BRNO UNIVERSITY OF TECHNOLOGY

Faculty of Electrical Engineering
and Communication

BACHELOR'S THESIS

Brno, 2017

Martin Horák

BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF CONTROL AND INSTRUMENTATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

DEFECT DETECTION

DETEKCE DEFEKTŮ

BACHELOR'S THESIS

BAKALÁŘSKÁ PRÁCE

AUTHOR AUTOR PRÁCE	Martin Horák
SUPERVISOR VEDOUCÍ PRÁCE	Ing. Miloslav Richter, Ph.D.

BRNO 2017

BRNO UNIVERSITY OF TECHNOLOGY FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

Bachelor's Thesis

Bachelor's study field **Automation and Measurement**
Department of Control and Instrumentation

Student: Martin Horák ***ID:*** 164286
Year of study: 3 ***Academic year:*** 2016/17

TITLE OF THESIS:

Defect detection

INSTRUCTION:

The topic of this thesis is detection of defects on the x-ray images of printed circuit boards. Topic of this thesis is from praxis at the student's request.

1. Study the basic properties of x-ray images and printed circuit boards with various kinds of defects.
2. Describe the characteristics of x-ray images and properties of defects. Describe the types of defects and their manifestation in the image. Prepare a set of images for testing of various defects.
3. Perform analysis of suitable programming languages and libraries for image processing and justify the selected programming tools.
4. Perform analysis and evaluation of methods suitable for the application and select the appropriate algorithms for implementation.
5. Choose three types of defects and implement suitable algorithms for their detection in the images.
6. From the resulting algorithms create commented library for defect detection. Evaluate the quality of implemented algorithms.

REFERENCE:

Žára J., Beneš B., Sochor J., Felkel P.: Moderní počítačová grafika, Computer Press, 1998, ISBN 80-251-0454-0
Hlaváč V., Šonka M.: Počítačové vidění,Grada, Praha 1992, ISBN 80-85424-67-3

Assigment deadline: 6. 2. 2017 ***Submission deadline:*** 29.5.2017

Head of thesis: Ing. Miloslav Richter, Ph.D.
Consultant:

[illegible]

doc. Ing. Václav Jirsík, CSc.
Subject Council chairman

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ Fakulta elektrotechniky a komunikačních technologií Ústav automatizace a měřicí techniky

ABSTRACT

Visual inspection of the printed circuit boards (PCB's) by human controllers is becoming impossible with the increasing complexity and miniaturization of the circuit boards. To achieve high reliability of the PCB's, manufacturers are forced to develop new methods of inspection that will ensure inline control of every circuit board and separate those that does not fulfill the quality requirements. One of the options is the automatic x-ray inspection method which is covered in this thesis.

KEYWORDS

Image processing, PCB defects detection, automated x-ray inspection

ABSTRAKT

Vizuální inspekce desek prošných spojů (DPS) lidmi se stává nemožná z důvodu zvyšující se komplexity a miniaturizace desek plošných spojů. K dosažení vysoké spolehlivosti DPS jsou výrobci nuceni vyvíjet nové metody inspekce, které by zajistily kontiunuální kontrolu každé desky a oddělily ty, které nesplňují požadavky na jakost. Jednou z možností je použít automatickou rentgenovou inspekci, kterou se zabývá tato práce.

KLÍČOVÁ SLOVA

Zpracování obrazu, detekce defektů na DPS, Automatická inspekce rentgenových snímků

HORÁK, Martin *Detekce defektů*: semestral project. Brno: Brno University of Technology, Faculty of Electrical Engineering and Communication, Ústav telekomunikací, Rok. 99 p. Supervised by prof. Ing. Miroslav Richter, Ph.D.

Vysázeno pomocí balíčku `thesis` verze 2.61; `http://latex.feec.vutbr.cz`

DECLARATION

I declare that I have written my semestral project on the theme of "Detekce defektů" independently, under the guidance of the semestral project supervisor and using the technical literature and other sources of information which are all cited in the project and detailed in the list of literature at the end of the project.

As the author of the semestral project I furthermore declare that, as regards the creation of this semestral project, I have not infringed any copyright. In particular, I have not unlawfully encroached on anyone's personal and/or ownership rights and I am fully aware of the consequences in the case of breaking Regulation § 11 and the following of the Copyright Act No 121/2000 Sb. of the Czech Republic, and of the rights related to intellectual property right and changes in some Acts (Intellectual Property Act) and formulated in later regulations, inclusive of the possible consequences resulting from the provisions of Criminal Act No 40/2009 Sb. of the Czech Republic, Section 2, Head VI, Part 4.

Brno ...24.05.2017...

[illegible]
author's signature

ACKNOWLEDGEMENT

First and foremost, I have to thank ELEDUS which is a company that offered me to work on the topic of this thesis. I also want to thank Ondřej Vičar, Jaroslav Malec and Patrik Predný for providing me with all the necessary facilities for the research and valuable remarks. Last but not least I would like to thank my thesis supervisor Ing. Miloslav Richter, Ph.D. who was willing to cooperate on this theses and for supporting me with many valuable comments and suggestions.

Brno 24.05.2017 .... [illegible]

author's signature

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

1 INTRODUCTION

Detection of defects during the manufacturing process of the printed circuit boards (PCBs) is a never-ending process that is driven by the desire to make the assembly process more efficient. Because of the great increase of demand of the electronic devices, manufacturers of PCBs have taken very important place. This great demand also introduced a lot of new challenges. They are for example expected to deliver more circuit boards, cheaper with higher quality. Because the lucrative market with the PCBs reached an estimated \$60.2 billion in value in 2014, a lot of effort is put to improve the process of manufacturing.[1] This thesis focuses on the inspection part of the manufacturing process especially on the visual inspection using the x-ray and machine vision because by using these technologies manufacturers can achieve high reliability and reduce repair costs. The goal of the inspection is not to only find fatal defects such as breakouts, bridges or missing conductors that will compromise the printed circuit board (PCB) performance during utilization, but also to find potential defects that can cause troubles in the longer time period such as over etching, under etching or voids in the ball grid array (BGA).

With the constant improvement of technologies manufacturers are pushed towards smaller, lighter and more functional devices where every millimeter on the board has to be fully utilized therefore the circuit boards have evolved into very complex, multilayer, high density boards. Testing of such a boards have become more challenging task because higher component and joint counts create more defect opportunities which lead to lower yields for a given defect level. Constant increase in quality demands and complexity of circuit boards are forcing manufacturers to develop new methods of inspection that will ensure inline control of every circuit board and separate those that does not fulfill the quality requirements. By using the inline inspection, manufacturer can use collected data not only for identifying the bad pieces but also as a feedback loop to improve the manufacturing process.

1.1 Inspection methods

In the early days of PCB manufacturing, all inspection was undertaken manually by humans with the magnification glass. Later, this method showed up to be very limited not only because the human factor, but also because of its speed and limited ability to spot defects that are invisible for human eye. Because of the rapid growth of the PCB manufacturing industry manufacturers invested into developing new methods that would keep up with the growing requirements of the customers.

[1] http://www.ipc.org/ContentPage.aspx?pageid=World-PCB-Production-in-2014-Estimated-at-60-2-Billion.

That resulted in a wide range of test and inspection strategies that will be briefly introduced in this chapter.

1. **Visual Inspection**
   Visual method is the traditional way of PCB inspection, but it is still widely used. It utilizes the magnifying glass or microscope and it is controlled by trained human controllers. The advantage of this approach is that there are low initial costs, but because of increasing complexity of circuit boards and miniaturization it is becoming more challenging. The great disadvantage of this approach is that the quality of control is depended on each individual, it is not effective in mass production and it is also more difficult to collect the data over time for analysis of the production. On top of that some defects can not be judged only by visual inspection system therefore it does not ensure 100% reliability.

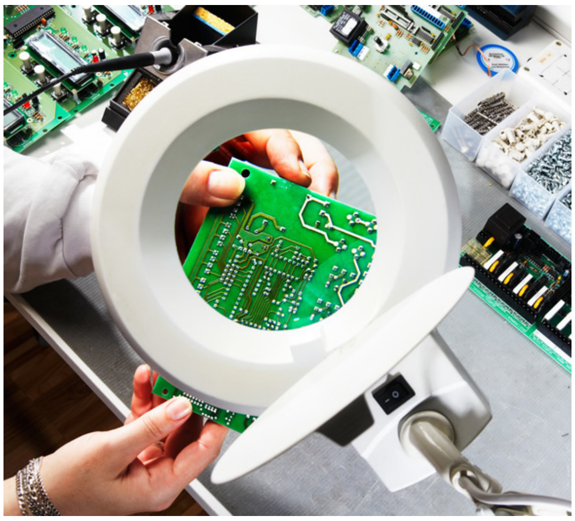

Fig. 1.1: Example of visual inspection [25]

2. **Solder Paste Inspection**
   Solder paste inspection (SPI) is a method that tests the solder paste deposit by measuring the volume of solder pads before the components are applied and the solder melted. This inspection method is necessary especially when the lead free solder pastes are used because it has been proven that they do not spread as well as tin lead solder pastes. Based on the image capturing method SPI can be divided into a 2D (area coverage) or 3D (volume coverage) inspection. [17]
3. **Functional Test (FT)**
   Functional test is the most straight forward way how to test a PCB board. It is little bit different from other forms of testing because it is not testing whether

the solder is there or not. It receives an assembled product and it simulates the electrical environment where should the circuit board work. For instance, if the final PCB is going to be a slot inside a computer than the slot will be simulated during the tests. It can be considered as a final quality control that will make sure that the end product meets the requirements. The drawback of this approach is that it is necessary to mimic the environment for every type of the PCB. Furthermore, it also might be very difficult to sufficiently test a systems with moderate complexity. A subcategory of the functional tests is *structural test* which does not test the PCB as a whole, but it tests individual partitions. This method allows to test the functionality more deeply because it can verify the corner cases, but it is difficult to reach necessary pins in the more complex or multiple layer PCBs.

4. **Boundary scan**
   Boundery scan (also known as JTAG boundary-scan) is a method for testing of interconnects on the PCB or it can be also used as debugging method to watch integrated circuit pin states. It is a way how to overcome challenges connected with mechanical access to the pins of the component by adding dedicated circuit tree to the system ICs. It uses special registers that allow to observe the signals or to input custom input and test its functionality. A drawback of this solution is that when the manufacturer wants to utilize this kind of test it is necessary include it to the design of the PCB.
5. **In-circuit test**
   In-circuit test (ICT) method is based on testing electrical parameters and performance of the circuit board to identify shorts, opens, resistance, capacitance, and other basic quantities which will show whether the assembly was correctly manufactured. There are two commonly used methods. The first one is the *Flying Probe method* which is basically several probes that are moved above the measured board by two axis system and test the circuit (see Figure 1.2). The second method is called *bed of nails test* which has numerous pins that are adjusted to the measured circuit board (see figure 1.3). On the one hand this method is more suitable to high volume production because it is much faster than *flying probe test*, but on the other hand it needs a fixture for each kind of PCBs and the cost is high. This method was very effective in the past, but with the increasing density of components this method is becoming restricted. Nowadays, these methods are not very effective due loss of physical access to the components and space constraints.
6. **Automatic optical inspection**
   Automatic optical inspection (AOI) is the advanced method that can be used to identify defects using the camera. It is non-contact method therefore it is

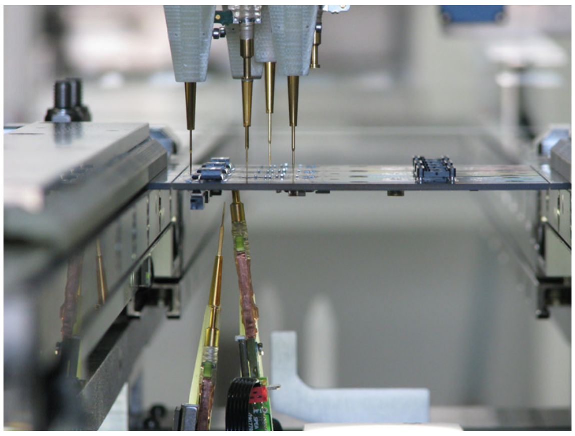

Fig. 1.2: Flying probe test [26]

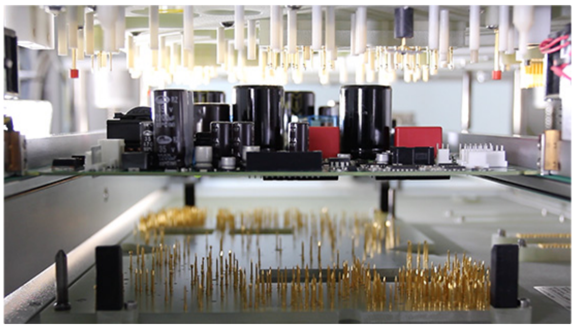

Fig. 1.3: Needle bed test [26]

very flexible because it does not need any fixture. Automatic optical inspection can work with both bare or assembled circuits and it can be used to find defects of the circuit board such as open-circuits, short-circuit defects, missing components, offset or incorrectly mounted parts. On top of that this method is fast enough to run online and do the proper testing of each circular board. The automatic optical inspection (AOI) consists of three steps. In the first step it obtains the image of the tested board. This image can be captured by charge coupled device (CCD) camera. The crucial part of capturing of the image is lighting because PCB has high reflection which will cause shadows of the object therefore in order to get good results it is essential to have very good light source. The captured image is processed and useful features are extracted. Finally the result can be processed by a computer program that will find defects automatically by using testing the geometrical rules of the PCB or by comparing with the referential circuit board. The disadvantage of this method is that it can test only the surface of the PCB therefore it can not reveal defects that does not manifest itself on the surface of the board.

7. **Automatic X-ray inspection**
   Automatic X-ray inspection (AXI) is very similar to the AOI. The only difference is the way how the image is captured. This method is measuring how much light got absorbed by the PCB at given point. The great advantage is that it is providing us more information about the PCB than the AOI. We can find for example bubbles (usually referred as voids) in the solder balls which is impossible to reveal using the CCD camera. On the other hand, in case of multiple layer boards, it might be more challenging to process the captured image due to the interference which is caused by components on the top and bottom layer of the PCB that are overlapping. The basic x-ray machine setup will provide a 2D image of the board, but more advanced machines can capture 2.5D image or even CT scan of the board (see Figure 1.4).

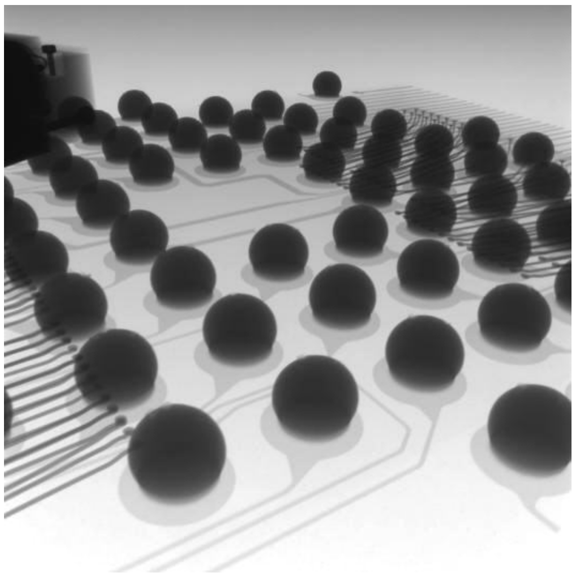
Fig. 1.4: CT scan of the BGA [27]

1.2 The best inspection method

There is no such thing as "the best inspection method" or method that can detect all possible defects. In fact choosing the right inspection method is very complex question and the answer is usually combination of multiple methods. Even picking between AOI and AXI can be tough question because is not just simple matter of looking at few characteristics. There are a lot of factors that have to be considered such as volume of the production, type of components, physical accessibility to the components, potential sources of defects and many others.

A lot of studies were done in order to provide some data that will help manufacturers to decide which method to pick. For example Stig Oresjo from Agilent Technologies Loveland in Colorado published study that was comparing efficiency of AOI, AXI and ICT methods after the steps of the PCB manufacturing process mentioned in the Table 1.1

Tab. 1.1: When each inspection method was used

Process step	AOI	AXI	ICT
Post pick-and-place, Pre-flow	X		
Post-reflow, Pre-wave	X	X	
Post-wave		X	X

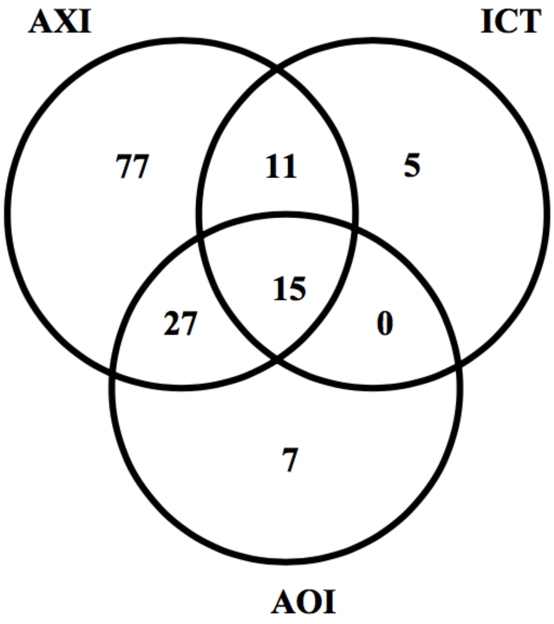

Fig. 1.5: Number of defects found by each method [18]

The Venn diagram on the Figure 1.5 shows how many defects were detected by each inspection method. You can see that the most effective method is AXI which detected 130 out of 142 defects. The Venn diagram also clearly shows that the AXI method is able to detect most of the defects detected by AOI, but not all of them. On the other hand there is a ICT method itself detected just 22% of the defects. [18]

1.3 Computer vision

Computer vision started in the late 1960s but the massive expansion started recently thanks to the increasing performance and decreasing price of the computers. Today, machine vision became a part of many fields starting with smartphones, through automotive industry up to PCB manufacturing industry trying to provide additional feedback about the surrounding environment.

X-ray inspection or optical inspection using the CCD camera will not be so beneficial without computer vision which goal is to mimic behavior of the human inspector. That means that the system should not only capture the environment in front of the camera, but also understand what is happening and make decisions based on the information extracted form the image. These systems have a lot of advantages over the human such as speed, no subjectivity or mistakes caused by fatigue and ability to work without any break. Even though the initial costs of the systems using the computer vision is very high it is beneficial in the long term especially in the high volume production.

1.4 What is defect?

The word *defect* is frequently used in this thesis therefore it is important to properly explain what is meant by this word. According to the Oxford Dictionary "defect" is defined as *"A shortcoming, imperfection, or lack"*. Which is very accurate definition even in the context of the printed circuit boards. In general, *defect* can be descried as a deviation from the norm that results in modification of the production process and prevent the defects that can affect the proper functionality or reliability of the PCB to prevent the costs connected with complaints. This implies that defect does not have to be necessarily something that prevent the PCB to work properly, but defect can be also an imperfection whose solving can make the overall manufacturing process more efficient and reliable.

2 AUTOMATED X-RAY INSPECTION

This chapter describes how is the x-ray image captured and characteristic properties of the automated x-ray inspection.

2.1 Basic properties of x-ray images of PCB

Automatic optical inspection is good for printed circuit boards with the visible joints, but recently a lot of PCBs are using technologies such as BGA, chip-scale package or integrated circuits where the potential defect is hidden under the body of the component. This is the result of miniaturization and the need for more pins on the integrated circuit. In these cases is AOI inadequate because it can check defects that are visible on the surface such as open-circuit, solder bridges or excess of solder whereas the AXI technology allows to check even part under the components. This is the reason why is the x-ray inspection required. The evaluation process is similar to the AOI, but the way how is the image obtained works on completely different principle. Instead of capturing reflected light from the surface of PCB AXI is measuring how much light got absorbed by the PCB. The x-rays are usually generated by the x-ray tube that pass through the object and on the other side is a sensor that captures the remaining x-rays and measures the intensity that is used to produce the x-ray image of the PCB. X-rays can interact with the materials through photoabsorption, Compton scattering or Rayleigh scattering. The strength of each effect depends both on the energy of the x-ray and the elemental composition of the material. The overall absorption deficient depends on the elemental compassion of the measured material, but it is not depended on the chemical properties. [19] In case of PCB scanning, the absorption coefficient of the solder paste is much higher than absorption coefficient of the board itself which manifests itself as the dark spots on the image.

Thanks to the rapid progress in the 2D X-ray inspection technology within last 10 years, today's systems are capable of extremely sharp and powerful X-ray sources with submicron feature recognition down to 0.1 micron or 100 nanometers. There is also a developmnent on the side of x-ray detectors that are providing resolution up to 16 Magapixels with 25 frames per second which results in faster and more precise detection capabilities. [8]

In my case I used x-ray machine from the company ELEDUS [1] called **SCIOX SMT** with the following specifications:

[1] `http://www.eledus.cz/sciox/`

Tab. 2.1: Specifications of the SCIOX SMT from ELEDUS

	SCIOX SMT
Utilization	Assembly control and development of electronics
Size (w x d x h)	710 x 760 x 1910 mm
The size of the scanned area	300 x 200 mm
The inner space for objects	560 x 560 x 560 mm
Detector type	CCD linear sensor with a high spatial and energy resolution
Detector resolution	15 lpm
Image resolution	18,3 Mpx
Used X-ray source	140kV / 3mA(focus 0,5 x 0,5 mm)
Weight	360 kg

This can be used to check if any of the solder balls is not missing or to check whether the solder ball contains voids which manifest itself on the resulting image as a brighter circular spots inside a solder ball.

2.2 3D X-ray Computer Tomography

Even thought Computer Tomography (CT) was invented in the 1971 by the Godfrey Newbold Hounsfield, it becomes recently more commonly used because the CT becomes cheaper and faster thanks to the increase in the computing speed capabilities. This method is very useful especially in case of multilayer PCBs because it can become very challenging for automated x-ray inspection. Components and conductors on each layer of the PCB cause interference which is caused because of the nature of the X-rays to penetrate through the all layers of the PCB. Especially looking for a shorts become very challenging on the 2D image of the multilayer PCB because the conductors on each layer overlap and the result image is chaotic. The solution to this problem is *3D X-ray Computer Tomography* which is a method that is commonly used in the medical field. The result of the Computer Tomography is a model that allows to create virtual cross sections, also called e-sections, at any plane of the PCB. This is providing a possibility to examine each layer of the PCB individually. The model of the board is made by capturing multiple 2D images from the known position and combining them using complicated mathematical formula. The detail of the model is depended on the number of captured images so it takes more time to get very precise model.

The typical setup of the computer tomography is visualized on the Figure 2.1. The scanned PCB is mounted between the x-ray source and the detector. By precise rotation of the PCB, multiple images are taken from the various angles. Afterwards all the images together with known position of the PCB are combined in the process called CT reconstruction. The result of that operation is a 3D density cloud that can be virtually sliced in order to get the required e-section of the circuit board. That will give us additional information that can be used to improve defect detection.

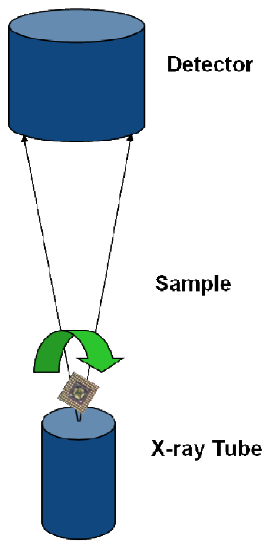

Fig. 2.1: Computed tomography setup [20]

2.3 AXI technology features

AXI inspection is usually placed after the soldering process so it can take advantage of it's ability to see through components and check the quality of soldering. It is able not only see through the chips, but it can also visualize the absorption deflection inside a solder ball. This way it can detect for example voids that are otherwise completely undetectable. So the x-ray method of screening is giving us additional information that can help manufacturers to ensure that the BGAs are being made up to the required standard.

Another feature of the x-ray inspection in the ability to measure parameters such as solder thickness and joint sizes or it can also visualize the heels of the joints on board which AOI systems can not see because they are hidden under the leads from the integrated circuit (see Figure 2.2). All these data can be captured and analyzed to reduce the fault level, improve the quality of the process or prevent the potential defects to occur. This is extremely useful when the new PCBs are manufactured to speed up the optimization process of the production and create less fault boards.

Last but not least AXI has become important part of the manufacturing process because of its ability to provide continuous, fast and in-depth feedback that is accurate enough to help maintain production of high quality reliable circuits.

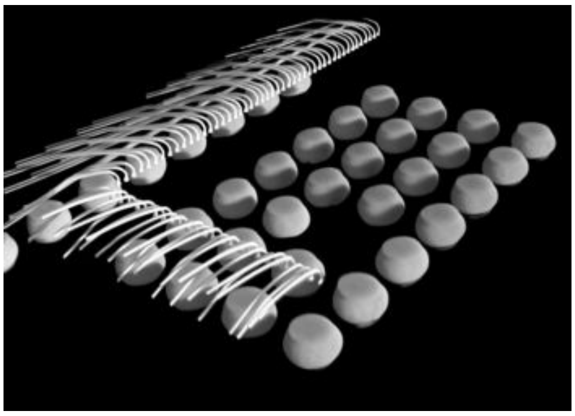

Fig. 2.2: CT reconstruction of BGA [21]

3 PCB DEFECTS

Due to the high density and smaller size of the components on the PCB, the process of manufacturing of these boards is more prone to the defects than in the past. In general, we can divide these defects into two categories. The first category of defects are *fatal defects* (sometimes called functional defects). This kind of defects can endanger the functionality of the PCB. This category includes for example short-circuit, open-circuit or missing hole. The second category are *potential defects* (sometimes called cosmetic defects). This group of defects do not affect the functionality right after the manufacturing process, but they can compromise the PCB performance during utilization due to for example over heating.

This chapter describes the most common defects that can occur on the PCB and how they manifest on the x-ray image. The chapter 3.1 will introduce defects that occur before the assembly and the chapter 3.2 will describe defects that occur after the assembly process. Because the biggest potential of the AXI technology can be utilized after the PCB is assembled so the bare PCB defects will be introduced just briefly.

3.1 Bare PCB defects

During the manufacturing process of the PCB (before the assembling) several defects can appear that are caused by an error in the process. This types of defects can be classified into defects caused by the missing copper and defects caused by the redundant copper. The source of this defects is usually dust, over etching, under etching or spurious metals.

The defects that are formed on the bare PCB are categorized into the categories described on in the table 3.1 and visualized in the Figure 3.2. [22]

FATAL	1 Breaks	1.1 Fracture
		1.2 Cut
		1.3 Scratches
		1.4 Cracks
	2 Shorts/bridges	
	3 Missing conductor	
	4 Incorrect hole dimension	
	5 Missing hole	
POTENTIAL	6 Partial Open	6.1 Mouse bit
		6.2 Nicks
		6.3 Pinholes
	7 Excessive spurious	7.1 Specks
		7.2 Spurs/protrusions
		7.3 Smears
	8 Pad violations	8.1 Under etching
		8.2 Over etching
		8.3 Breakout
	9 Variations between the printed lines	9.1 Small thickness wiring
		9.2 Large conductors
		9.3 Excessive conductors
		9.4 Incipient short (conductor too close)

Fig. 3.1: Classification of the bare PCB defects [22]

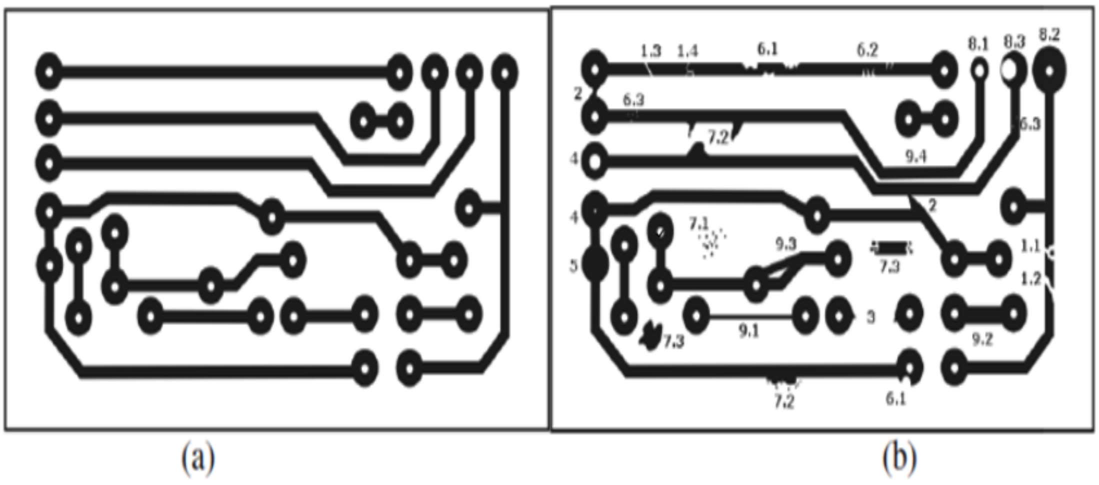

Fig. 3.2: Example of bare PCB defects [22]

These kinds of defects can be detected by using both AXI or AOI technology. In case of multilayer PCBs AOI is able to check only the outer sides of the board and in case of using the AXI technology it is possible to test all the layers separately by using the CT method. Depending on the complexity of the board it is also possible to use simple 2D x-ray image, but more complex boards will cause chaotic image caused by the overlapping of the conductors on each layer.

There are two ways how to detect these kind of defects. User can compare the test image with some reference image and analyze the differences or it is also possible to check the geometry of the board. Detection methods are analyzed into more details in the chapter 5.

3.2 Assembly defects

Assembly defects is a group of defects that occur during the assembly of components to the PCB. The detection of these defects is crucial because it usually prevents the end product to meet its criteria. Thought understanding of the root causes of the defect manufacturers can improve the quality of all assemblies. According to industry statistics, the top 3 PCB assembly defects which account for 74% of all manufacturing defects are opens, solder bridging, and component shift. [1]

3.2.1 Open Solder Joints

Open Solder Joint (sometimes called cold solder joint) occurs when there is no connection or poor connection between the lead and the pad which is causing an open connection. Some of these connection does not work at all, but some of them works but they are unreliable. The solder bond will be poor and the cracks may develop in the joint over time.

Detection of the is difficult using the top-down view so in order to detect open solder joint it is necessary to use a 3D CT or to change the viewing angle from top-down to angle between 55 and 70 degrees. On the figure 3.3 you can see how does the open solder joint manifest itself on the oblique x-ray image. The red arrows are pointing toward open pins and the green arrows identify good pins.

This kind of defect accounts for 34% of all assembly defects and it can be caused by lack of solder paste, gap between the PCB and component or by corrosion at component lead. It can be prevented by trying to avoid paste contamination or ensuring proper heated solder paste. It can be repaired by re-heating the joint with a hot iron until the solder flows or adding more solder.

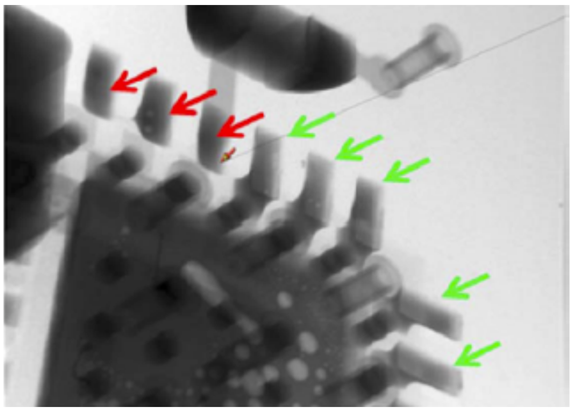

Fig. 3.3: Open solder joints of the QFN package [28]

3.2.2 Solder bridges (shorts)

Solder bridges can occur in case of melting of two solder joints together which cause unintended connection. Shorts can be microscopic so they can be very difficult to detect visually and it can potentially cause serious damage to the components such as burn-out or it can damage a PCB by burning-out of the trace and causing open circuit. For detection of this kind of defect it is possible to use 2D top-down view. For example, on the figure 3.4 you can clearly see that two solder balls in the BGA melted together underneath the component package.

This defect can be caused by applying too much solder on the pads, misalignment between the stencil and PCB or because of the soldering pads are too big in compassion to the gab. In some cases this defect can be repaired manually by scraping out unwanted connection or dragging the tip of a hot iron between the two solder joints. If there is too much solder, a solder sucker or solder wick can help get rid of the excess.

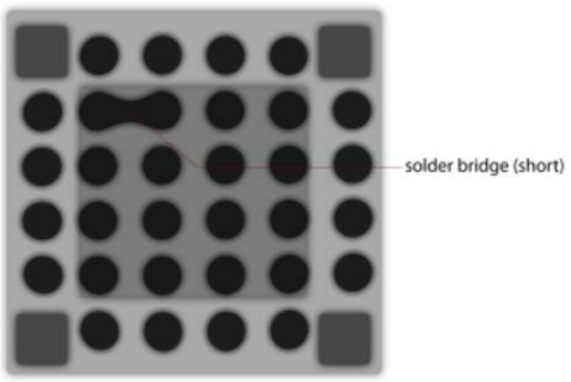

Fig. 3.4: Solder bridge of two solder balls [?]

3.2.3 Component Shift

Component shift manifests itself as a misalignment between the component and the solder pad. It usually happens during the reflow soldering due to the components ability to float on the molten solder. It can be caused because of the oxidation of component leads, bent leads, vibrations and in case of placing small component next to the large component where the heated gas is directed from the side of big component towards the smaller ones. Component shift can be prevented by minimizing of the movement of the unreflowed assembly boards, using more aggressive flux and compliance the requirement temperature and humidity.

On the figure 3.5 you can see example of shifted component. This kind of defect can be detected by comparison of top-down view of the reference image of the PCB with the tested image.

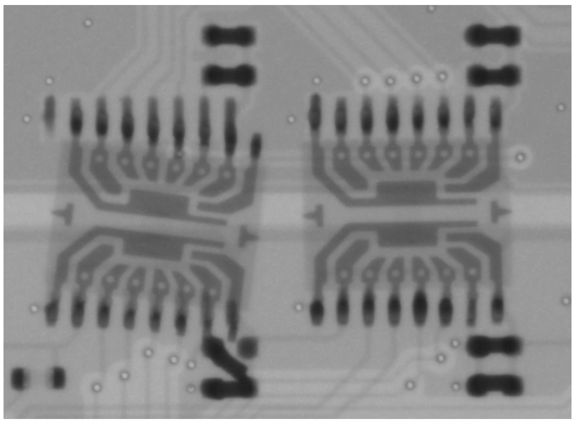

Fig. 3.5: Shifted component

3.2.4 Tombstone

Tombstone is an extreme version of component shift. It occurs when one side of the surface mount technology (SMT) component is properly soldered to the the PCB pad and the other one stands up vertically. There are several reasons why it can occur, one of them is different wedding speeds that can cause imbalanced torque on each side of the component. The uneven oven temperature, nitrogen presence or the uneven solder paste printing can increase the occurrence of the tombstone defect.

On the figure 3.6 you can see example of the tombstone defect. This kind of defect can be identified using the top-down view of the board because it also usually cause shift of the component to the side, but the better option is to use the CT or oblique x-ray image.

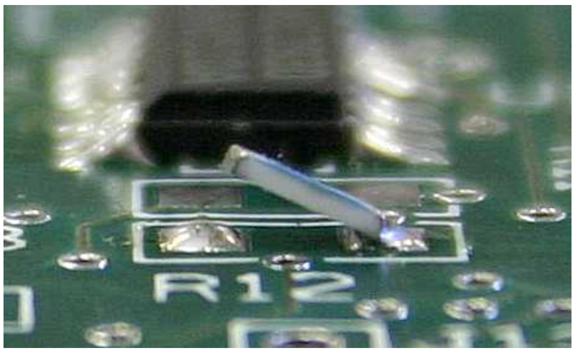

Fig. 3.6: Tombstone [29]

3.2.5 BGA Voids

BGA voids or voids in general occur when using the surface mount technology during the rewlow process. Enclosed voids can cause displacement of electrical or heat paths that can lead to local overheating. Gas voids usually form balls inside the solder ball which could lead to tilting of the component. The void occurrence can be affected by the e.g. a good wettability of metallization, solder pastes with special adopted solvents or an adequate preheating profile.

As you can see on the figure 3.7 BGA voids manifest itself on the x-ray image as a bright spots inside the solder ball. This kind of defect can be detected by segmenting each solder ball and calculating the ratio the surface between the solder ball and the void area. This can be done by using the top-down image of the PCB, but that will give us just the area of the void so for the more precise calculation of the volume of the void it is necessary to use the CT scan of the board. For more information about the BGA void detection read the chapter 6.1 where is the problem described into more detail.

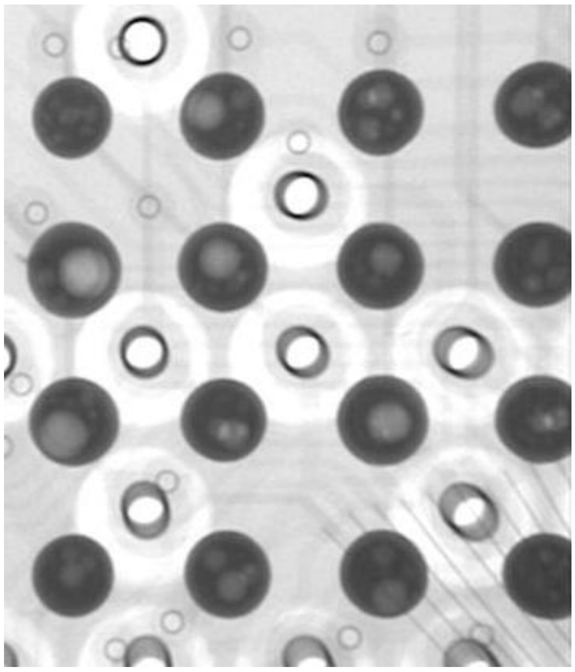

Fig. 3.7: Voids inside the solder balls

4 PROGRAMMING LANGUAGES FOR COMPUTER VISION

This chapter will introduce programming languages that are mostly used in the field of the computer vision. It does not make any sense to mark one of these languages as the best one for the machine vision because it mainly depends on the task and how it will be implemented in the praxis.

4.1 Matlab

Matlab is a standard in the academic world due to its easy syntax and debugging so it is great for prototyping of algorithms. The great advantage of being an academic standard is enormous amount of code that can be found from other researchers. On top of that Matlab has one of the best documentation with many examples so it is very easy use.Matlab is a matrix engine. The great advantage of that approach is that when you get use to the coding in "Matlab style", which is different from the programming style of general purpose languages, you will get very good performance with just few lines of code. In case you you do not write code the MATLAB way, your code can become extremely slow.

Matlab is a great tool when it comes to the university researches, but it very complicated and expensive to put it to the production. It is very expensive to buy a license for commercial use of the Matlab for example if you would like to write this thesis in Matlab and sell it, you will have to buy a Matlab R2016b license for 2 000 EUR, Computer Vision System Toolbox for 1 250 EUR which requires Image Processing Toolbox for 1 000 EUR. And in case you want to compile it to stand alone executable you have to pay 8500 EUR. So this approach makes sense for big companies or universities. [4]

There is also a memory or performance problem which might be a big problem especially in the computer vision field which is very memory and power demanding. Generally a typical Matlab program runs many times slower than C++ program, but it is possible to compensate it by using the *MEX Files* that will enable to write computationally intensive parts in C, C++ or Fortran and call them as if they were built-in functions. [3]

Matlab supports a lot of libraries (toolboxes) that are focused on computer vision and machine learning. For instance *Image Processing Toolbox* that provides comprehensive set of basic algorithms for image processing such as image enhancement, image segmentation, geometric transformations and many more. Another toolbox connected with image processing is *Computer Vision System Toolbox* which provides advanced function e.g. feature detection, extraction, matching, object tracking, object recognition and also it supports 3D point cloud processing. On top of that Matlab also brings possibility to interface with *OpenCV* using the Matlab's OpenCV interface. [5]

4.2 C++

This language is widely used in production-grade computer vision projects because most of the machine learning and computer vision libraries are supporting this language. It also has a great performance results when it comes to "loop on all pixels" so together with the OpenCV library it is very common combination for the production version.

Although quite few libraries that are focused on computer vision such as Halcon, Matrox Imaging Library, Open eVision, Adaptive Vision Library or Common Vision Blox the unofficial standard on this field is OpenCV mainly because it is completely free for commercial applications, you can view the source code and you do not have to open source your project. OpenCV library has more than 2500 optimized algorithms which includes basic image transformations, object recognition and tracking, produce 3D point clouds from stereo cameras or classify human actions in videos. It also have a big community and supports other programming languages such as C, Python and Java. OpenCV 3 also supports *Transparent API* which is a way to add hardware acceleration and some of the algorithms supports *CUDA*.

The disadvantage of C++ is that it is difficult and time consuming to develop new algorithms especially when it is used by inexperienced coder. Also the documentation of OpenCV is bad so sometimes you need to have a good understanding of the function or read some paper to find out effect of some parameter of the function to the result.

4.3 C sharp

C# is a general-purpose, object-oriented high level programming language. Considering computer vision it is very similar Python because it offers a wrapper around the C++ OpenCV classes/functions. It also offers another computer vision libraries such as *AForge.NET* which is a computer vision and artificial intelligence library developed by Andrew Kirillov and *Accord.NET* which is a machine learning framework.

4.4 Python

Python is somewhere in the middle ground between the Matlab and C++. With libraries such as *numpy*, *scipy*, *scikit-learn* and *matplotlib* Python provides a powerful environment for both commercial use and research on universities which results in the great active community. Language itself is easy to learn (especially in comparison with C++), it is fast for prototiping of new algorithms and on top of that you will write much less code in compassion with the C++.

Considering the performance it is not so efficient as a program written in the the C++. But since the Python's OpenCV is just a wraper around the original C/C++ the difference between calling OpenCV function in Python and C++ is minimal. The main performance difference will occur when the user needs to write some function that is not implemented in OpenCV the performance of the Python program will get considerably slower especially when it will use looping through the big arrays instead of array manipulation facilities available in *Numpy.* There is also a possibility to use Python's version of OpenCL called *PyOpenCL* which lets you access the OpenCL parallel computation API from Python.

Disadvantage of python in comparison with the Matlab is that a lot of the libraries are poorly documented so it might be sometimes difficult to understated it. In these cases user have to find the answer on forums or some papers that deal with the same problem. But in contrast to the Matlab, Python also offers much more libraries that are focused on computer vision such as *mahotas*, *scikit-learn*, *ilastik*, *SimpleCV* and already mentioned *OpenCV*. Very interesting library is SimpleCV which should make learning curve with the computer vision much faster for beginners. [5]

4.5 Why Python?

I did not choose the Matlab for this project because my script will be used in the commercial product and the licenses for Matlab would add unnecessary cost moreover the user interface is written in the C# so it would be necessary to integrate it into a C# application or translate it to different language.

Implementation of the user interface in the C# offers the possibility to write the script also in the C#. The reason why I did not picked this language is that I am not familiar with this programming language and also C# is not commonly used for computer vision together with the OpenCV which could result in a lot of troubles because I do not have a deep knowledge of the computer vision or the OpenCV.

If we compare the C++ and the Python from the performance point of view, C++ is the number one choice. Even though at the start of this thesis I had similar experience with both of the languages I picked Python because of it's syntax, the coding style and mainly because of it's learning curve because at the beginning of this thesis I had no idea how difficult this project is so I wanted to make sure that I will not struggle with both programming language and computer vision.

5 DEFECT DETECTION METHODS

There are a lot of ways how to detect defects on the PCB. This chapter will discuss the most common approaches of defect detection using x-ray inspection. The Figure 5.1 shows how are the methods structured. There are three main categories of the methods based on the source of the information that leads to identification of the defect. The first category of defect detection methods is called *Reference comparison* that compares the tested image with some image that is considered as perfect. The second category is *Non-reference comparison* that uses usually some general design rules of the PCB and uses them to look for parts of the board that violate these rules.

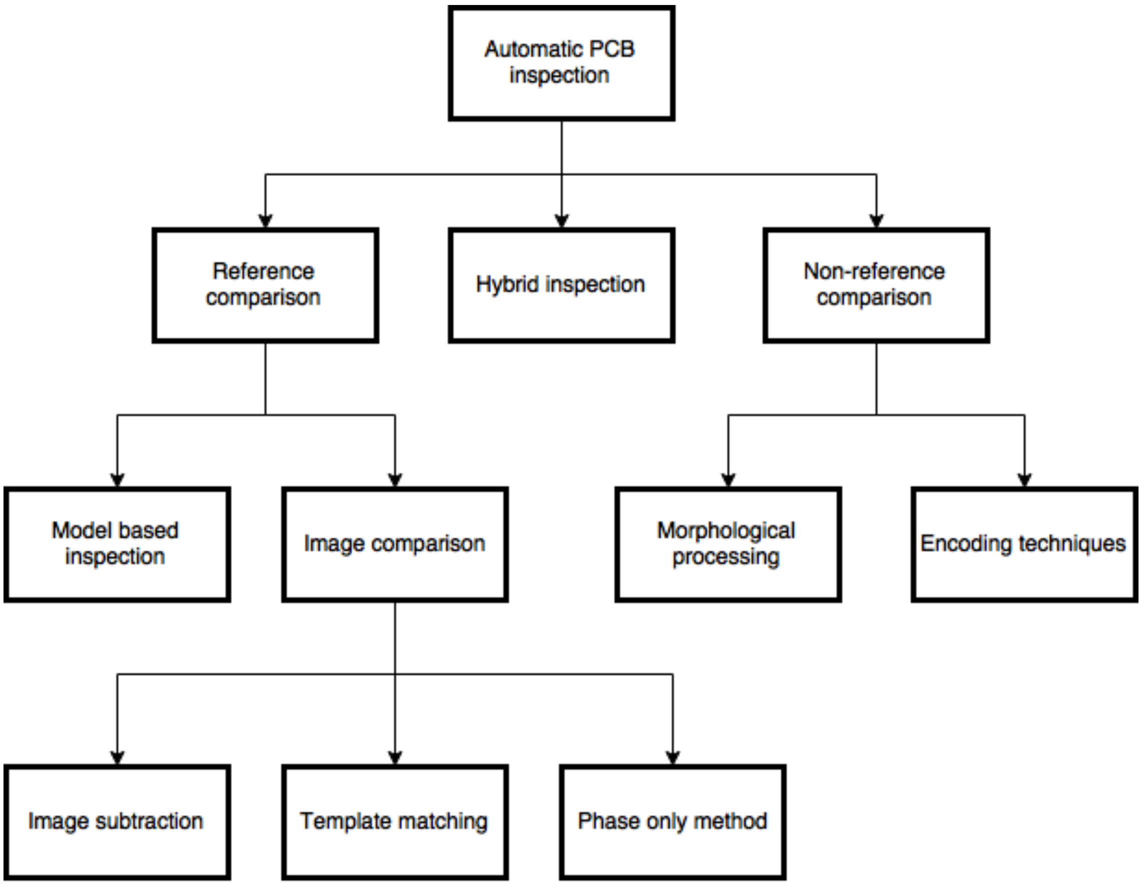

Fig. 5.1: Automatic PCB inspection methods [23]

5.1 Reference comparison

Reference comparison methods are based on comparing a referential image of the PCB with the image of the PCB that we want to test. Reference image of the image of PCB that passed all tests and does not contain any significant defect. This image is afterwards compared with the test image and the difference is considered as a potential defect and can be used for the further classification. The most crucial part of the comparison process is the image alignment which have to be very accurate because every misalignment will be source of the false defects.

The reference comparison method can be used to detect the following defects:

1. Missing component or solder paste
2. Over-etching/Under-etching of conductors
3. Bridging faults
4. Oversized or undersize solder paste
5. Dislocation faults
6. Deformed pads

There are several ways how to get a referential image. The most straightforward method is to find out the PCB board without any defect using other defect detection methods and use its image as a reference that can be used for the comparison. The second approach is to take advantage of the designing process of the PCB and use some of the resulting CAD files as a reference. The most used is a Gerber file which is an open ASCII vector format for 2D binary images. The advantage of this approach is that you will always get the most accurate reference image and no testing is necessary to make sure that the reference image is without any defect.

This approach was used in the function that is described in the chapter 6.3.

5.2 Non-reference comparison

Non-reference comparison methods does not need any referential image instead of that it uses general designing rules of the PCB to determinate whether the image has defects. This method works with the assumption that features are simple geometrical shapes and the defects cause irregularities. These method typically use morphological technique such as erosion and dilation as a basic operation. This method can be used to find voids in the BGA or to recognize some components based on their characteristic geometric shape.

For more information about this method read the description of the function that is described in the chapter 6.1 which describes the implementation of void detection in the BGA.

5.3 Hybrid inspection

The hybrid method is the combination of the both methods mentioned above. It takes advantages of both approaches to overcome some shortcomings. Hybrid systems make use of both methods and complement each other so they are able to cover large variety of defects.

6 IMPLEMENTATION

This chapter will describe the implementation of three defect detectors. The first two of them (BGA void detection and QFN void detection) are using the *non-reference comparison* method. These algorithms are autonomous and after initial setup of input parameters it can be used to automatically check the solder balls on the PCBs without any help of the operator. The last algorithm is using the *reference comparison* method. This method is not fully autonomous and needs operator to decide whether there is a defect or not. The best use of this algorithm is as a auxiliary tool that can help to reveal the defects faster.

Because the images from different x-ray machines or images of different PCB boards can have different properties I decided to use configuration files for entering the parameters that can affect the result of the program. In this way the program becomes very flexible because instead of changing all the arguments for each image, user can easily create configuration file for each configuration and change just the path to the configuration file.

6.1 BGA void detection

The goal of this function is to find all solder balls in the BGA package and calculate a ratio between the area of the solder ball and void for each solder ball in the ball grid array. The function is divided into two steps. In the first step it is necessary to implement robust function that will segment the solder balls from the image. This function have to be very robust because solder balls on the one side of the board can overlay with the components on the other side of the board. In the second step the program will iterate through each solder ball and calculates the ration between the solder ball area and the void area.

The example of the voids in solder ball is on Figure 6.1. On the left image you can see easily visible voids, but the right image has poor contrast so the voids are very difficult to detect for the human eye. The algorithm that is described in this chapter can work even with the images with poor contrast.

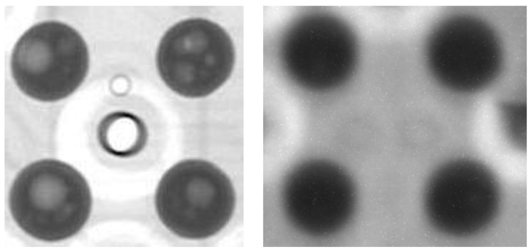

Fig. 6.1: Example of the voids on the x-ray image

6.1.1 Solder ball segmentation

Because the solder paste has a high absorption coefficient, solder balls appear on the x-ray image as a dark circles whereas the conductors and the board itself does not absorb so much x-ray so they are much lighter on the x-ray image. The initial task in the solder ball segmentation is to separate components (including solder balls) from the board itself which means that it is necessary to find some threshold value of the pixel intensity to separate dark regions from the bright ones. The most common method to separate background from the foreground is the thresholding. Because the 2D x-ray image of the PCB can be considered as the image with the bi-modal histogram which means that if we look at the histogram of the gray scaled image of the PCB we can see the same pattern that is repeating. The histogram has two peaks, the first one is around intensity 35 that represents dark pixels and the second peak that is usually has much higher number of pixels is around intensity 170 (see 6.2). These two peaks represent dark pixels of solder balls and components and bright pixels are the conductors and the board itself. For this image we can approximately take a value that is in the middle of these peaks. This can be done automatically by threholing method called *Otsu's Binarization.* [11]

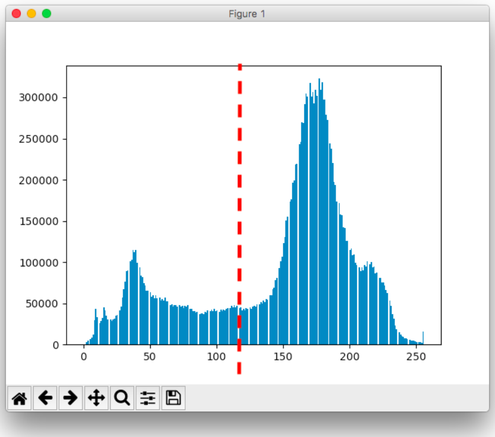

Fig. 6.2: Histogram of gray scaled xray image

Afterwards we have to find all circles on the resulting image and decide if they are solder balls or not. To find solder balls on the image, user can use the function called *find_solderballs(...)* which will get the image that contains the the BGA as the only mandatory parameter and several other parameters that are not mandatory, but they might be useful in case of using the x-ray image with different properties than the image obtained from the *SCIOX* x-ray machine.

1. **input_image** is the only mandatory parameter. It should be color image that contains BGA
2. **method** is an optional parameter that will decide what fuction will be used for detection of circles on the image. The default value is *CircleDetectionMethod.BoundaryRectangles*. The method have to be entered using the custom enum type called *CircleDetectionMethod*.
   At this point two methods are implemented. The conventional circle detection algorithm is Hough transform that can be used using the *HoughtCircles*. This circle detection method is a basic technique in the computer vision for detecting the circular objects in the image. [6] [7] The disadvantage of this method is it's low performance for images that contains a lot of potential circles after initial thresholding. Another disadvantage is that this function has four input parameters[1] that can affect the both computation time and the accuracy. This method would work much faster with known radius of the circles that it is looking for because the parameter space dimension would increase to 2D.
   The second function that could be used to find a circles on the image is called *BoundaryRectangles*. This function is taking the binary image that was ob-

[1]Read more about the parameters of the HoughCircles on `http://docs.opencv.org/3.0-beta/modules/imgproc/doc/feature_detection.html#houghcircles`

tained by thresholding and finds a contours of the components on the PCB. Afterwards it will find boundary rectangles of these contours and compares the ratio between the width and height of the rectangle. In case the ratio between the width and height is approximately 1 we can consider the object inside as a square or as a circle. This method is very simple and extremely fast (see Table 6.1) even with the high resolution images and the number of false detection is much better that using the Hought transform because Hought transform is able to find also circles that are partly covered by some other object. Because we would filter out these partly visible circles anyway the second method is the recommended one to use.

3. **closing_kernel_size** is a parameter from configuration file. It is used for filtration of the binary image that was obtained by the thresholding function. The default value is tuple (5, 5) that should work in most cases. This parameter defines the size of kernel of the morphological function closing that is used to remove small holes (dark regions) in the image.
4. **closing_iterations** is a parameter from configuration file. It is also used for the morphological closing. The default value is 2 and it defines how many times will be morphological closing used.
5. **radius_histeresis** is a parameter from configuration file. It is used for a filtration of small and big circles on the image. The algorithm that is implemented will find all circles on the image. The default value is 0.2 which means that after performing initial filtering, it will calculates median radius of the circles and remove circles that are smaller than *0.8*average_radius* and circles that are bigger than *1.2*average_radius.*
6. **intensity_ratio** is a parameter from configuration file. This parameter is used for the filtering purposes. The default value is 25 which means that it checks if every detected circle has ratio between the white and black pixels inside the circle and if the circle has more than 25% of the white pixels it is removed because the solder ball should be black.
7. **fxy** is an optional parameter that is used for visualization purposes. Big images might not fit to the screen so it is necessary to resize the image by the fxy which is a scale ratio that will be used to resize the width and height.
8. **debug_mode** is an optional parameter that is used for debugging. The default value is *False*, but when it is set *True* it will show every step of the image processing. This can be used when user have problems with detection of the circles on the image because it will help you to find in which step of the image processing is the problem.

One of the steps during the segmentation is a filter that is removing false solder balls. This filter can be tuned using the arguments *intensity_ratio* and *ra-*

Tab. 6.1: Comparison of the Hough transform method and the Boundary Rectangles method

Image size [px]	**Hough transform [sec]**	**Boundary rectangles [sec]**
600x600	0.3137	0.0029
4000x4000	73.080	0.0572
2300x2300	13.689	0,0188

dius_histeresis. False detected solder balls are usually caused by square components (in case of the *BoundaryRectangle* method), by vias (that are aslo circular, but they does not absorb so much x-rays so they does not appear so dark on the image) or by other components on the board. Both methods for circle segmentation usually find several false solder balls. Especially the method *HoughCircles* returns a lot of false solder balls so it is necessary to come up with some criteria that will filter out obviously false solder balls. There are implemented three conditions. For the filtering out the circles that are not surrounding solder balls the following set of rules is used.

1. The whole contour of the circle have to be in the image.
2. The number of white pixels can not be higher than number of black pixels. User can define manually the ratio between the white and black pixels or leave the default value which is set to 25%
3. After using two previous filters the median of circles found is calculated and all circles that are much smaller or bigger are deleted. The default hysteresis of circle size set to 20% which means that circles that have smaller radius than 80% of average radius and circles that have bigger radius than 120% of the radius are deleted.

At this stage most of the true solder balls should be found, but some PCBs have a lot of components on the other side of the desk that are interfering with the BGA and that can force user to set the filters too strict so there will be some missing solder balls. To ensure the robustness the algorithm is taking advantage of the fact that the solder balls are placed in the 90° grid. To take advatage of that the Delaunay triangulation is used (see Figure 6.4). It will take all centers of solder balls and connects them with triangles such that no point is inside of any triangle. Than the lines that connects the centers of the solder balls and rotate them three times 90 degrees and test if any of this position already contains some circle and in case it does not, then I check if the whole contour of the circle lies on the image and the ratio of black and white pixels.

On the Figure 6.4 and Figure 6.3 you can see the intermediate results of each step of solder ball segmentation. The first image is the output 2D image that is captured by the x-ray machine. The second image is the binary image after the thresholding and the third image is result of morphological closing that removes small holes in the image. In the fifth image you can see all circles that were detected on the image and in the next step the filtering is used to remove circles that does not surrounds solder ball. On the penultimate image you can see the triangles that ware obtained using the Delaunay triangulation and in the last image is the result of the solder ball segmentation.

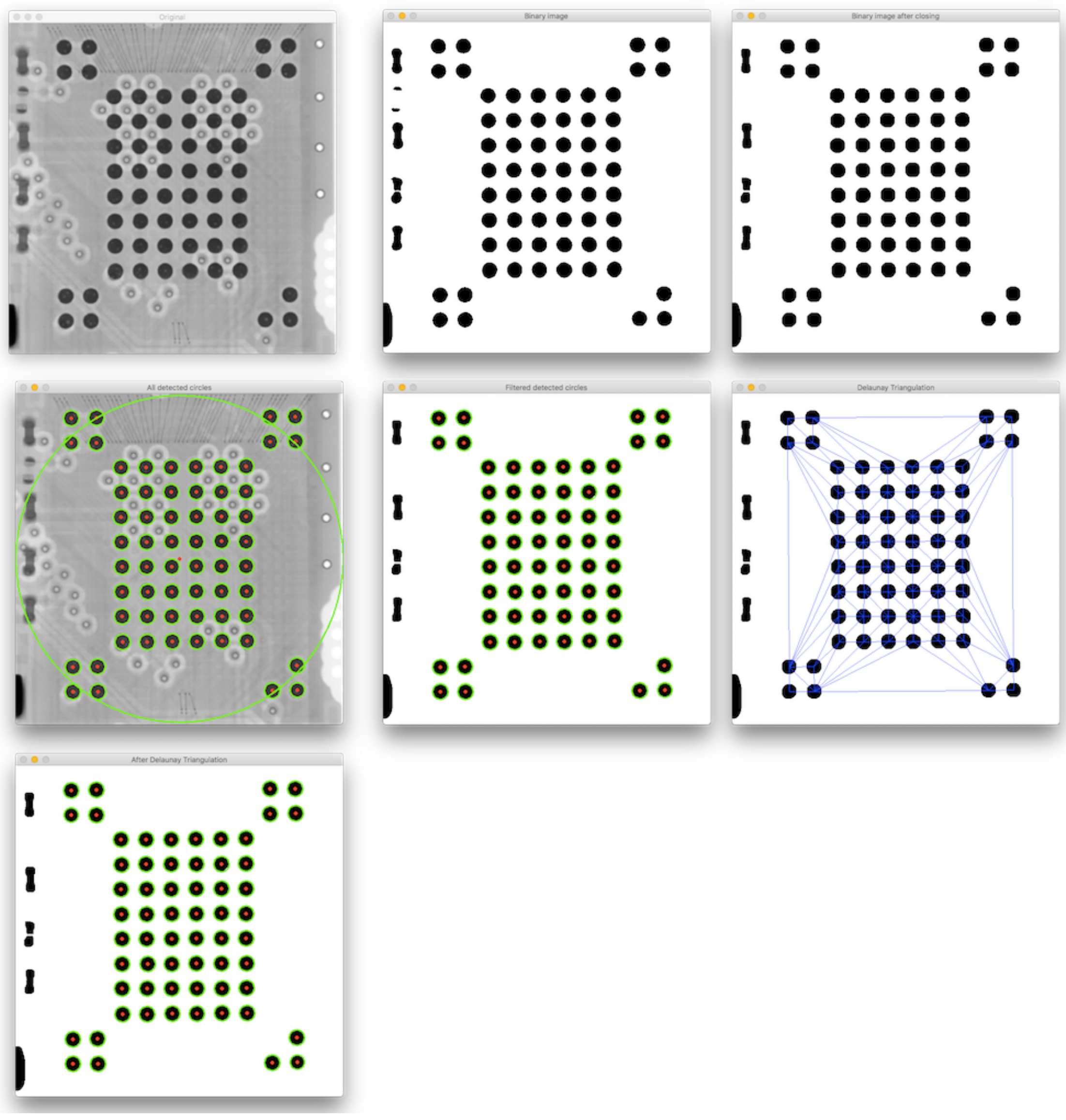
Original
Binary image
Binary image after closing
All detected circles
Filtered detected circles
Delaunay Triangulation
After Delaunay Triangulation

Fig. 6.3: Step by step process of the solder ball segmentation using the *BoundaryRectangle* method

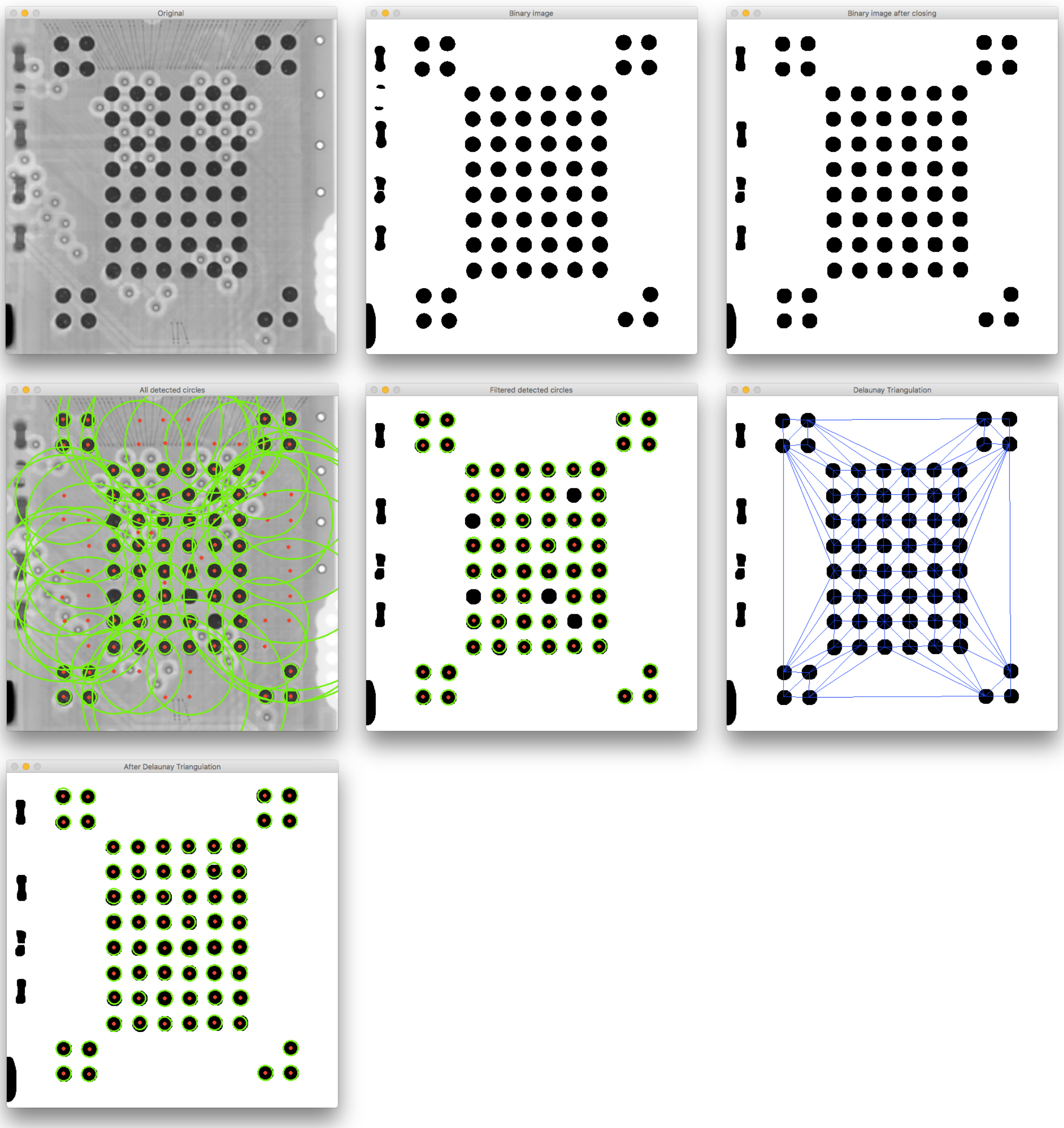
Original
Binary image
Binary image after closing
All detected circles
Filtered detected circles
Delaunay Triangulation
After Delaunay Triangulation

Fig. 6.4: Step by step process of the solder ball segmentation using the *HoughtCircles* method

6.1.2 Void area calculation

The process of image processing of the void detection can be very tricky due to low contrast and possible interference with the components on the other side of the board. To find voids inside each solder ball user can use function called *solderballs.analyze_voids(...)* which has two mandatory arguments and 6 optional arguments.

1. **image** is the first mandatory argument that contains a color image that will be used for segmentation of voids

2. **method** is the second mandatory argument that receives object that will represent the method that will be used to get the contour of the void and parameters for that method. I decided to use this approach because each method that could be used receives different arguments. If I would use conditional logic (e.g. user would input a method name using the string, number or enum) and than I will use if-else statement, it would lead to a lot of arguments that will do anything. You can imagine that on the following example. If I implemented *method1* with aruments *arg1* and *arg2* and *method2* with arguments *arg3* and *arg4* and user would pick *method1*, the arguments *arg3* and *arg4* would be useless. On top of that if you would want to tune this function, you would have to find out what arguments really affect the method that was picked. In order to make this choice more user friendly I replaced conditional logic with polymorphism which means that I implemented a class for each method that contains the necessary arguments for the particular method and method itself for detection of the edge of the void. So the user have to create the object with the necessary variables for particular edge detector using *MethodLoG(gaus_blur_kernel, laplace_kernel)*, *MethodCanny(threshold1, threshold2, apertureSize, L2gradient)* or *MethodDoG(blur_kernel1, blur_kernel2)* and afterwards input that object to the function *analyze_voids()*.
3. **clahe_clip_limit** is a parameter from the configuration file that is used for enhancing contrast. The default value is 4 and it affects the contrast limit.
4. **clahe_tile_grid_size** is a parameter from the configuration file that is also used for enhancing of the contrast. The default value is tuple (5, 5) and it defines in how many blocks (titles) will be the image divided.
5. **closing_ksize** is a parameter from the configuration file that is used for morphological closing. The default value is tuple (2, 2) which defines the size of the kernel that is used for removing of the holes (dark spots) on the image.
6. **closing_iterations** is a parameter from the configuration file that is used for morphological closing. The default value is 2 which defines how many times will be morphological closing used.
7. **fxy** is an optional parameter that is used for visualization purposes. High resolution images that will not fit to the screen have to be resized and the fxy is a scale ratio that will be used to resize the width and height.
8. **debug_mode** is an optional parameter that is used for debugging. The default value is *False*, but when it is set *True* it will show every step of the image processing. This can be used when user have problems with detection of voids inside the solder balls because it will show you in which step of the image processing is the problem.

The images that I have available are mostly low contrast and suffers from salt and pepper noise. So the first step in the image processing is using the median filter that is very effective in suppression of the salt and pepper noise. The second step is to increase the contrast of the image. To enhance the contrast of the image the global contrast enhancement alone is not sufficient so I have decided to use the Contrast Limited Adaptive Histogram Equalization (CLAHE) which is an image processing technique to improve local contrast in images. The main advantage of CLAHE is that unlike the ordinary histogram which calculates the transformation based on the image histogram of the whole image, this method computes a transformation from a histogram derived from the neighborhood of the pixel so it is more effective to enhance the local contrast. On the Figure 6.5 you can see step by step process of the image image void segmentation and on the Figure 6.6 you can see the result.

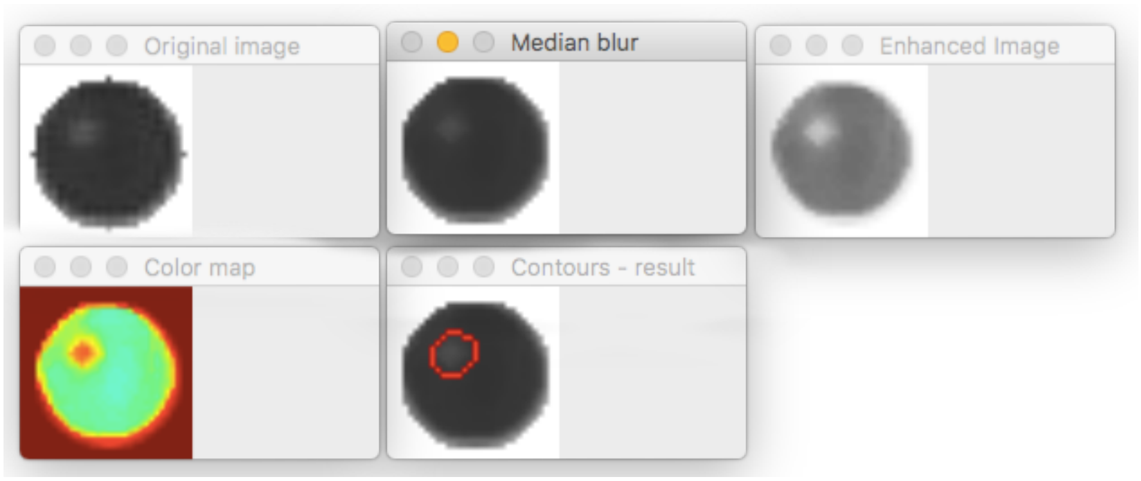

Fig. 6.5: Image processing steps of the void segmentation

After image enhancement we have to find the borders of the voids. Voids are typically circular bright spots inside the solder balls so to find their border we can use one of the many edge detection algorithms. I have implemented Canny edge detector, Difference of Gaussians (DoG) and Laplacian of Gaussian (LoG) and got the best results with the LoG algorithm with the proper size of the filter. This edge detector usually detects also the borders of the noise so after the edge detection I use the morphological closing to get rid of the detected noise. In the last step the area of all voids inside each circle is calculated and divided by the area of the solder ball to get the ratio between void and solder ball area.

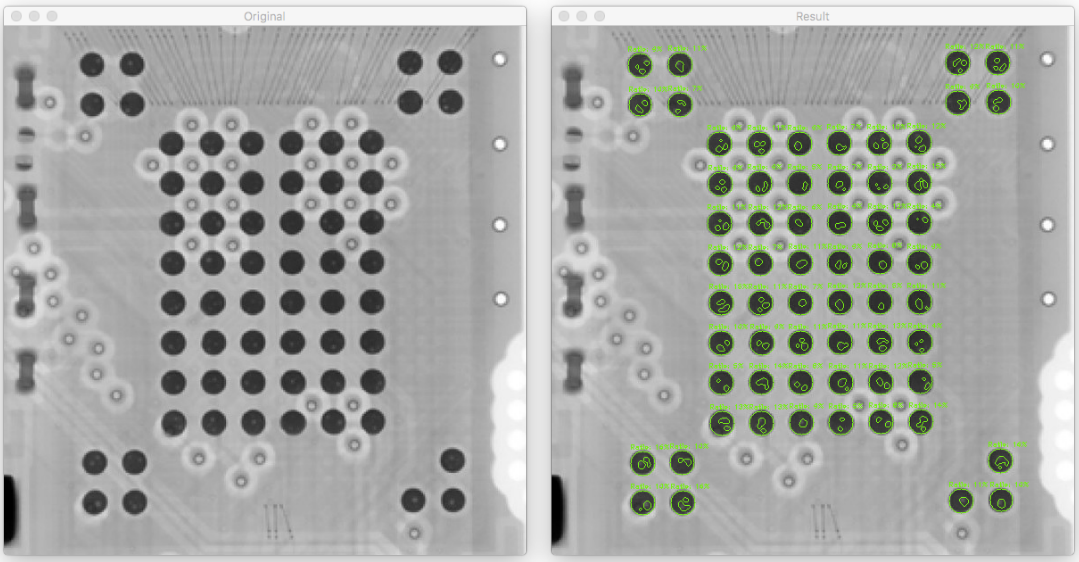

Fig. 6.6: The result of the void segmentation

6.1.3 Program usage

The solder ball detection works in three steps. In the first step user has to call a function *find_solderballs(...)* that can be found in the python module */utilities/find_functions.py*. This function look for the solder balls on the x-ray image and returns an *SolderBalls* object that inherits from the *Component* object and stores all necessary variables that defines the location of the solder ball. Afterwards user can use the the method of the object *SolderBalls* called *find_solderballs(...)* that has two mandatory arguments. Image that will be used to find voids and the method that will be used to find void inside the solder balls (for more information about the arguments of this function read the chapter 6.1.2). This function will find the voids inside each component and adds the contours of the voids and it's hierarchy to the *SolderBalls* instance. The last step is the vizualization of the voids using the method *show_all_voids(...)* where user has to input the image that will be used for the visualization and the *error_ratio* that will visualize all solder balls with the void ration higher than the entered *error_ratio* red and writes the percentage above each solder ball.

The working example of the void detection algorithm can be found in the root directory of the program in the folder *root/example/voids_inside_solderballs.py*. All available images that can be used by this algorithm can be found in the attached folder *Images/Solder balls/Original images*. On top of that you can find more results of this algorithm in the chapter .2 or in the enclosed file *Images/Solder balls/Results* where you can find input and result images together with the configuration file so you can test the program yourself without spending too much time tuning the input arguments.

```
import cv2
from Ultron.enums import CircleDetectionMethod
from Ultron.utilities import find_functions
from Ultron import edge_detection


image_path = '.../images/figure.jpg'
config_file_path = '.../images/config.ini'
image = cv2.imread(image_path)
solderballs = find_functions.find_solderballs(
    image,
    method=CircleDetectionMethod.BoundaryRectangles,
    configuration=config_file_path,
    debug_mode=False,
    fxy=1)
method = edge_detection.MethodLoG(
    gaus_blur_kernel=5,
    laplace_kernel=15)
solderballs.analyze_voids(
    image,
    method=method,
    configuration=config_file_path,
    fxy=3,
    debug_mode=False)
solderballs.show_all_voids(image, error_ratio=25)
```

Listing 6.1: Solderball segmentation and void detection

6.2 Thermal pad void detection in the QFN package

Quad Flat No-leads package also known under the shortcut QFN is a package type that connects integrated circuits to the PCB using the surface-mount technology. This kind of chip is encapsulated in the plastic package with a planar copper lead frame substrate. The package can have rectangular or square shape with the pins that provide electrical connection on the perimeter of the package (see Figure 6.7). In order to enhance the electrical performance of the integrated circuits it is necessary to effectively remove the heat from the package. This is achieved by the soldering the thermal pad to the PCB with the minimum of the voids. This is very difficult to achieve especially because of the large size of the thermal pad and the presence of thermal vias. Because the heat removal is necessary for achieving the high reliability of the resulting PCB it is essential to ensure the minimum void area. This can be done only by using the AXI methods. [12] [13]

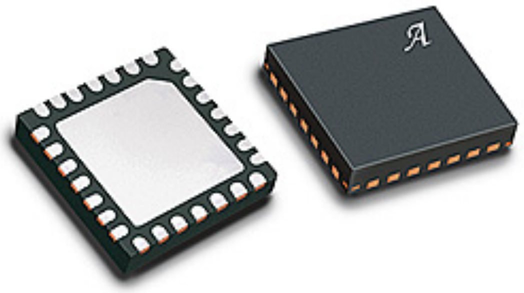

Fig. 6.7: QFN package

6.2.1 QFN package segmentation

The segmentation of the QFN package consists of two steps. In the first step it is necessary to filter out the background including the board itself and the small components. This is done in the very similar way as the segmentation of the solder balls that was described in the previous chapter so the process of component segmentation will be described very briefly. Because the x-ray image of PCB has a bi-modal histogram we can use OTSU thresholding for background suppression. The only difference between the segmentation of solder balls and the QFN package is that the morphological closing is used with large kernel and high number of iterations. This parameter is dependent on resolution, contrast of the image and the type of x-ray machine that was used for the capturing of the image. In case of the SCIOX x-ray machine we use kernel size eleven and the number of iterations also eleven. This way we can eliminate small components on the board (see Figure 6.8).

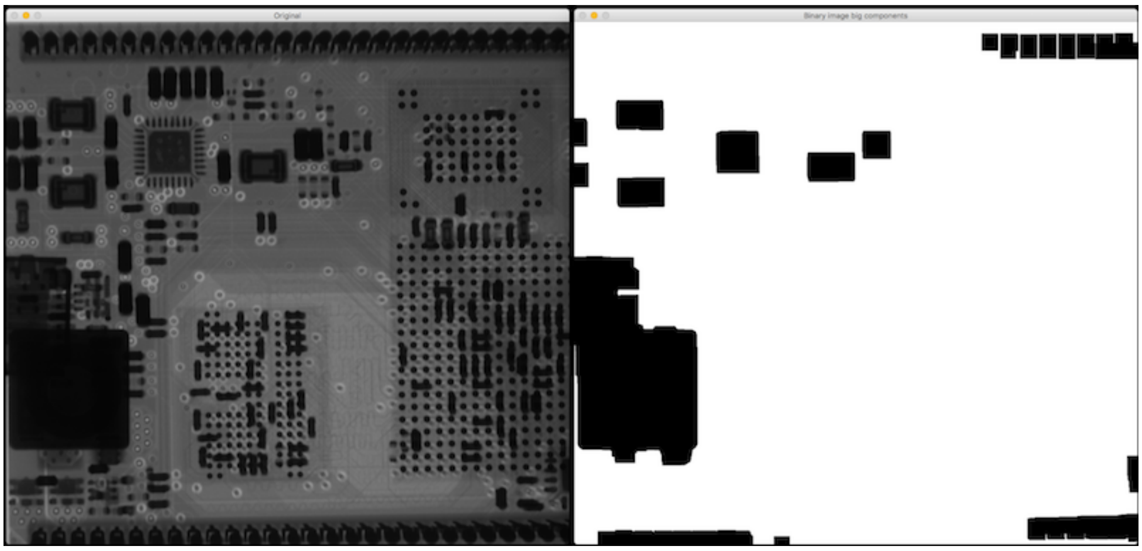

Fig. 6.8: Segmentation of big components on the PCB

In the final step of the QFN package segmentation, the program iterate through each component that was left on the board and test whether it meets all conditions that are defined by the user.

1. The default condition that has to be met is that the center part of the QFN package has to be surrounded by pins. This is tested by taking the center part of the package that was detected in the previous step and look for the pins in the close neighborhood. This is tested by counting number of intersections of line that is formed by the centre of pin and the centre of the component with the each edge of the potential QFN package (e.g. it counts how many times this line intersects with the top edge of the component, bottom edge of component etc. see Figure 6.9). If the number of intersections of each edge is equal or larger than minimum number of intersections defined by user using the argument *min_intersection_count* in the function *find_functions.find_qnf_package(...)* the condition is fulfilled.
2. The optional condition that can be set by the user is the shape of the QFN package. If the qfn package that the user wants to inspect is square we can set the *is_square_package* argument to *True* and the algorithm will find just packages with the square shape.
3. The last optional argument is called *is_bevelled_edge*. Some of the QFN packages have one bevelled edge. In this case we use template matching function that is a part of the OpenCV library and compare the shape of the component with the template image of the QFN package. If the similarity level is equal or lower than the threshold similarity, that can be set by the user, the condition is fulfilled.

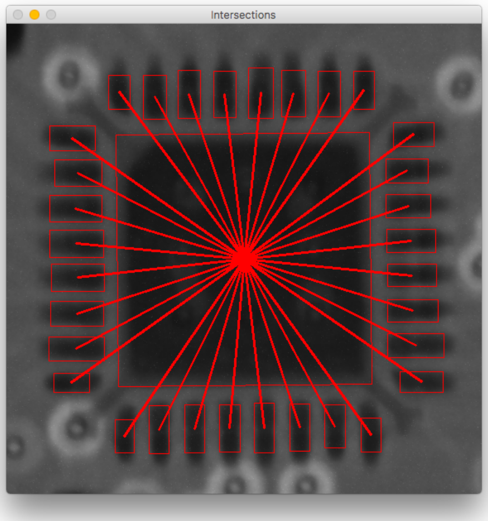

Fig. 6.9: Visualization of intersections of the pins and the center of the component

To segment the QFN packages from the x-ray image user can use the function *find_functions.find_qnf_package(...)* that have the following arguments:

1. **original_image** is a mandatory parameter which is a input color image that will be used for QFN package segmentation
2. **is_square_package** is a parameter from the configuration file that defines if the QFN package that we are looking for has a square shape.
3. **is_bevelled_edge** is a parameter from the configuration filer that defines if the QFN package that we are looking for has a beveled edge.
4. **min_intersection_count** is a parameter from the configuration file that defines minimum number of pins on each side of the QFN package. The default value is 3.
5. **min_number_of_pins** is a parameter from the configuration file that defines minimum number of pins that are surrounding the QFN package. It the default value is None therefore it is not using this condition. User can define any positive integer.
6. **k_size** is a parameter from the configuration file that defines the kernel size of the morphological closing that is used to remove small objects (holes) in the image. The default value is 11.
7. **iterrations** is a parameter from the configuration file that defines how many times will be the of the morphological closing applied to the image to remove small objects (holes) in the image. The default value is 11.
8. **debug_mode** is an optional parameter that can be used for debugging purposes. The default value is False. It can be useful to find for tuning the input

arguments.

9. **fxy** is an optional parameter that is used to resize the output image by the fxy ratio that is applied to both width and height.

6.2.2 QFN void detection

The void detection in the thermal pad of the QFN package is very similar process to the void detection in the solder ball. The only difference is that most of the QFN packages have vias underneath the package which cause a lot of troubles because it destroys the homogeneity of the void (see Figure 6.10). To get rid of these vias it is necessary to use very strong Gaussian blurring which works very good in this case because the voids have very good contrast otherwise by using strong bluing on image with low contrast we would also loose the voids. The rest of the algorithm is exactly the same as the algorithm used for searching the voids inside the solder balls because both share the same method that is inherited from the super class called *Component*.

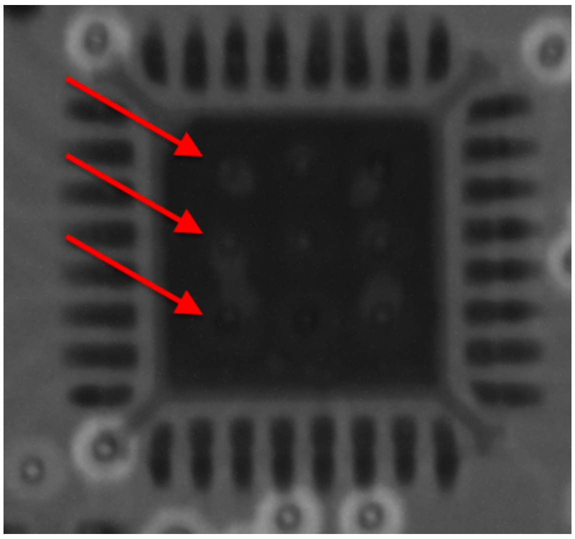

Fig. 6.10: Arrows pointing at three out of nine vias on under the QFN package

To find voids inside the QFN packages user can use the method of the *QFNPackage* object called *qnf_packages.analyze_voids(...)* the same input arguments as the void detection inside the solder ball that are described into a detail in the chapter 6.1.2. The only deference between the solder ball void detection and the QFN void detection is the process of cropping the contour. After cropping, it calls the same method from the super class from which it inherits. See the result of the void

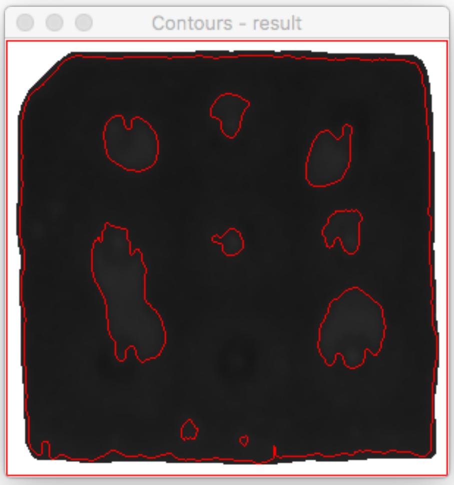

Fig. 6.11: Detected voids inside QFN package

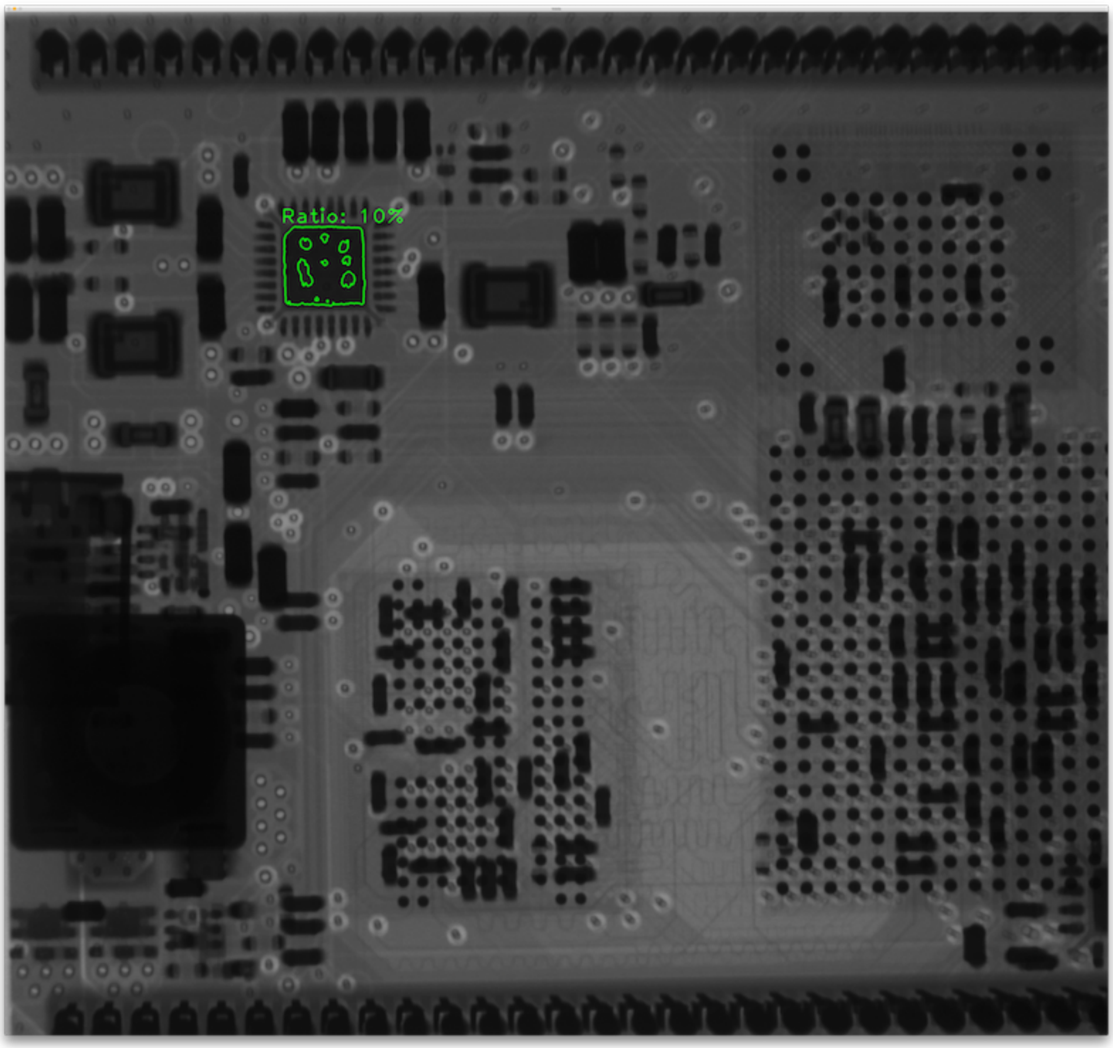

Fig. 6.12: Result of the void detection

6.2.3 Program usage

Analyzing the voids inside the QFN package consists of three steps. In the first step user calls a *find_qnf_package()* that is located in the python module located in */utilities/find_functions.py* which will return the *QFNPackage* object. This object contains method *analyze_voids(...)* that will find the voids inside each QFN package. Afterwards user can use another method from *QFNPackage* object called *show_all_voids(...)* that will mark all QFN packages with the void ratio higher than *error_ratio* with red color and the rest with green.

The working example of the void detection under the QFN package can be found in the root directory of the program in the folder *root/example/void_inside_QFN.py*. All available images that can be used by this algorithm can be found in the attached folder *Images/QFN package/Original images*. On top of that you can find more results of this algorithm in the chapter .3 or in the enclosed file *Images/QFN package/Results* where you can find input and result images together with the configuration file so you can test the program yourself without spending too much time tuning the input arguments.

```
import cv2
from Ultron.utilities import find_functions
from Ultron import edge_detection


image_path = '.../images/figure_with_qfn.jpg'
config_file_path = '.../images/config.ini'

image = cv2.imread(image_path)

qnf_packages = find_functions.find_qnf_package(
    image,
    configuration=config_file_path,
    debug_mode=False,
    fxy=1)

method = edge_detection.MethodLoG(gaus_blur_kernel=51, laplace_kernel
   =11)
qnf_packages.analyze_voids(
    image,
    configuration=config_file_path,
    method=method,
    debug_mode=False,
    fxy=1)
qnf_packages.show_all_voids(image, error_ratio=25, fxy=1)
```

Listing 6.2: Solderball segmentation and void detection

6.3 Image subtraction

In this category of defect detection methods, there are two major techniques. The first one is direct image comparison. This method is based on pixel-by-pixel comparison of the PCB image with some referential image of the circuit board that is stored in the image database. The referential image can have various forms it can be an image of the PCB board that can be considered as a perfect board or the Gerber file can be used. Gerber file is an open ASCII vector format for 2D binary images that is generated by the CAD system and it contains all necessary data for manufacturing. [9] It is basically a text file that specifies the shape, dimensions and coordinated of the solder pads. The main advantage of Gerber file over the image of referential board is that it is very accurate because it describes designed properties of the board. In most of the cases it is also easier to obtain Gerber file because it can be exported from the software that was used to design the PCB whereas the referential board must undergo many tests to ensure that it does not contain any defect.

Image comparison is usually done by using the XOR operator on the binary image of the images (see Figure 6.13). This approach is based on assumption that every difference between the reference board and the test board is a defect.

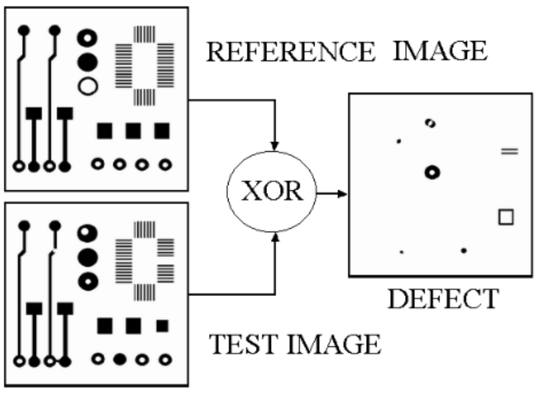

Fig. 6.13: Example of the XOR subtraction [24]

Another approach is to subtract the pixel values on the same position of the reference image from the test image. The result of this subtraction is a positive and negative image. These two images provides more information about the defect, than the simple XOR operator, which can be used for the defect classification. For example because the components manifest in the image as a darker regions, the positive image contains the missing components on the tested board and the negative image contains things that are redundant on the tested image such as bridges (see Figure 6.14).

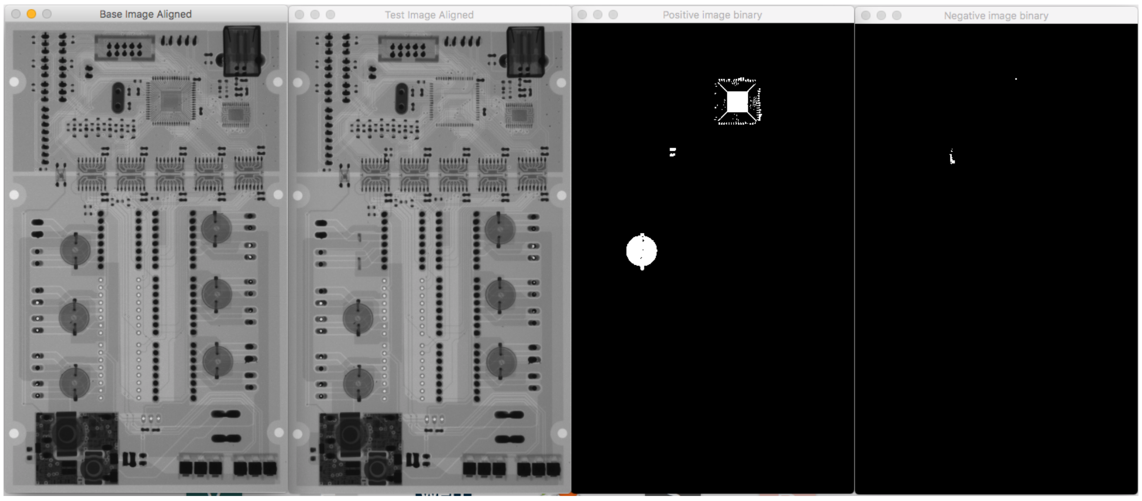

Fig. 6.14: Image subtraction

$$I_{negative} = I_{test} - I_{reference} \tag{6.1}$$

$$I_{positive} = I_{reference} - I_{test} \tag{6.2}$$

This method is very easy to implement, but the pixel-by-pixel comparison requires high location accuracy. If both of the images are not aligned properly, a lot of the false warning will occur.

Another type of the reference comparison methods is model based methods, which match the pattern under the inspection with the predefined template models. The most used techniques for the template matching are *connectivity based technique*, *N-tuple technique* and *Run Length Encoding* based technique. The best one is the Run Length Encoding technique because it not only finds defect but also get the location of the defect. [10]

6.3.1 Image alignment

Image alignment (also referred as image registration) is a process of determination of a right transformation that aligns a source and target image. In a typical image alignment problem we have two images of the same scene and they are related by the motion model. So to be able to align two images we have to determinate the motion model parameters. OpenCV is able to represent following models:

1. **Translation** model means, that the images are just shifted. To describe this transformation we need just two parameters (x,y)

2. **Euclidean** model covers the translation model that can be also rotated by some angle. To describe this transformation we need three parameters (x,y, angle)
3. **Affine** model includes all previous and also scale and shear.
4. **Homography** motion model is able to describe some of the 3D effects in the space. This transformation is described by eight parameters [14]

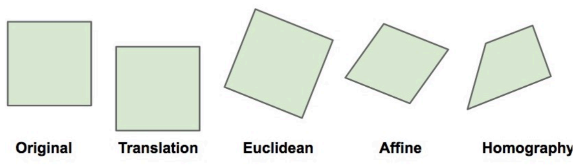

Fig. 6.15: Motions models [14]

There are two major approaches for image alignment. The first one is called *"Intensity based"* (also called direct alignment) which takes whole image or it's part in case if the images are just partly overlapping and estimates geometric warp between the source and target image with the minimum cost function for the pixels. Example of cost function is a function that calculates sum of square differences of intensity values.

The second approach is a *feature-based* which finds a feature points (sometimes called key-points) on both images and finds the most similar one and based on their descriptors decide the transformation matrix. There is a lot of key points detectors that can be used e.g. SIFT (Scale-Invariant Feature Transform), SURF (Speeded-Up Robust Features), FAST (Features from accelerated segment test) or ORB (Oriented FAST and Rotated BRIEF). After finding the feature points we have to find matches between the feature points on the source and the target image using for example Brute-Force Matching or FLANN (Fast Library for Approximate Nearest Neighbors). After finding the same points on both images the affine transformation matrix (2 x 3 matrix) or Homography matrix (3 x 3 matrix) can be calculated and used to align the target image.

In case of alignment of two PCBs I implemented my own rough alignment algorythm because after cropping of the PCB the program will return upright rectangle (that means that the rectangle will either standing or laying). So I have decided to rotate the rectangle 4 times by 90° and subtract both source and target images and compare them using the *cv2.absdiff(img1, img2)* that will calculate the per-element absolute difference between two images. The rotation that is the most similar is taken as a the result of the rough alignment. Afterwards, I use Enhanced Correlation Coefficient (ECC) for precise image alignment. ECC is a new similarity measure method introduced in the OpenCV 3 which brings two advantages over the traditional intensity based methods. This method is invariant to photometric distortions in contrast, brightness and it is fast because of the optimization. The small disadvantage is that if the target image undergo strong displacement there might be a need for an initial rough transformation which is exactly what I did in the previous step. [14] [15] [16]

For image alignment of two x-ray images of the printed circuit boards user can utilize the function called *allign_images(...)* that has two mandatory arguments and three optional:

1. **base_img** is a mandatory argument. It is the template image that will be used as the reference.
2. **test_img** is a mandatory argument. It is the image that will be compared with the template image.
3. **method** is an optional argument. User can choose which method will be used for the image alignment by entering the enum type called *AlignMethod*. At this point user can choose from the ECC method by entering *AlignMethod.ECC* or SIFT method by entering *AlignMethod.SIFT*. The default value method is ECC because it is more accurate.
4. **fxy** is an optional parameter that is used for visual purposes. Big images might not fit to the screen so it is necessary to resize the image by the *fxy* factor that will be used to resize the width and height.
5. **debug_mode** is an optional parameter that is used for debugging. The default value is *False*, but when it is set *True* it will show every step of the image processing.

Before the alignment itself it is recommended to crop the PCB because it will speed up process and also prevents a problem with the alignment caused by some errors in the captured image. The image cropping is done by function that can be found in the python module located in the */utilities/image_operations.py*. The fucntion has six arguments:

1. **input_image** is the mandatory parameter that is an image with the PCB that user wants to crop
2. **pcb_thresholding** is an optional parameter that divides the pcb board from the background. The default value is 190
3. **closing_kernel** is an optional parameter. It defines the size of the kernel that will be used for the morphological closing in order to remove holes from the image. The default value is 5.
4. **closing_iterations** is an optional parameter. It defines the number of iterations of the morphological closing in order to remove holes from the image. The default value is 3.

5. **fxy** is an optional parameter that is used for visualization purposes. Big images wont that wont fit to the screen have to be resized and the fxy is a scale ratio that will be used to resize the width and height.
6. **debug_mode** is an optional parameter that is used for debugging. The default value is *False*, but when it is set *True* it will show every step of the image processing. This can be used when user have problems with the segmentation of the PCB.

On the figure 6.16 you can see example of alignment and cropping of two PCBs. The top left image is the template and the top right image is the tested board that has to be aligned with the tested board. Underneath of each of this image is the result of cropping and alignment.

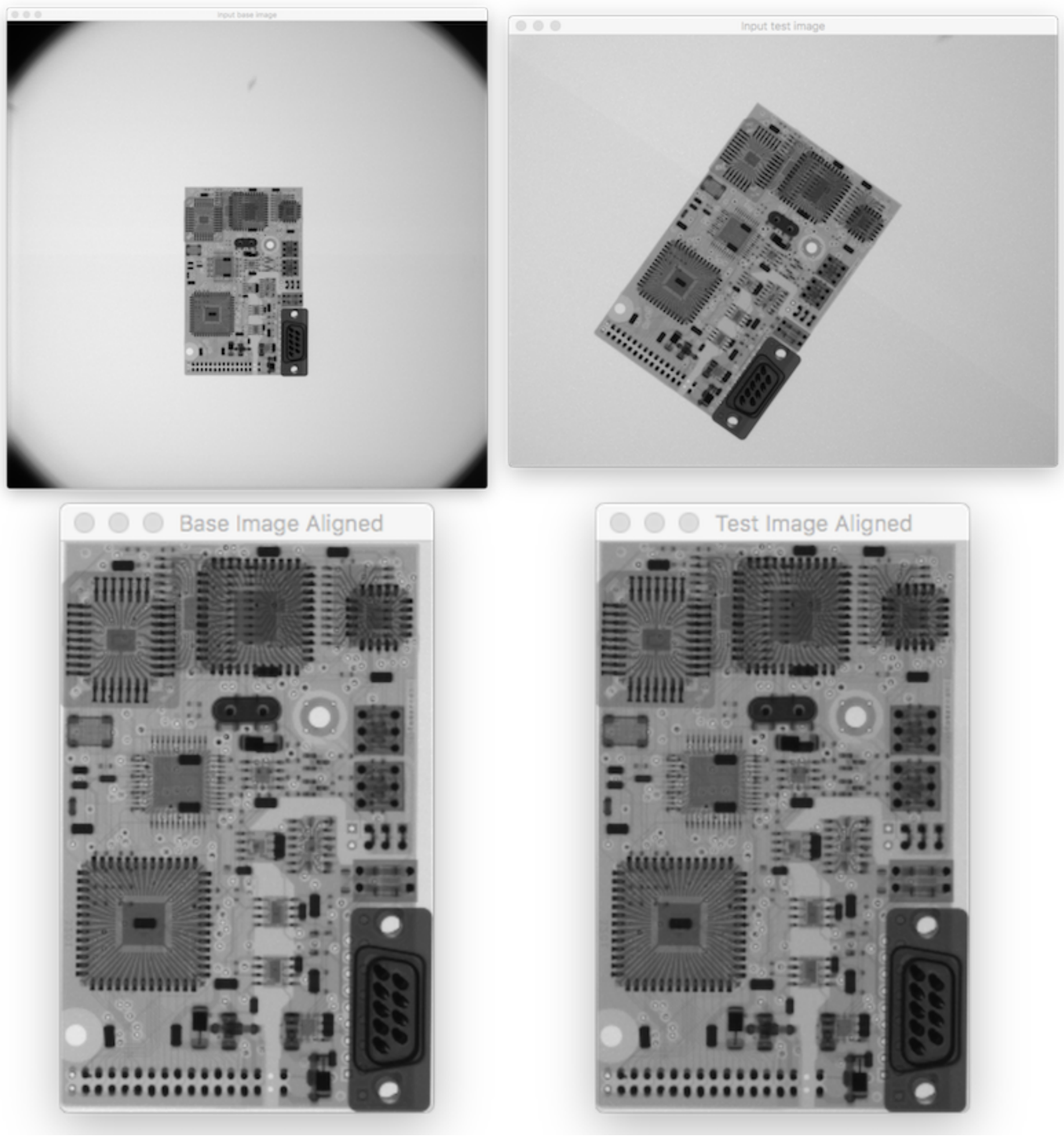

Fig. 6.16: Alignment of two PCBs

6.3.2 Defect classification

This method is not fully automatic like the other two defect detection methods because the classification of the defects using just the template image and the tested image is very complex task. The problem is that the amount of the displacement that can be considered as a defect is different for each type of the component which raise the need to precisely segment each component and recognize the type of the component. This is a challenging task to do automatically because a lot of the ICs does not have homogeneous body as the QFN package so it is very difficult to segment it. On top of that there is an interference from the components on the other side of the board. Instead of writing complex script for the classification of each component (which most likely will not be reliable due to the interference) it is much easier to use the Gerber file to determinate the position and shape of the component and decide if it is shifted.

In my case I classified the defects only into two categories and visualized the result with the different colors. The first category are things that are missing on the tested image such as missing components. The contours of these kind of defects are marked with the red color. The second category are things that are redundant on the tested image such as bridges. These kind of defects are marked with the blue color (see Figure 6.17).

This function can be utilized as a auxiliary tool to help operator to spot the potential defects faster and evaluate whether it is a critical defect or not.

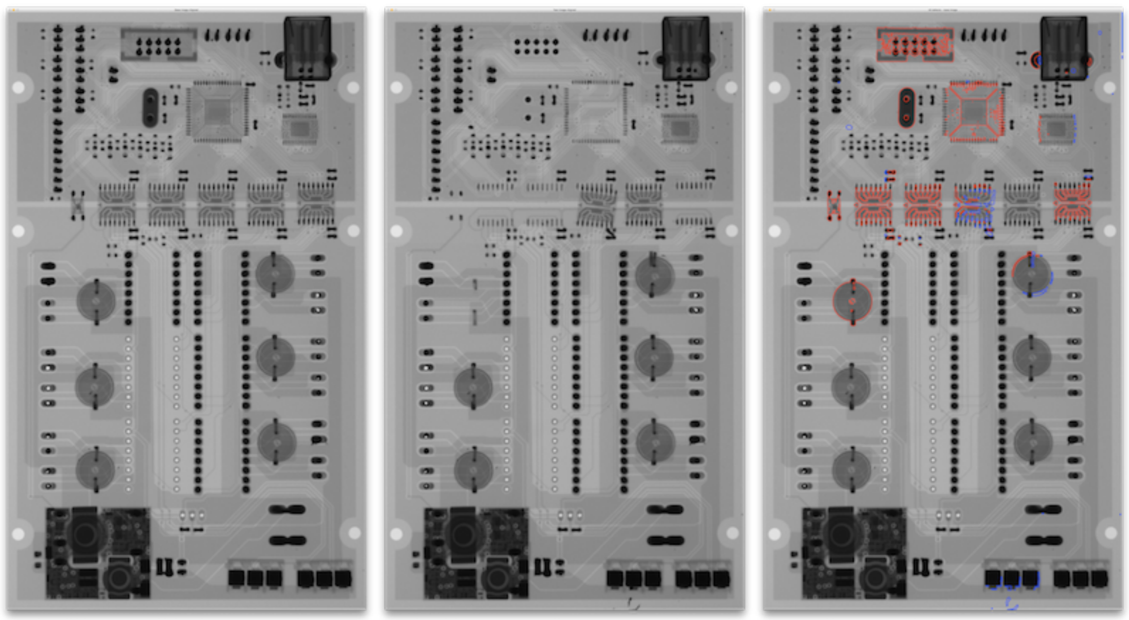

Fig. 6.17: Classification of the image subtraction

6.3.3 Program usage

Compassion of two PCBs consists of two steps. In the first step user uses the function called *crop_and_align_images(...)* that can be found in the python module in */utilities/image_operations.py*. This function will crop and align both x-ray images of PCB. User can also use the functions *crop_pcb(...)* and *allign_images(...)* separately in case of necessity to do some modification of the image before alignment.

The second step is a visualization of the results. At this point there are implemented 3 functions for visualizing the differences between the template and tested image. The first one is *show_missing_components(...)* that will show only things that are missing in the tested image. The function *show_redundant_components(...)* will visualize only the redundant components and the last function called *show_all_defects(...)* will visualize both missing and redundant things with different colors.

The working example of the image subtraction algorithm can be found in the root directory of the program in the folder *root/example/comparison_method.py*. All available images that can be used by this algorithm can be found in the attached folder *Images/Image Subtraction/Original images*. On top of that you can find more results of this algorithm in the chapter .4 or in the enclosed file *Images/Image Subtraction/Results* where you can find input and result images together with the configuration file so you can test the program yourself without spending too much time tuning the input arguments.

```python
import cv2
from Ultron.utilities import output
from Ultron.utilities import image_operations
from Ultron.enums import AlignMethod

source_image_path = '.../images/source.jpg'
target_image_path = '.../images/target.jpg'
config_file_path = '.../images/config.ini'

source_image = cv2.imread(source_image_path)
target_image = cv2.imread(target_image_path)

output.debug_show("Input source image", source_image, fxy=1)
output.debug_show("Input target image", target_image, fxy=1)

base_img_cropped, test_img_cropped = image_operations.
    crop_and_align_images(
     source_image,
     target_image,
     align_method=AlignMethod.ECC,
     configuration=config_file_path,
     debug_mode=False,
     fxy=1)

output.debug_show("Base Image Aligned", base_img_cropped, fxy=1)
output.debug_show("Test Image Aligned", test_img_cropped, fxy=1)

output.show_all_defects(
     base_img_cropped,
     test_img_cropped,
     debug_mode=True, fxy=1)
```

Listing 6.3: Solderball segmentation and void detection

7 CONCLUSION

This thesis covers implementation of the 3 methods to find a defects on the x-ray image of the printed circuit board.

The first method focused on the segmentation of solder balls and the detection of voids inside. The segmentation part using the boundary rectangles is very fast even with the images with high resolution. The only drawback of the segmentation process is the triangulation which is very slow especially if the BGA consists of hundreds of solder balls. It would be very beneficial to come up with faster function that will check the regular grid of the BGA. But overall segmentation function is reliable and robust. The second part is a void detection. The algorithm that was implemented works also with the x-ray images with low contrast because it uses local contrast enhancement using the CLAHE. The drawback of CLAHE algorithm is that it can enhance the noise in the homogeneous areas if the input parameters are chosen wrongly. The advantage of this implementation is that it can work fully autonomously without the need to evaluate the results by the operator. I can imagine this method to be used in the manufacturing process to automatically separate the PCBs that might be unreliable.

The detection of the second defect is similar to the first one in many ways. The biggest challenge of this task was to segment the QFN package. The problem is that shape of different types of the QFN packages can vary. It can be rectangular or square shape, some of them have beveled edge, but all of them are surrounded by the pins which I used as a primary condition. The algorithm itself works fine, but it has problems to identify QFN package if it is partially overlapping with some component on the other side of the board that will significantly modify the geometry of the package. This kind of interference is very difficult to overcome and in my opinion the only way how to make this segmentation reliable in all cases is by using some additional information from Gerber file. The void detection itself is also very similar to the void detection in the solder balls. The only difference is that a lot of the QFN packages have vias underneath their body which is challenging for the void detection. To solve this problem I used strong Gaussian blur that suppressed the strong edges of the vias on the image and allowed me to find the contour of the whole void. The overall reliability of this function is not sufficient due to the interference of the other components so I can not imagine this method to work in the real manufacturing process. I believe that by using some hybrid inspection method instead of relying just on the geometrical rules would result in much better reliability.

The last defect detection method which I implemented is using the reference comparison method. The algorithm consists of two steps. In the first step I wrote an algorithm for image alignment. I used the ECC algorithm for alignment together with my rough alignment algorithm which is very fast (approximately 0.5 seconds for image 1600x1000px) and precise. The problem with automatic checking of the PCB using this method is that it is very challenging to evaluate the results automatically. The main bottleneck is the segmentation and recognition of each component because unlike solder ball or QFN package other components does not have homogeneous body with high absorption coefficient which is a major problem for evaluation of the results. For that reason this algorithm is not fully automated and needs someone to evaluate the results. This problem can be solved by using Gerber file or any other source of position and shape information of the components because that would allow the program to create an output that will be much easier to analyze. Furthermore, this method can be very helpful in connection with the manual inspection because it can help the operator to highlight the areas with the potential defects.

BIBLIOGRAPHY

[1] Hooks, Nick 3 Most Common PCB Assembly Defects. (n.d) from Optimum Design Associates Web Site: http://blog.optimumdesign.com/3- most-common-pcb-assembly-defects

[2] "How to Prevent the Tombstone and Open Defects during the SMT Reflow Process." How to Prevent the Tombstone and Open Defects during the Reflow Process. Bittele Electronics Inc, n.d. Web. 21 Feb. 2017.

[3] "Documentation." Introducing MEX Files - MATLAB & Simulink. Accessed February 21, 2017. https://www.mathworks.com/help/matlab/matlab_external/introducing-mex-files.html.

[4] "New License for MATLAB R2016b." New License for MATLAB R2016b. Accessed February 21, 2017. https://www.mathworks.com/store/link/products/standard/new?s_iid=htb_buy_gtwy_ct

[5] Mallick, Satya. "Home." Learn OpenCV. October 30, 2015. Accessed February 21, 2017. https://www.learnopencv.com/opencv-c-vs-python-vs-matlab-for-computer-vision/.

[6] Somchai Nuanprasert, Sueki Baba, Takashi Suzuki, (2015). A Simple Automated Void Defect Detection for Poor Contrast X-ray Images of BGA, 3rd International Conference on Industrial Application Engineering 2015. Department of Systems Innovation, Osaka University, 1-3 Machikaneyama-cho, Toyonaka 560-8531, Japan bBEAMSENSE Co.Ltd., 2-19-16 Izumi-cho, Suita 564-0041, Japan URL: https://www2.ia-engineers.org/iciae/index.php/iciae/iciae2015/paper/viewFile/549/426

[7] A. F. Said, B. L. Bennett, L. J. Karam and J. Pettinato, "Robust automatic void detection in solder balls," 2010 IEEE International Conference on Acoustics, Speech and Signal Processing, Dallas, TX, 2010, pp. 1650-1653. doi: 10.1109/ICASSP.2010.5495524 URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5495524&isnumber=5494886

[8] Evstatin Krastev, John Tingay , "RECENT ADVANCES IN THE X-RAY INSPECTION TECHNOLOGY WITH EMPHASIS ON LARGE BOARD COMPUTER TOMOGRAPHY AND AUTOMATION", Evstatin Krastev and John Tingay Nordson DAGE Concord, CA, USA doi: 10.1109/ICASSP.2010.5495524 URL: http://www.nordson-at.com/technology/up_img/1428036789-103009.pdf

[9] Wikipedia contributors. Gerber format [online]. Wikipedia, The Free Encyclopedia. 2017 Mar 27, 20:19 UTC [cited 2017 Apr 3]. Available from: https://en.wikipedia.org/w/index.php?title=Gerber_format&oldid=772526836.

[10] IBRAHIM, Zuwairie, Syed ABDUL RAHMAN AL-ATTAS a Zulfakar ASPAR. Model-based PCB Inspection Technique Using Wavelet Transform. Singapore, 2002. Universiti Teknologi Malaysia.

[11] Image Thresholding [online]. OpenCV team, 2016 [cit. 2017-03-28]. http://docs.opencv.org/3.2.0/d7/d4d/tutorial_py_thresholding.html

[12] Wikipedia contributors. Quad Flat No-leads package [online]. Wikipedia, The Free Encyclopedia; 2016 May 12, 16:25 UTC [cited 2017 Apr 5]. Available from: https://en.wikipedia.org/w/index.php?title=Quad_Flat_No-leads_package&oldid=719922003.

[13] ATMEL, Atmel Corporation. QFN Package Mounting Guidelines AT88RF1354. 2009. 8583A–RFID–3/09.

[14] MALLICK, SATYA. Image Alignment (ECC) in OpenCV (C++ / Python) [online]. [cit. 2015-07-01]. http://www.learnopencv.com/image-alignment-ecc-in-opencv-c-python/

[15] Motion Analysis and Object Tracking [online]. OpenCV dev team, 2014. http://docs.opencv.org/3.0-beta/modules/video/doc/motion_analysis_and_object_tracking.html#findtransformecc

[16] MALLICK, SATYA. Homography Examples using OpenCV (Python / C ++) [online]. OpenCV dev team, 2016 [cit. 2017-04-08]. http://www.learnopencv.com/homography-examples-using-opencv-python-c/

[17] MOHANTY, Rita a Vatsal SHAH. Solder Paste Inspection Technologies: 2D-3D Correlation [online]. Speedline Technologies, Inc Franklin, MA. Dostupné také z: https://www.smtnet.com/library/files/upload/Solder-Paste-Inspection-APEX2008.pdf

[18] ORESJO, Stig. WHEN TO USE AOI, WHEN TO USE AXI, AND WHEN TO USE BOTH [online]. Agilent Technologies Loveland, Colorado, 2002 [cit. 2017-04-10]. http://www.keysight.com/upload/cmc_upload/All/When_Use_AOI_AXI_Both.pdf?&cc=CZ&lc=eng

[19] Wikipedia contributors. X-ray [online]. Wikipedia, The Free Encyclopedia; 2017 Apr 10, 05:34 UTC [cited 2017 Apr 10]. https://en.wikipedia.org/w/index.php?title=X-ray&oldid=774708625.

[20] KRASTEV, Evstatin a John TINGAY. RECENT ADVANCES IN THE X-RAY INSPECTION TECHNOLOGY WITH EMPHASIS ON LARGE BOARD COMPUTER TOMOGRAPHY AND AUTOMATION [online]. Nordson DAGE Concord, CA, USA, 2014 [cit. 2017-04-10]. http://www.nordson-at.com/technology/up_img/1428036789-103009.pdf

[21] Cost-Effective 3D X-Ray Inspection [online]. Metris USA, Inc., Brighton, MI [cit. 2017-04-10]. http://www.us-tech.com/RelId/674885/ISvars/default/Cost-Effective_3D_X-Ray_Inspection.htm

[22] PAL, Ajay, Singh CHAUHAN a Sharat CHANDRA BHARDWAJ. Detection of Bare PCB Defects by Image Subtraction Method using Machine Vision [online]. 2011 [cit. 2017-04-10]. https://www.researchgate.net/publication/265319589_Detection_of_Bare_PCB_Defects_by_Image_Subtraction_Method_using_Machine_Vision

[23] KAUR, Kamalpreet. Various Techniques for PCB Defect Detection [online]. Lecturer, ECE Department, Thapar Polytechnic College, Patiala: An International Journal of Engineering Sciences, 2016 [cit. 2017-04-11]. http://ijoes.vidyapublications.com/paper/Vol17/23-Vol17.pdf

[24] IBRAHIM, Zuwairie, Syed ABDUL RAHMAN AL-ATTAS a Zulfakar ASPAR. Model-based PCB Inspection Technique Using Wavelet Transform [online]. Singapore, 2002 [cit. 2017-04-18]. Dostupné z: https://pdfs.semanticscholar.org/f200/d23b93c46b6e12f07e6eefd0aa9045495c9c.pdf. Universiti Teknologi Malaysia.

[25] BRISSETTE, Peter. Introduction to IPC Inspection Guidelines [online]. 2015 [cit. 2017-04-20]. https://bayareacircuits.com/introduction-to-ipc-inspection-guidelines/

[26] SPEA 3030 Operatorless Test Cell. Maximum competitiveness, maximum savings [online]. 2015 [cit. 2017-04-20]. http://www.spea.com/Portals/0/SPEANewsTemplates/Generic.aspx?pItemId=268&pModuleId=845&pViewType=AllPress

[27] XT V 160 Electronics X-ray system [online]. [cit. 2017-04-20]. http://www.nikonmetrology.com/en_EU/Products/X-ray-and-CT-Inspection/X-ray-systems-for-electronics-inspection/XT-V-160-Electronics-X-ray-system/(key_features)

[28] Primer on 2D and 3D X-Ray. Datest [online]. [cit. 2017-04-21]. http://www.datest.com/resources-boardtestmeth-primer2d3d.php

[29] PCB inspection using X-ray inspection technologies [online]. 2016 [cit. 2017-04-21]. https://www.ourpcb.com/x-ray-inspection.html

LIST OF SYMBOLS, PHYSICAL CONSTANTS AND ABBREVIATIONS

PCB	printed circuit board
PCBs	printed circuit boards
BGA	ball grid array
AOI	automatic optical inspection
CCD	charge coupled device
AXI	automatic X-ray inspection
ICT	in-circuit test
SPI	solder paste inspection
FT	functional test
SMT	surface mount technology
CGA	column grid array
CSP	chip size package
CLAHE	Contrast Limited Adaptive Histogram Equalization

LIST OF APPENDICES

.1 Additional information

You can find all following images in the original resolution in the attached files. There is also a configuration file so you can easily test each method yourself.

.2 BGA void detection

.2.1 test1

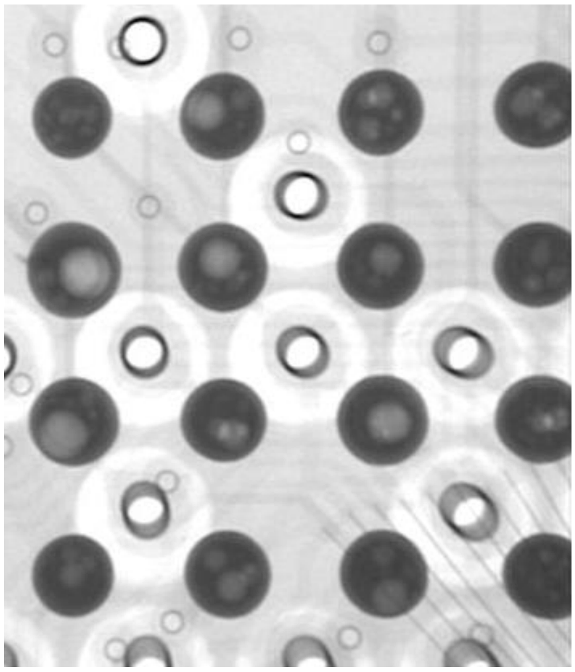

Fig. 1: Input image

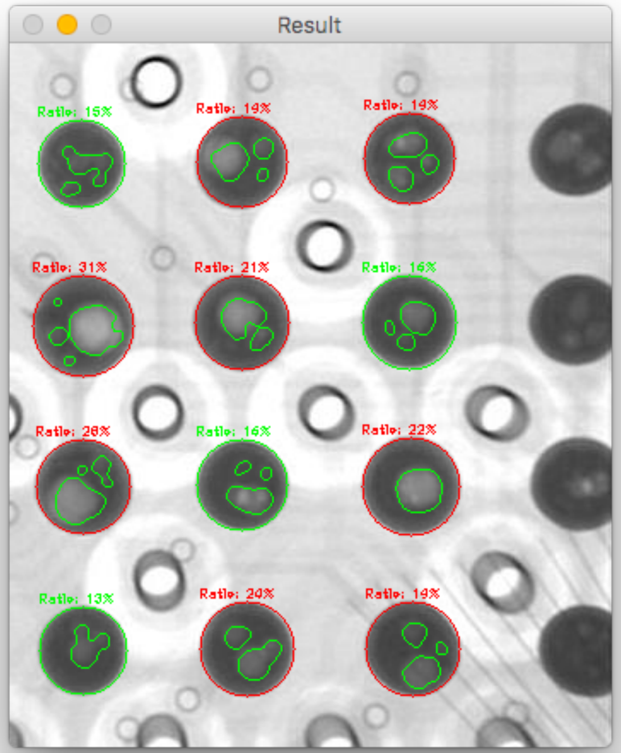

Fig. 2: Result image

.2.2 test2

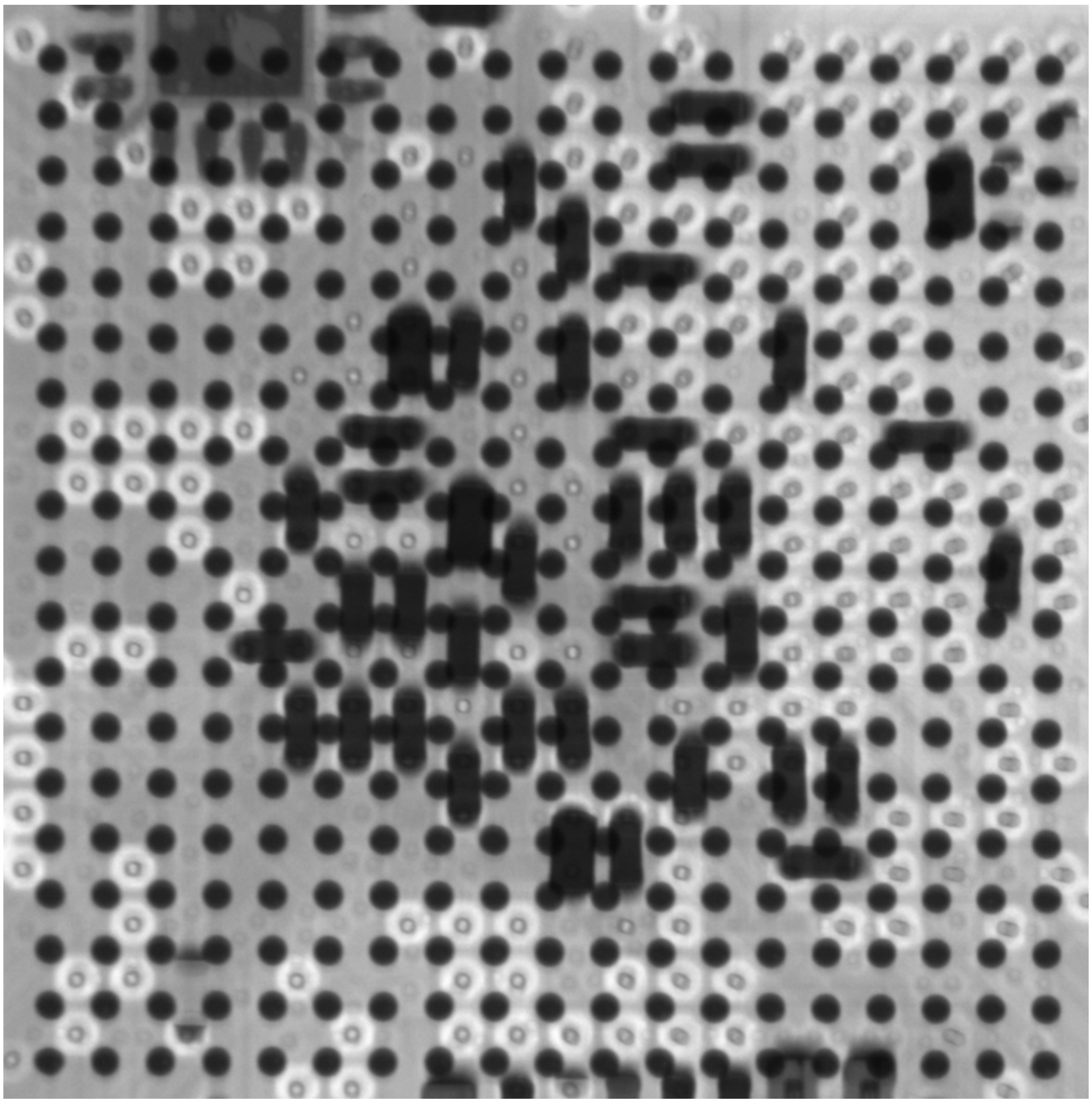

Fig. 3: Input image

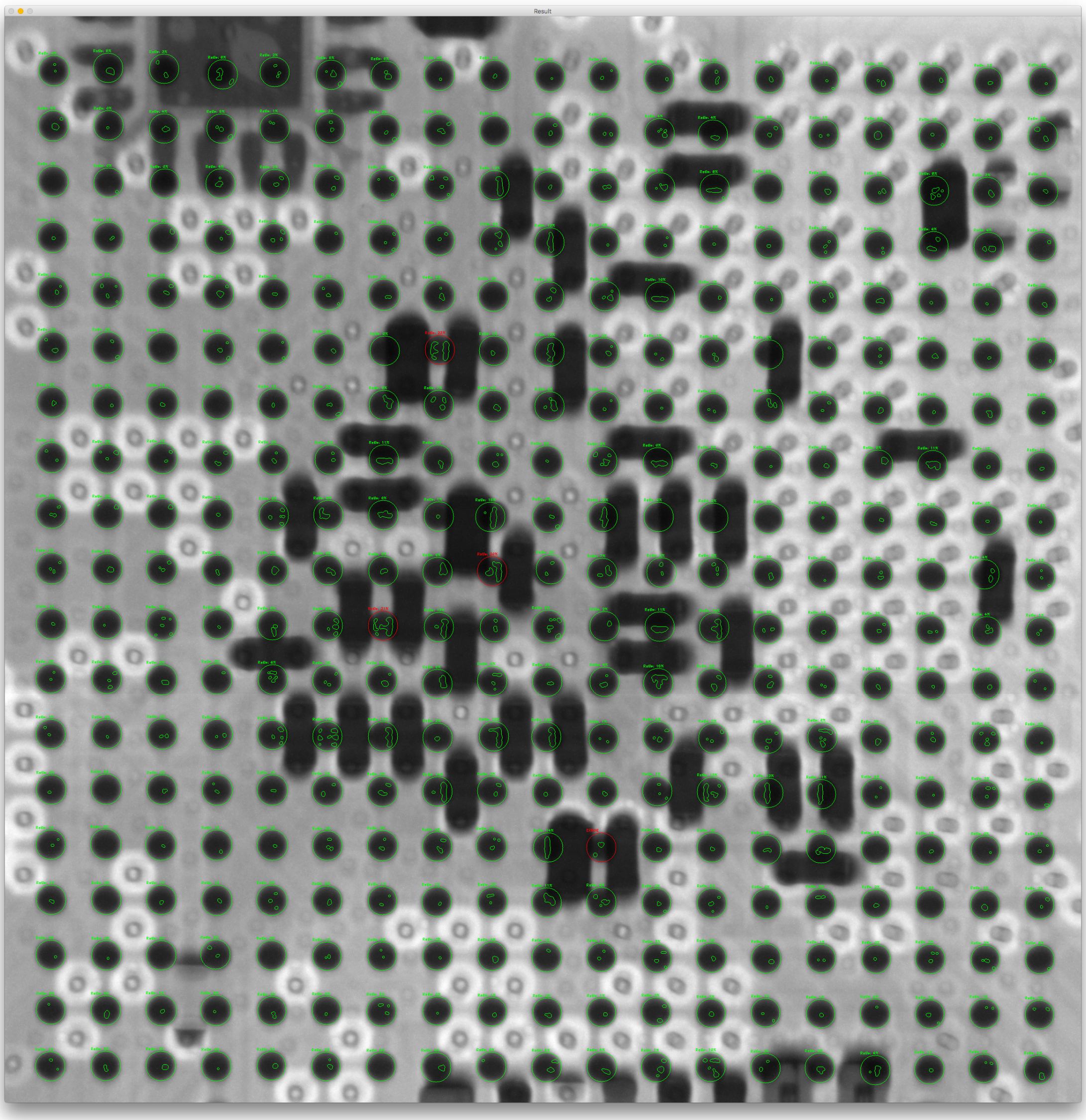

Fig. 4: Result image

.2.3 test3

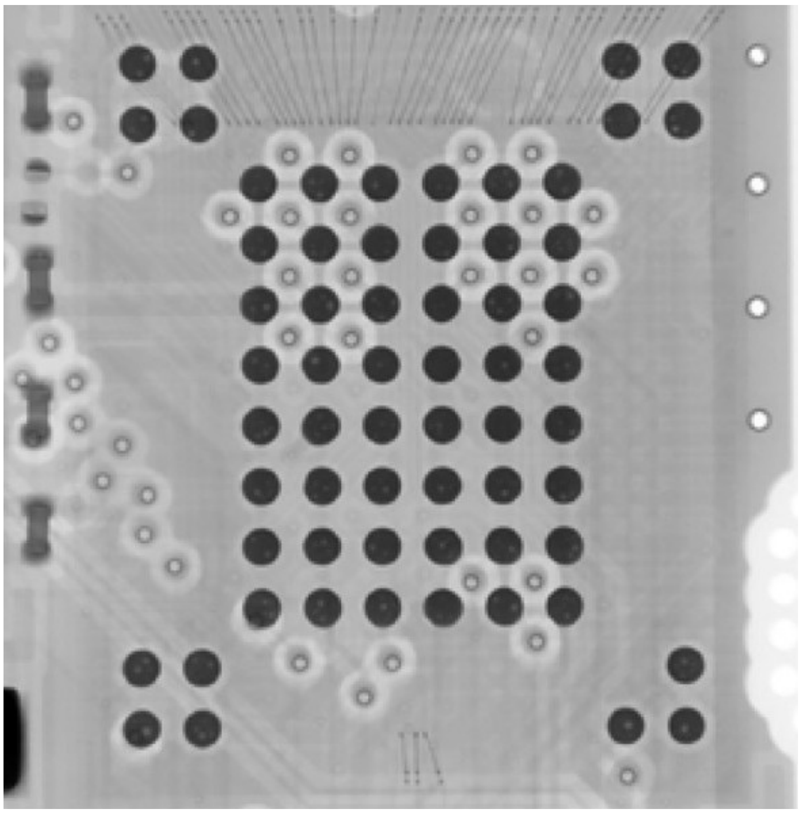

Fig. 5: Input image

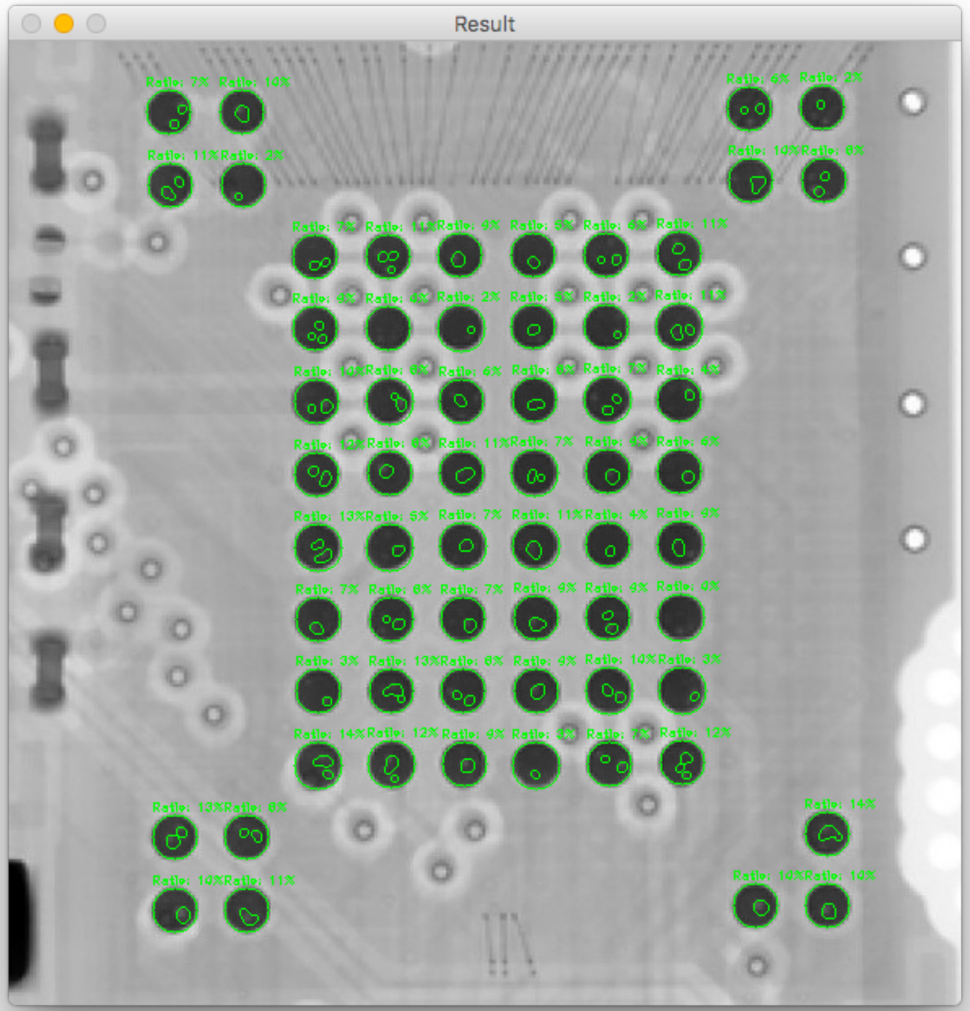

Fig. 6: Result image

.3 QFN void detection

.3.1 test1

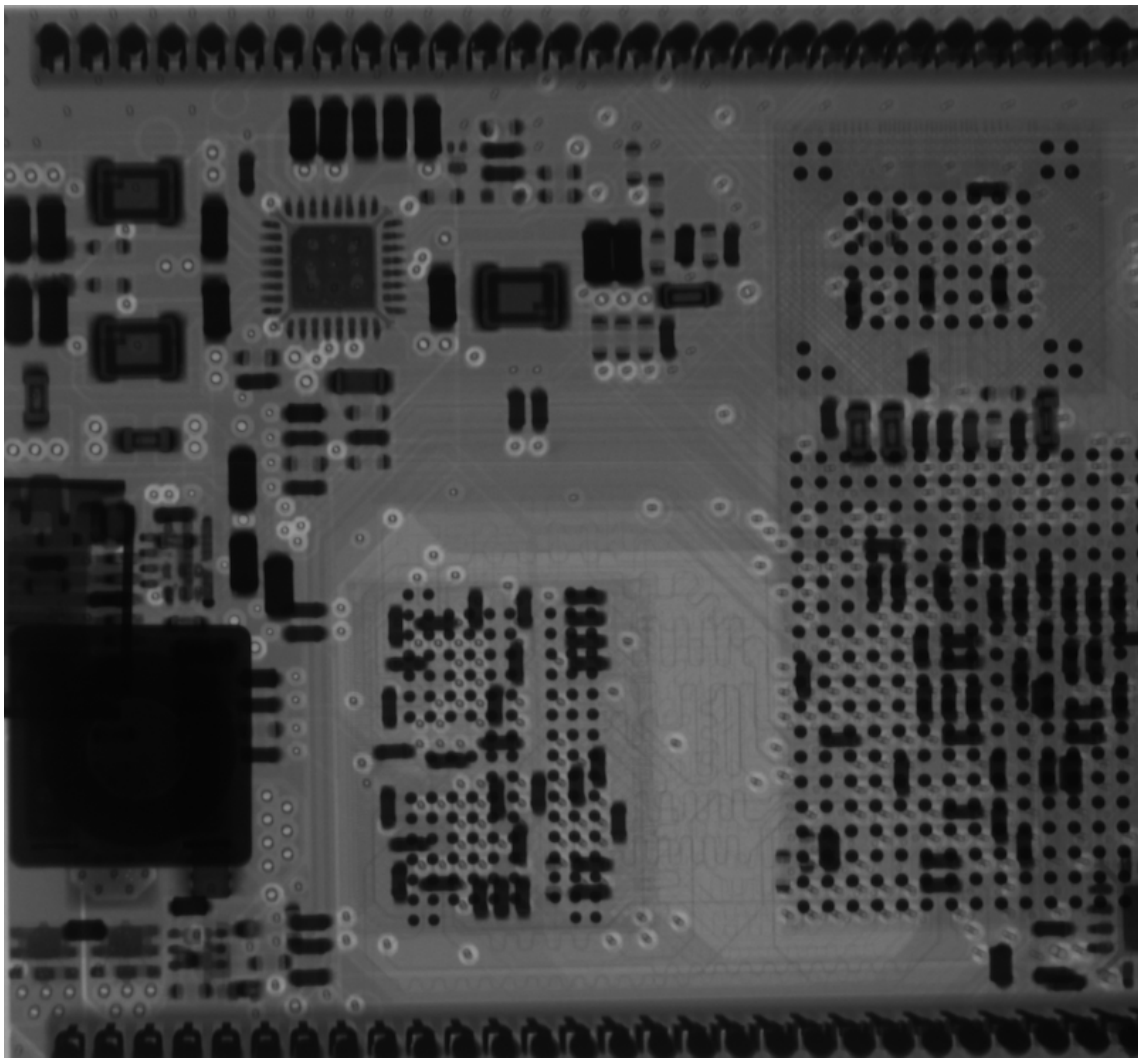

Fig. 7: Input image

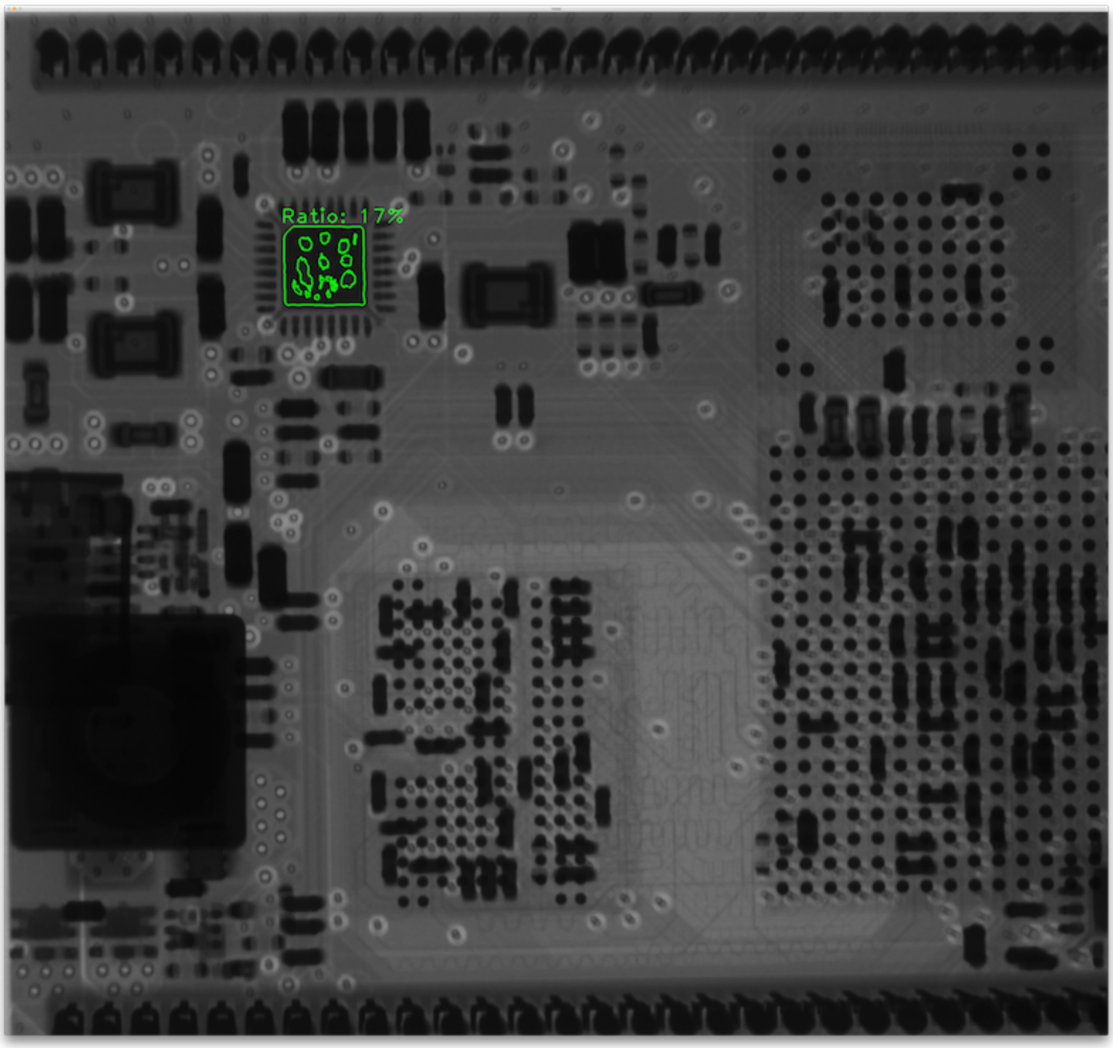

Fig. 8: Result image

.3.2 test2

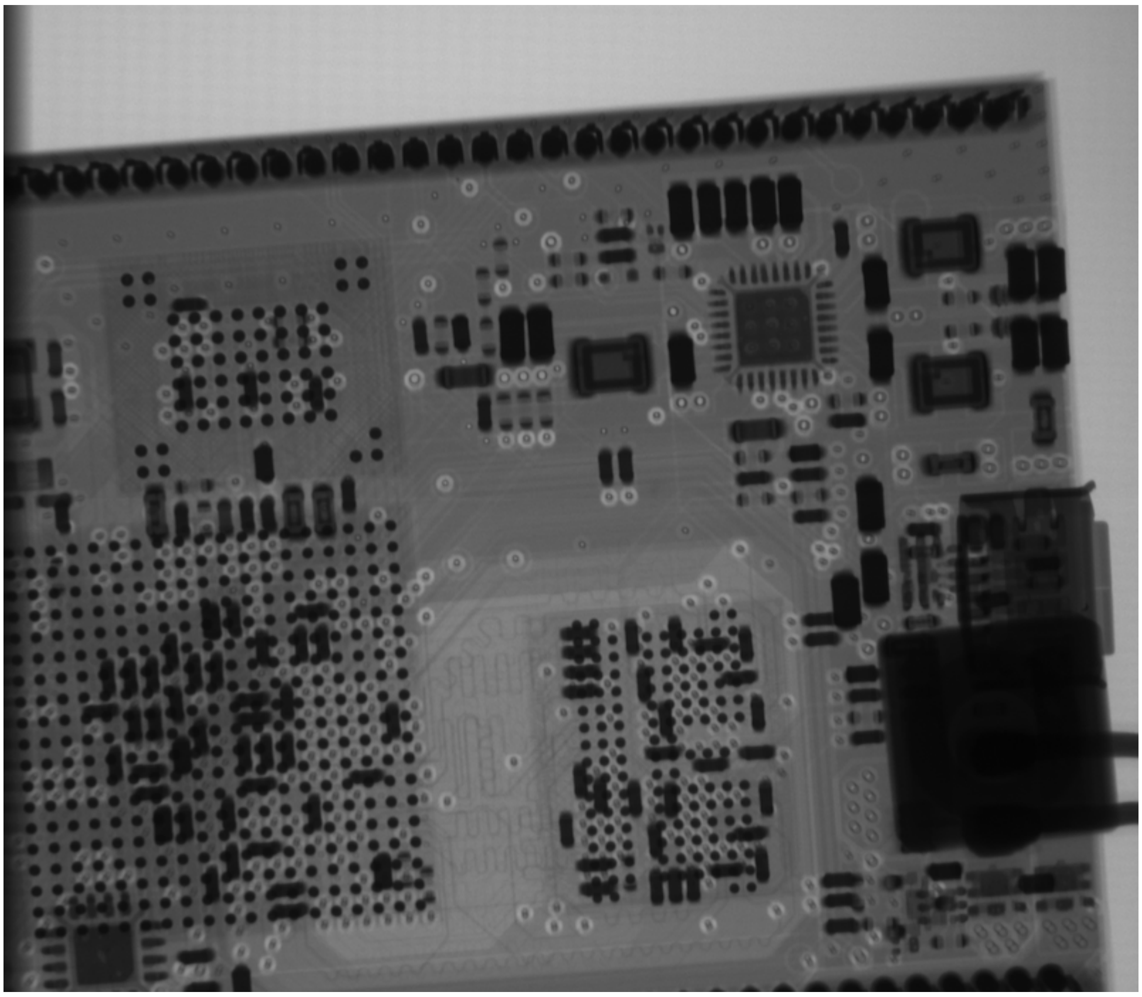

Fig. 9: Input image

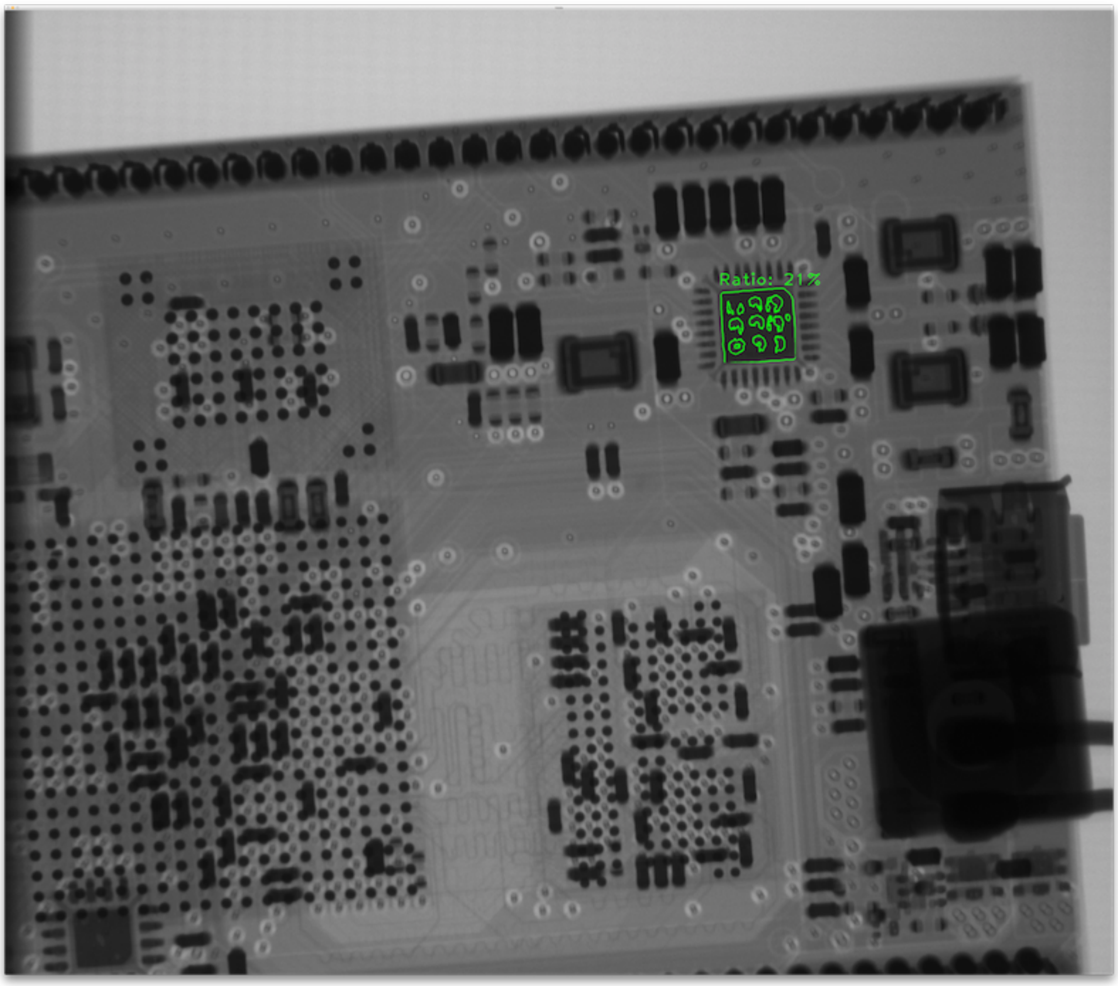

Fig. 10: Result image

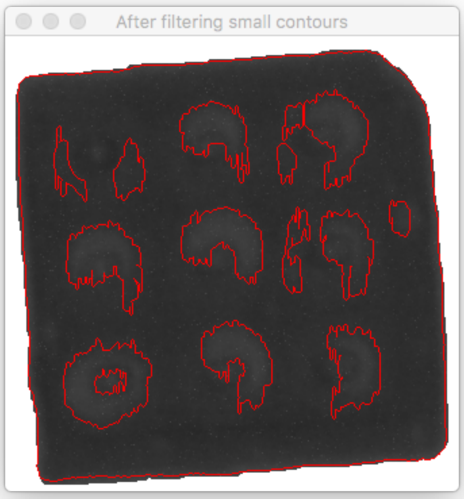

Fig. 11: Detail of the result image

.4 Image subtraction

.4.1 PCB1 test1

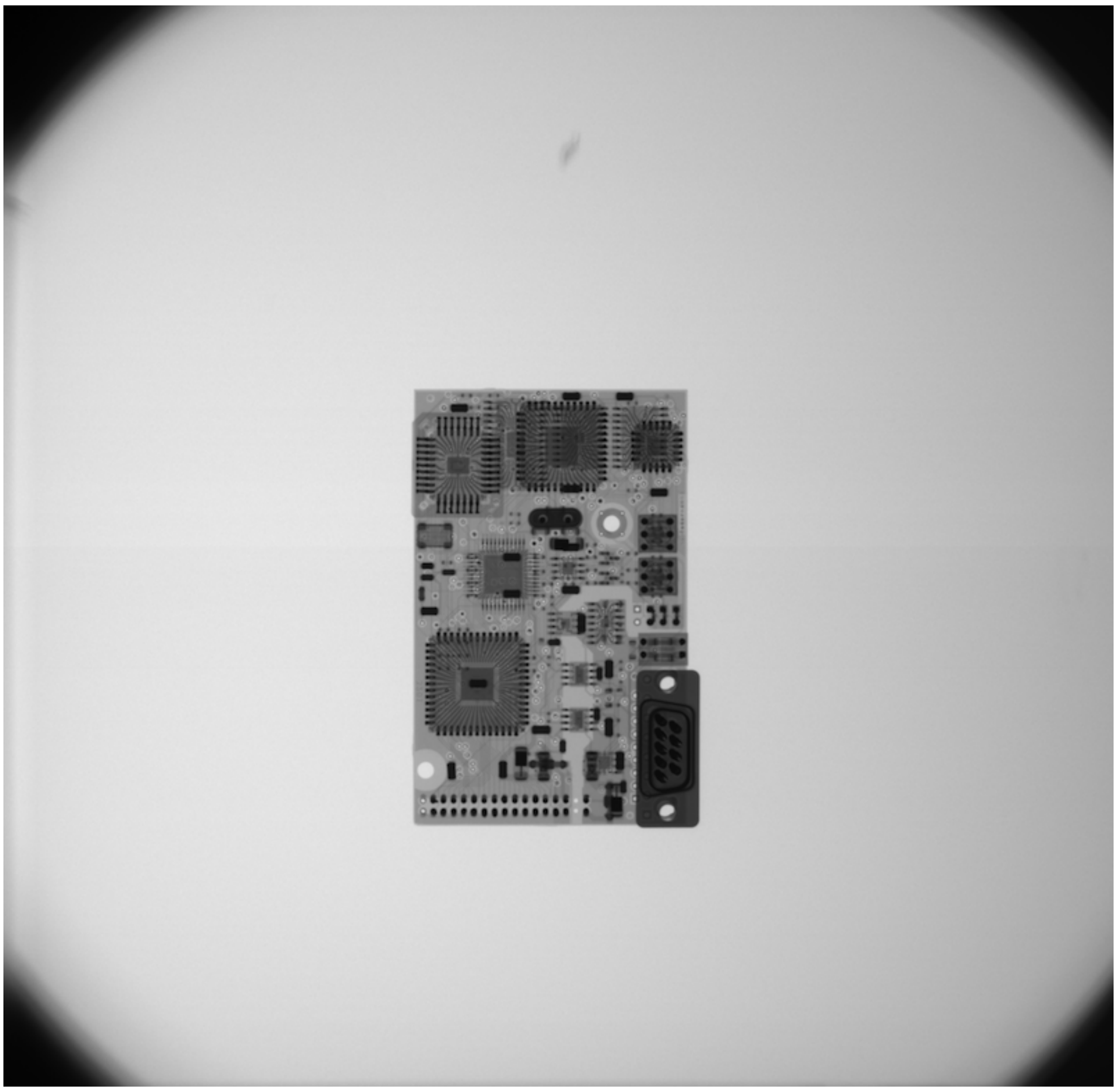

Fig. 12: Base image

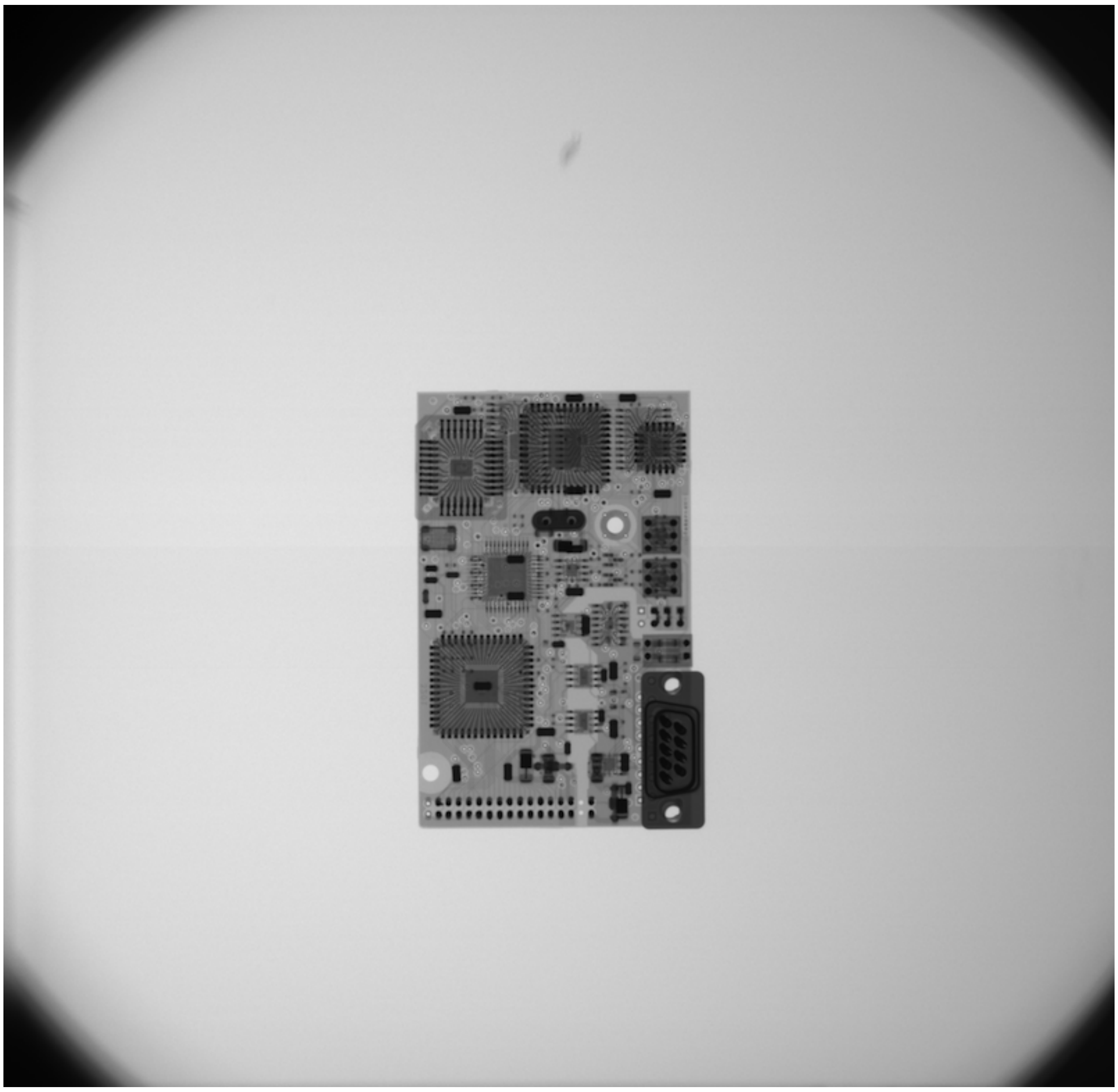
Fig. 13: Tested image

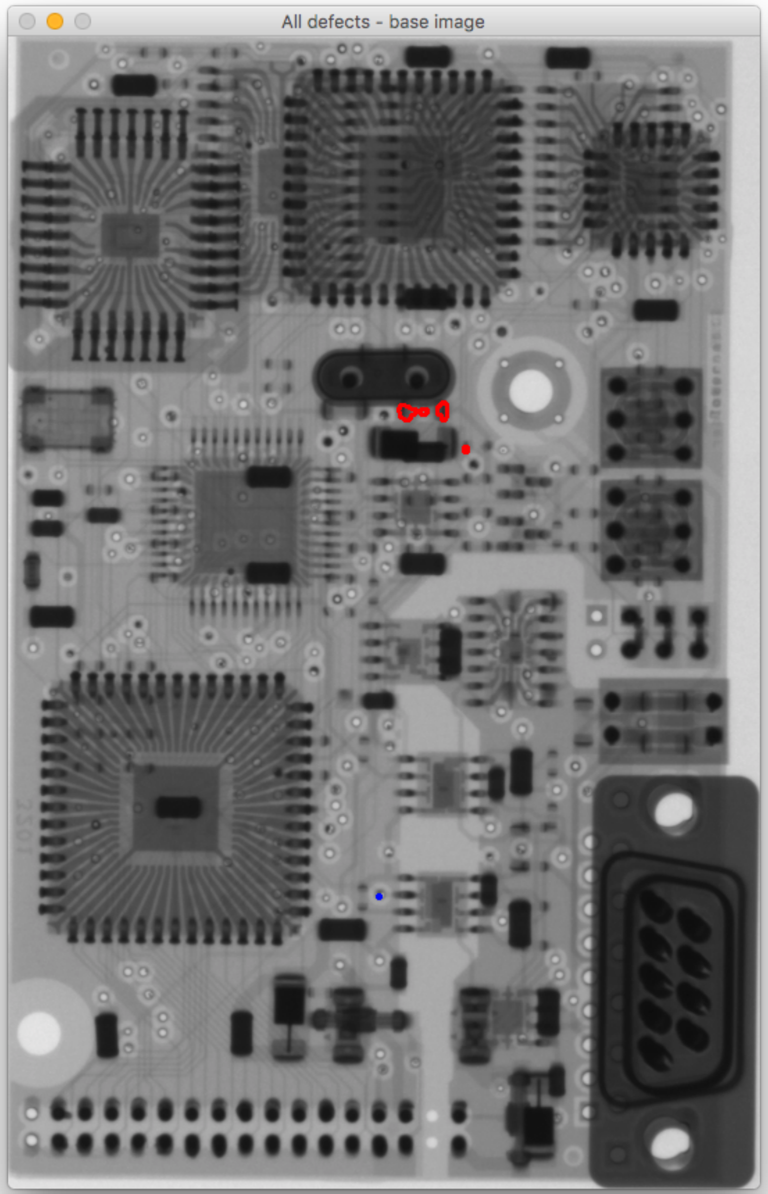

Fig. 14: Result image

.4.2 PCB1 test2

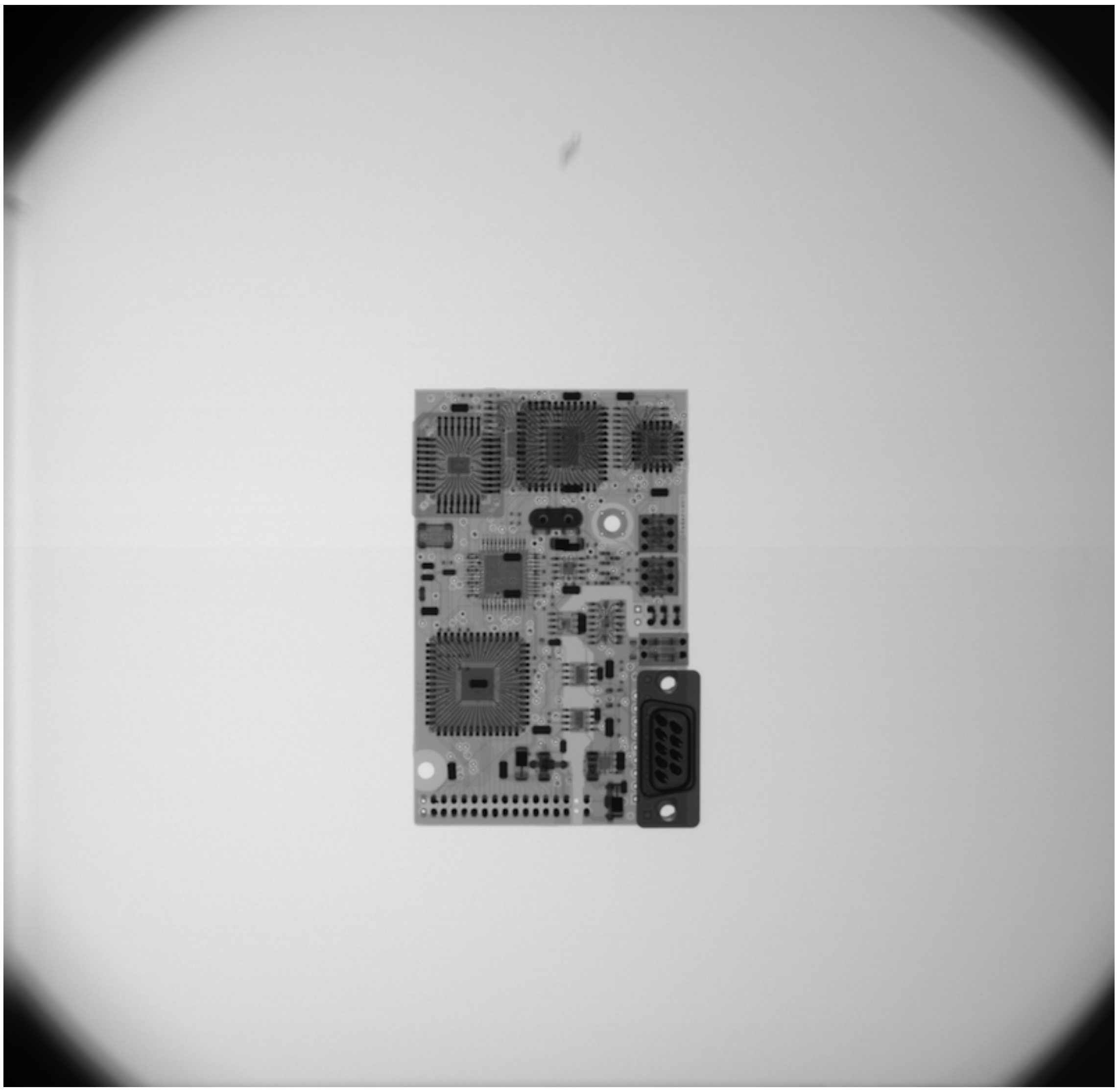

Fig. 15: Base image

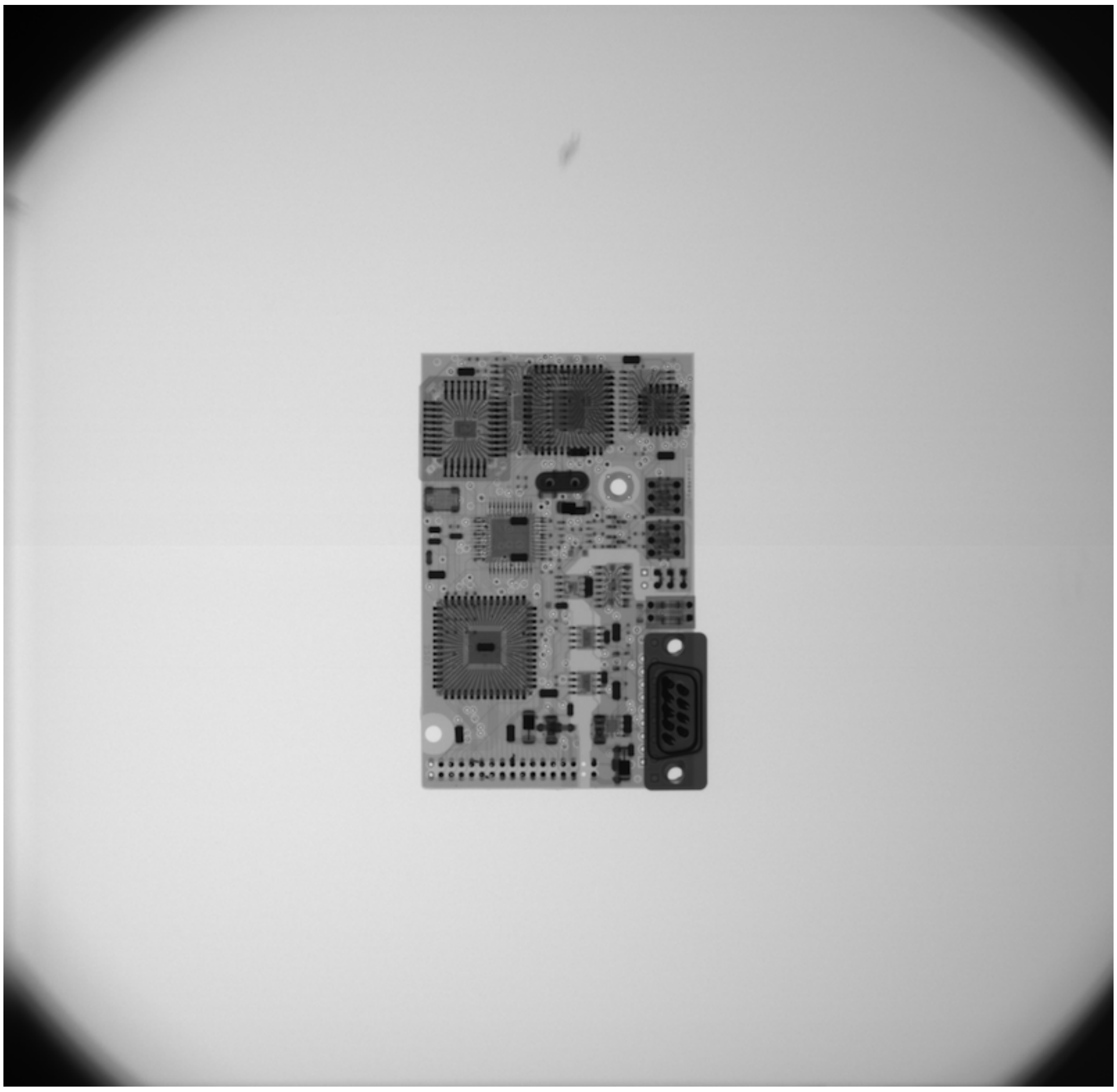

Fig. 16: Tested image

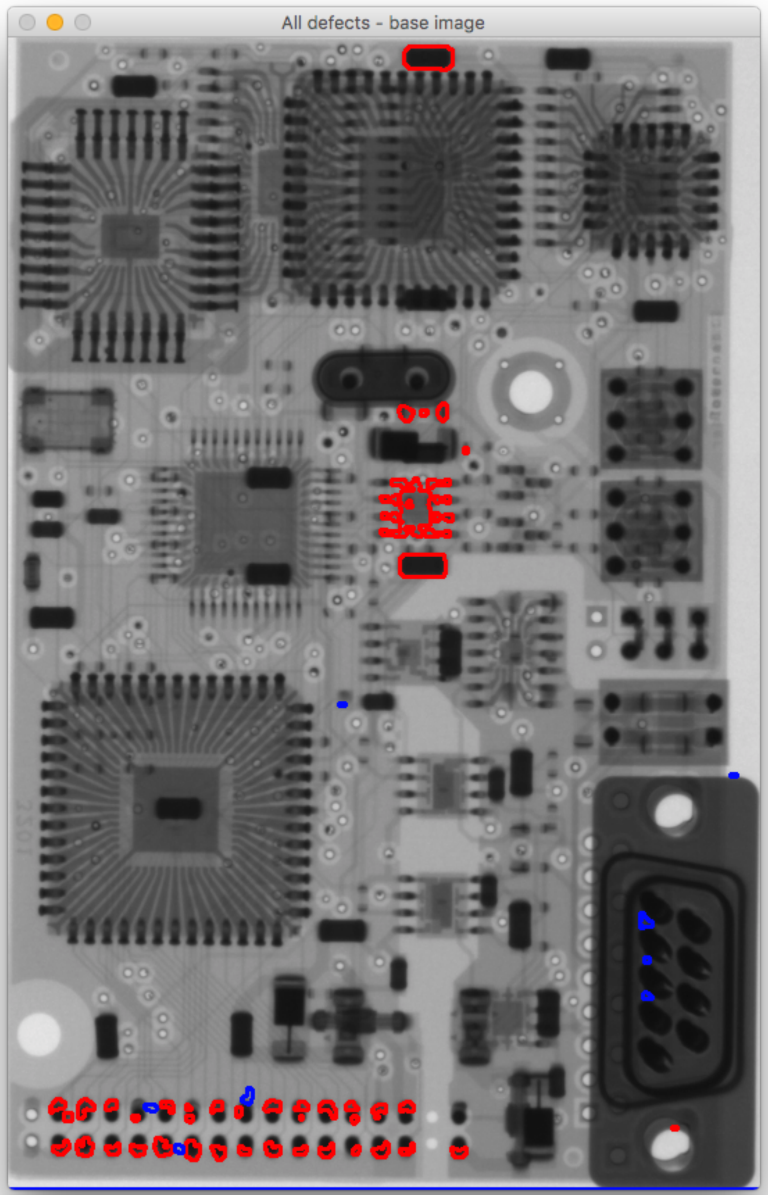

Fig. 17: Result image

.4.3 PCB1 test3

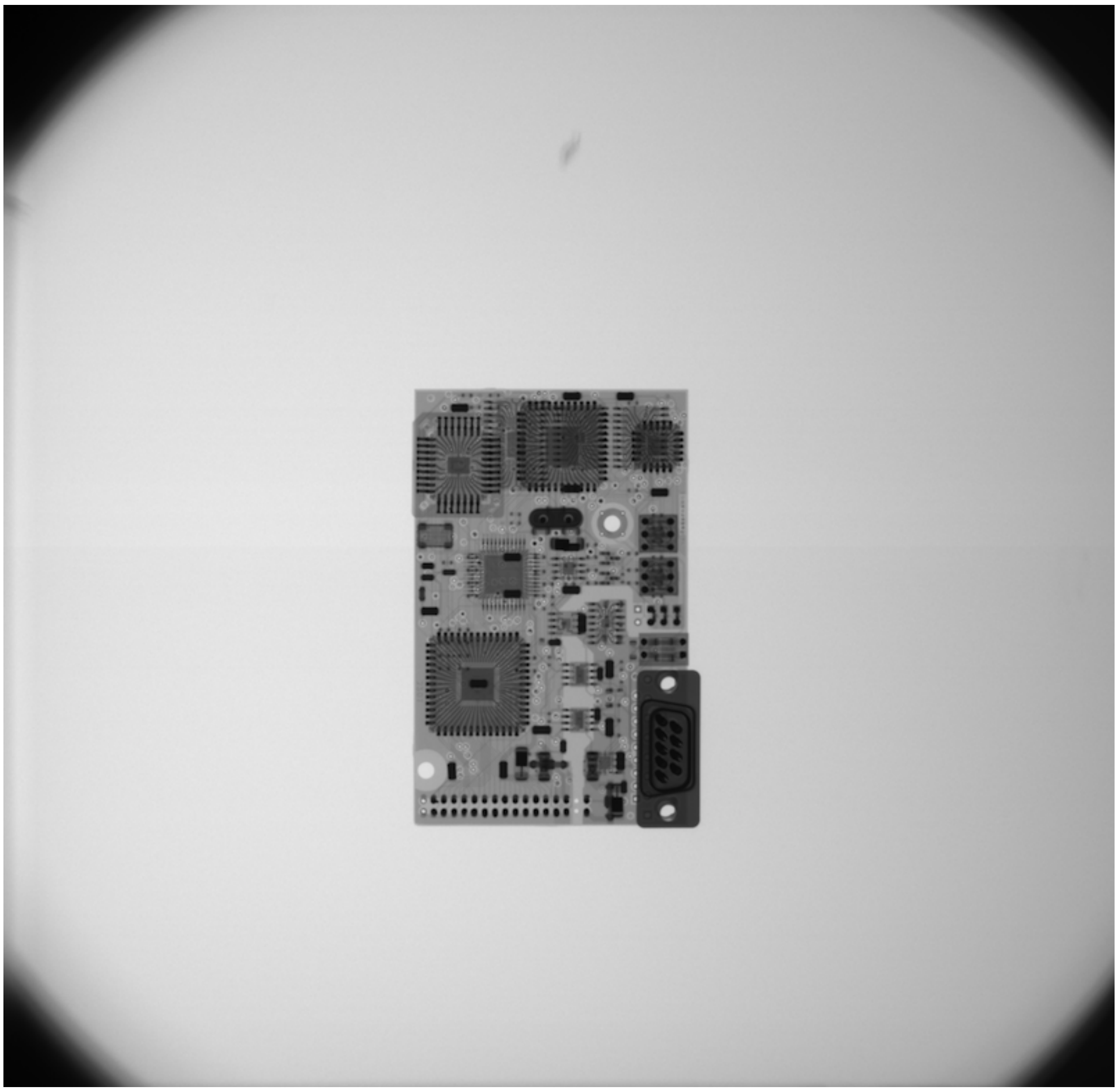

Fig. 18: Base image

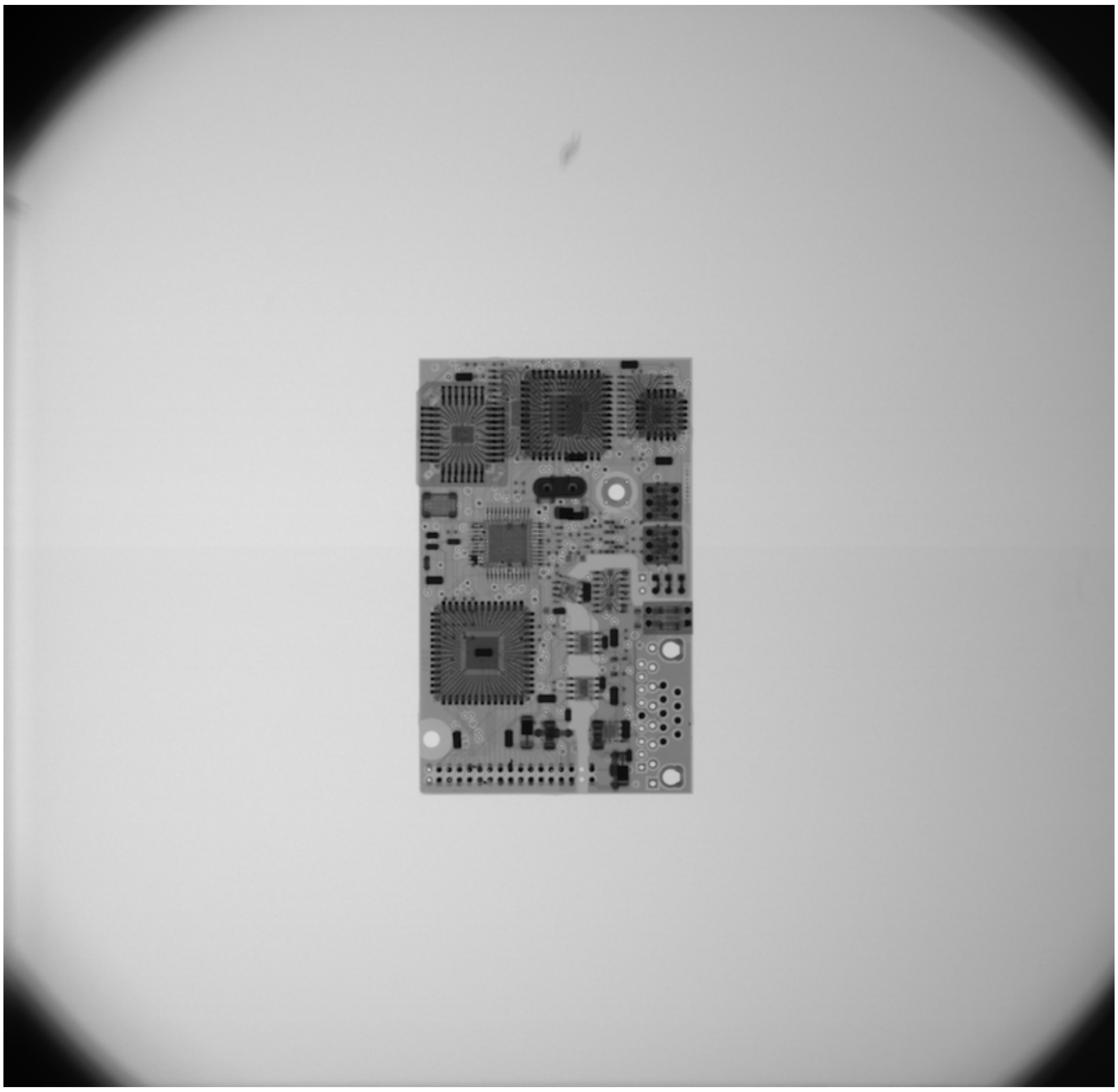

Fig. 19: Tested image

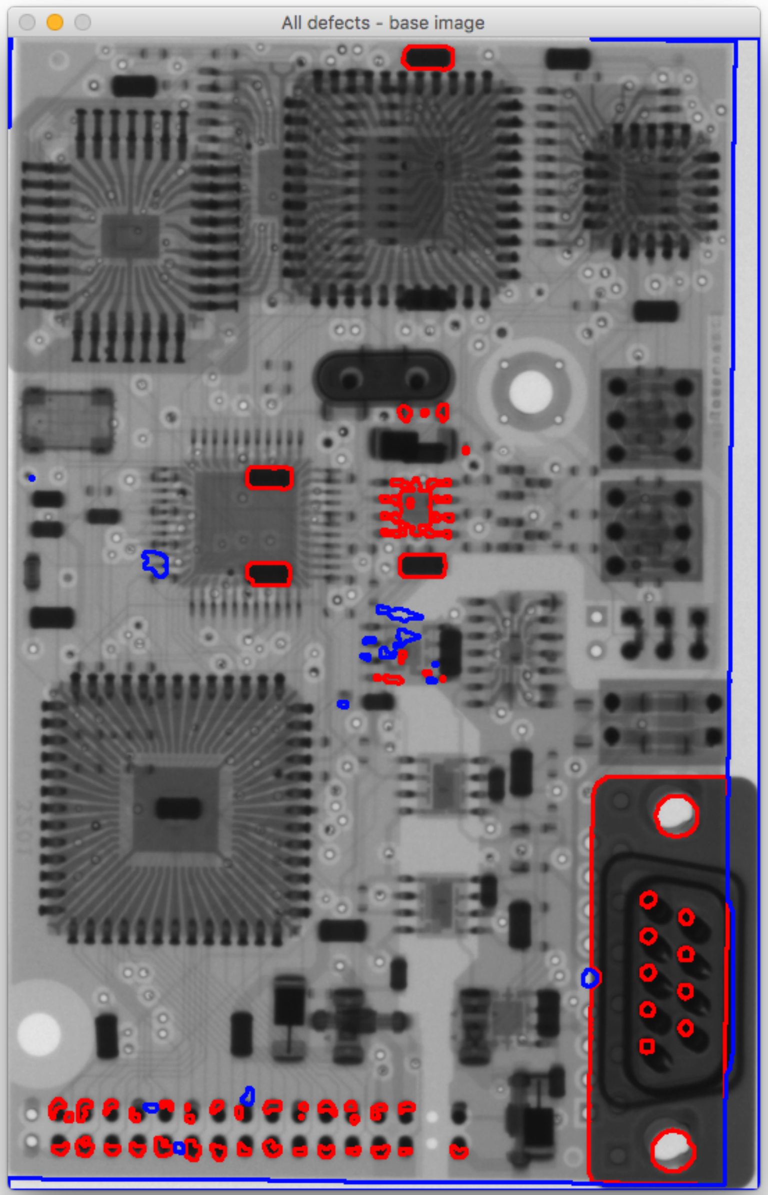

Fig. 20: Result image

.4.4 PCB2 test1

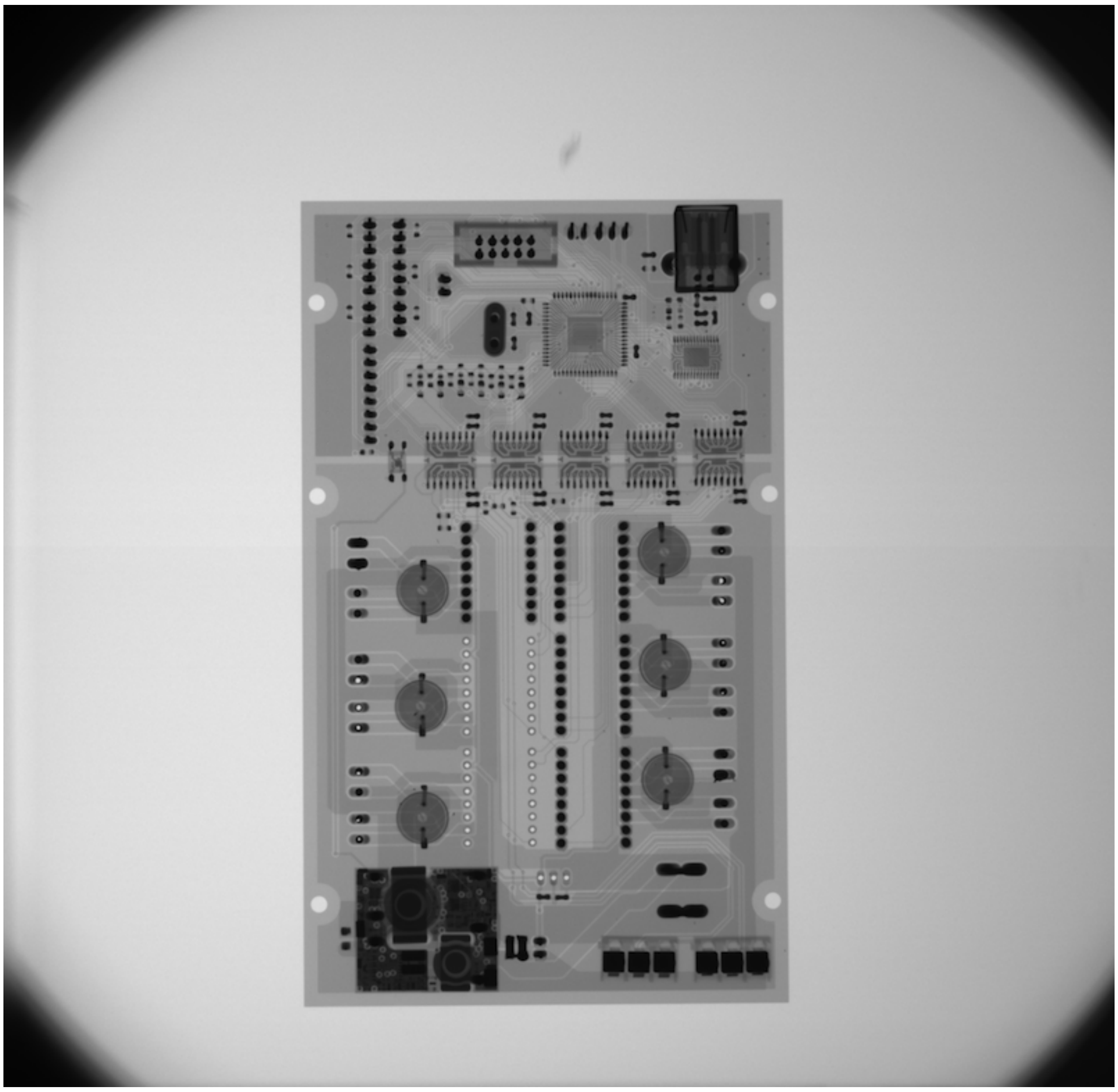

Fig. 21: Base image

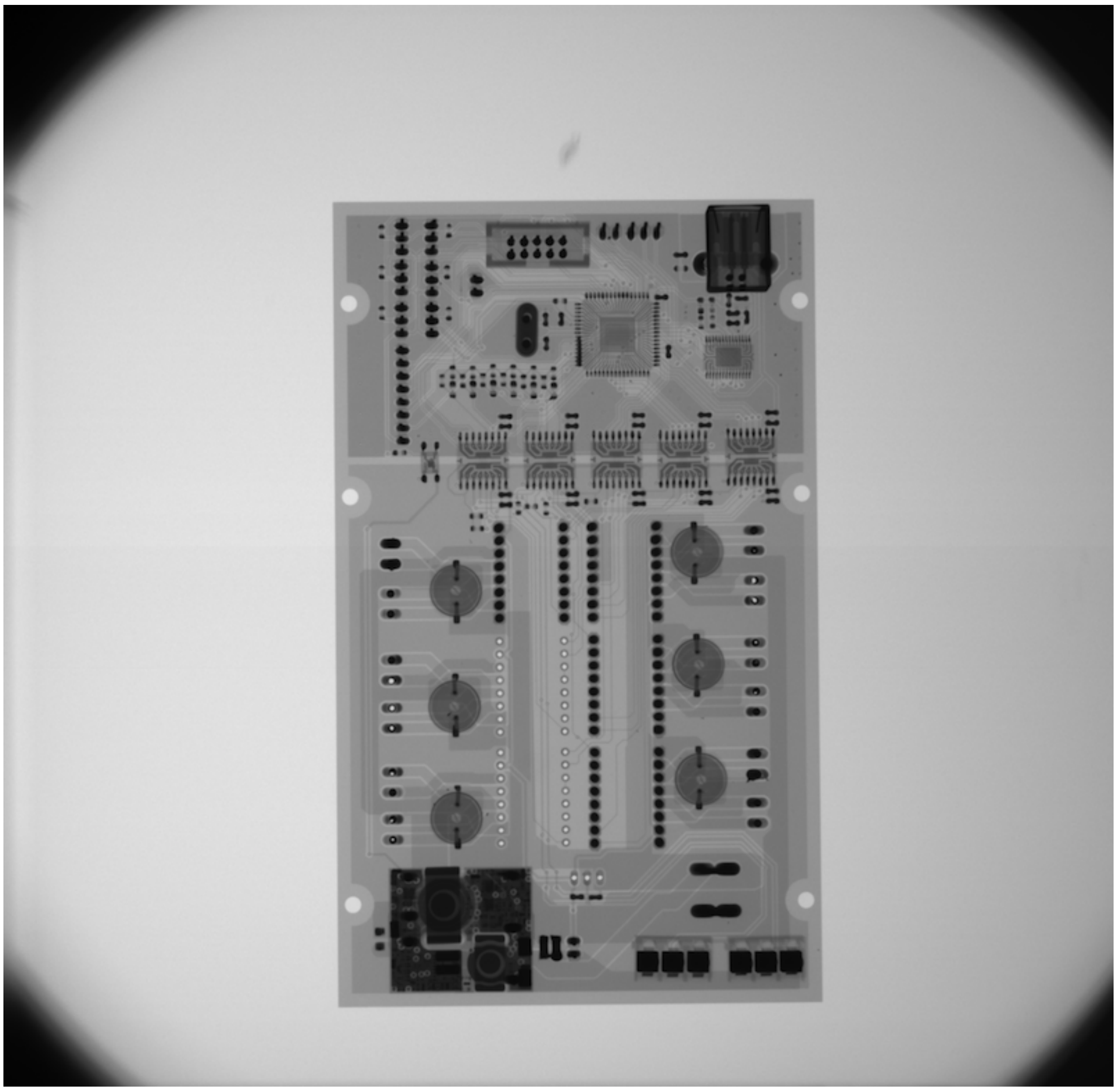

Fig. 22: Tested image

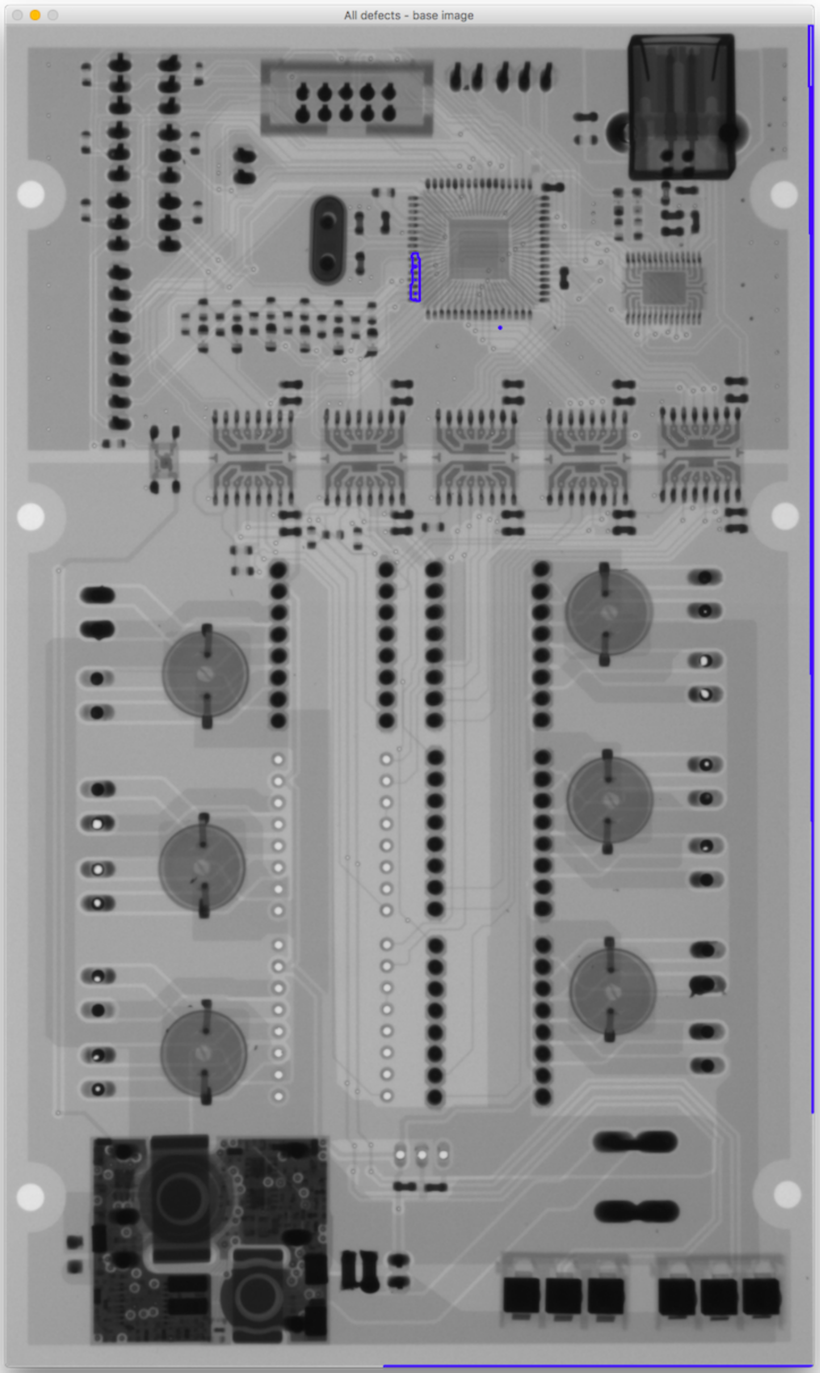

Fig. 23: Result image

.4.5 PCB2 test2

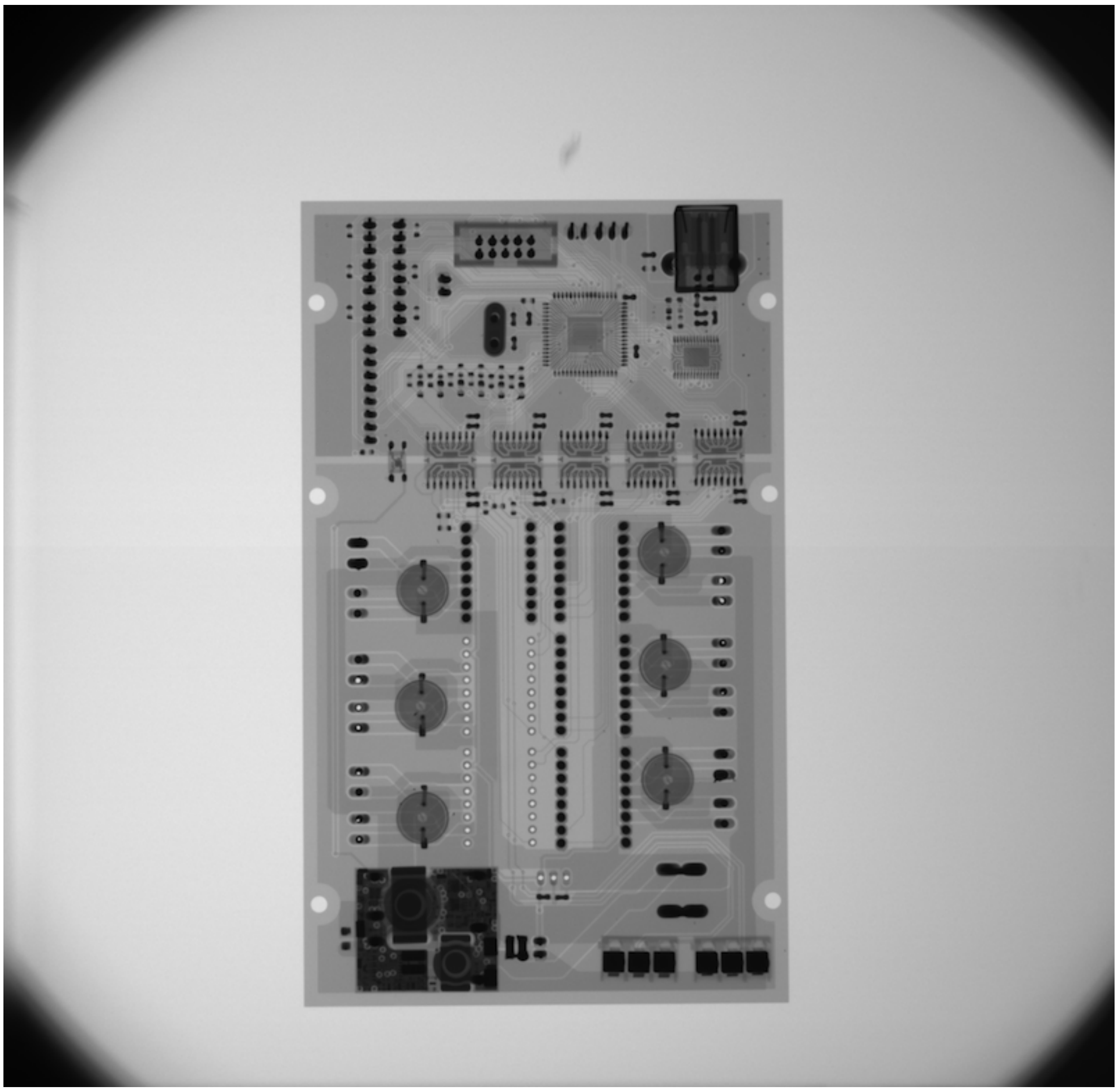

Fig. 24: Base image

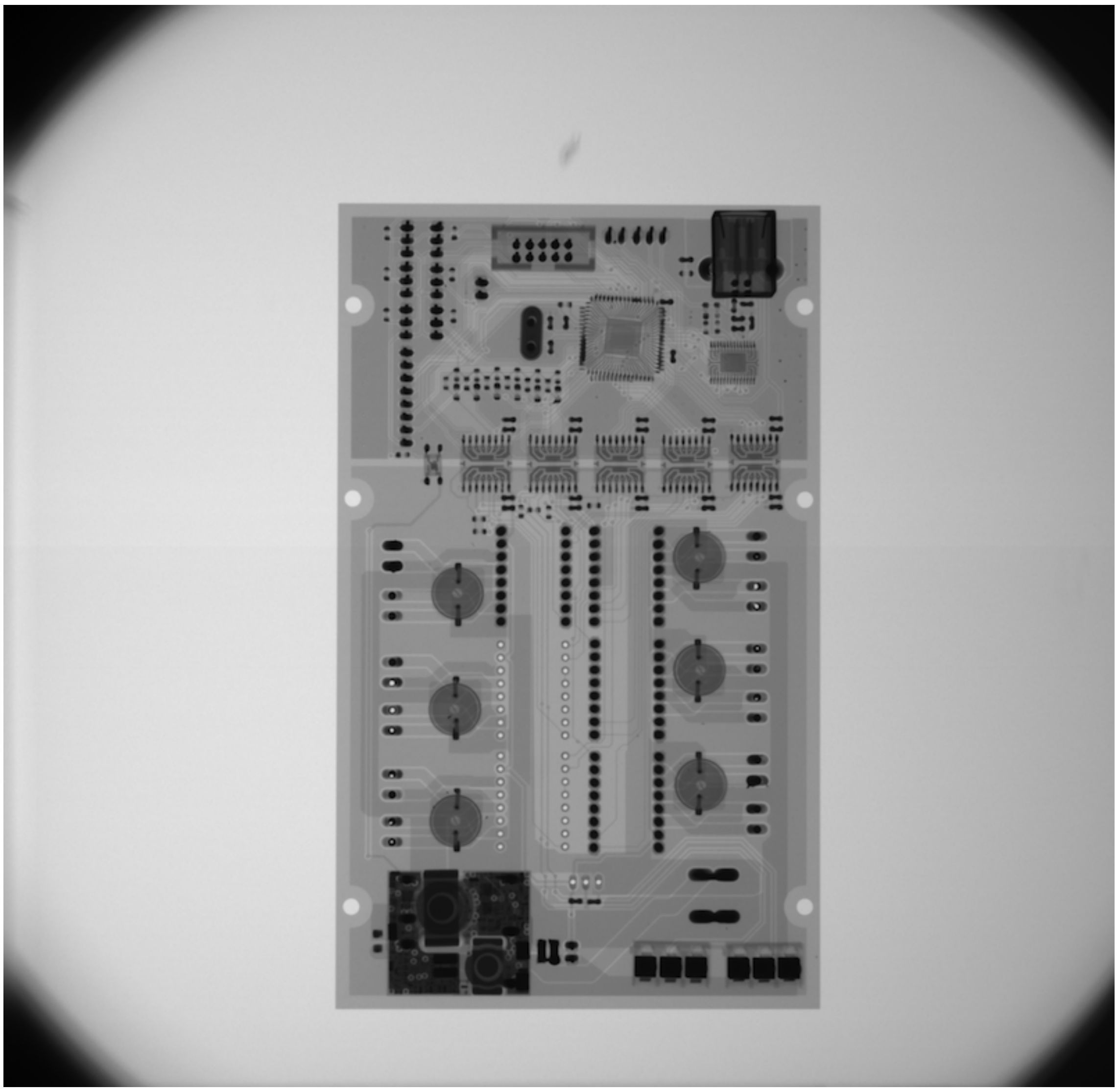

Fig. 25: Tested image

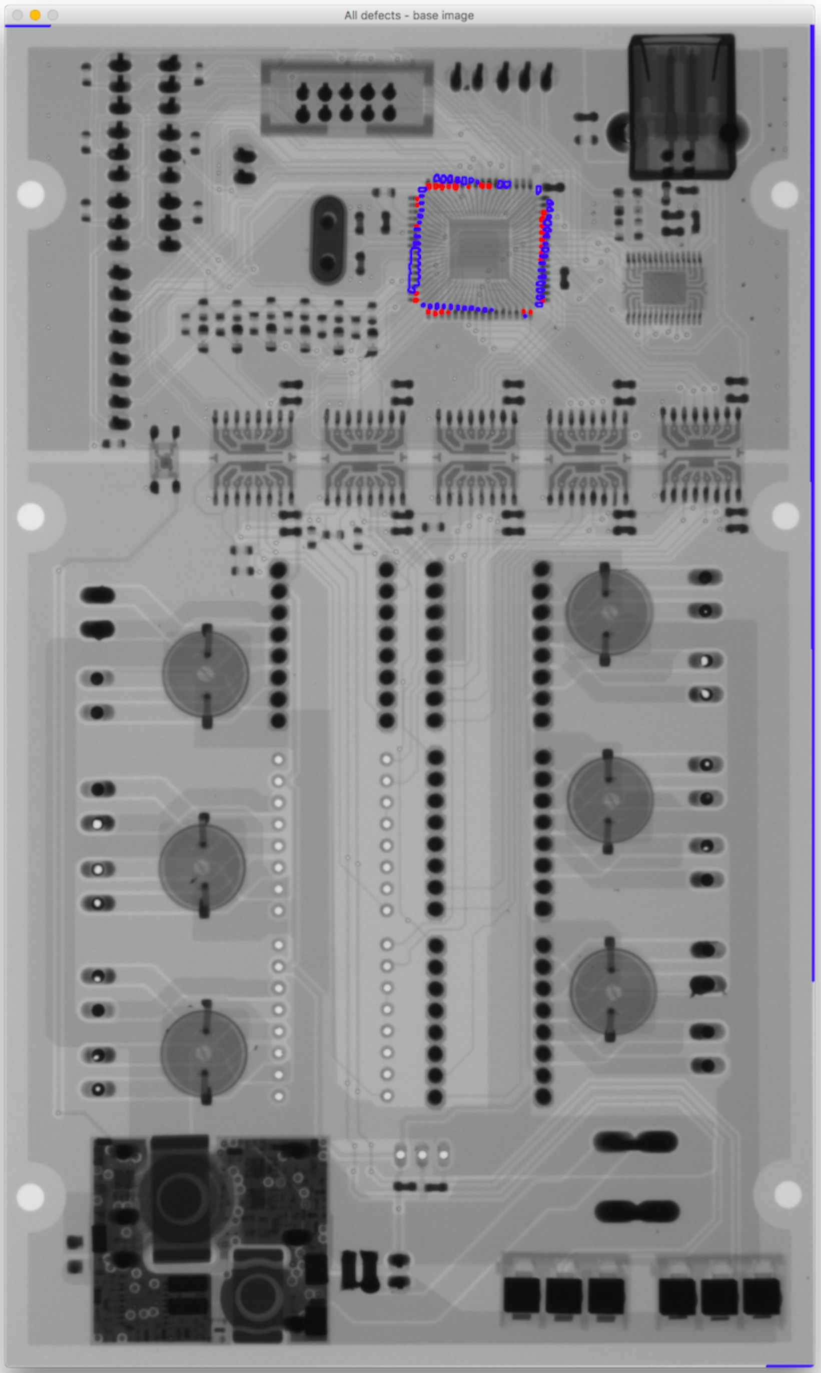

Fig. 26: Result image

.4.6 PCB2 test3

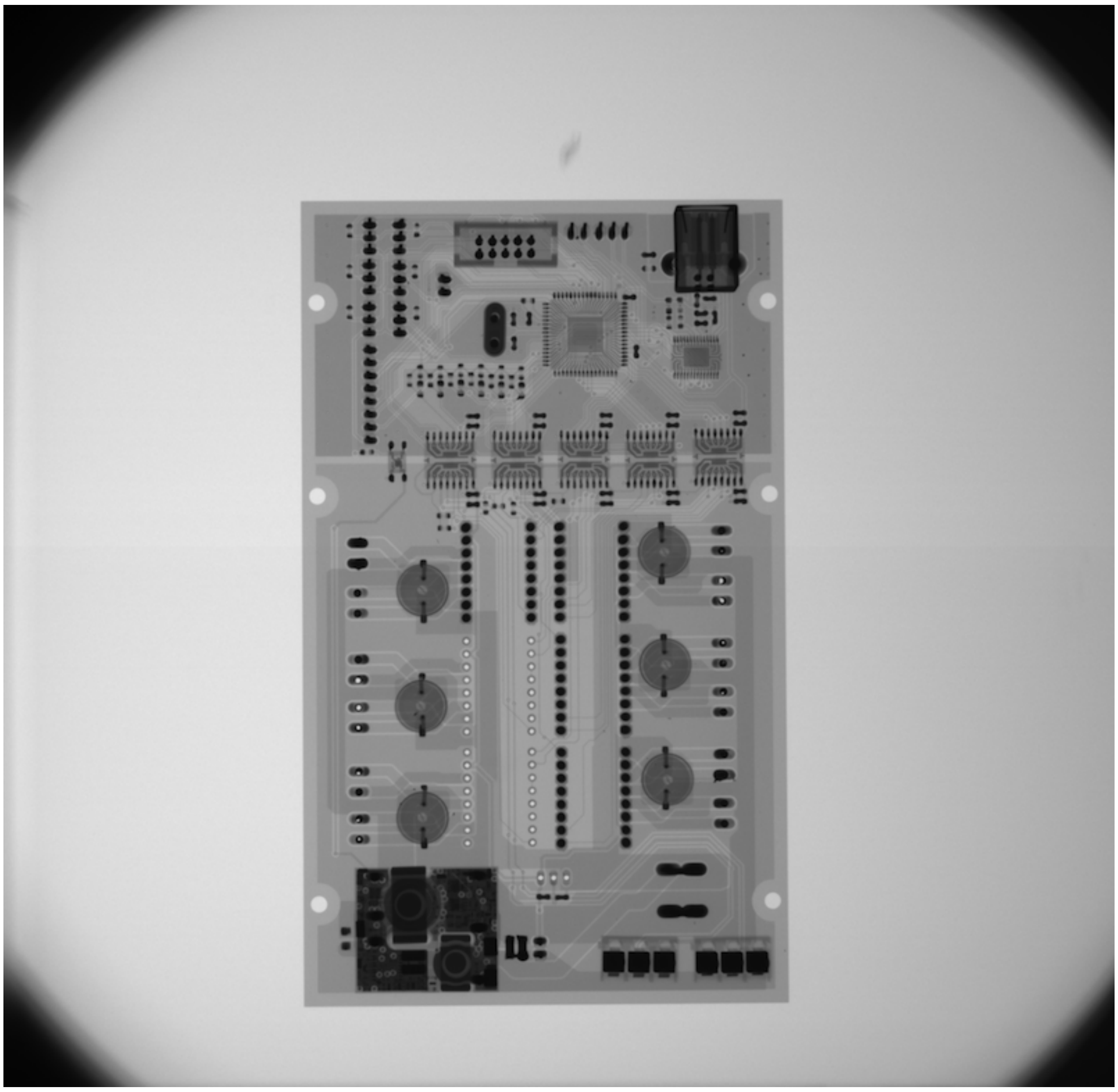

Fig. 27: Base image

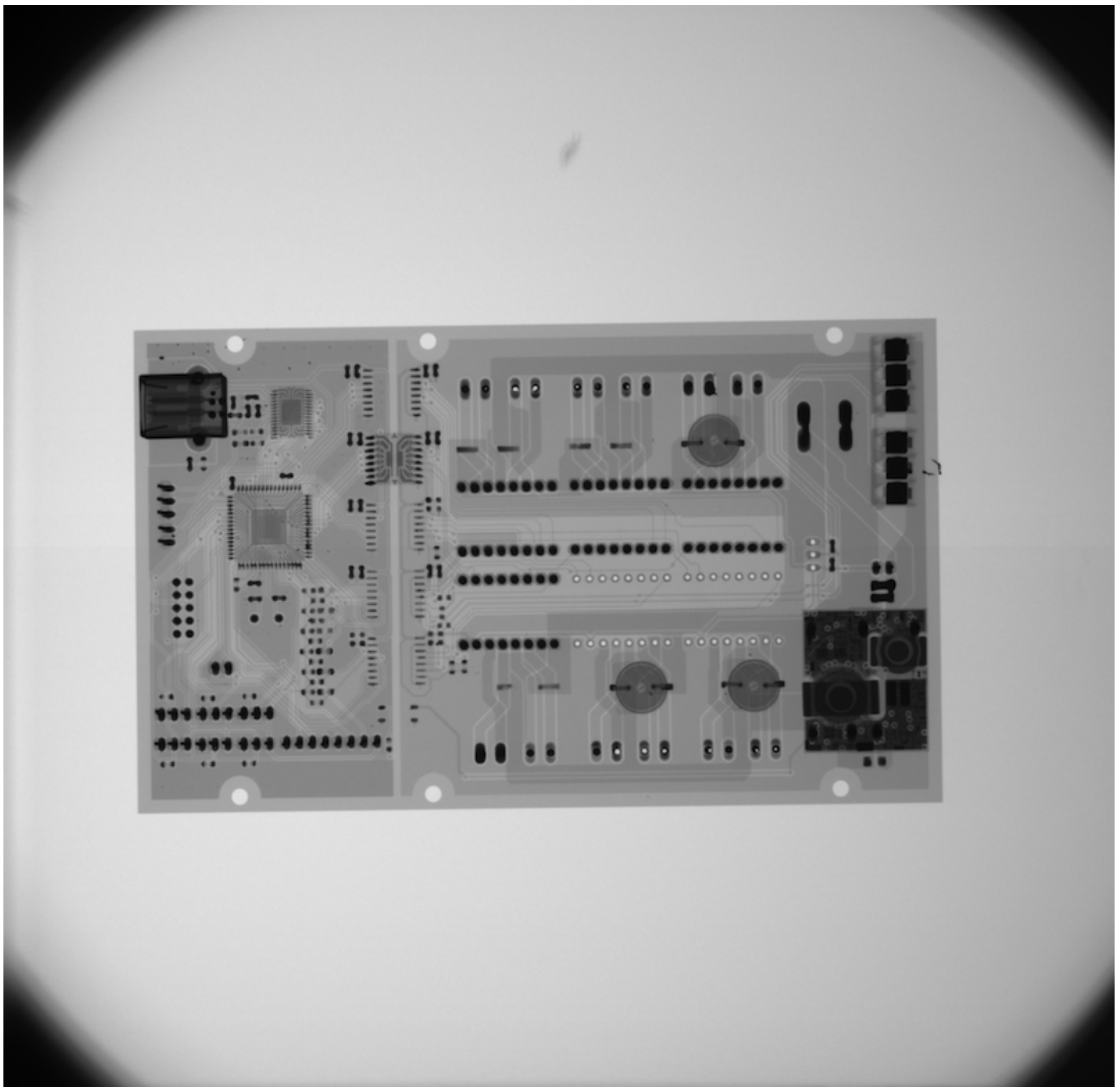

Fig. 28: Tested image

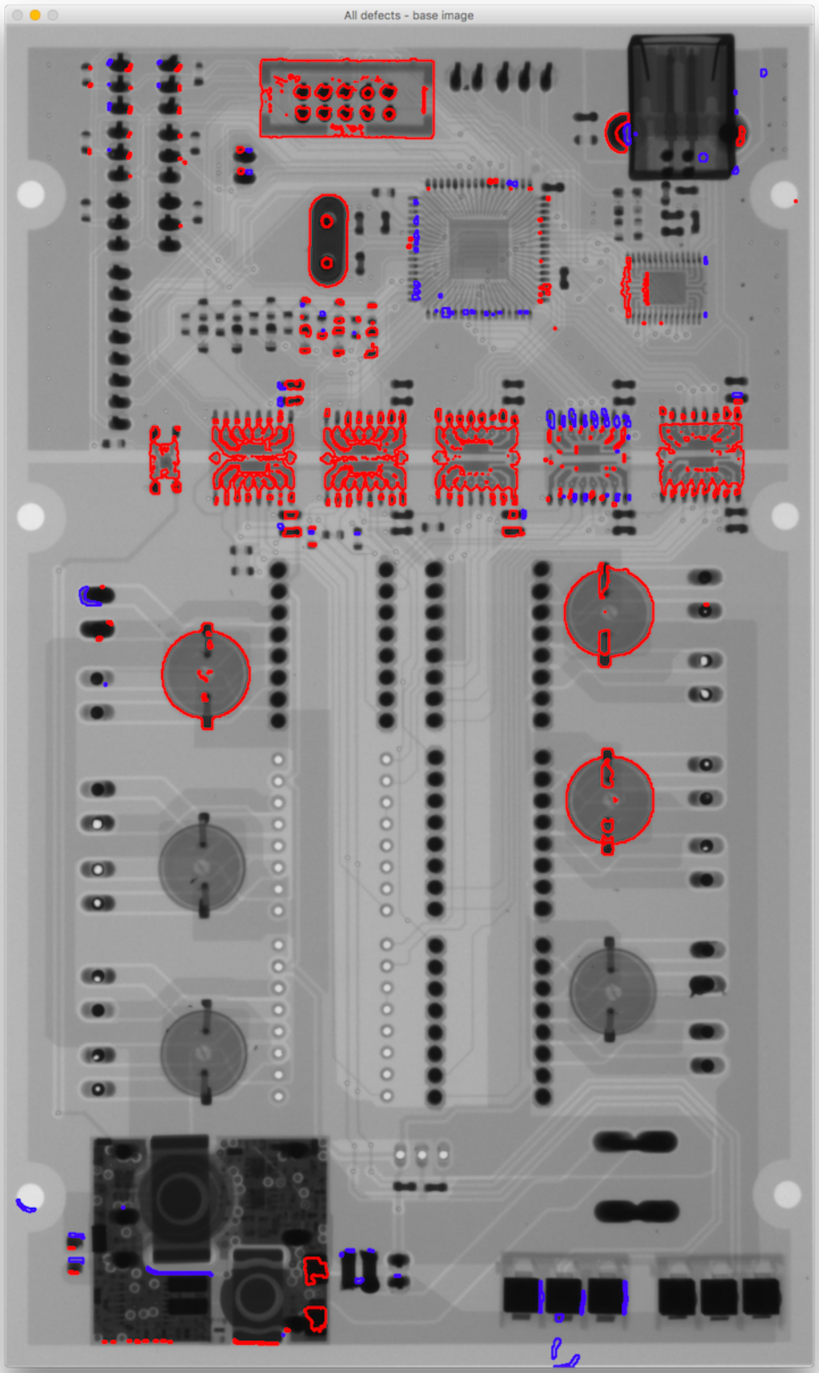

Fig. 29: Result image