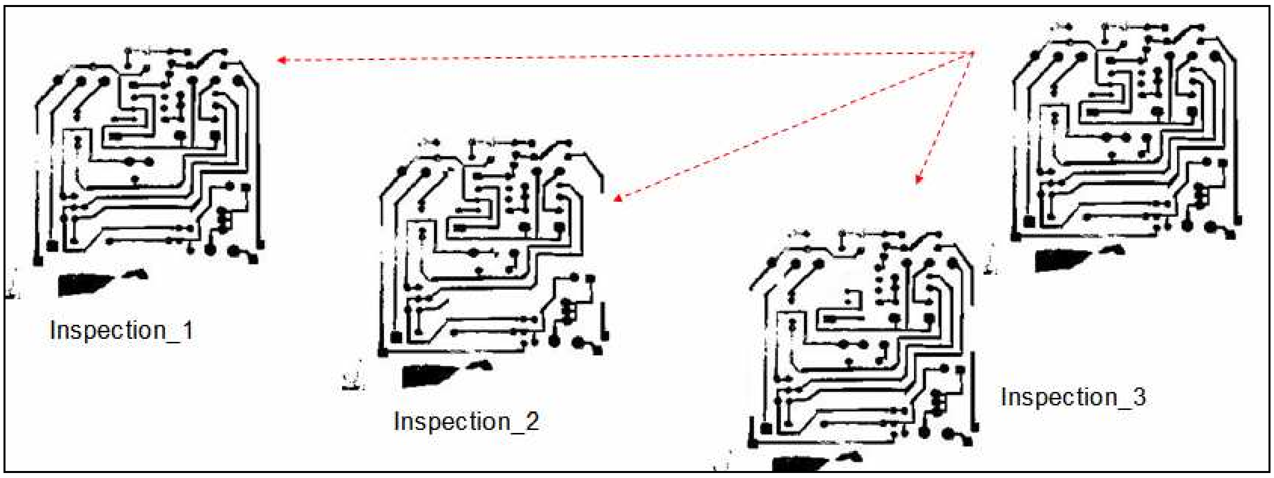


Figure 8.33 – PCB's defected images to the sample 1

- The scanned PCB's sample images with defects to the sample 2 are the following:

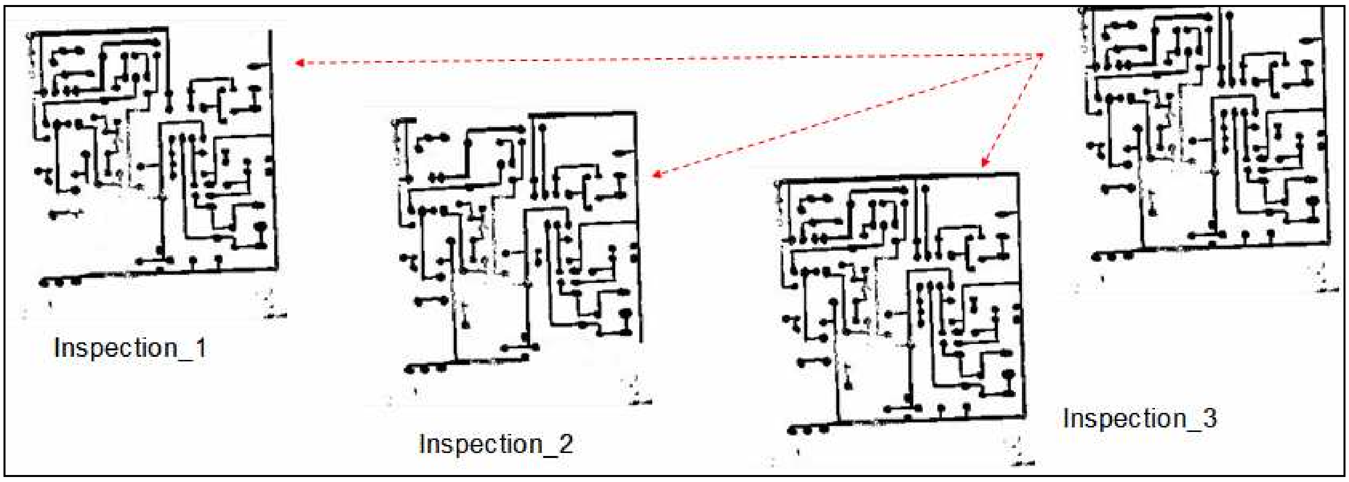


Figure 8.34 – PCB's defected images to the sample 2

- The results in relation to the image time processing to the sample 1, during the inspection 1, 2 and 3 are the graphics bellow:

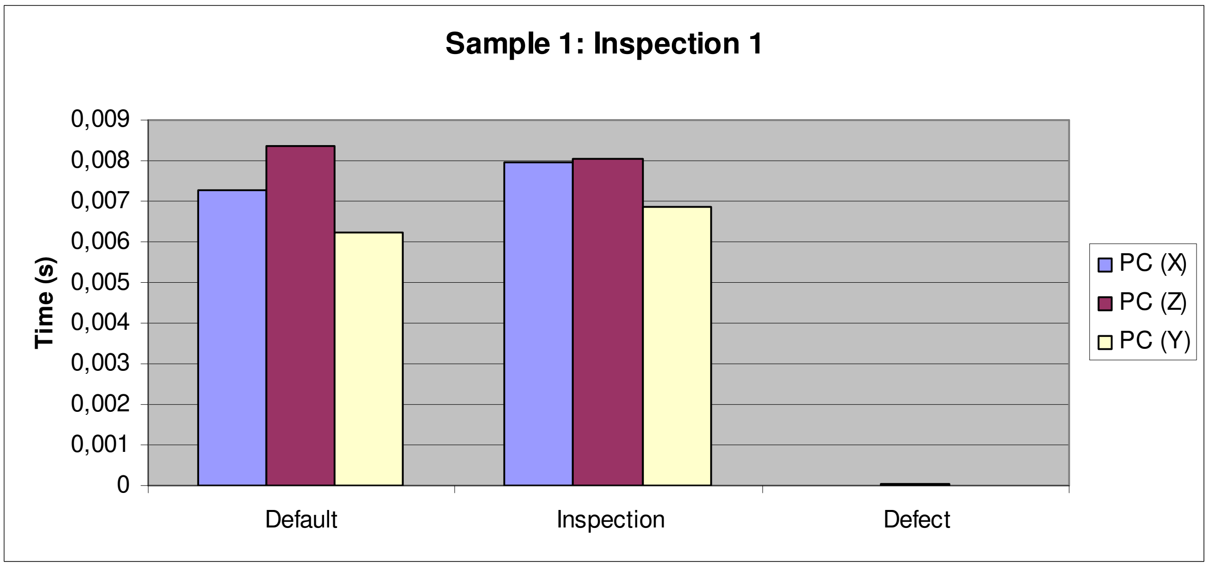


Figure 8.35 – Graph sample 1, inspection 1

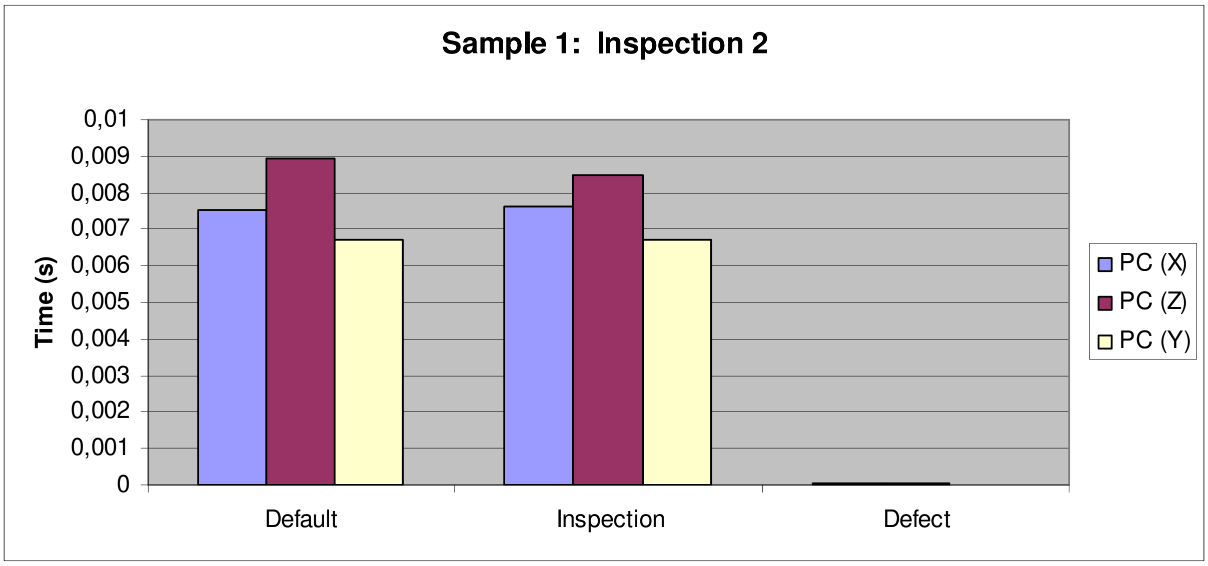


Figure 8.36 – Graph sample 1, inspection 2

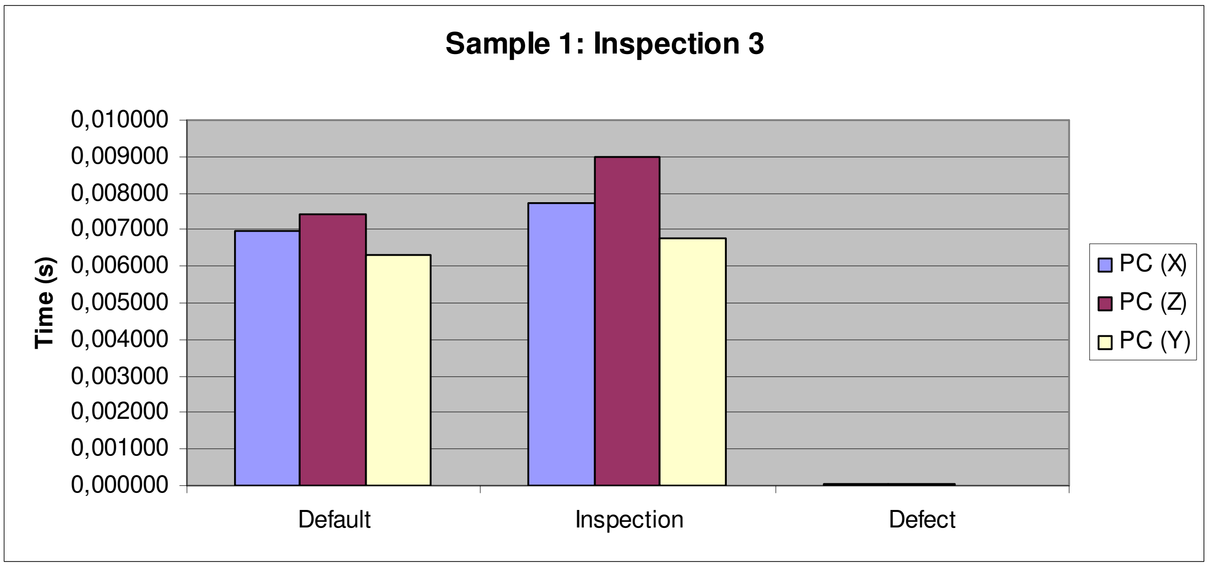


Figure 8.37 – Graph sample 1, inspection 3

- The results in relation to the image time processing to the sample 2, during the inspection 1, 2 and 3 are the graphics bellow:

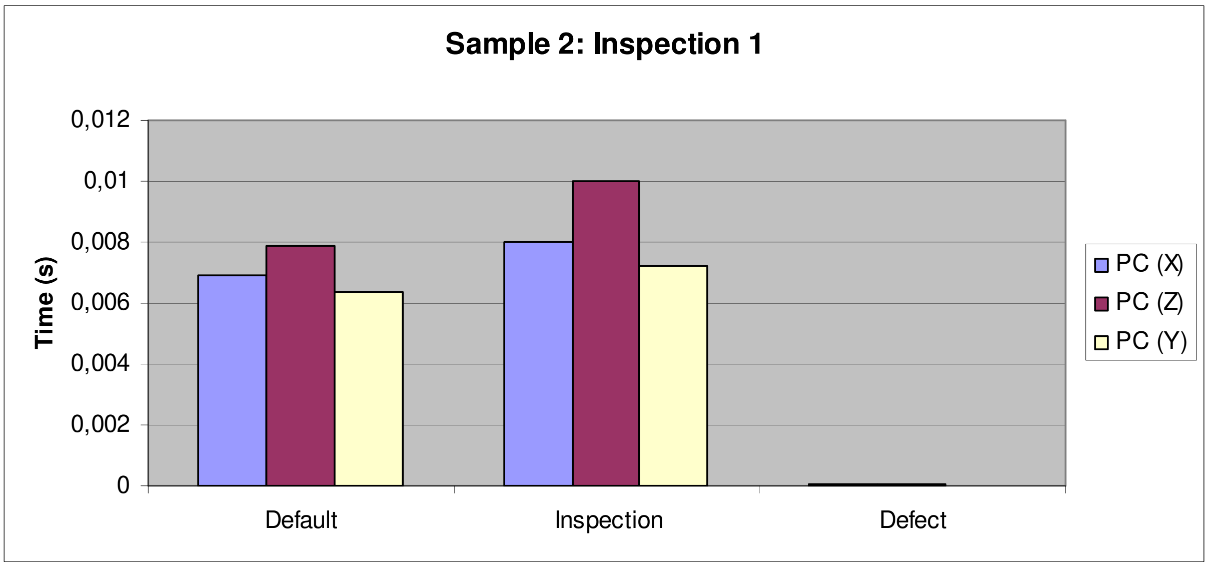


Figure 8.38 – Graph sample 2, inspection 1

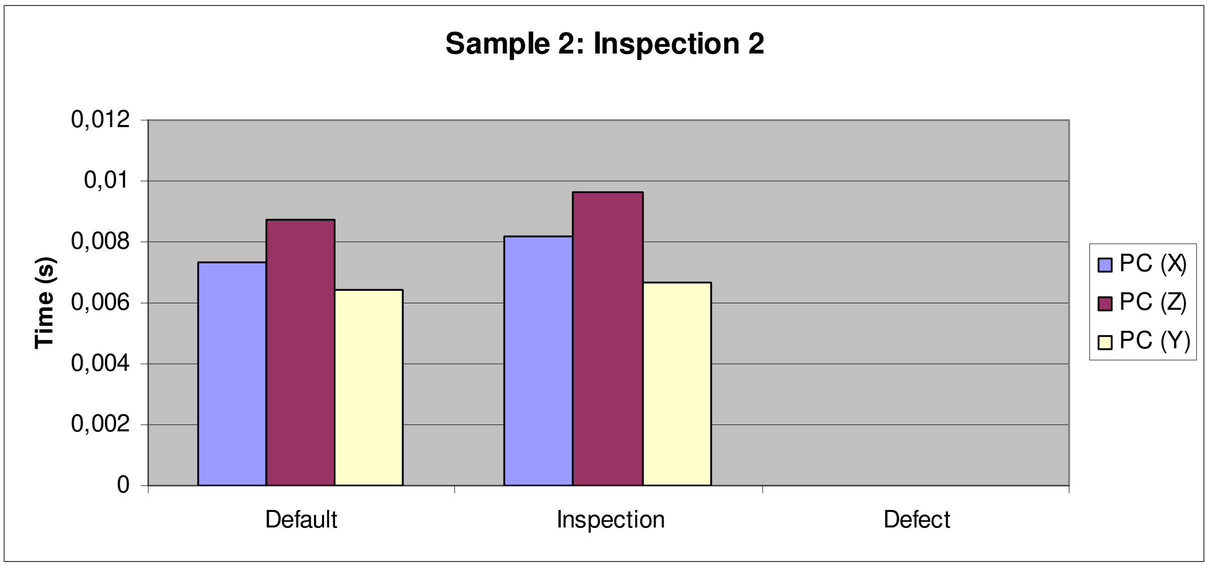


Figure 8.39 – Graph sample 2, inspection 2

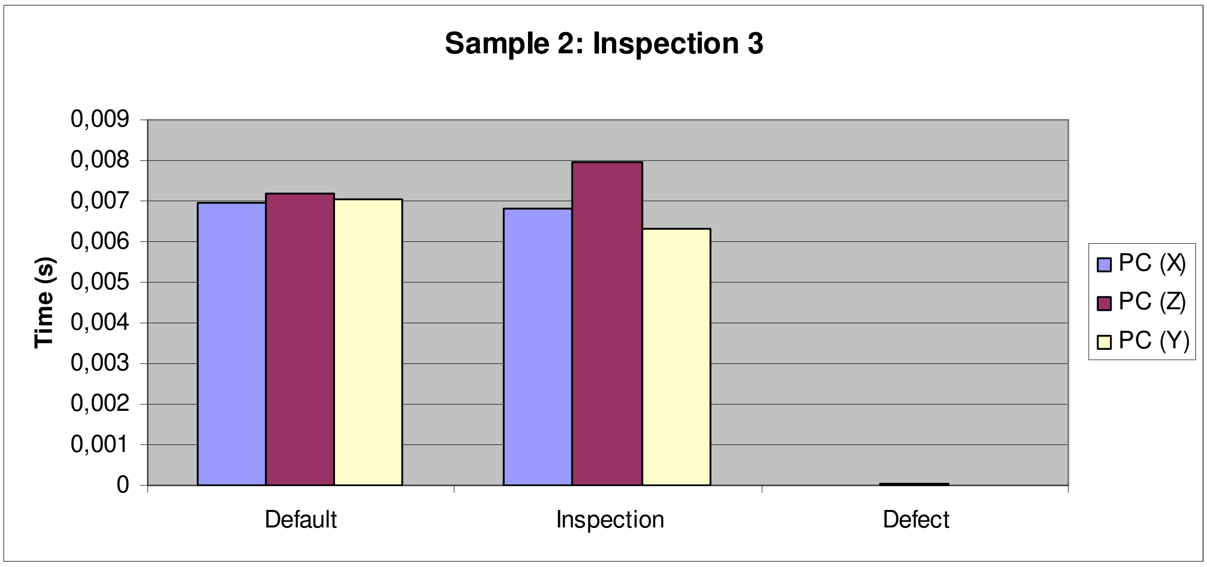


Figure 8.40 – Graph sample 2, inspection 3

8.5 RESULTS RESUME

It is important observe through Figures 8.5 to 8.10, Figures 8.15 to 8.20, Figures 8.25 to 8.30 and Figures 8.35 to 8.40 this there is significatives variations in relation to the time image processing algorithm, in the case of the implementation of FWT algorithm are maintained in order of 10^{-3}s and 10^{-5}s . Those results indicate the efficiency of FWT algorithm with values dependent of image inspection complexity.

It's observed also observed this due the formulation made in the Chapter 5, the level of actuation of FWT algorithm will be more severe as much as be the complexity of the bare PCB samples inspections. Therefore, in the conditions of low complexity (conditions detailed in the Chapters before), the control system is low solicited, due to the low complexity of the bare PCB samples inspection. In the condition described in the Chapter 6, it is possible notate the FWT algorithm controller actuate in a more efficient way, avoiding to capture insignificant defects and focusing in capture the whole board condition. It is also observed this in this case, the difference between the limier of compression image tax between cases of linear models, because the image XOR operation occurs in the linear case using the MATLAB software. Finally, in the condition where sample bare PCB images are analyzed after implementation of FWT algorithm, the algorithm actuate in a severe way, allowing defect detections of singularities in the samples of bare PCB's digital layout's. The difference between the severity of FWT algorithm actuation can be explained because of dependence between gains of FWT algorithm controller and the inspection velocity of the system.

Therefore, we can plot the graphs in respect samples 1, 2 and 3 of scanned images and see the percentage difference between two methods: first is the image time inspection of reference, testing and defect detected in the scanned images without the FWT algorithm and second, is the image time inspection of reference, testing and defect detected in the scanned images with the FWT algorithm developed in MATLAB. The results appoints an enhance about 45% up to 97% in respect of time optimization in the second method using the FWT algorithm.

- The results in relation to the image time processing percentage value to the sample 1, in relation to the usage of FWT algorithm for time optimization in the scanned images, during the inspection 1, 2 and 3 are the graphics bellow:

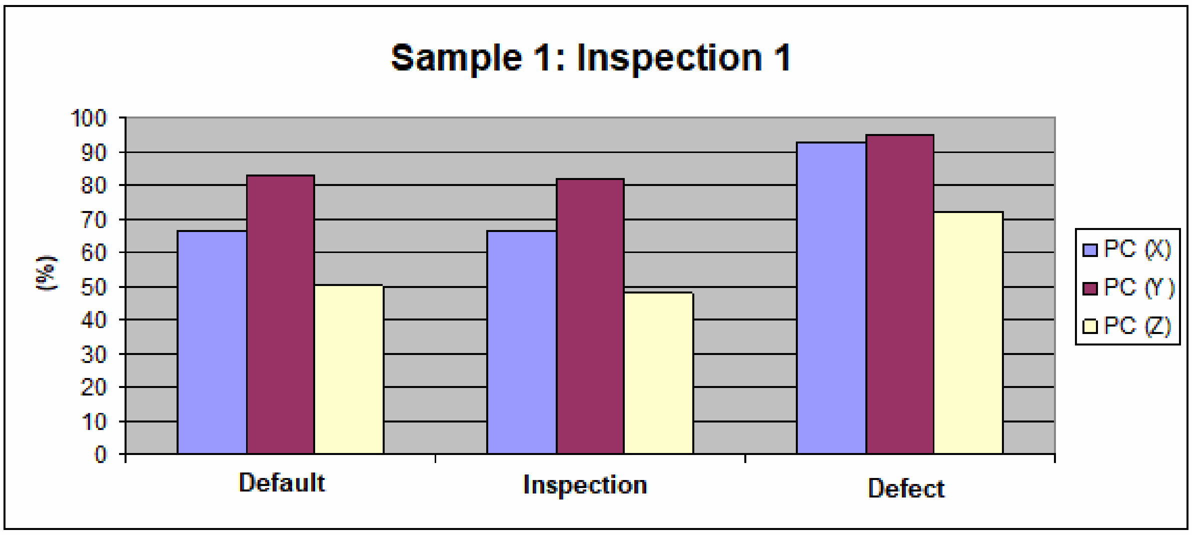


Figure 8.41 – Graph percentage sample 1, inspection 1

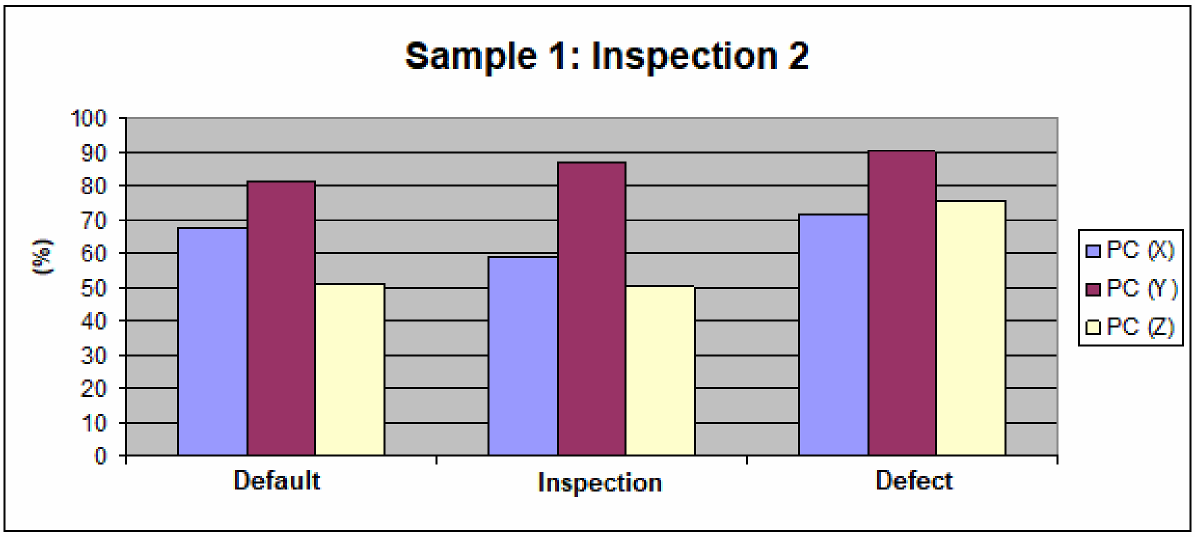


Figure 8.42 – Graph percentage sample 1, inspection 2

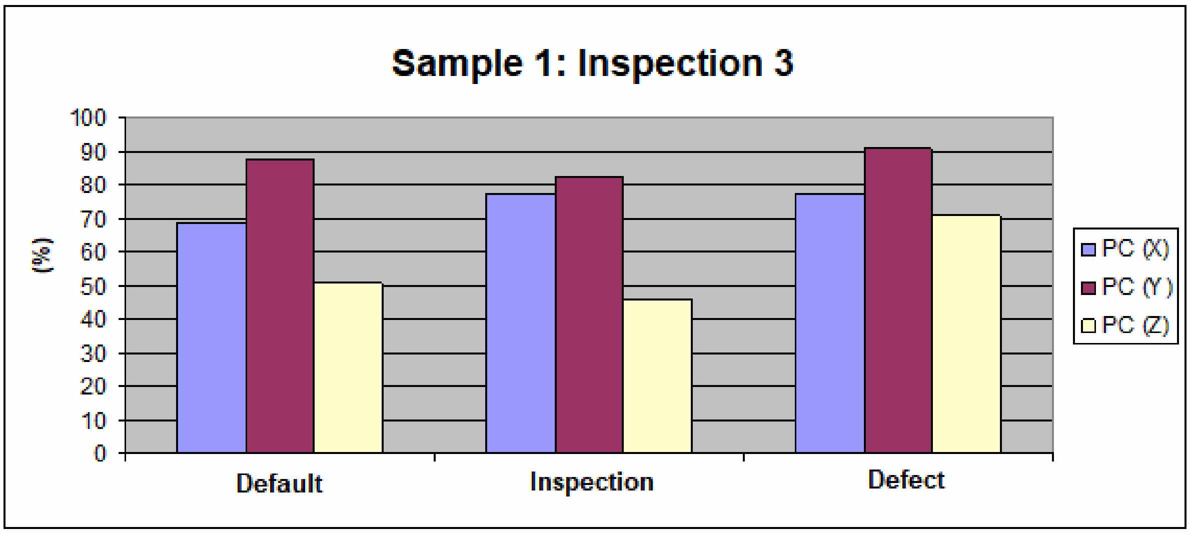


Figure 8.43 – Graph percentage sample 1, inspection 3

- The results in relation to the image time processing percentage value to the sample 2, in relation to the usage of FWT algorithm for time optimization in the scanned images, during the inspection 1, 2 and 3 are the graphics bellow:

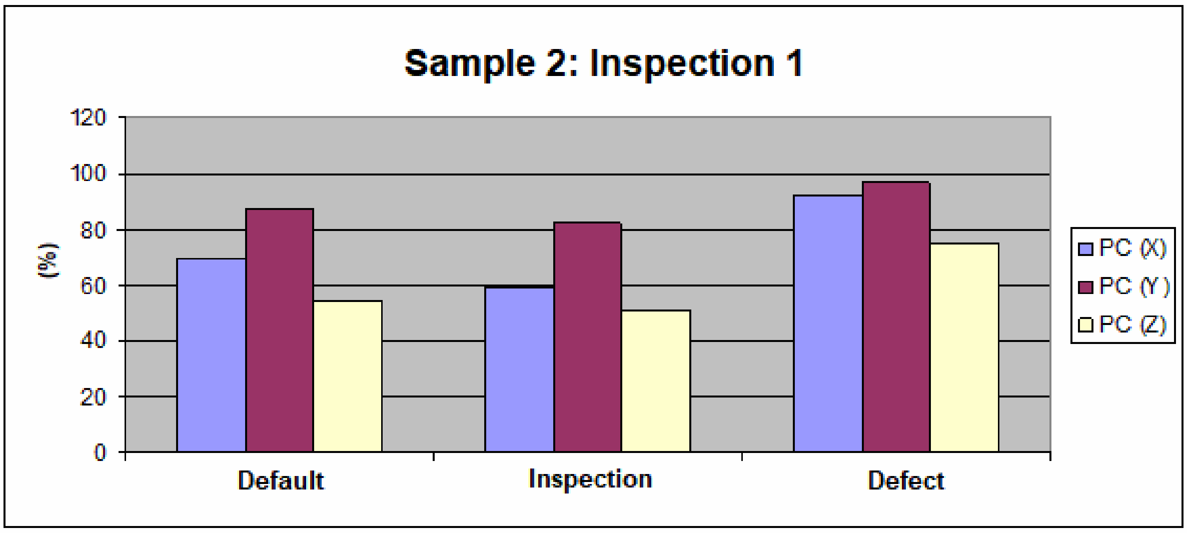


Figure 8.44 – Graph percentage sample 2, inspection 1

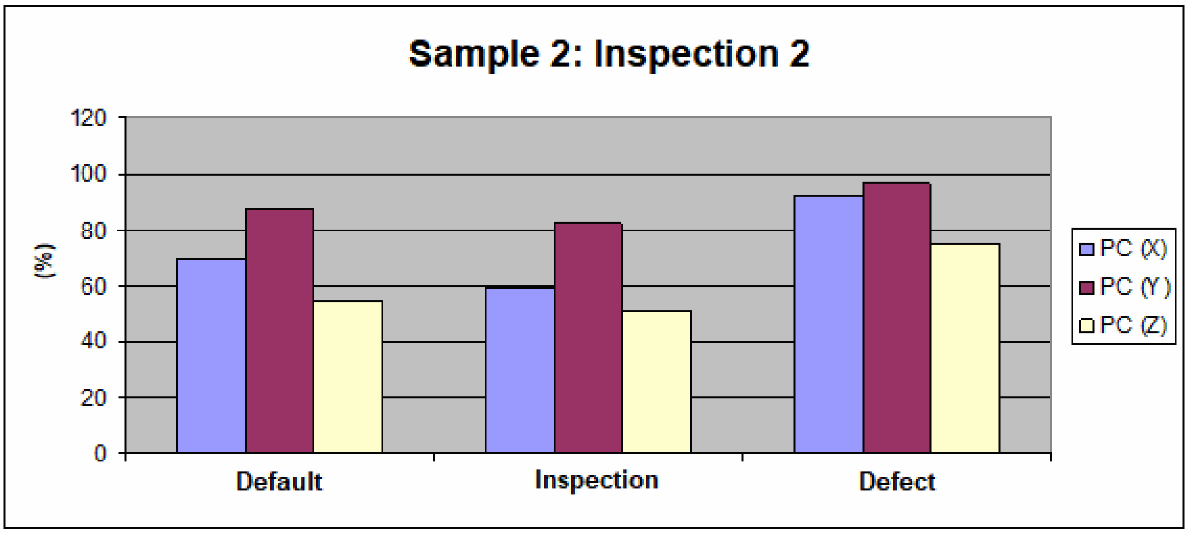


Figure 8.45 – Graph percentage sample 2, inspection 2

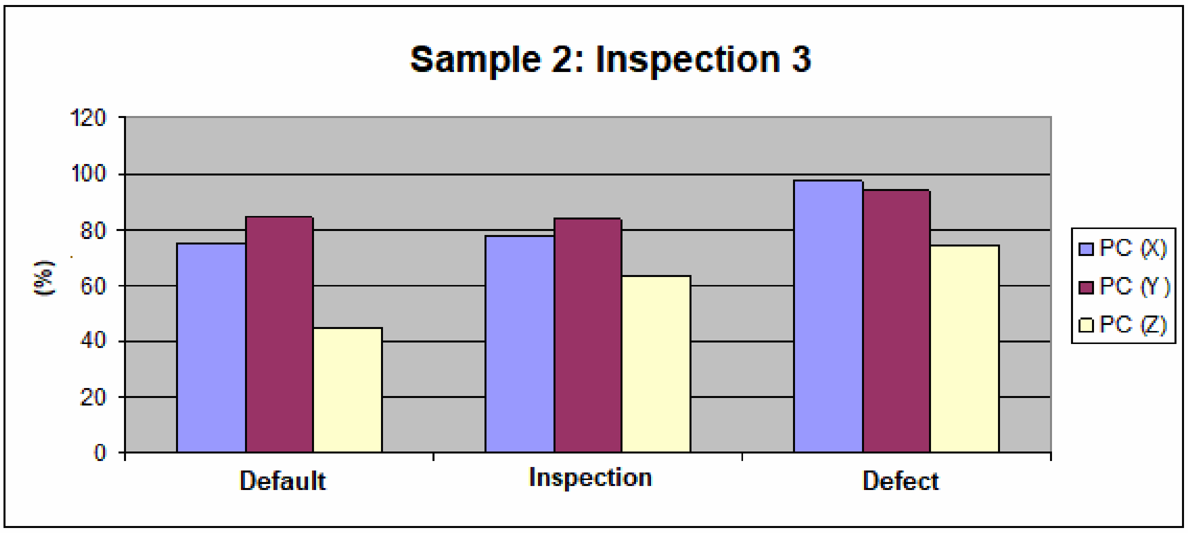


Figure 8.46 – Graph percentage sample 2, inspection 3

9 CONCLUSION

In this work was deduced a commonly model utilized in the field of image inspection techniques to represent the image subtraction technique and time reduction appointments during the inspection of PCBs templates. Once defined the inputs and outputs of the inspection system, which would be utilized in a practical case, was defined implementation strategies of image subtraction techniques with gains in relation to the time and image classification efficiency depending on the implementation of FWT algorithm into the image inspection system. After definition of a metrics of optimization through FWT, was verified the dependence of the terms in the algorithm of time and image classification inspection, and was preferable to choose an algorithm with constant gains in the MATLAB software. With this procedure, was possible also to obtain gains in relation to the image classification of the defects detection on the PCBs layouts production samples. After this step, were tested different simulations in the MATLAB with production PCBs reference and testing layouts, and the performance of the FTW algorithm introduced to the inspection system was compared with and without software scanned images in the step of pre-processing.

The results obtained in the Chapter 8, together with other data previously presented in the Chapter 5, allow appointing the following main conclusions:

- Observing the graphics of the inspection times of the computers, for each bare PCB production sample, is concluded the method of PCB time inspection before and after using the FWT algorithm is highly precision, because the PCB inspection scale time, after the implementation of FWT algorithm, will kept in the order of greatness of 10^{-6}s;
- For this presented cases, the simulations from the FWT algorithm are fit enough to be well approximated by linear relations in the linear simulations in MATLAB, though in this cases is there also the possibility to be represented in a in/out system using SIMULINK in MATLAB as well, because is treat about an image inspection system variables of entrance and output;
- Due to the characteristics of image precision inspection of the FWT algorithm implemented in MATLAB simulations, the image inspection time for the 11 samples of PCBs production layouts also was kept in a region, in the all cases linear, in mode of the

approximation of small values of time inspection for reference, testing and defect detection images would be enough for precision values of lower and medium complexities (as showed from the comparisons between various simulations this were described in the Chapter 8);

- The FWT algorithm has been acted in an efficient form in the cases of *Testing image* and *Reference image*; in the cases were there was not risk of missing time, the function of counting time of the FTW algorithm not affected in a significative form the summary of the time, being in the lower times accounting, its changing was not significant; in the more higher accounting times, was not concluded which is the variation of small decimal places, once was concluded the PCBs production layout of the image inspection system would have an influence about the time accounted during the inspection algorithm of FWT in the case of *Detection defect*; in this cases, the actuation of the FWT algorithm was more efficient in the cases of image classification efficiency, indicating this linearization of the time of the PCBs production layouts of the image inspection system results in a optimistic estimative of this behavior;
- The coefficients of the FWT algorithm can be adjusted in a manner to increase the actuation of the filters of the algorithm; in this case, the time inspection of the bare PCBs layouts production would be a little bit affected during the execution of the FWT algorithm (because the gains of the FWT counter are small in this case), and would be possible increase the number of decimal places of the algorithm during the running in MATLAB because of the precision accuracy;
- The usage of time gains was showed a useful tool in the development of the FWT algorithm applied to the inspection of the PCBs digital layouts of bare production samples; in this mode, was possible to realize the computational calculus of the infinite possibilities to the time accounting of the bare production samples PCBs, to the only who in a given linear case, from which is possible to calculate gains of time inspection, in the PCB inspection system, to after realize a interpolation of any time image PCB inspection desired in the system.

Then, is believed the time inspection accounting of the bare PCBs digital layouts of production samples observed through FWT algorithm and image XOR subtraction techniques can be utilized in the praxis, actuating mainly in the cases of most practical interest (looking the time inspection PCBs of the bare production samples and the image subtraction defect detects efficiency showed in the FWT algorithm are not so common in the reality). Although, to the practical implementation, some steps must be given in the direction of the development and physical validation before the realization of the physical tests in an assembly PCB production line.

In this mode, follow some suggestions for the future works:

- Utilize a robust algorithm (like FWT algorithm here presented), in conjunction with commercial softwares to the image inspection of PCBs with components in a production assembly line, to the validation in a independent system and of a higher complexity;
- Utilization of models more complex of image difference inspection, this allows the a higher variation of image classification, to study the behavior of robustness of the image inspection algorithm facing to the variations of PCBs components of this parameter;
- To vary the function objective of the FWT algorithm to verify the influence of each state in the characteristics of image inspection and image classification of the algorithm;
- To study the characteristics related to the measurement devices of the production PCB line and actuation for this can be an analysis of viability to the implementation in a line of large scale (together with the FWT algorithm), taking in account as would be done its implementation in a physics environment (sensors, actuators and digital processing).

REFERENCES

1] Thibadeau, R., “Printed circuit board inspection.” Carnegie Mellon University, 1981.

[2] Ortega et al., “PCB Inspection Using Image Processing and Wavelet Transform.” ResearchGate, 2007.

[3] Gonzalez, R.C., Woods, R.E., “Digital Image Processing – 3rd edition.” Pretince Hall 2010; 122 pages.

[4] Kumar and Verma, “Color-to-grayscale conversion to maintain discriminability.” Proceedings of SPIE – The International Society for Optical Engineering, 2004.

[5] Senthilkumaran, N., and Vaithegi S., “Image Segmentation By Using Thresholding Techniques For Medical Images.” Semantic Scholar, 2016.

[6] Jayaraman et al., “Scilab Textbook Companion for Digital Image Processing”. Tata McGraw 2010; 112 pages.

[7] Suhasini, A., et al., “PCB Defect Detection Using Image Subtraction Algorithm.” International Journal of Computer Science and Technology, 2015.

[8] Santoyo, et al., “PCB Inspection Using Image Processing and Wavelet Transform.” Mexican International Conference on Artificial Intelligence, 2007.

[9] Tatibana, M.H., Lotufo R., “Novel automatic PCB inspection technique based on connectivity.” Computer Graphics and Image Processing, 1997.

[10] Batchelor, B.G., Whelan, P.F., “Intelligent vision systems for industry.” Springer 1997; 457 pages.

[11] INMETRO (2012). “Sistema Internacional de Unidades.” Rio de Janeiro: INMETRO.

[12] Fikret Ercal, Filiz Bunyak, Lei Zheng., “Fast Modular RLE-Based inspection scheme for PCBs.” Published in Other Conferences, 1997. DOI: 10.1117/12.294439.

[13] Madhav Moganti, Fikret Ercal, Shou Tsunekawa., “Automatic PCB Inspection Algorithms: A Survey.” Published in Computer Vision and Image Understanding, 1996. DOI: 10.1006/cviu.1996.0020.

[14] D. T. Pham, R. J. Alcock., “Smart Inspection Systems: Techniques and Applications of Intelligent Vision.” Academic Press, 2002.

[15] Rafael C. Gonzalez, Richard E. Woods., "Digital Image Processing, 2nd Edition" Prentice Hall, 2010.

[16] Mark Nixon ., "Feature Extraction & Image Processing, 2nd edition" Academic Press, 2008.

[17] Rafael C. Gonzalez, Richard E. Woods., "Digital Image Processing, 1st edition" Prentice Hall, 1992.

[18] Sergios Theodoridis, Konstantinos Koutroumbas., "Pattern Recognition, 4th edition" Academic Press, 2009.

[19] Sanz, J. L. C. and Jain, A.K., "Machine-Vision Techniques for Inspection of Printed Wiring Boards and Thick-Film Circuits", Journal of Optical Society of America A, Vol. 3, No. 9, pp. 1465-1482, September 1986.

[20] Mohit Borthakur, Anagha Latne, Pooja Kulkarni., "A Comparative Study of Automated PCB Defect Detection Algorithms and to Propose an Optimal Approach to Improve the Technique", International Journal of Computer Applications, Vol. 114, No. 6, pp. 27-33, March 2015.

[21] Er. Amit Paul, Er. Gaurav, Er. Poonam Verma., "Defect Detection on Printed Circuit Board using Local Binary Pattern", International Journal of Innovative Research in Computer and Communication Engineering, Vol. 4, No. 8, pp. 14457-14464, August 2016.

[22] Ismail Ibrahim, Zuwairie Ibrahim, Kamal Khalil, Musa Mohd Mokji, Syed Abu Bakar., "An Algorithm for Classification of Five Types of Defects on Bare Printed Circuit Board", International Journal of Computer Sciences and Engineering Systems, Vol. 5, No. 3, pp. 201-208, July 2011.

[23] Namita Kalyan Shinde, S.S. Morade., "PCB Inspection System Using Image Processing", International Journal of Science, Engineering and Technology Research, Vol. 4, No. 4, pp. 1009-1012, April 2015.

[24] Sonal Kaushik, Javed Ashraf., "Automatic PCB Defect Detection Using Image Subtraction Method", International Journal of Computer Science and Network, Vol. 1, No. 5, October 2012.

[25] Zuwairie Ibrahim, Syed Abdul Rahman Al-Attas, Zulfakar Aspar., "Model-based PCB Inspection Technique Using Wavelet Transform", The 4th Asian Control Conference, September 2002.

[26] Joaquín Santoyo, J. Carlos Pedraza, L. Felipe Mejía, Alejandro Santoyo., "PCB Inspection Using Image Processing and Wavelet Transform", Research Gate, 2007.

[27] Amit H. Choksi, Ronak Vashi, Mayur Sevak, Kaushal Patel., "Printed Circuit Board Defect Detection using Wavelet Transform", International Journal of Current Engineering and Technology, Vol. 4, No. 4, pp. 2647-2652, August 2014.

[28] Stéphane Mallat., “Wavelets for a Vision”, Proceedings of the IEEE, Vol. 84, No. 4, April 1996.

[29] Zuwairie Ibrahim, Syed Abdul Rahman Al-Attas., “Wavelet-based printed circuit board inspection algorithm”, Integrated Computer-Aided Engineering, Vol. 12, No. 1, pp. 201-213, 2005.

[30] Syed Abddul Rahman Bin, Syed Abu Bakar., “Automatic Search for Break Points in Printed Circuit Board Using Intelligent Visual System Based on DSP Algorithms”, Universiti Teknologi Malaysia, 1998.

[31] Zuwairie Ibrahim, Syed Abdul Rahman Al-Attas., “Wavelet-Based Printed Circuit Board Inspection System”, International Journal of Signal Processing, Vol. 1, No. 2, pp. 73-79, 2004.

[32] Charbel Szymanski., “Desenvolvimento de Técnicas de Processamento Digital de Imagens para Inspeção de Placas de Circuito Impresso Produzidas em Pequenas Séries”, Universidade Federal de Santa Catarina, 2014.

[33] Camila Pontes Brito da Costa., “Desenvolvimento de um Sistema de Diagnóstico de Defeitos na Montagem de PCI Baseado em Redes Bayesianas”, Universidade Federal de Santa Catarina, 2014.

[34] Prachi P. Londe, Atul N. Shire., “A Review on Automatic PCB Defects Detection and Classification”, International Journal of Emerging Trends in Engineering and Development, Vol. 1, No. 4, pp. 380-388, January 2014.

[35] Surenda Khushwaha, Dinesh Goyal, Rahul Kumar., “Computer Vision System for Automatic PCB Inspection & Quality Analysis with Auto Rejection”, International Journal of Digital Application & Contemporary Research, Vol. 4, No. 2, September 2015.

[36] Ang TeohOng, Zulkifilie Bin Ibrahim, Suzaimah Ramli., “Computer Machine Vision Inspection on Printed Circuit Boards Flux Defects”, American Journal of Engineering and Applied Sciences, Vol. 6, No. 3, pp. 263-273, 2013.

[37] Ismail Ibrahim, Zuwairie Ibrahim, Kamal Kahlil, Musa Mohd Mokji, Syed Abdul Rahman, Syed Abu Bakar, Norrima Mokhtar, Wan Khairunizam, Wan Ahmad., “An Improved Defect Classification Algorithm for Six Printing Defects and its Implementation on Real Printed Circuit Board Images”, International Journal of Innovative Computing, Information and Control, Vol. 8, No. 5, pp. 3239-3250, May 2012.

[38] Suhasini A, Sonal D Kalro, Pathiksha B G, Meghashree B S, Phaneendra H D., “PCB Defect Detection Using Image Subtraction Algorithm”, International Journal of Computer Science Trends and Technology, Vol. 3. No. 3, pp. 1-6, May-June 2015.

APPENDIX

Appendix A – Algorithm of image binarization before the implementation of FWT transform.

```
function varargout = PreGui(varargin)
% PREGUI MATLAB code for PreGui.fig
%      PREGUI, by itself, creates a new PREGUI or raises the existing
%      singleton*.
%
%      H = PREGUI returns the handle to a new PREGUI or the handle to
%      the existing singleton*.
%
%      PREGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in PREGUI.M with the given input arguments.
%
%      PREGUI('Property','Value',...) creates a new PREGUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before PreGui_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to PreGui_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help PreGui

% Last Modified by GUIDE v2.5 15-Apr-2018 10:44:10

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @PreGui_OpeningFcn, ...
                   'gui_OutputFcn',  @PreGui_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
```

```
end
% End initialization code - DO NOT EDIT


% --- Executes just before PreGui is made visible.
function PreGui_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to PreGui (see VARARGIN)

% Choose default command line output for PreGui
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes PreGui wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = PreGui_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executa leitura da imagem + redimensiomaneto
function [I_Ref] = openImage(I)
I_Ref=imread(I);
% I_Ref=imresize(I_Ref,[200 200]);

% --- Aplica greyscale
function [I_Grey] = imageToGrey(handles)
if ndims(handles.I_Ref)> 2
    I_Grey=rgb2gray(handles.I_Ref);
else
    I_Grey=I_test;
end

% --- Aplica threshold
function [I_Threshold] = greyToThreshold(handles)
I_Threshold = imbinarize(handles.I_Grey, 'adaptive');
I_Threshold = imcomplement(I_Threshold);

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
addpath(genpath(pwd));
cd
I=imgetfile;
[I_Ref]=openImage(I);
initTime=@() openImage(I);
endTime=timeit(initTime);

axes(handles.axes1);
imshow(I_Ref);title(sprintf('Reference Image - %f s', endTime))
handles.I_Ref=I_Ref;

[I_Grey]=imageToGrey(handles);
initTime2=@() imageToGrey(handles);
endTime2=timeit(initTime2);

axes(handles.axes2);
imshow(I_Grey);title(sprintf('Grey Image - %f s', endTime2))
handles.I_Grey=I_Grey;

[I_Threshold]=greyToThreshold(handles);
initTime3=@() greyToThreshold(handles);
endTime3=timeit(initTime3);

axes(handles.axes3);
imshow(I_Threshold);title(sprintf('Threshold Image - %f s', endTime3))
handles.I_Threshold=I_Threshold;
guidata(hObject, handles);

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
F = getframe(handles.axes2);
Image = frame2im(F);
filter = {'*.jpg;*.png'};
[file, path] = uiputfile(filter);
if file > 0
  filename = fullfile(path,file);
  imwrite(Image, filename);
end

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
F = getframe(handles.axes3);
Image = frame2im(F);
filter = {'*.jpg;*.png'};
[file, path] = uiputfile(filter);
if file > 0
  filename = fullfile(path,file);
  imwrite(Image, filename);
end
```

ANNEX

ANNEX A – Discrete Wavelet Transform (DWT) mathematic conception

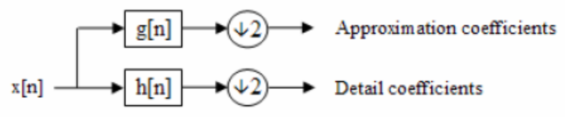
According to (GONZALEZ AND WOODS, 2010):

One level of the transform

The DWT of a signal x is calculated by passing it through a series of filters. First the samples are passed through a low pass filter with impulse response g resulting in a convolution of the two:

$$y[n] = (x * g)[n] = \sum_{k=-\infty}^{\infty} x[k]g[n-k]$$

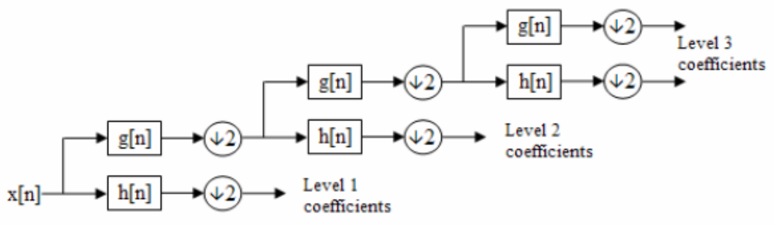
The signal is also decomposed simultaneously using a high-pass filter h. The outputs giving the detail coefficients (from the high-pass filter) and approximation coefficients (from the low-pass). It is important that the two filters are related to each other and they are known as a quadrature mirror filter.



Block diagram of filter analysis

Cascading and filter banks

This decomposition is repeated to further increase the frequency resolution and the approximation coefficients decomposed with high and low pass filters and then down-sampled. This is represented as a binary tree with nodes representing a sub-space with a different time-frequency localisation. The tree is known as a filter bank.



A 3 level filter bank

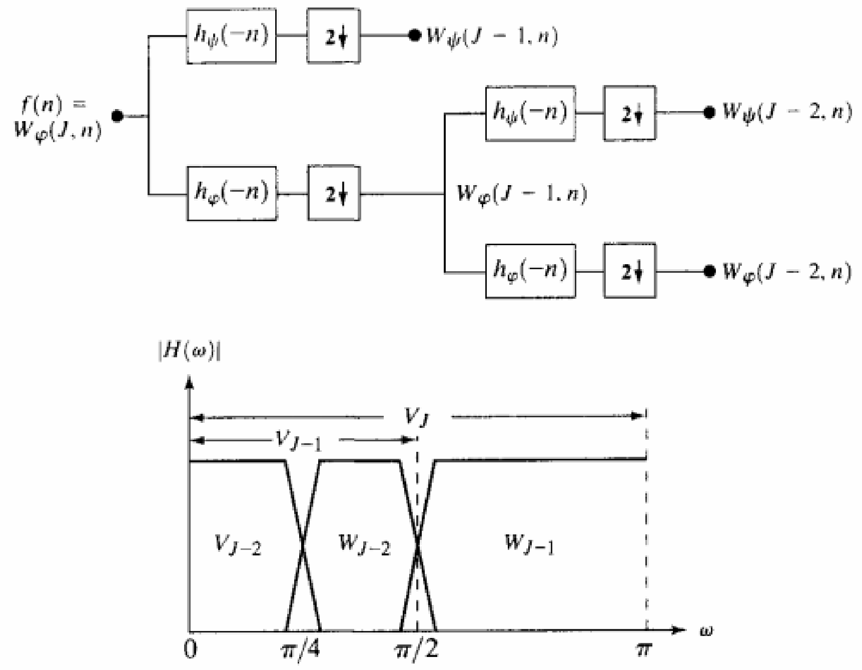
At each level in the above diagram the signal is decomposed into low and high frequencies. Due to the decomposition process the input signal must be a multiple of 2^n where n is the number of levels.

For example a signal with 32 samples, frequency range 0 to f_n and 3 levels of decomposition, 4 output scales are produced:

Level	Frequencies	Samples
3	0 to $f_n/8$	4
	$f_n/8$ to $f_n/4$	4
2	$f_n/4$ to $f_n/2$	8
1	$f_n/2$ to f_n	16

ANNEX B – Fast Wavelet Transform (FWT) mathematic conception

According to (GONZALEZ AND WOODS, 2010):



a
b

FIGURE 7.16 (a) A two-stage or two-scale FWT analysis bank and (b) its frequency splitting characteristics.

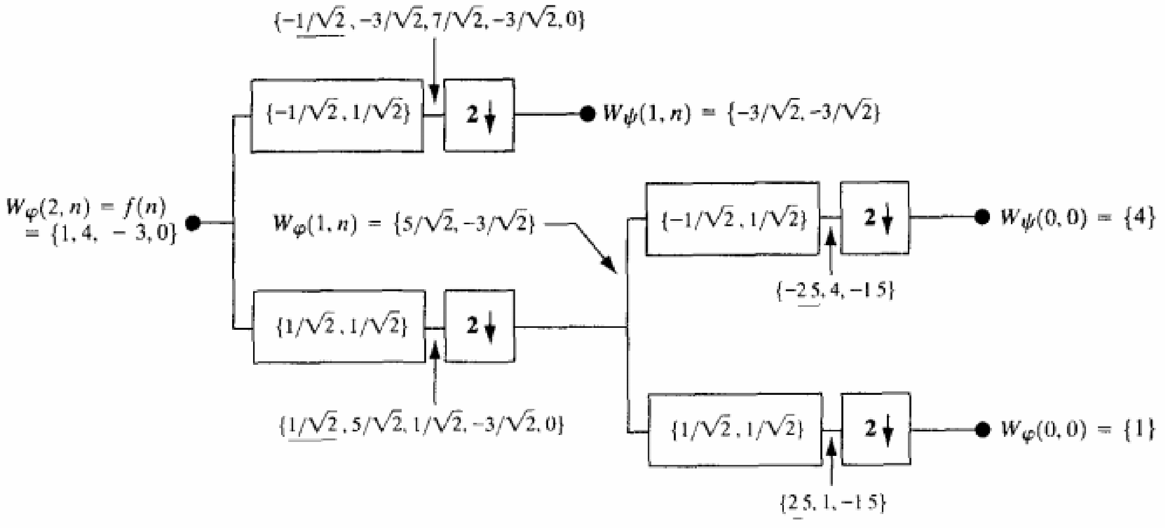

$\{-1/\sqrt{2}, -3/\sqrt{2}, 7/\sqrt{2}, -3/\sqrt{2}, 0\}$
$\{-1/\sqrt{2}, 1/\sqrt{2}\}$
$2\downarrow$
$W_\psi(1, n) = \{-3/\sqrt{2}, -3/\sqrt{2}\}$
$W_\varphi(2, n) = f(n) = \{1, 4, -3, 0\}$
$W_\varphi(1, n) = \{5/\sqrt{2}, -3/\sqrt{2}\}$
$\{1/\sqrt{2}, 1/\sqrt{2}\}$
$\{1/\sqrt{2}, 5/\sqrt{2}, 1/\sqrt{2}, -3/\sqrt{2}, 0\}$
$W_\psi(0, 0) = \{4\}$
$\{-2.5, 4, -1.5\}$
$W_\varphi(0, 0) = \{1\}$
$\{2.5, 1, -1.5\}$

FIGURE 7.17 Computing a two-scale fast wavelet transform of sequence $\{1, 4, -3, 0\}$ using Haar scaling and wavelet vectors.