People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

University M'Hamed BOUGARA – Boumerdes

Institute of Electrical and Electronic Engineering

Department of Electronics

Final Year Project Report Presented in Partial Fulfilment of

The Requirements for the Degree of

MASTER

In: **Electronics**

Option: Computer Engineering

Title:

Ceramic tiles classification using OpenCV under Embedded Linux

Presented by:

- BOUAKAZ Djihad Nacereddine

Supervisor:

Dr. MAACHE Ahmed

Contents

Dedication

This project is dedicated my patents. My father who did not only raise and nurture me but also taxed himself dearly over the years of my education and intellectual development, thank you for being my biggest support. My mother. Who has been a source of happiness, love and strength during moments of despair and discouragements, her motherly care and support have been shown in incredible ways. I would also like to dedicate my work to my brothers Amine, Islem, Salah and to my two sisters Hiba and Ines. A special thanks to my wife and my life partner for the unlimited support and for the great help during my months of work on this project.

I would also like to thank my friends for being by my side during these great five years.

Acknowledgments

All praise and thanks giving to Allah the most powerful and most merciful for everything and especially for giving me the patience and the knowledge to continue my work ALHAMDOLILAH.

I especially thank my supervisor, Dr. MAACHE Ahmed, for being a great support and for helping me throughout this project. I am grateful to have you as a supervisor and as a teacher, I would also like to thank SAFCER company for being a great host and for giving me the opportunity to do my test and for the amount of information that they gave me.

We would like to acknowledge the donation of the DE10 board done by Intel University Program.

List of figures

List of tables

List of abbreviations

AMBA- Advanced Microcontroller Bus Architecture.

ARM- Advanced RISC Machines.

ASIC- application-specific integrated circuit.

AXI- Advanced extensible Interface.

CPU- Central processing unit.

DDR- Double Data Rate.

DSP- digital signal processing.

FPGA- field programmable gate array.

GB- gigabyte.

GCC- GNU Compiler Collection.

GND- ground.

GPIO- General Purpose Input/Output.

HDL- hardware description language.

HPS- hard processor system.

HW- hardware.

I/O- input/output.

IR- infrared.

LDR- Light Dependent Resistor.

LED- light emitting diode.

LXDE- Lightweight X11 Desktop Environment.

MHz- MegaHertZ.

OpenCV- Open-Source Computer Vision.

PIO- parallel input output.

RAM- random-access memory.

RGB- Red, Green, and Blue.

RISC- Reduced Instruction Set Computer.

SD- Secure Digital.

SoC- System on Chip.

SoPC-	System on Programmable Chip.
SW-	software.
USB-	Universal Serial Bus.
VHDL-	VHSIC Hardware Description Language.
VHSIC-	Very High-Speed Integrated Circuit Program.
VGA-	Video Graphics Array.

Abstract

The proposed project consists in developing a real-time system for automatic inspection of moving ceramic tiles and detection and identification of their surface defects at high-speed using OpenCV and FPGA HPS. Based on vision techniques, the system will consist of a lighting device (LEDs) and a LDR to provide the suitable lighting environment, camera, and embedded Linux running on FPGA DE10 board to create an image processing hardware and software environment. It will allow the visualization of the surface and by means of a data base, it will detect and proceed to the identification of the defects of surface during the production cycle. This process classifies ceramic tiles automatically and allows for making the necessary corrections in due time.

Introduction

In ceramic tile manufacturing processes, great importance is given to the surface condition and inspection possibilities of the products during their production. The simple visual inspection is in fact unable to follow the product which is in movement even at low speed. The inspection of the surface can only be carried out as a sampling which obviously remains non-exhaustive and where error is inevitable. End-of-process inspection is not the ideal solution because it can only trace the history of the process and provide information on its trends. Consequently, defects in the final product that are not detected and corrected lead to downgrading of the products and induce additional costs, also a certain classification in the choice of the product may help in terms of fixing the price and the size of the final product.

In recent years, image processing is playing key role in finding solution to many problems in fields such as medical, industry, security, remote sensing applications and so on. But in some applications system performance has become bottle neck. Most of the image processing systems are developed based on Desktop PC which is more generic for image processing applications and these systems may not meet the requirement of real time performance. Along with the development of advanced and computationally intensive image processing algorithms, computational platforms based on FPGA chip and HPS processor are also developed. This project presents an image processing system for ceramic tile classification based on an FPGA chip and HPS processor. In this system the Cyclone V is used as image processing core and a DE-10 FPGA chip is used for image sampling and display.

Organization of the report

To understand the functionality of the system along with its features, we first have to understand the problem, and to do so, chapter one explains the main procedures in the ceramic tiles’ production and the existing classification methods with their advantages and disadvantages. Chapter two gives more information about the theoretical background for both software part (VHDL, image processing, C++, OpenCV, and Linux) and hardware part (DE10 board and every other component used in this project). Chapter three includes the design, the codes, the flowcharts, and the used circuits. Chapter four deals with the implementation of the system and shows the interaction between the software and the hardware and how to use the system efficiently.

CHAPTER I

Ceramic tiles production

1-1- Introduction

This chapter deals with everything that has a relation with ceramic tile production, and since most of the calculation and the tests were made during my internship in SAFCER company, a brief introduction about it and different kinds of models they produce, in addition the used classification method in this company.

1-2- SAFCER company

SAFCER is the result of more than half a century of experience in the field of construction and production mainly:

- Ceramic tiles for floor, wall and ventilated facades in red and white body,

- Building bricks and slabs.

Figure 1.1: SAFCER GROUP LOGO [1].

1-3- Production procedures

In the ceramic tiles production, several processes are required before being finalized and before being presented to the costumer:

1 - first step is called weighing and grinding, where the raw material is the slip and the atomized powder, where chemically it consists of different molecules and each the weighting is approximated and can’t be fixed due to the large quantity used.

2 - Second step is the pressing step where the automized powder is being pressed using specific machines in the shape of a rectangle, the dimensions are fixed based on the dimension of the tile

that they want to get, for example they use a 64.75cm x 64.75cm pressed tile to get a 60cm x 60cm as a final product.

3 - Drying step, where the tile is dried it from water to be more solid.

4 - Enameling and englobing step, where the add the two materials (enamel and englobe), after this step all of the tiles have the same size and the same dimension.

5 - Printing step, a drawing will be printed in the tile, depending on the model of the tile.

6 - Step number 6, the tiles have to be cooked in a big oven at a high temperature (>150°).

7 - Control and classification, here the tiles are being controlled to see if they meet the minimum requirement to be sold, and classify them based on their quality and dimension.

1-4- Deflect causes

Even though the company uses high technology and tries to keep the product as clean and good as possible, certain problem can occur during the production, these problems prevent the ceramic tiles to be as the desired ones:

- The final chemical composition from a tile to another can't be identical 100%, due to the variety in the chemical composition of the atomized powder, so when the tile is cooked, some materials take longer than others, so the shape of the final tiles can't be the same, some are bigger than the others and some won't have the shape of a perfect rectangle.
- During the printing step, the printer can make some problems, a slight difference in the colors or a color may be printed twice, some places won't printed on some times, a lack in the color in the printing machine can make a difference in the tiles.
- When a tile passes through the oven, black stains can be seen at the surface of the tile.
- The atomized powder coming from outside the company can have different colors, so it affects the intensity of the colors in the final product.

1-5- Type of tiles produced

SAFCER as every other ceramic company, has a lot of models with different shapes and different dimensions, in this part we are going to see the basic models of the company:

1/ Unicolor: a tile that has only one color printed on it, where the designing departments chooses the colors that has to be printed:

Figure 1.2: Design of a unicolor tile[1]

2/ Geometric tile: a tile that has a specific geometric drawing on it, the shape of the drawing and the colors must be preserved on the tile:

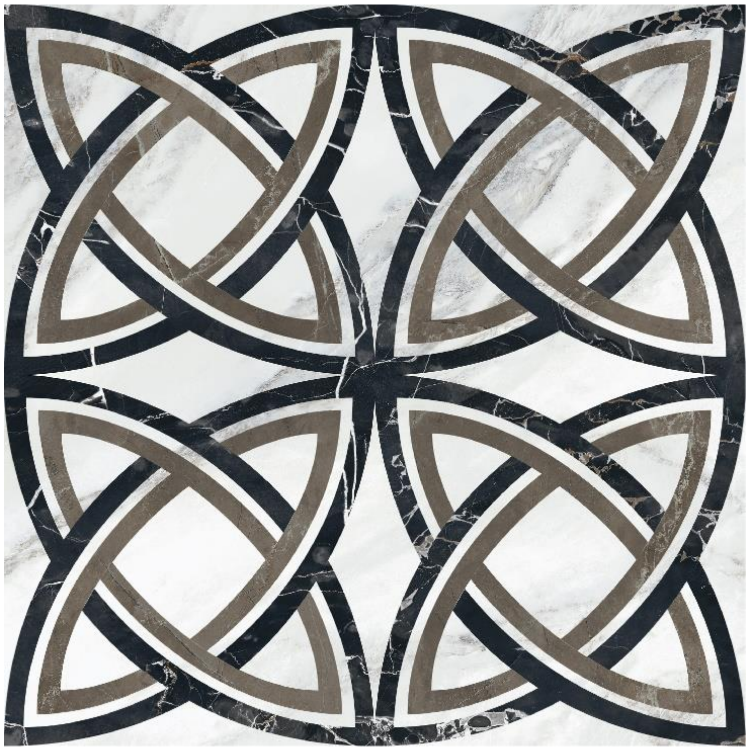


Figure 1.3: Design of a geometric tile.[1]

3/Random selection: the computer graphics department gives a giant image that is much bigger than the diameter of the tile, the specific machine selects randomly part of this image and prints it on the tile.

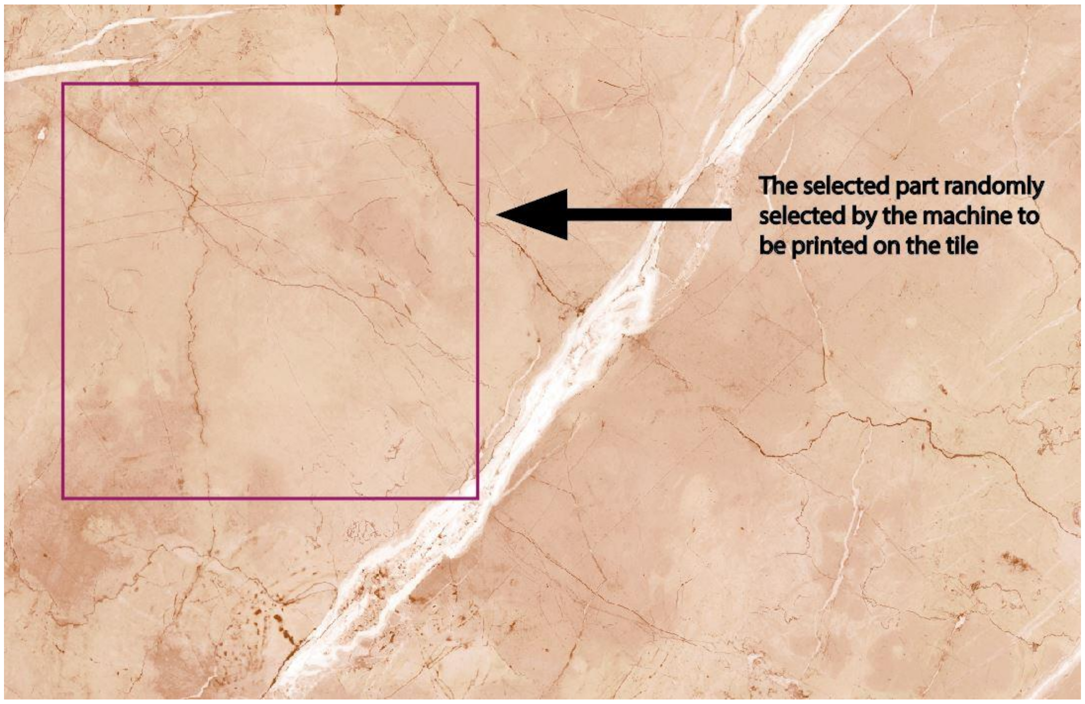

The selected part randomly selected by the machine to be printed on the tile

Figure 1.4: The whole design and the specific area that has to be printed on the tile.[1]

The tiles differ also in terms of shapes and dimensions (60cm x 60cm, 58cm x 58cm, 60cm x 40cm, 60cm x 30cm, 15cm x 15cm …etc).

1-6- Classification

During the production of a ceramic tile, the problems mentioned previously may happen, so a certain classification of the product has to be done in terms of dimension, quality and form:

- Classification based on the dimension of the tile has to be done, the tiles that have the same dimension are grouped together so when the costumer buys it won't have different tiles:

Table 1: Classification based on dimension [2]

Choice of tile	Class	Length error		Width error		Height
		from	to	from	to	
Premium	A	-0.25%	-0.125%	-0.25%	-0.125%	10.4cm
Premium	B	-0.125%	+0.125%	-0.125%	+0.125%	10.4cm
Premium	C	+0.125%	+0.375%	+0.125%	+0.375%	10.4cm
Premium	D	+0.375%	+0.5%	+0.375%	+0.5%	10.4cm
Commercial	/	>-0.25% and <+0.5%		>-0.25% and <+0.5%		10.4cm

- During the cooking of the tile, a curvature (convexity or concavity) may be noticeable so they have to be classified as:
  *Perfect tiles (flat tile) as a premium choice tile.
  *A tile that suffers from a little curving commercial-choice tile.

- The tile may also suffer from a problem in the tonality of the color, a break in the corners, a slight shift at the sides, an excess or a lack of Granilia in the tile, a sting that may cause a black point, stains caused by the oven, a rift in the surface of the tile, a distortion visible to the naked eye, a difference in the color, a difference in the class of dimensions (for example length class A, width class C).

and based on these mistakes we can classify the tiles as follow:

- First-choice: a perfect tile.
- Commercial-choice: a tile that suffers from one or more than one problem from the problems that we've sited in previously, but it doesn't have to exceed a specific range so that we can classify it in this choice instead of the third choice.
- Third-choice: a none-useable tile that can have either:
  - a large crack on the surface of the depositing tile.
  - A big break in the surface.
  - a big difference in the colors.

- Big printing error.

These are all the classification criteria that we base on to classify a tile as a premium-choice (dimension a, b, c, or d), commercial-choice or a third-choice (none-usable tile).

1-7- Existing classification method

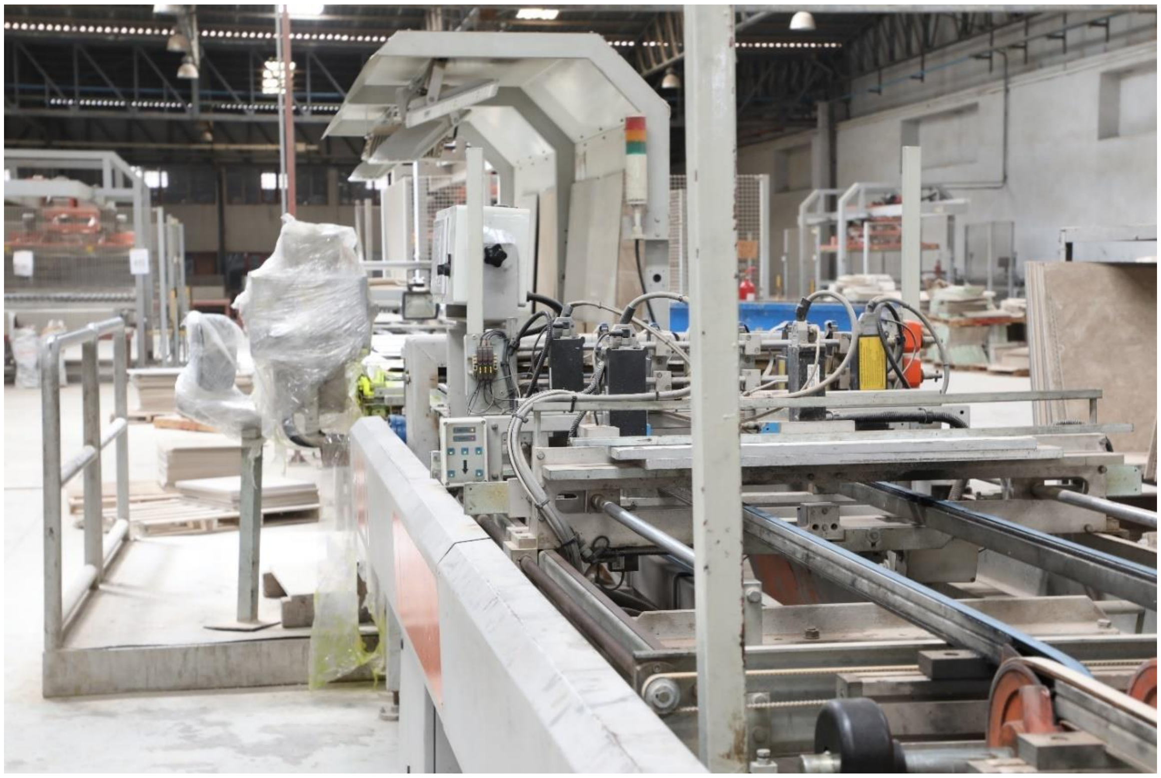


Figure 1.5: Classification method used.

The existing classification method in the company is divided into two parts:

Visual classification: two employees are classifying the tiles visually, if any seen deflect in the surface of the tile, they mark it to be classified with the commercial choice, and if the deflect is too big, the directly take it out of the line to be classified as a third-choice product.

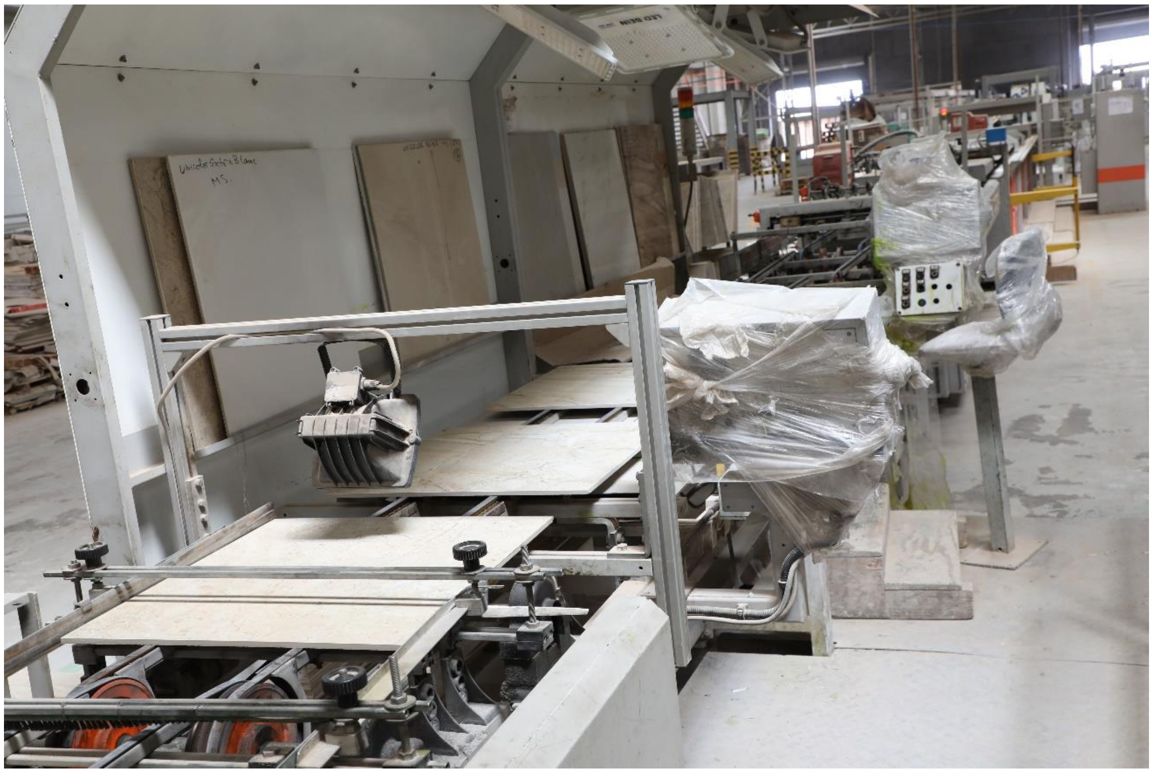


Figure 1.6: Visual classification.

Automatic classification: the company uses some developed machines to measure the dimension of the tiles to classify them based on the values mentioned in table 1, the marks made by the employee in the visual classification, and the flatness of the tile.

the automatic check is based on a system that uses 4 Infrared sensors to measure the distance and get the dimension of the tile, and 7 other Infrareds to measure the percentage of the error of the flatness.

This system contains also a mark reader, it reads the marks made by the employee in the tiles that has a small visual deflect in the surface.

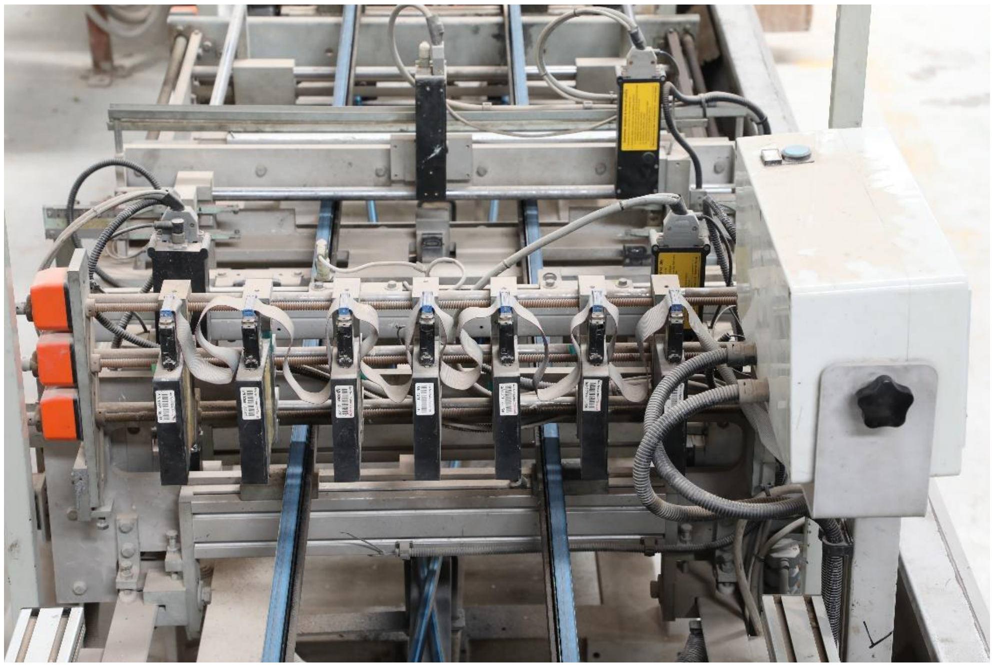


Figure 1.7: Flatness automatic check.

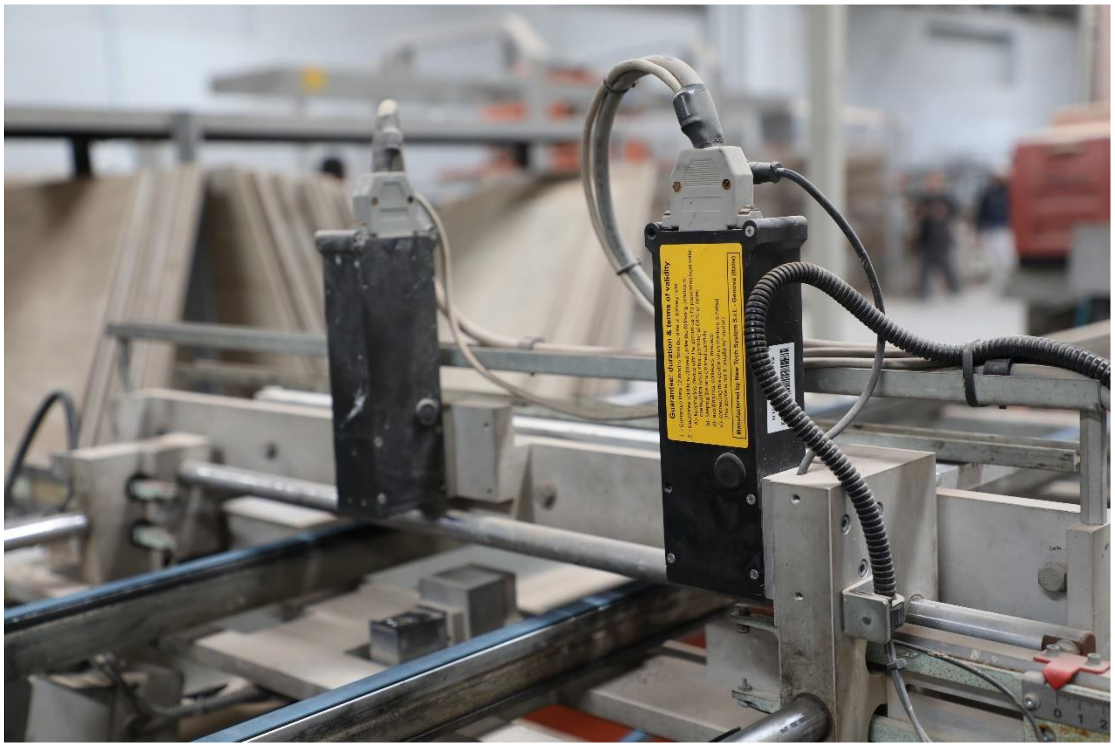


Figure 1.8: Dimension automatic check.

1-8- Disadvantages of the current classification method

Generally, The employees in the visual classification method work 8 hours a day, and this is a bit of a problem, because as the time passes, the employee gets tired and percentage of the error starts to increase, they may skip some errors and let tiles with error passes as a premium choice product.

Also, this technique takes a lot of time to detect the error visually, the worker stops the line, and sometime has to call his supervisor to get his opinion about the deflect, this process wastes a lot of time and for a company that produces 17,000 m^2 a day, a slight millisecond delay per tile counts.

1-9- Conclusion

In this chapter we dealt with different products made by the SAFCER company, the classification criteria and the existing methods with their disadvantages, we understood the process of the production and the problems that may happen, actually, we can integrate our system to optimize time and it increases the accuracy in the deflect detection.

CHAPTER II

Theoretical background

2.1 Introduction

This chapter presents the background knowledge of the systems referred to in this thesis and the development environment. The first section presents the characteristics of FPGAs, including their internal structures, core technology and DE10 standard board and the HPS system. The second section of this chapter briefly introduces the field of computer vision and image processing and describes the background to this technology. The next section focuses on the comparison between FPGAs and architectural approaches. Finally, the software and hardware component used in this project are briefly presented.

2.2 field-programmable gate array (FPGA)

A **field-programmable gate array** is an integrated circuit designed to be configured by a customer or a designer after manufacturing – hence the term "field-programmable". The FPGA configuration is generally specified using a hardware description language (HDL), similar to that used for an application-specific integrated circuit (ASIC). Circuit diagrams were previously used to specify the configuration, but this is increasingly rare due to the advent of electronic design automation tools.

FPGAs contain an array of programmable logic blocks, and a hierarchy of "reconfigurable interconnects" allowing blocks to be "wired together", like many logic gates that can be inter-wired in different configurations. Logic blocks can be configured to perform complex combinational functions, or merely simple logic gates like AND and XOR. In most FPGAs, logic blocks also include memory elements, which may be simple flip-flops or more complete blocks of memory. Many FPGAs can be reprogrammed to implement different logic functions, allowing flexible reconfigurable computing as performed in computer software. FPGAs have a remarkable role in embedded system development due to their capability to start system software (SW) development simultaneously with hardware (HW), enable system performance simulations at a very early phase of the development, and allow various system partitioning (SW and HW) trials and iterations before final freezing of the system architecture.[3]

2.2.1. DE-10 standard development board

The DE10-Standard Development Kit presents a robust hardware design platform built around the Intel System-on-Chip (SoC) FPGA, which combines the latest dual-core Cortex-A9 embedded cores with industry-leading programmable logic for ultimate design flexibility. Users can now leverage the power of tremendous re-configurability paired with a high-performance, low-power processor system. Intel's SoC integrates an ARM-based hard processor system (HPS) consisting of processor, peripherals and memory interfaces tied seamlessly with the FPGA fabric using a high-bandwidth interconnect backbone. The DE10-Standard development board is equipped with high-speed DDR3 memory, video and audio capabilities, Ethernet networking, and much more that promise many exciting applications.[4]

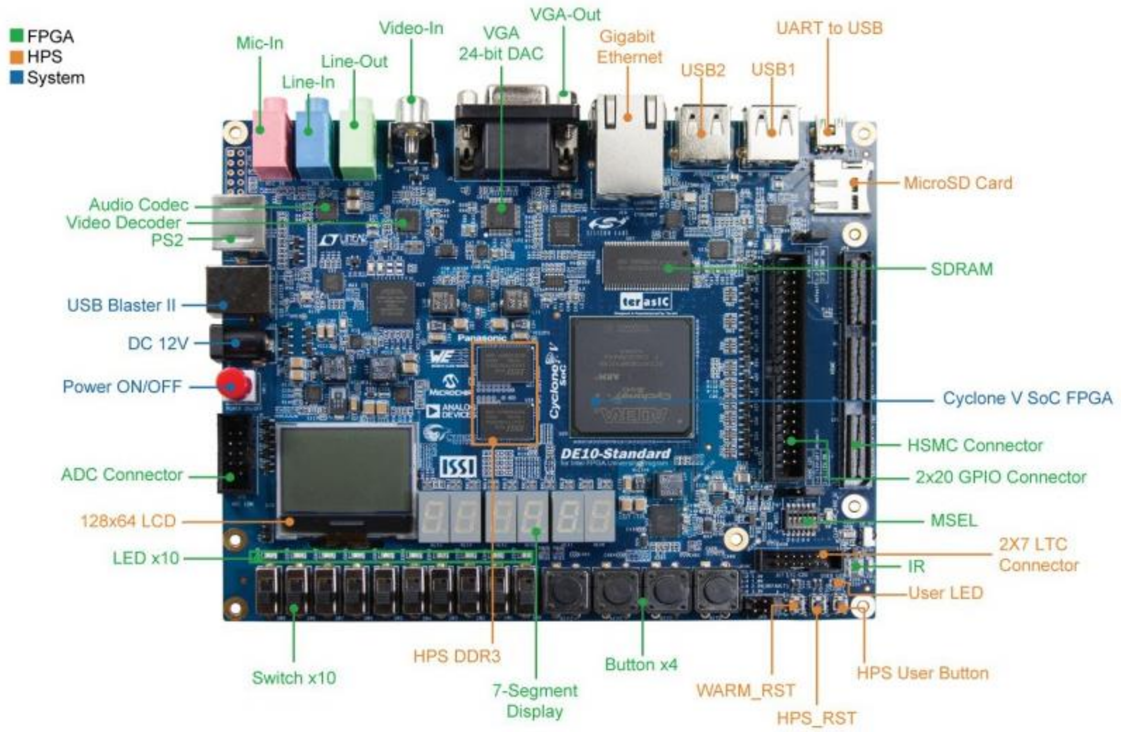


Figure 2.1: DE10-Standard Development Board [4].

2.2.2. Hard Processor System (HPS)

Intel's SoPC integrates an ARM hard processor system (HPS) consisting of processor, peripherals and memory interfaces tied seamlessly with the FPGA fabric using a high-bandwidth interconnect backbone, the figure bellow shows the HPS-FPGA Bridges. The HPS has three bridges that use memory-mapped interfaces to the FPGA based on the Arm Advanced Microcontroller Bus Architecture (AMBA) and Advanced extensible Interface (AXI) to allow communication between the FPGA and the ARM processor [5].

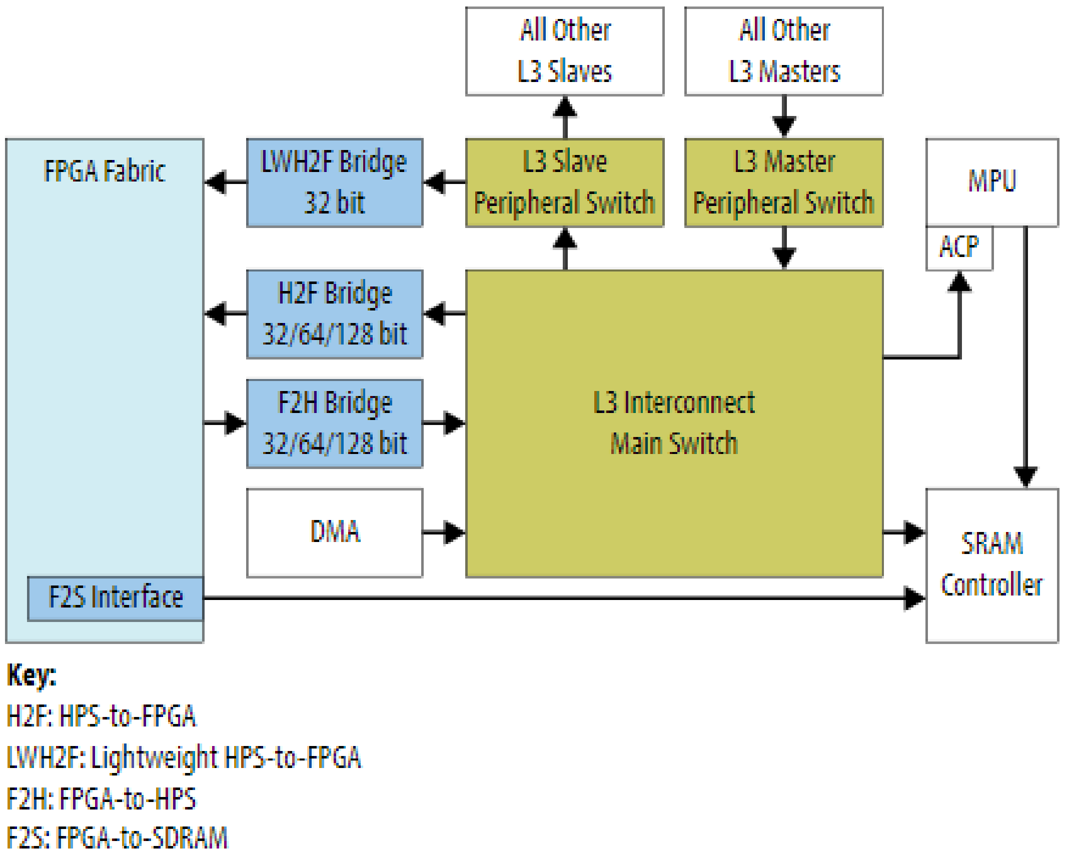


Figure 2.2: HPS-FPGA Bridges [5].

2.2.3. DE-10 ARM based HPS

The Cyclone® V system-on-a-chip (SoC) is composed of two distinct portions- a single- or dual-core Arm Cortex-A9 hard processor system (HPS) and an FPGA. The HPS architecture integrates a wide set of peripherals that reduce board size and increase performance within a system. The SoC features the FPGA I/O, which is I/O pins dedicated to the FPGA fabric.[6]
And the FPGA arm processor consists of:

- 925 MHz, Dual-Core ARM Cortex-A9 MPCore Processor
- 512 KB of Shared L2 Cache
- 1 GB DDR3 RAM.
- 64 KB of Scratch RAM
- Multiport SDRAM Controller with Support for DDR2, DDR3, LPDDR1, and LPDDR2
- 8-Channel Direct Memory Access (DMA) Controller

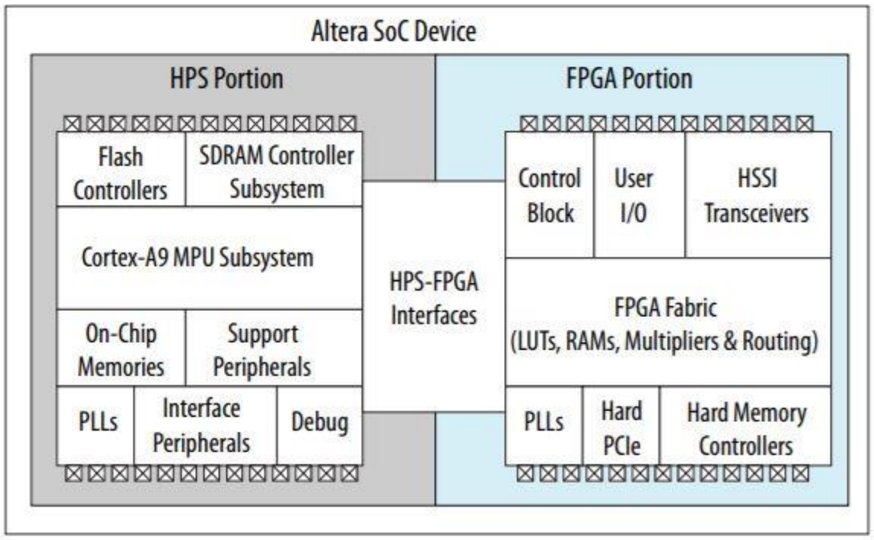


Figure 2.3: Intel SOC device block diagram [7].

2.3 Image processing

Image processing is a method to perform some operations on an image, in order to get an enhanced image or to extract some useful information from it. It is a type of signal processing in which input is an image and output may be image or characteristics/features associated with that image. Nowadays, image processing is among rapidly growing technologies. It forms core research area within engineering and computer science disciplines too.

Image processing basically includes the following three steps:

- Importing the image via image acquisition tools;
- Analyzing and manipulating the image;
- Output in which result can be altered image or report that is based on image analysis.

There are two types of methods used for image processing namely, analogue and digital image processing. Analogue image processing can be used for the hard copies like printouts and photographs or in our case printing drawings on the tiles. Image analysis use various fundamentals of interpretation while using these visual techniques. Digital image processing techniques help in manipulation of the digital images by using computers. The three general phases that all types of data have to undergo while using digital technique are pre-processing, enhancement, and display, information extraction.[8]

2.3.1. Digital image processing

Digital image processing is the use of a digital computer to process digital images through an algorithm. As a subcategory or field of digital signal processing, digital image processing has many advantages over analog image processing. It allows a much wider range of algorithms to be applied to the input data and can avoid problems such as the build-up of noise and distortion during processing. Since images are defined over two dimensions (perhaps more) digital image processing may be modeled in the form of multidimensional systems. The generation and development of digital image processing are mainly affected by three factors: first, the development of computers; second, the development of mathematics (especially the creation and improvement of discrete mathematics theory); third, the demand for a wide range of applications in environment, agriculture, military, industry and medical science has increased.[9] or in industry in our case.

2.3.2. Real time image processing

Real-time image processing systems involve processing vast amounts of image data in a timely manner for the purpose of extracting useful information, which could mean anything from obtaining an enhanced image to intelligent scene analysis.[10]

2.3.3. Images in computer science

Graphics on a screen (images specifically) are made up of tiny blocks called pixels. The more pixels on the screen, the higher the resolution and the better the quality of the picture will be.[11] and each pixel is represented by three values in colored images and in one value in grayscale images, the three values in the colored images or RGB images are (RED, GREEN, and BLUE), the concept of RGB was taken from real life, where the eye perceives the color using three types of sensors (cones) sensitive to three ranges of wavelengths.

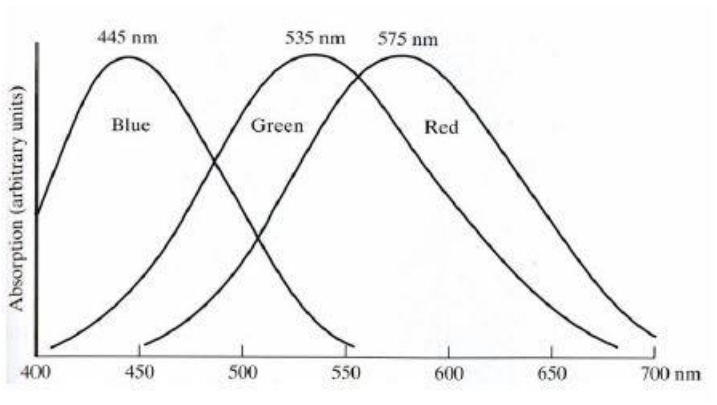


Figure 2.4: Absorption of light by the cones in the human eye [12].

Any color can be constructed from these three colors, and the figure 2.5 demonstrates that:

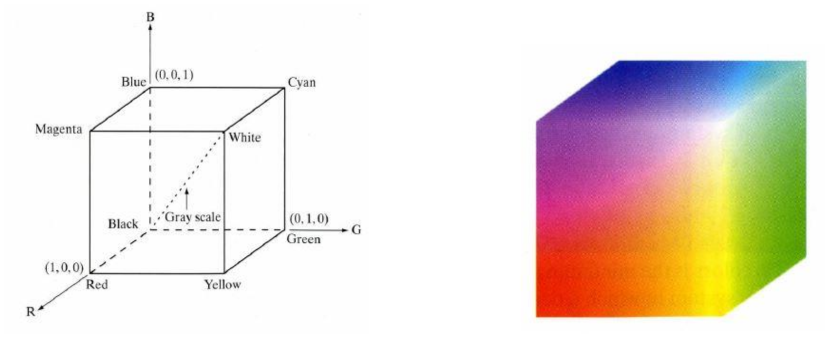


Figure 2.5: **24 bits** RGB cube [12].

So, any image we see by our eye is just the combination of the three images and each of these images is represented by pixels that has values of the concentration of only one of the previously sited colors (RGB), as we can see in the example shown in figure 2.6.

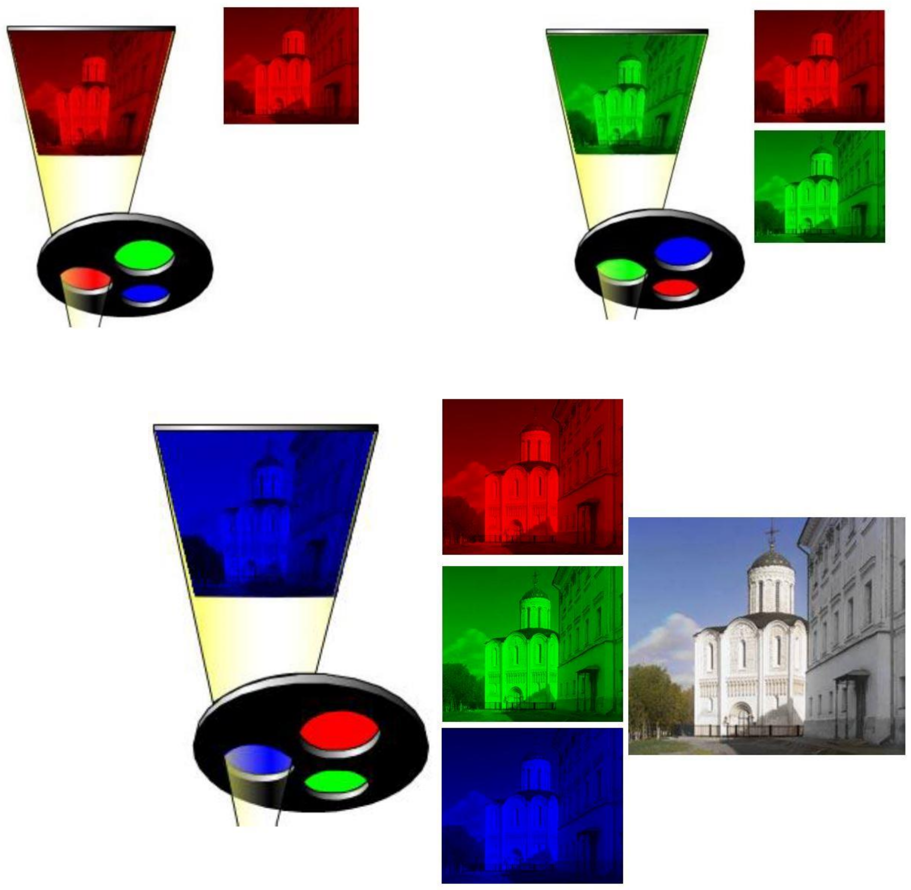


Figure 2.6: Image representation in RGB [12].

2.3.4. Types of filters used

RGB to Grayscale: this filter is used to convert an image from a colored one to a gray one, and this is done by summing the three values (R, G and B) and dividing them by 3, so instead of having a matrix of arrays of three element, we will have a normal matrix having only one value for each pixel.

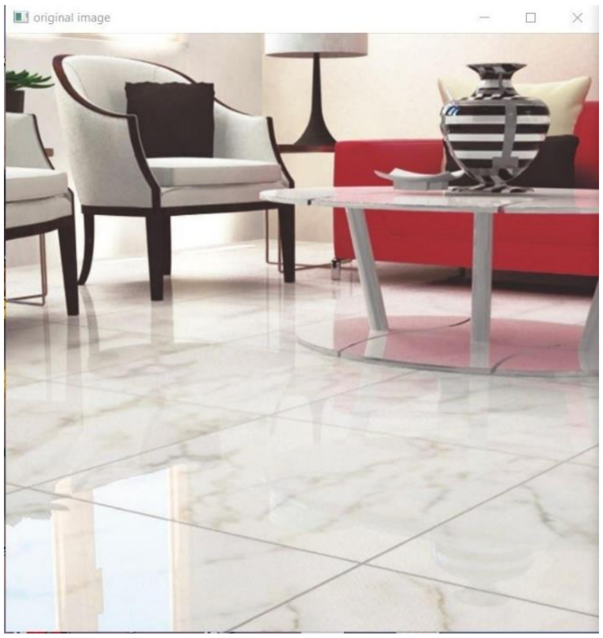


Figure 2.7: Example from RGB to gray scale (before).

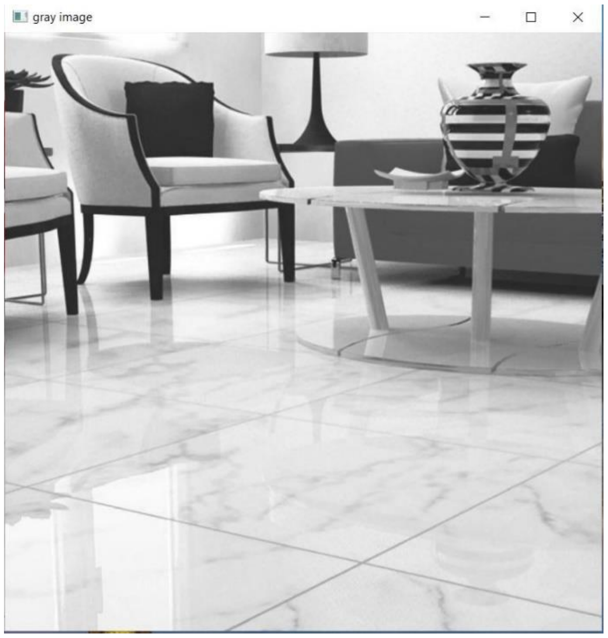


Figure 2.8: Example from RGB to gray scale. (after).

Gaussian Blur filter: this filter is typically used to reduce image noise and reduce detail. The visual effect of this blurring technique is a smooth blur resembling that of viewing the image through a translucent screen, The Blur filters soften a selection or an image and are useful for retouching. They smooth transitions by averaging the color values of pixels next to the hard edges of defined lines and shaded areas. Blur. Eliminates noise where significant color transitions occur in an image, it uses the following equation:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)} \qquad (2.1)$$

where x represents the distance from the origin in the horizontal axis, y is the distance from the origin in the vertical axis, and σ is the standard deviation. To implement the Gaussian function using a convolution method, it is necessary to calculate the convolution mask for the Gaussian function. A convolution (kernel) mask is usually much smaller than the actual image. As a result, the mask is slid over the image and applied at each location in the image. The larger the kernel size is, the lower the detector's sensitivity to noise. The larger the value of σ, the result becomes more smoothing with less noise. However, with more smoothing of the image, the less edges will be detected by the detector.

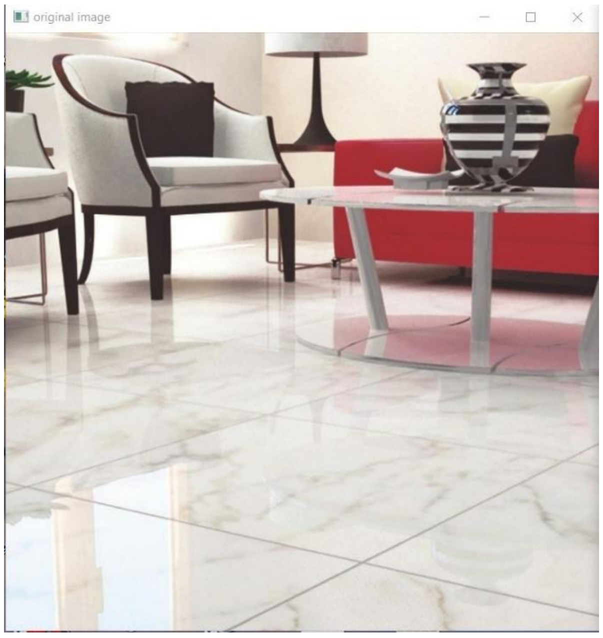


Figure 2.9: Example of applying gaussian blur (before).

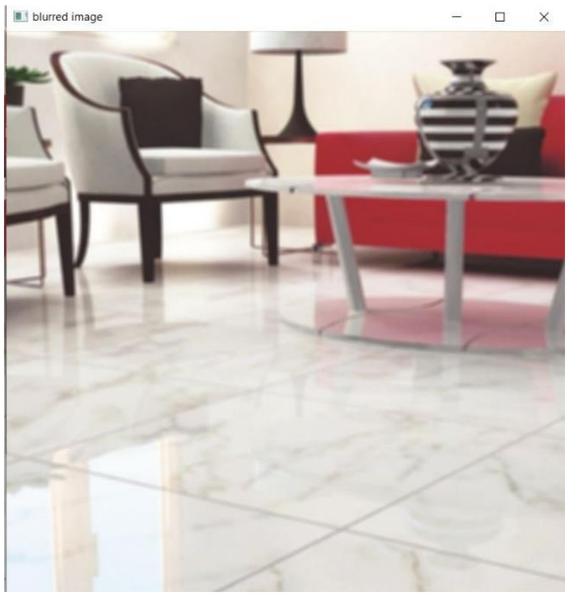


Figure 2.10: Example of applying gaussian blur (after).

Canny filter; this filter is considered as the most important one, it helps to detect the edges (edge detection filter), the two filters discussed previously are used to help this filter, in other words they are used as preprocessing to this filter, so after converting the image to grayscale and after reducing the noise and smoothing the image, the next step is to find the intensity gradient of the image by performing standard Sobel edge detection [13]. The Sobel operator uses a pair of 3x3 convolution masks, one estimating the gradient in the vertical direction (x) and the other estimating the gradient in the horizontal (y) which are shown in the equation bellow:

$$G'_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad G'_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{2.2}$$

Then, it can calculate the approximate absolute gradient magnitude (edge strength) at each pixel as shown by the equation:

$$|G'| = \sqrt{G_x'^2 + {G'_y}^2} \qquad (2.3)$$

Simultaneously, the gradient angle of each pixel can be calculated as:

$$\theta = arctan\left(\frac{G'_y}{G'_x}\right) \qquad (2.4)$$

[14] After the magnitude and gradient angles of each pixel are calculated, the result may contain thick edges which contains spurious results on the edges. The use of non-maximum suppression for each pixel in the previous result image is to sharpen the edges.

Finding the direction for each pixel, for a 3x3 image result, has 4 range of directions which is decided by the angle of each pixel:

- If the angle is 0-22.5 degrees or 157.5-180 degrees, the direction of the pixel is horizontal.
- If the angle is from 22.5 degrees to 67.5 degrees, the direction of the pixel is positive diagonal.
- If the angle is from 112.5 degrees to 157.5 degrees, the direction of the pixel is negative diagonal.
- If the angle is from 67.5 degrees to 112.5 degrees, the direction of the pixel is vertical [15].

As shown in Figure 2.11, each dark area represents an edge pixel.

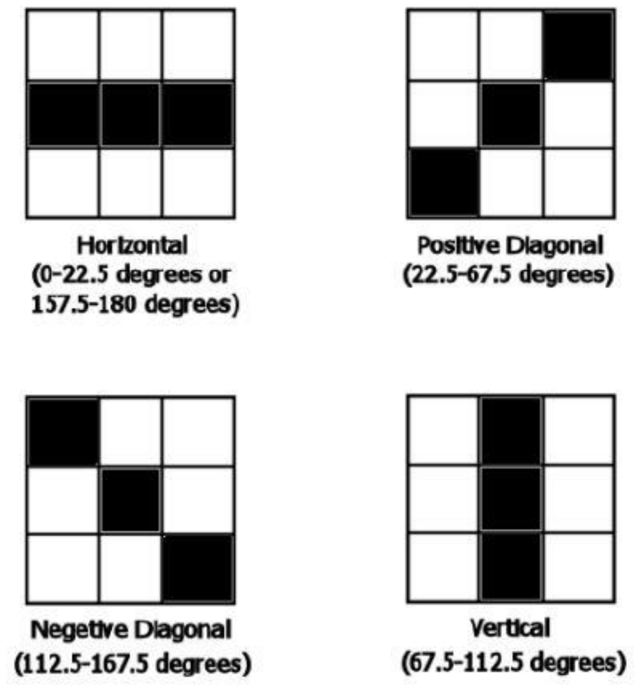


Figure 2.11: Four possible directions of the edges [15].

Once the orientation of each pixel has been calculated, the second step is to compare the magnitude value of each pixel with the two pixels next to it but in different directions. If the magnitude of the current pixel is the largest compared to the other 2 pixels this pixel will be preserved as a thin edge. Otherwise, the value will be suppressed. Figure 2.12 shows the pixels (marked in red) that need to be compared for each condition, this step is known as non-maximum suppression.[16]

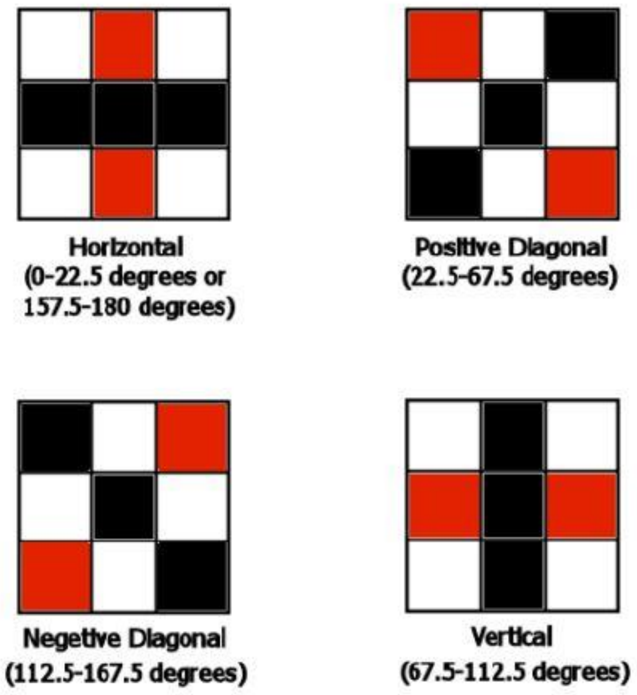


Figure 2.12: Compared pixels. [15]

After application of non-maximum suppression, the remaining edge pixels provide a more accurate representation of the real edges in an image. However, some edge pixels remain which are caused by noise or color variations. To account for these spurious values, it is essential to filter out edge pixels with a low gradient value and preserve edge pixels with a high gradient value. This is accomplished by selecting high and low threshold values. If an edge pixel's gradient value is higher than the high threshold value, it is marked as a strong edge pixel. If an edge pixel's value is smaller than the low threshold value, it will be suppressed. Canny recommends that the ratio of the high to low limit be in the range two or three to one, based on predicted signal-to-noise ratios. Pixels which are between the two thresholds are accepted if they are connected to a strong edge pixel.

So, the whole canny filter is summarized in figure 2.13:

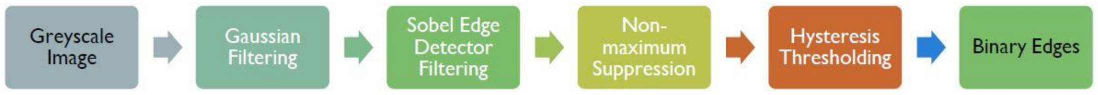


Figure 2.13: Stages of canny edge detection process.

The desired result of the full Canny Edge detection processing is shown in the example shown in figures 2.14 and 2.15:

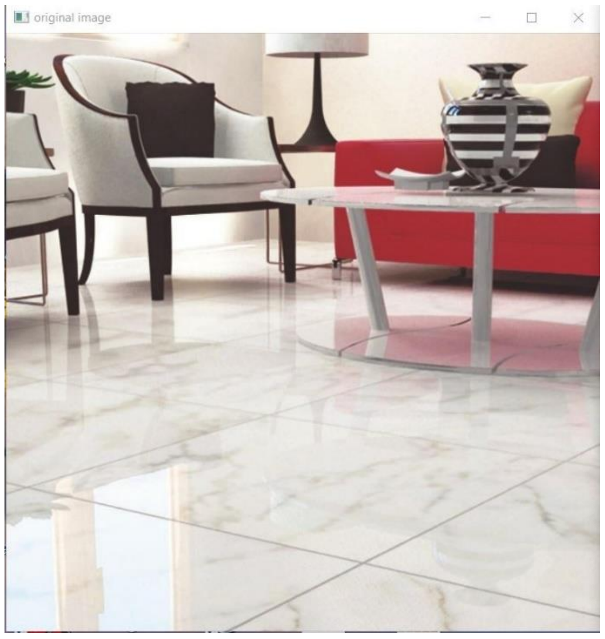


Figure 2.14: Example of applying canny filter (before).

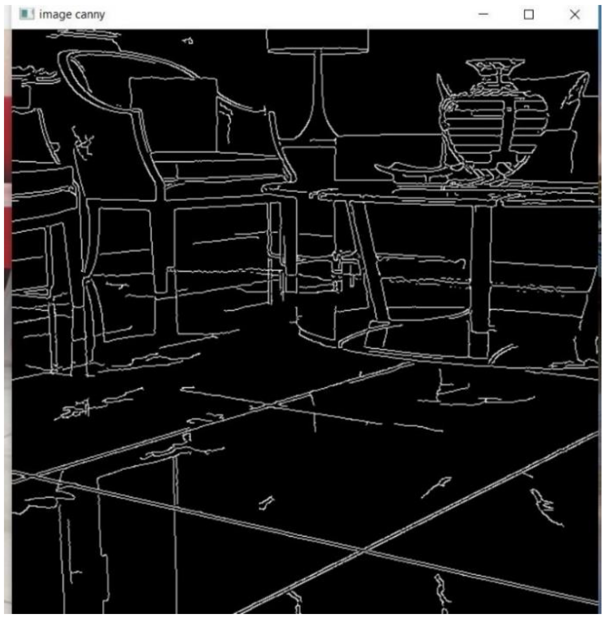


Figure 2.15: Example of applying canny filter (after).

Noise remover: this filter is used to just remove the noise from an image, the difference between this filter and the blur filter is that this filter removes the noise without blurring the image, the concept is to replace the values of the pixels that has a big difference between the neighbors' pixels values, an example is shown in figures 2.16 and 2.17:

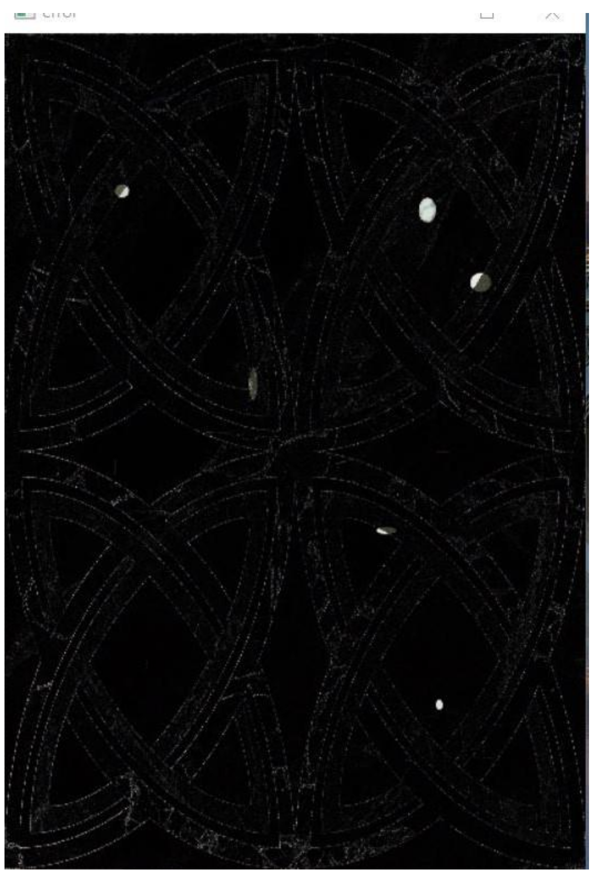


Figure 2.16: Noise removal example. (before)

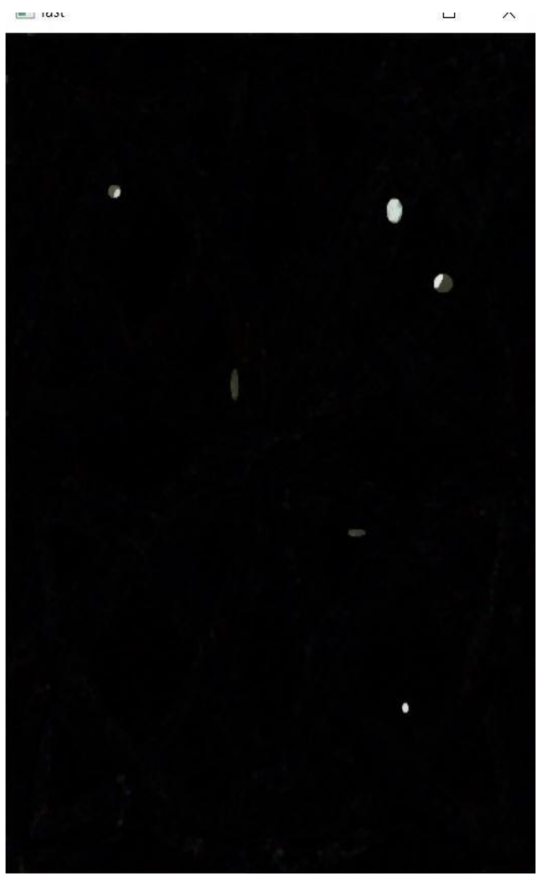


Figure 2.17: Noise removal example. (after)

2.4 OpenCV

OpenCV (Open-Source Computer Vision) is an Open-Source library C/C ++ which contains over than 500 image processing functions and computer vision. It can process real-time still images, video files or video streams from cameras [17] OpenCV was made to ease work for programmers while trying to write a code for an image processing project, where it has several filters to manipulate and that are saved inside its libraries and micro libraries, and can be invoked by just writing a one-line instruction.

2.5 Linux LXDE

LXDE, which stands for Lightweight X11 Desktop Environment, is a desktop environment which is lightweight and fast. It is designed to be user friendly and slim, while keeping the resource usage low. LXDE uses less RAM and less CPU while being a feature rich desktop environment. Unlike other tightly integrated desktops LXDE strives to be modular, so each component can be used independently with few dependencies. This makes porting LXDE to different distributions and platforms easier.[18]

2.6 Hardware component

2.6.1 Ultrasonic sensor

An ultrasonic sensor is an electronic device that measures the distance of a target object by emitting ultrasonic sound waves, and converts the reflected sound into an electrical signal. Ultrasonic waves travel faster than the speed of audible sound (i.e. the sound that humans can hear). Ultrasonic sensors have two main components: the transmitter (which emits the sound using piezoelectric crystals) and the receiver (which encounters the sound after it has travelled to and from the target).

In order to calculate the distance between the sensor and the object, the sensor measures the time it takes between the emission of the sound by the transmitter to its contact with the receiver. The formula for this calculation is:

$$D = \frac{1}{2} T \times C \qquad (2.5)$$

(where D is the distance, T is the time, and C is the speed of sound ~ 343 meters/second). [19]

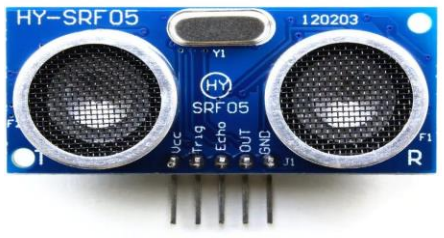


Figure 2.18: Ultrasonic sensor.

2.6.2 Infrared sensor

An infrared (IR) sensor is an electronic device that measures and detects infrared radiation in its surrounding environment. There are two types of infrared sensors: active and passive. Active infrared sensors both emit and detect infrared radiation. Active IR sensors have two parts: a light emitting diode (LED) and a receiver. When an object comes close to the sensor, the infrared light from the LED reflects off of the object and is detected by the receiver. Active IR sensors act as proximity sensors, and they are commonly used in obstacle detection systems (such as in robots). [20]

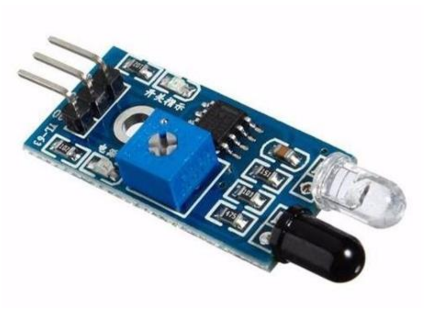


Figure 2.19: InfraRed sensor.

2.6.3 Light dependent resistor (LDR)

Photo resistors, also known as light dependent resistors (LDR), are light sensitive devices most often used to indicate the presence or absence of light, or to measure the light intensity. In the dark, their resistance is very high, sometimes up to 1MΩ, but when the LDR sensor is exposed to light, the resistance drops dramatically, even down to a few ohms, depending on the light intensity. LDRs have a sensitivity that varies with the wavelength of the light applied and are nonlinear devices. [21]

Figure 2.20: LDR sensor.

2.6.4 USB camera

USB webcam is a video camera that feeds or streams an image or video in real time to or through USB cable that is connected to a pc or in this project to the USB port of the FPGA.

Figure 2.21: USB webcam.

2.7 Conclusion

This chapter dealt with all the theoretical background needed to understand the implementation of this project, it gave all the necessary information about the DE10 board, why FPGA will be suitable in this case, and definitions needed of the used components and software tools.

CHAPTER III

Hardware and software system Design

3.1 Proposed solution

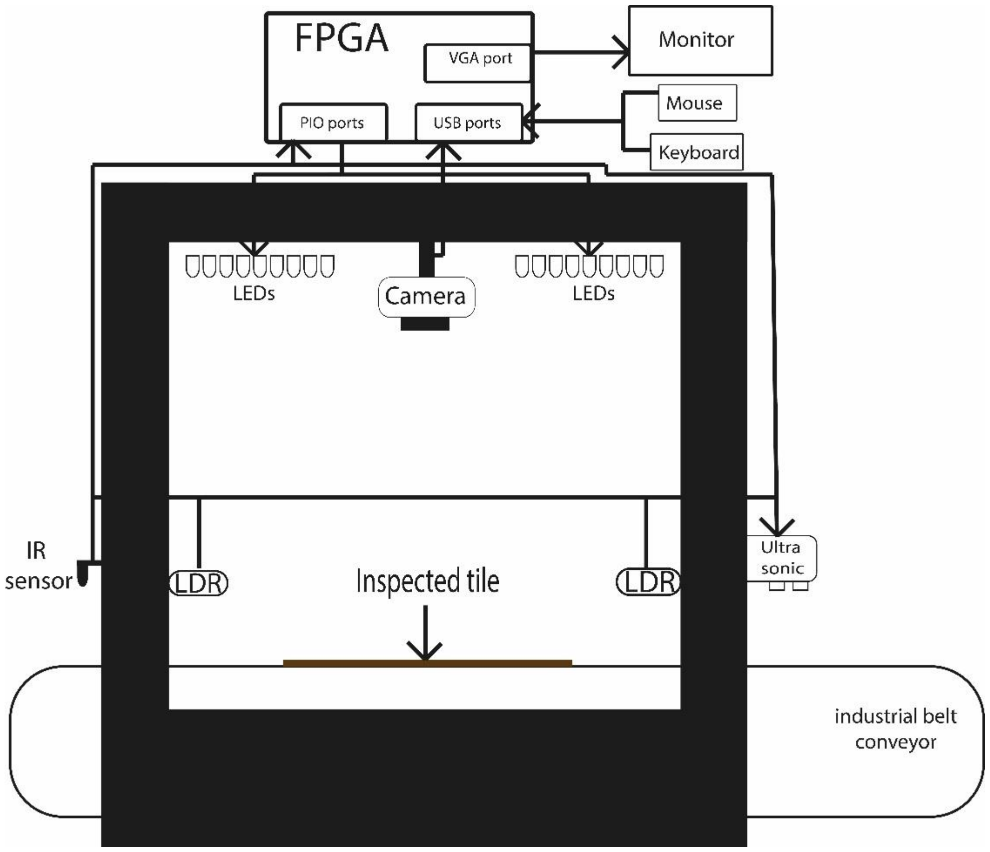


Figure 3.1: Overall system design.

The solution is to make a system that classifies tiles automatically and in a short time with high accuracy. The machine is designed to inspect ceramic tiles for mechanical, glaze and decorative defects and classify them according to their quality and size.

The inspection of the tiles takes place from different angles and allows the detection of different defects such as:

-Surface defects

- Mechanical defects (corner, edge)
- contaminations.
- Classification based on the dimension.
- Color tone.
- Curvature.

This system replaces human inspection by identifying defects in a wide range of floor tiles at high speeds 3 seconds for each tile.

The automatic final inspection of the tiles allows to increase productivity and performance while obtaining many advantages: high inspection rate, less downtime, quality benefits, uniform inspection, reduction of complaints, and continuous monitoring of production defects; this allows targeted improvements to be made to the production process to increase quality.

For the lighting of the area to be inspected, controlled LED lights. The LEDs are controlled by an LDR, which allows the machine to provide excellent illumination, improving image quality.

Everything that needs to be known, such as the quality of each tile inspected, the condition of the lot, the percentages of tiles in each quality and size category, can be viewed on the screen as the inspection proceeds. New product can be quickly configured and necessary adjustments made in real time.

The system is equipped with Ultrasonics to measure the flatness of the tile, it is also equipped with an IR sensor to detect when the tile is coming so the camera would take a photo and process the image, the figure 3.1 shows the overall system design:

3.2 Classification method

The tiles are classified as follows:

-Premium-choice product

a- Dimension: see table 1 in Chapter 1 section 1.6 (classification).

A tile that has a deviation of 0.3% in the length of a category will not be classified as premium-choice product.

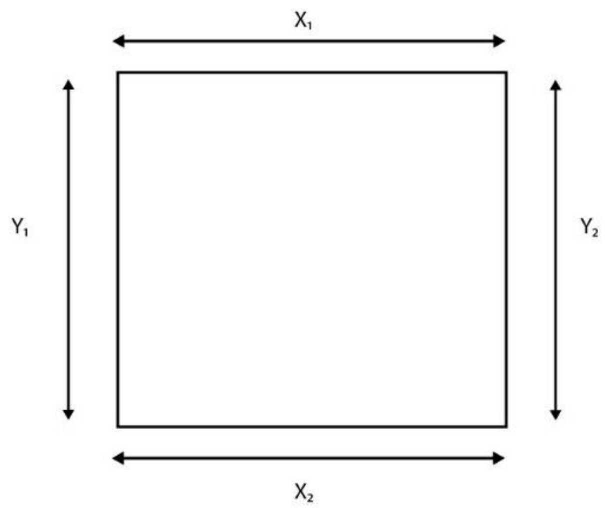


Figure 3.2: Tile dimensions.

$|X_n - X_m| < 0.3\%$ and $|X_n - Y_m| < 0.3\%$ and $|Y_n - Y_m| < 0.3\%$ (where m,n = {1,2})

b-Flatness: The system is equipped with 5 ultrasonic sensors; each ultrasonic sensor takes 5 values (total of 25 values) the values of the flatness of 25 points that are shown it the figure 3.3.

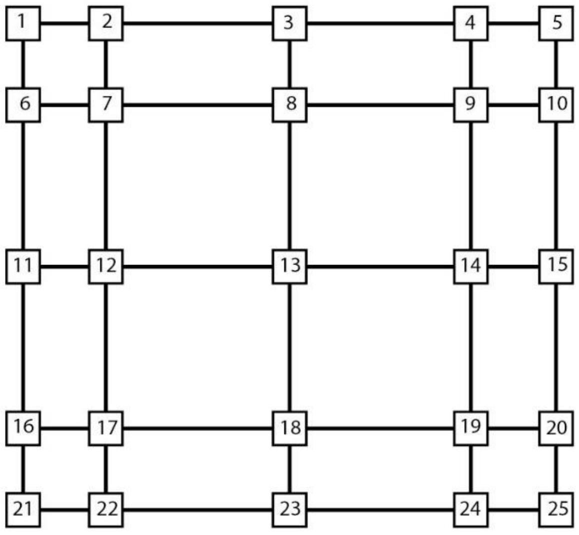


Figure 3.3: Points that the ultrasonic will take.

Based on these values, a certain calculation will be made and it will ensure that the flatness of the tile is maintained and does not exceed a certain percentage, the calculation will be based on:
- Overall flatness coefficient: the average of the 25 calculated values must be less than 3.5mm.
- Concavity of the corners: in the four points closest to the corner, the concavity of each corner is calculated and all values must be less than 1.1 mm.- Convexity in the corners: in the four points closest to the corner, the convexity of each corner is calculated and all values must be less than 1.8 mm.
- Edge concavity: at the edge points, the concavity of each edge is calculated and all must be less than 1 mm.
- Convexity of the edges: in the points of edge, the convexity of each edge will be calculated and all will have to be lower than 1 mm.

c- Health of the surface: the first-choice tile may contain some cracks, but these cracks must be less than 1cm.

d- Color intensity: the color intensity of the tile should not be significantly different from the standard color and should not exceed a certain percentage that will be visible to the naked eye.

e- Design condition: small spots can be accepted, but many of them will affect the quality, so the size and number of spots, printing deviations don't have to exceed 1cm combined.

-Commercial choice

-Defect of an area between 1cm and 5cm, or sum of the areas of defects between the mentioned range, this can come from Slight break to the sides, Slight offset of screen printing, Stains coming from the oven, A crack on the surface of the tile.

-Color tone difference less than 0.5% can be caused by Excess or lack of granilia load on the tile, difference in the painting supplied to the printer.

-A curved tile (based on the criteria mentioned in the previous part).

-(See figure 3.2) $|X_n – X_m| > 0.3\%$ or $|X_n – Y_m| > 0.3\%$ or $|Y_n – Y_m| > 0.3\%$ (where $m,n = \{1,2\}$)

-3rd choice

-Big visual defect that ranges from 5cm to 50% of the surface of the tile can be classified as a non-sellable tile that can be rectified where they cut the damaged part and keep the good part and sell it as a small sized tile.

-In the case where the defect damage more than 50% of the surface, the tile is considered as non-usable and directly thrown.

3.3 Software part

The software part is divided into three parts, the first part deals with the tile's classification, while the second part shows how to add a new tile model as a reference which will be used when comparing the tiles in the first part, the last part talks about light control and how to provide the perfect lighting environment.

The C++ and VHDL codes will be discussed and demonstrated by flowcharts, thanks to OpenCV libraries the filters discussed on chapter one part 3.2.4 are already designed and ready to be used:

3.3.1. Classification system

This part will discuss the classification process. The IR sensor detects the tile when entering the machine. The system waits until the tile is in the center of the machine and the camera then takes an image and sends it to the FPGA. This captured image will be processed to detect the defects on the surface of the tile to check its health by comparing it to the reference image that is already saved on the system. It will also read the dimensions of the tile, then the ultrasonic will be responsible of checking the flatness of the tile as shown in the flowchart in Figure 3.4:

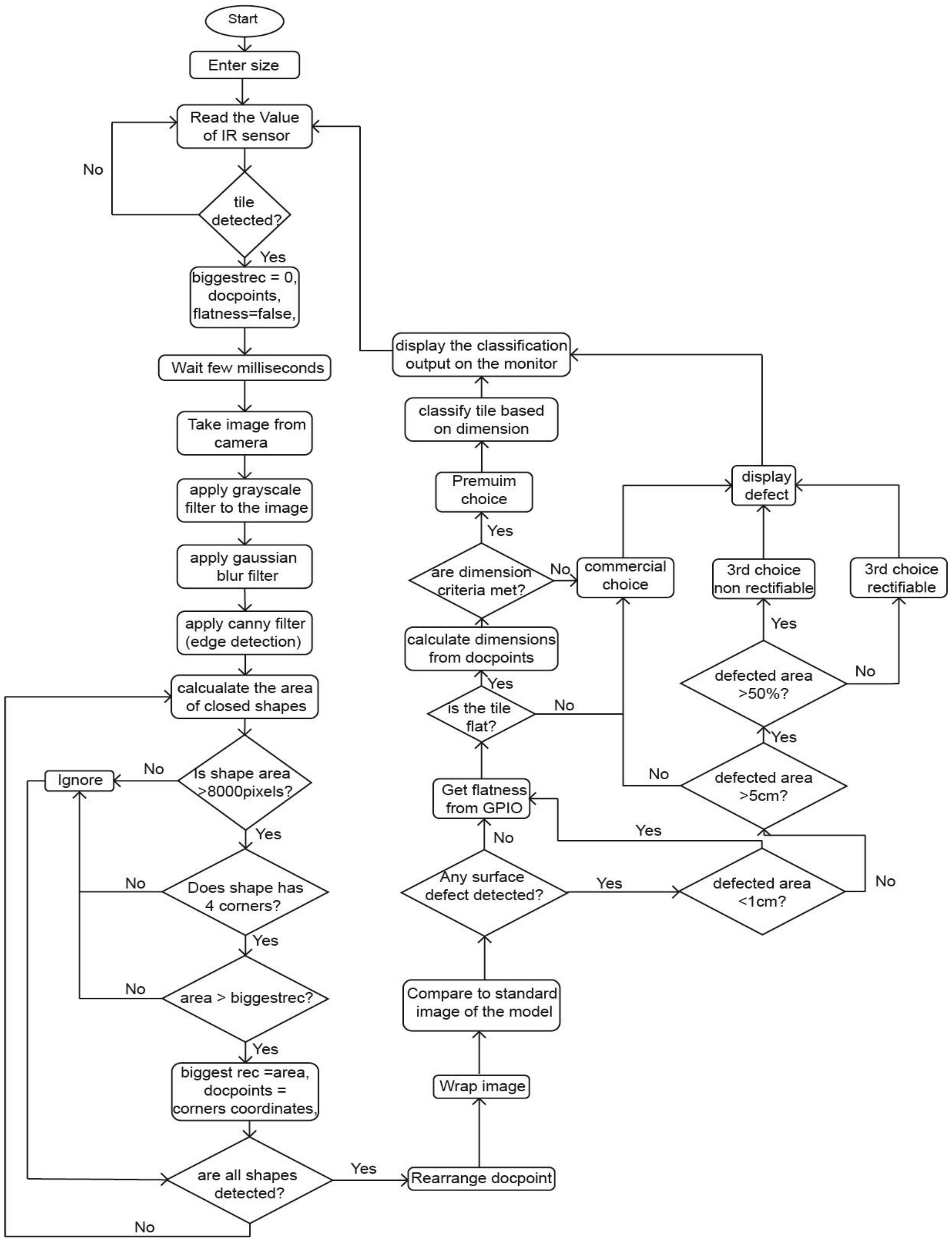

Start
Enter size
Read the Value of IR sensor
tile detected?
No
Yes
biggestrec = 0, docpoints, flatness=false,
Wait few milliseconds
Take image from camera
apply grayscale filter to the image
apply gaussian blur filter
apply canny filter (edge detection)
calcualate the area of closed shapes
Is shape area >8000pixels?
Ignore
Does shape has 4 corners?
area > biggestrec?
biggest rec =area, docpoints = corners coordinates,
are all shapes detected?
Rearrange docpoint
Wrap image
Compare to standard image of the model
Any surface defect detected?
Get flatness from GPIO
Is the tile flat?
calculate dimensions from docpoints
are dimension criteria met?
Premuim choice
classify tile based on dimension
display the classification output on the monitor
commercial choice
defected area <1cm?
defected area >5cm?
defected area >50%?
3rd choice non rectifiable
3rd choice rectifiable
display defect

Figure 3.4: Classification system flowchart.

When starting the program, it will tell the human operator to enter the model and the size of the tile that it is going to be classified. After entering the values and the name of the model, the IR sensor will check whether the tile has arrived or not by measuring the distance. When a tile is detected, the camera will wait until the tile is in the center of the machine then it will take a picture of it. The image then will be processed by applying first the grayscale filter and the gaussian blur filter. This will help to apply the canny filter to detect the edges. Once the edges are detected, it will detect all the closed shapes. The system will neglect the small shapes to gain time and it will only get the shapes that has 4 corners (rectangles) and get the biggest one out of all of these shapes. This will be our tile.

After detecting the tile, its image will be wrapped after rearranging the points (corners) to be compared to the standard image that is already saved in the system. By comparing the two images the area of the defects, the dimensions of the tile and the flatness rate (this value will be set by the Ultrasonic sensors) will be calculated. The system will classify the tiles based on the criteria mentioned in section 3.2 (classification methods) and display the output with the defects detected from the tile on the monitor.

3.3.2. Adding new tile model

This part is made to add a new tile model (new product) as a reference to be compared with. It works with the same method as the previous part, where the IR sensor will detect the tile and take an image wrap the tile from the image using the same filters discussed above and save it by the name defined by the operator.

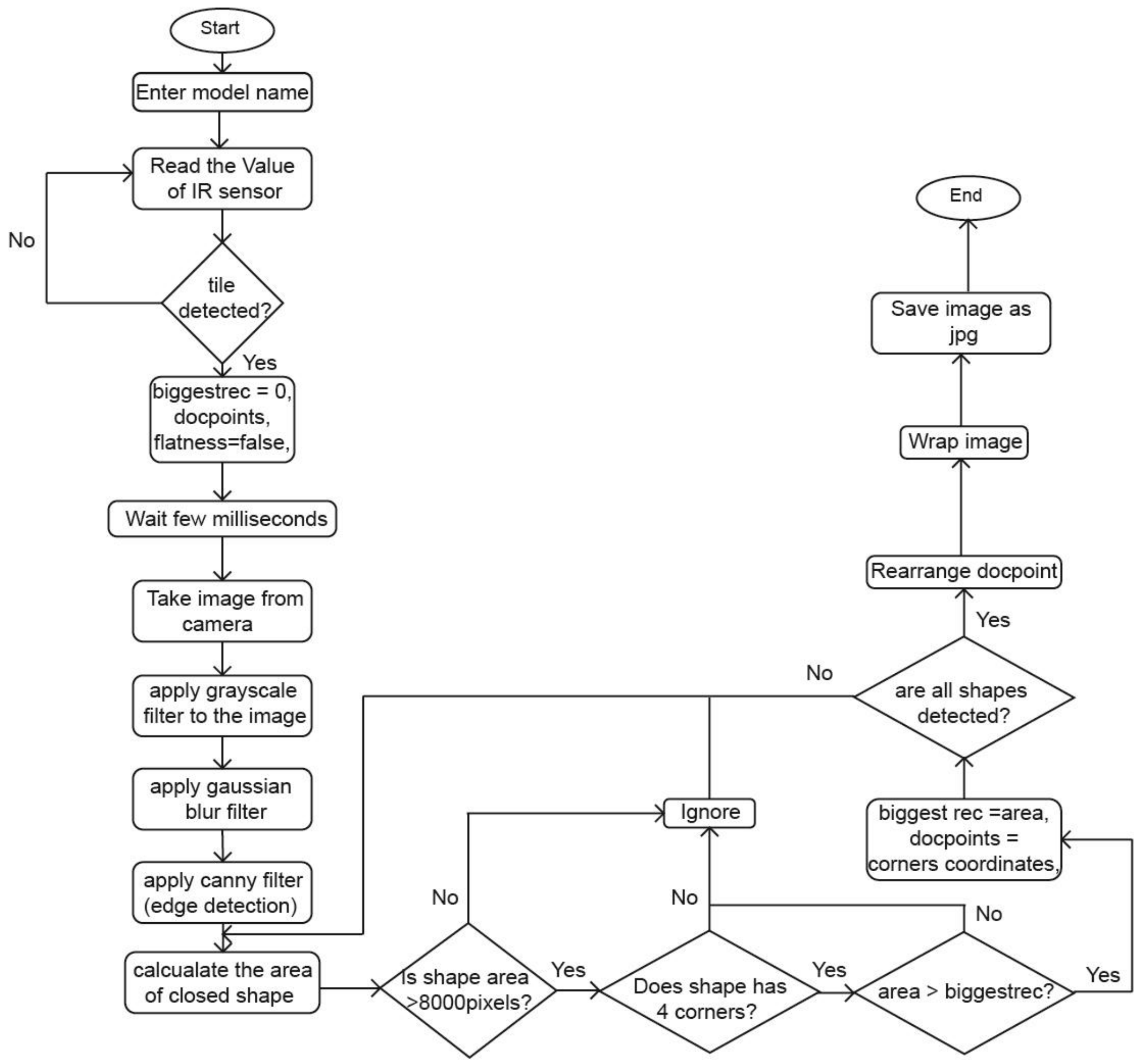

Start
Enter model name
Read the Value of IR sensor
tile detected?
No
Yes
biggestrec = 0, docpoints, flatness=false,
Wait few milliseconds
Take image from camera
apply grayscale filter to the image
apply gaussian blur filter
apply canny filter (edge detection)
calcualate the area of closed shape
Is shape area >8000pixels?
No
Yes
Ignore
Does shape has 4 corners?
No
Yes
area > biggestrec?
No
Yes
biggest rec =area, docpoints = corners coordinates,
are all shapes detected?
No
Yes
Rearrange docpoint
Wrap image
Save image as jpg
End

Figure 3.5: Adding new model flowchart.

3.3.3. Light control

The camera needs a perfect lighting environment, the LDR sensor read the light value, if it is less or more it will send a signal to the FPGA to increase or decrease the voltage supplied to the LEDs to regulate the light.

The perfect lighting environment for a camera to take an image is 3,000 lumens, [22] this value is converted to ohms based on the resistances used, a range of 200 lumens is given to stabilize the system.

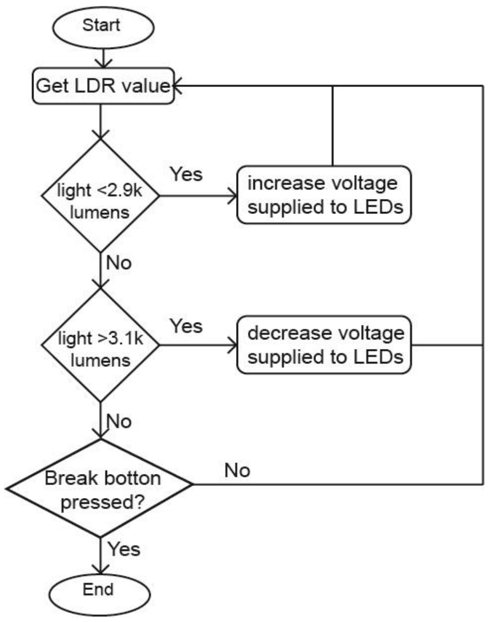


Figure 3.6: Light control system flowchart.

3.4 Hardware

The system consists of two hardware pars. The first one is the flatness checker which consists of five ultrasonic sensors, and each ultrasonic sensor calculates five values. Each ultrasonic has four pins (Vcc, Gnd, Echo and Trig) the Vcc of each ultrasonic is connected to the 5V Vcc of the FPGA, and the ground is also connected to the ground of the FPGA. The five echo pins of the five sensors are connected to the GPIO pins of the FPGA and are declared as outputs to send the sound signal, and the five trig pins are connected to other five GPIO pins and declared as inputs to read the signal coming from the obstacle (in our case the tile).

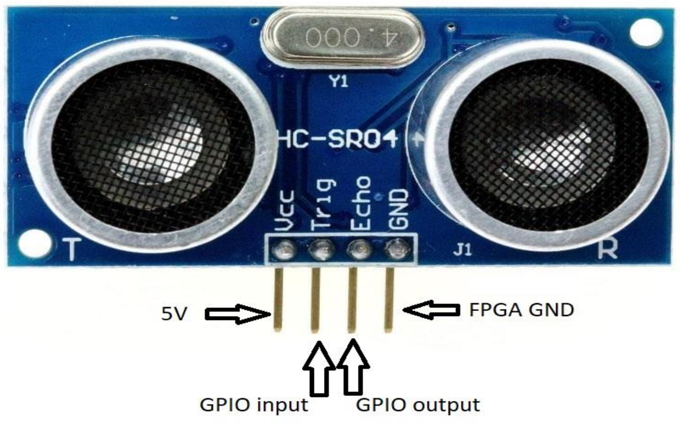


Figure 3.7: Ultrasonic sensor connections.

The second circuit, which is the light control one, the circuit, is shown in the figure 3.7:

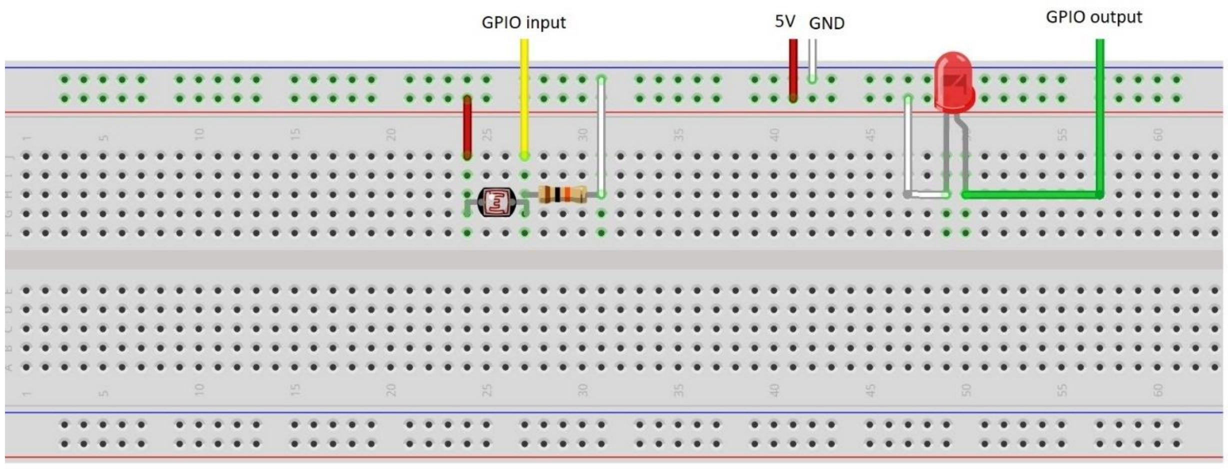
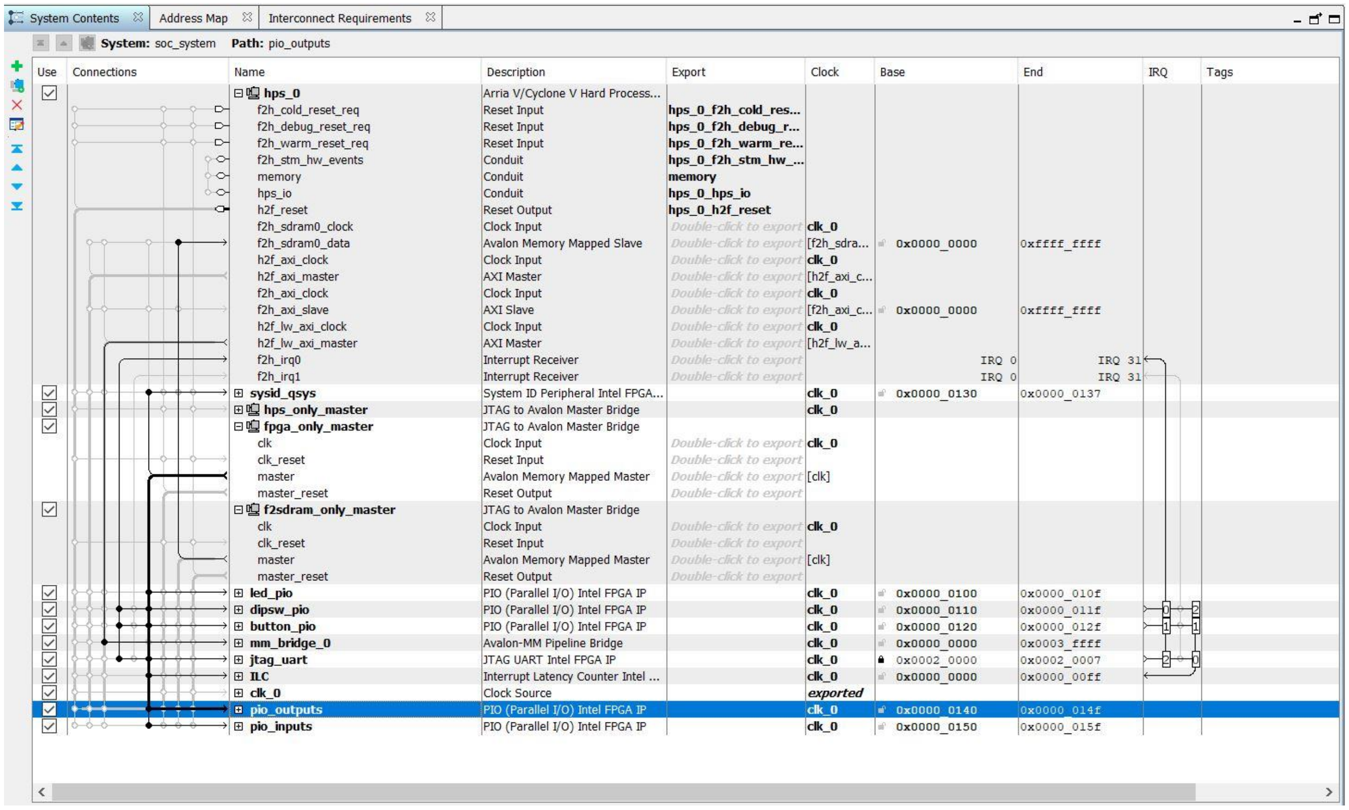


Figure 3.8: light control circuit

The value of read by the LDR sensor will be transmitted to the FPGA, and from this value the FPGA can control the LEDs by supplying the right voltage to the LEDs.

3.5 QSYS design

PIO ports are connected to the FPGA, and we are working with HPS, so we have first to enable the exchange of data between FPGA and HPS by enabling the lightweight-AXI-FPGA-to-HPS-bridge and declare the inputs and outputs of the system as we can see on figure 3.7:


System Contents | Address Map | Interconnect Requirements

System: soc_system **Path:** pio_outputs

Use	Connections	Name	Description	Export	Clock	Base	End	IRQ	Tags
✓		⊟ **hps_0**	Arria V/Cyclone V Hard Process...						
		f2h_cold_reset_req	Reset Input	**hps_0_f2h_cold_res...**					
		f2h_debug_reset_req	Reset Input	**hps_0_f2h_debug_r...**					
		f2h_warm_reset_req	Reset Input	**hps_0_f2h_warm_re...**					
		f2h_stm_hw_events	Conduit	**hps_0_f2h_stm_hw_...**					
		memory	Conduit	**memory**					
		hps_io	Conduit	**hps_0_hps_io**					
		h2f_reset	Reset Output	**hps_0_h2f_reset**					
		f2h_sdram0_clock	Clock Input	Double-click to export	**clk_0**				
		f2h_sdram0_data	Avalon Memory Mapped Slave	Double-click to export	[f2h_sdra...	0x0000_0000	0xffff_ffff		
		h2f_axi_clock	Clock Input	Double-click to export	**clk_0**				
		h2f_axi_master	AXI Master	Double-click to export	[h2f_axi_c...				
		f2h_axi_clock	Clock Input	Double-click to export	**clk_0**				
		f2h_axi_slave	AXI Slave	Double-click to export	[f2h_axi_c...	0x0000_0000	0xffff_ffff		
		h2f_lw_axi_clock	Clock Input	Double-click to export	**clk_0**				
		h2f_lw_axi_master	AXI Master	Double-click to export	[h2f_lw_a...				
		f2h_irq0	Interrupt Receiver	Double-click to export		IRQ 0	IRQ 31		
		f2h_irq1	Interrupt Receiver	Double-click to export		IRQ 0	IRQ 31		
✓		⊞ **sysid_qsys**	System ID Peripheral Intel FPGA...		**clk_0**	0x0000_0130	0x0000_0137		
✓		⊞ **hps_only_master**	JTAG to Avalon Master Bridge		**clk_0**				
✓		⊟ **fpga_only_master**	JTAG to Avalon Master Bridge						
		clk	Clock Input	Double-click to export	**clk_0**				
		clk_reset	Reset Input	Double-click to export					
		master	Avalon Memory Mapped Master	Double-click to export	[clk]				
		master_reset	Reset Output	Double-click to export					
✓		⊟ **f2sdram_only_master**	JTAG to Avalon Master Bridge						
		clk	Clock Input	Double-click to export	**clk_0**				
		clk_reset	Reset Input	Double-click to export					
		master	Avalon Memory Mapped Master	Double-click to export	[clk]				
		master_reset	Reset Output	Double-click to export					
✓		⊞ **led_pio**	PIO (Parallel I/O) Intel FPGA IP		**clk_0**	0x0000_0100	0x0000_010f		
✓		⊞ **dipsw_pio**	PIO (Parallel I/O) Intel FPGA IP		**clk_0**	0x0000_0110	0x0000_011f	0 2	
✓		⊞ **button_pio**	PIO (Parallel I/O) Intel FPGA IP		**clk_0**	0x0000_0120	0x0000_012f	1 1	
✓		⊞ **mm_bridge_0**	Avalon-MM Pipeline Bridge		**clk_0**	0x0000_0000	0x0003_ffff		
✓		⊞ **jtag_uart**	JTAG UART Intel FPGA IP		**clk_0**	0x0002_0000	0x0002_0007	2 0	
✓		⊞ **ILC**	Interrupt Latency Counter Intel ...		**clk_0**	0x0000_0000	0x0000_00ff		
✓		⊞ **clk_0**	Clock Source		***exported***				
✓		⊞ **pio_outputs**	PIO (Parallel I/O) Intel FPGA IP		**clk_0**	0x0000_0140	0x0000_014f		
✓		⊞ **pio_inputs**	PIO (Parallel I/O) Intel FPGA IP		**clk_0**	0x0000_0150	0x0000_015f		

Figure 3.9: QSYS design of the system.

To control the PIO from the HPS we first have to get the virtual memory address since the LXDE maps the actually physical address to a new virtual memory address, and to do so, embedded shell application has been used.

After getting the memory addresses of the PIOs we can use them as pointers to send and receive data.

CHAPTER IV

Implementation and results

4.1 Introduction

This chapter presents implementation part of the project. The first section is about how to set up lightweight Linux on the FPGA. The second part is about how to use and how to program on Linux LXDE, the third one will mainly talk about the C++ code, the filters used, and the flowcharts. Finally, the last section will focus on the hardware part of the system.

4.2 Setting up Linux on FPGA

The procedure to boot Linux on DE10-Standard Board are:

-Download the BSP image file from http://de10-standard.terasic.com/cd.

-Create a Linux booting microSD card by using Win32 Disk Imager utility to write the image file into a microSD card.

-Insert the microSD-to-microSD socket on the DE10-Standard.

-Make sure MSEL [4:0] switch on DE10-Standard is set to proper position.

-Connect a VGA monitor, an USB keyboard, and a USB mouse to the DE10-Standard.

-Power on DE10-Standard.

-The LXDE Desktop will appear on the VGA monitor.

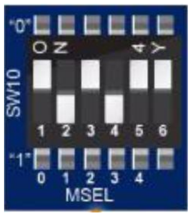


Figure 4.1: MSEL configuration mode switch.

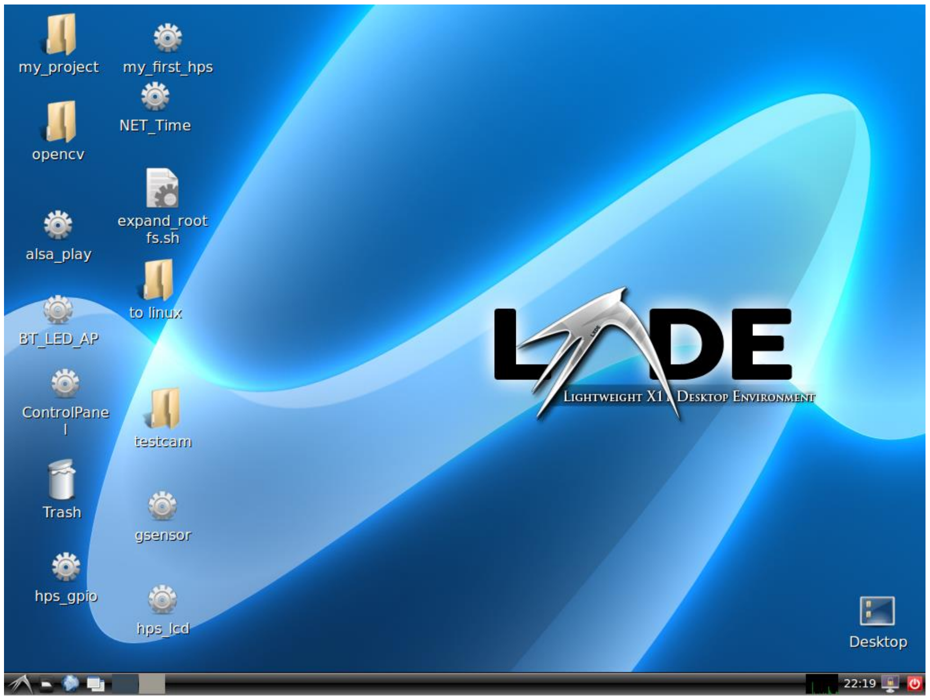


Figure 4.2: Booted Linux on desktop.

4.3 Linux LXDE usage

4.3.1 Importing file to LXDE desktop

To copy paste files from personal laptop or any other peripheral, there exist two ways to do so:

- Upgrading Linux:

  Connect the FPGA to a source of internet using the ethernet cable, then execute the following instruction on the terminal: “sude apt-get upgrade”, wait until the installation is finished, then all of the files on the SD card will appear.

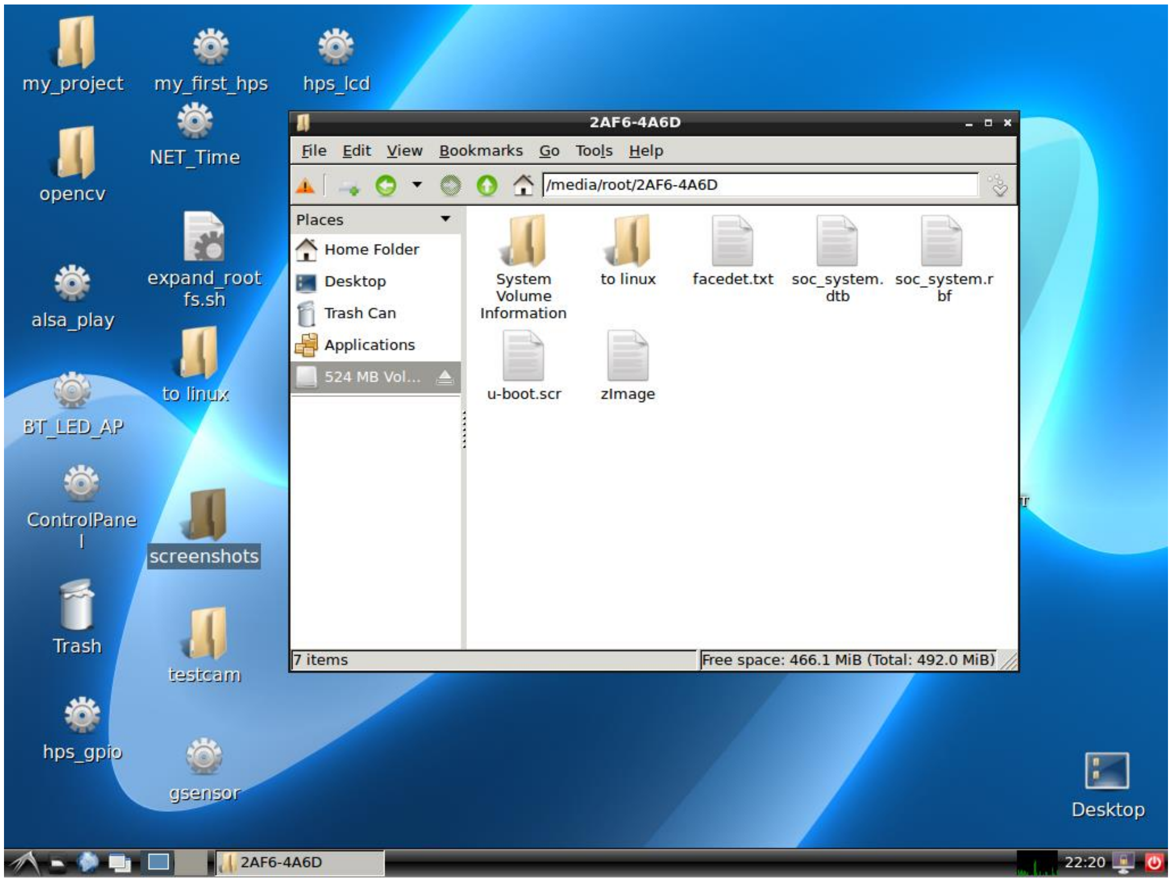
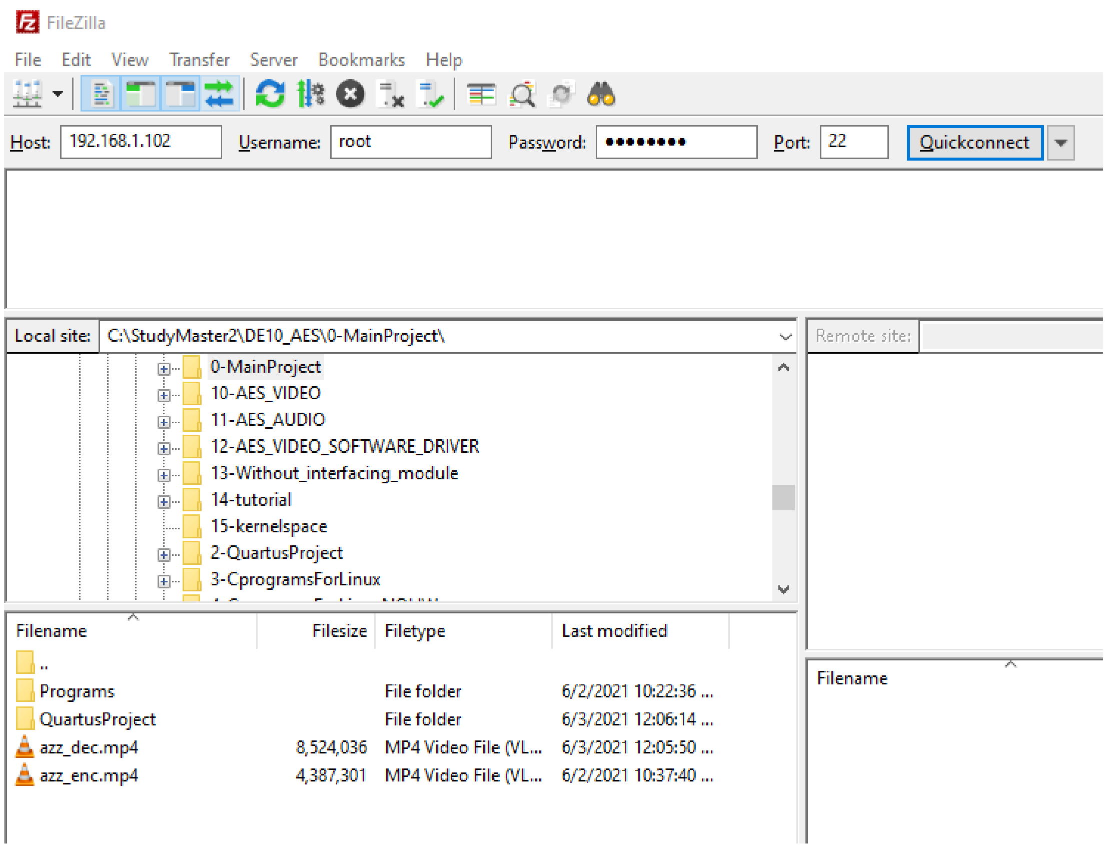


Figure 4.3: SD-card content.

- Copying file from laptop via ethernet cable:

  This is done by connecting an ethernet cable to both FPGA and personal laptop, change IP address of the DE10 standard board on your laptop to 192.168.1.200 mask 255.255.255.0, and use the command "sudo ifconfig eth0 192.168.1.201 netmask 255.255.255.0" on the Linux terminal on the FPGA. fileZila is then used to connect to host 192.168.1.102, username to root, password to password, and port to 22.


FileZilla

File Edit View Transfer Server Bookmarks Help

Host: 192.168.1.102 Username: root Password: ●●●●●●●●● Port: 22 Quickconnect

Local site: C:\StudyMaster2\DE10_AES\0-MainProject\

Remote site:

```
0-MainProject
10-AES_VIDEO
11-AES_AUDIO
12-AES_VIDEO_SOFTWARE_DRIVER
13-Without_interfacing_module
14-tutorial
15-kernelspace
2-QuartusProject
3-CprogramsForLinux
```

Filename	Filesize	Filetype	Last modified
..			
Programs		File folder	6/2/2021 10:22:36 ...
QuartusProject		File folder	6/3/2021 12:06:14 ...
azz_dec.mp4	8,524,036	MP4 Video File (VL...	6/3/2021 12:05:50 ...
azz_enc.mp4	4,387,301	MP4 Video File (VL...	6/2/2021 10:37:40 ...

Filename

Figure 4.4: Copying file to FPGA via ethernet cable.

4.3.2 Coding in LXDE

The LXDE does not offer any sophisticated development tool, and the code has to be written in a non-user-friendly generic text editor. However, the LXDE BPS offers the required toolchain GCC for the direct development, compilation, execution and debugging of OpenCV applications, without the need for cross-compile.

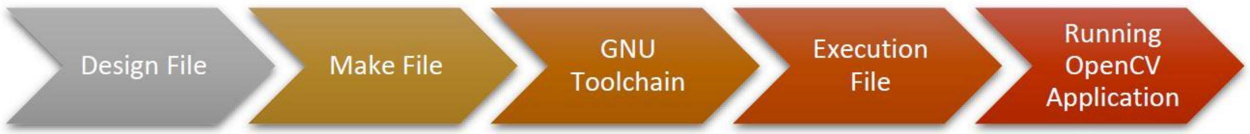
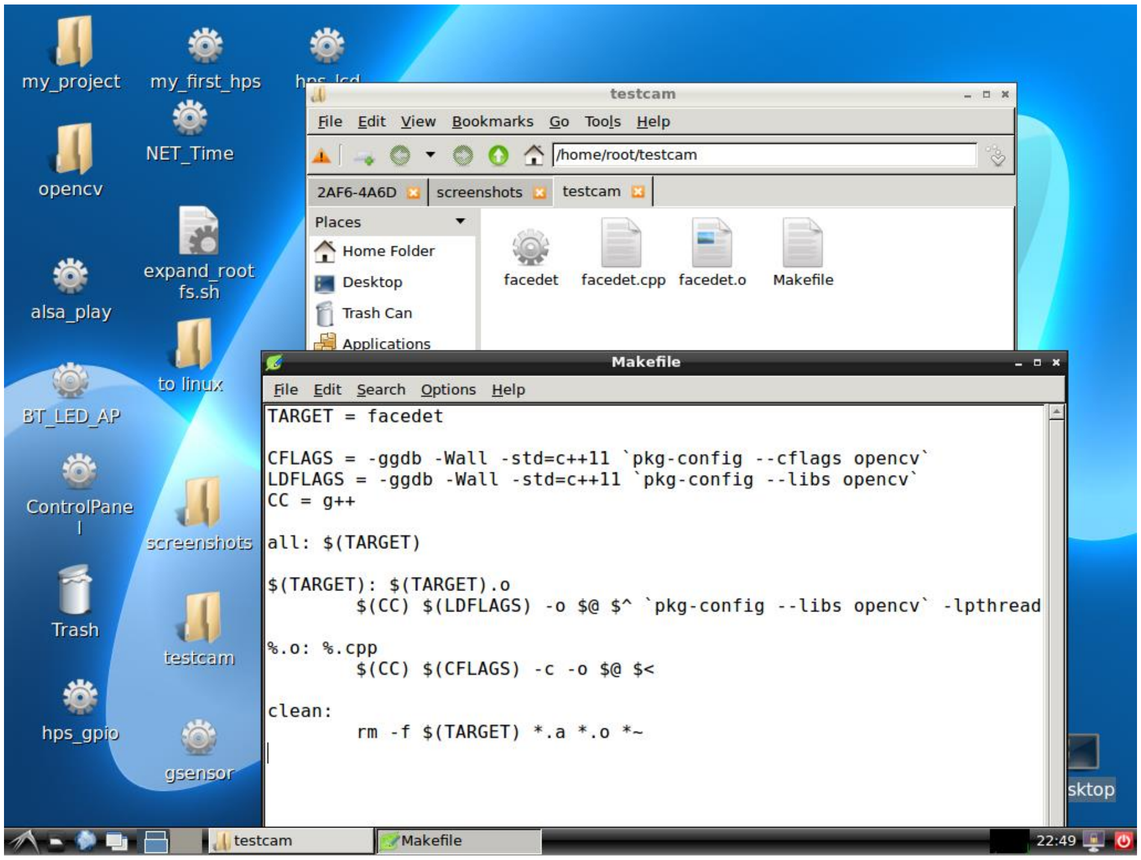


Figure 4.5: Design flow in LXDE.[23]



```
TARGET = facedet

CFLAGS = -ggdb -Wall -std=c++11 `pkg-config --cflags opencv`
LDFLAGS = -ggdb -Wall -std=c++11 `pkg-config --libs opencv`
CC = g++

all: $(TARGET)

$(TARGET): $(TARGET).o
        $(CC) $(LDFLAGS) -o $@ $^ `pkg-config --libs opencv` -lpthread

%.o: %.cpp
        $(CC) $(CFLAGS) -c -o $@ $<

clean:
        rm -f $(TARGET) *.a *.o *~
```

Figure 4.6: Executable program and Makefile example.

4.4 Setting up the camera

Generally, normal USB camera work directly on light Linux without installing any driver. In our case, the intel series camera driver is installed in “dev/Video0”. To check the camera, it first has to be tested using ‘cheese’ software under linux. In the first test, the camera did not display any results, since the HPS is quite weak and the RAM is only 1GB it is difficult for the FPGA to get the camera working of a resolution of 640 x 480 or more, the resolution has to be decreased to 320 x 240. So while coding we have to set the resolution of the camera to 320 x 240 using OpenCV instructions:

Camera_name.set(CAP_PROP_FRAME_WIDTH,320);

Camera_name.set(CAP_PROP_FRAME_HEIGHT,240);

The camera needs also some time to stabilize by giving it few milliseconds before taking a picture as shown in the flowcharts in the figures 3.4 and 3.5.

4.5 Steps of the comparison method

The first step will be taking the image from the camera. The image that will be entered (in the worst-case scenario) will look like the image demonstrated in figure 4.7. It will have some defect on the surface and will not have the shape of a perfect rectangle.

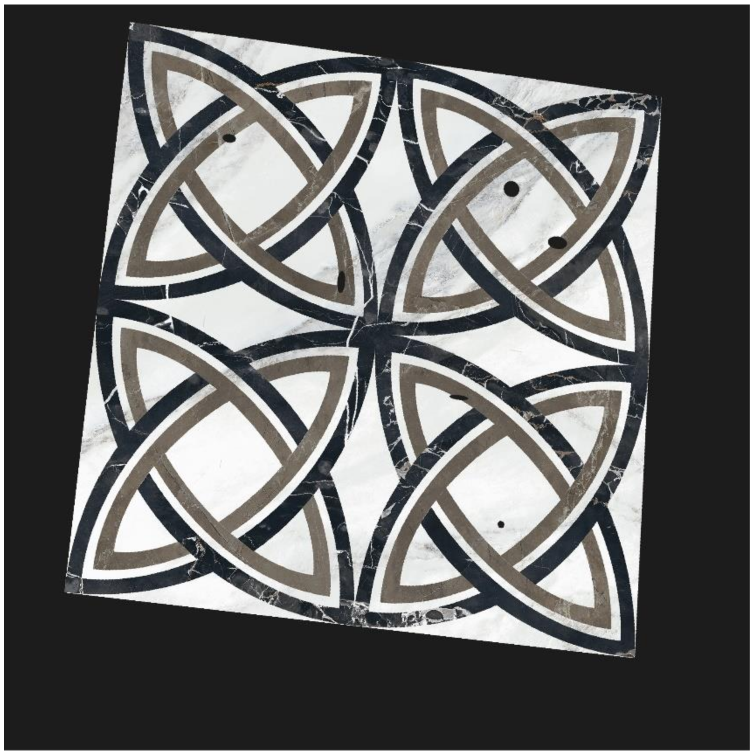


Figure 4.7: Example of image taken from the camera.

Grayscale filter will be applied to this image and the output image will be as in figure 4.8:

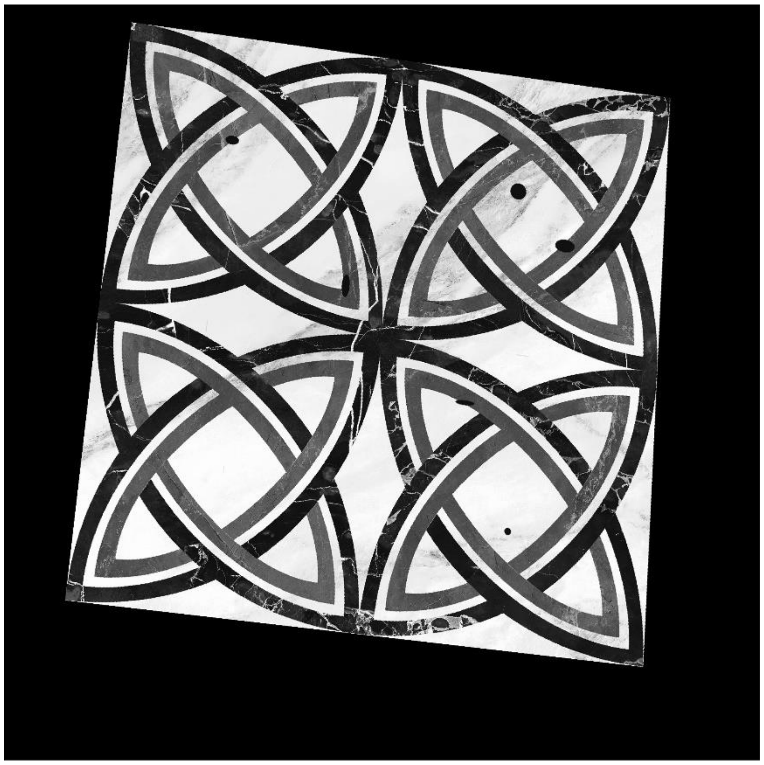
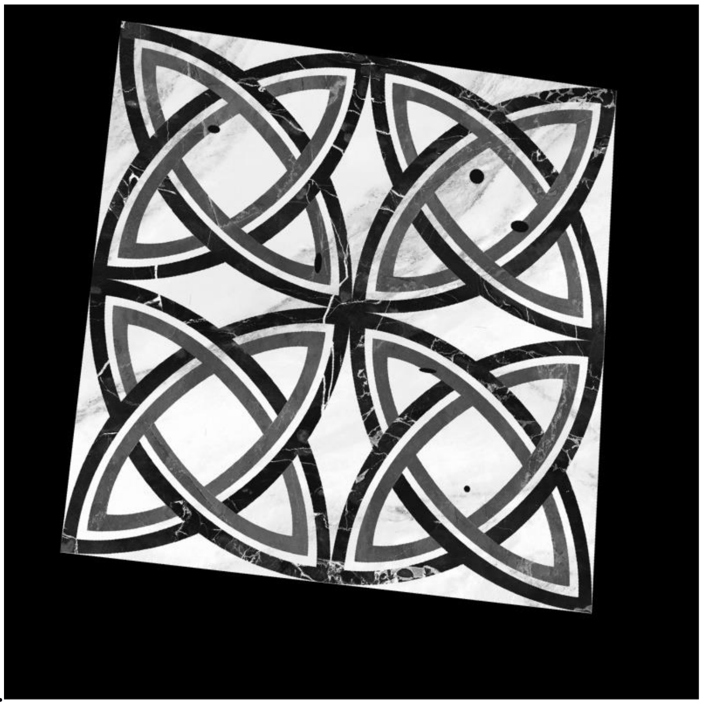


Figure 4.8: Applying grayscale filter to the image

The blur filter will be applied in the next step to reduce the noise:



.

Figure 4.9: Applying gaussian blur to the image

The canny filter (edge detection) will be applied in the next step:

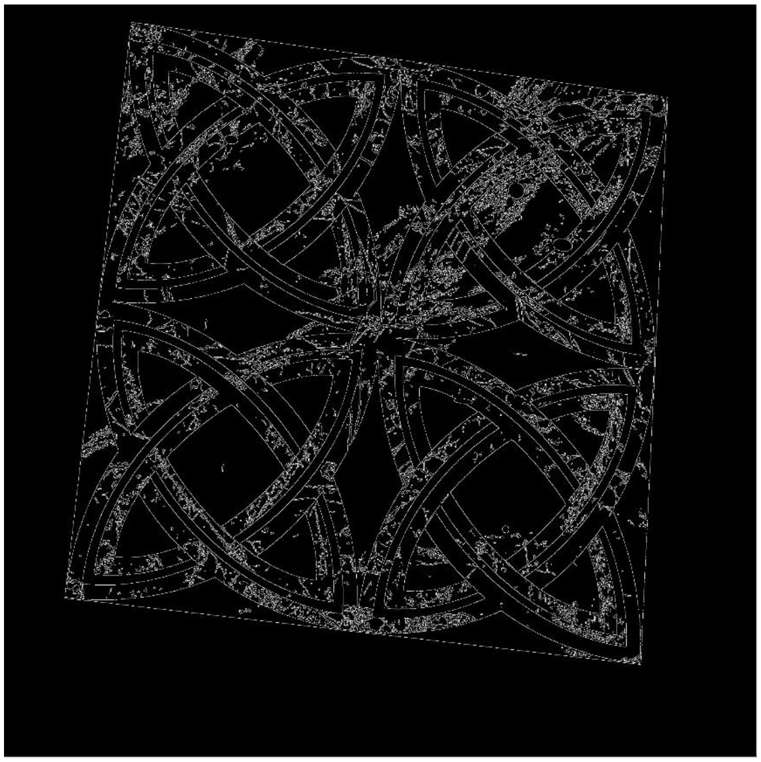


Figure 4.10: Applying canny filter to the image

The edges in this image are very thin, so we have to dilate them:

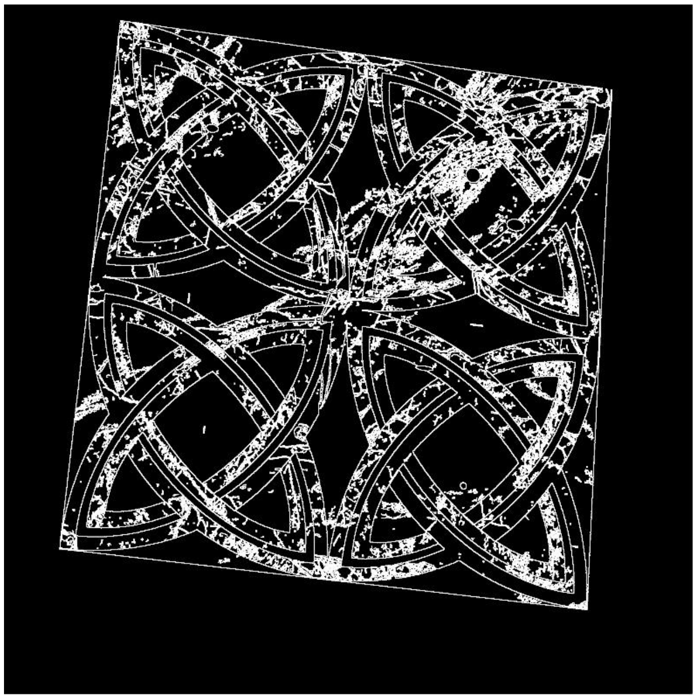


Figure 4.11: Dilating the edges

Next step is to get the biggest rectangle. In this part, the biggest rectangle is obviously the tile here. The software gets the coordinates of its corners:

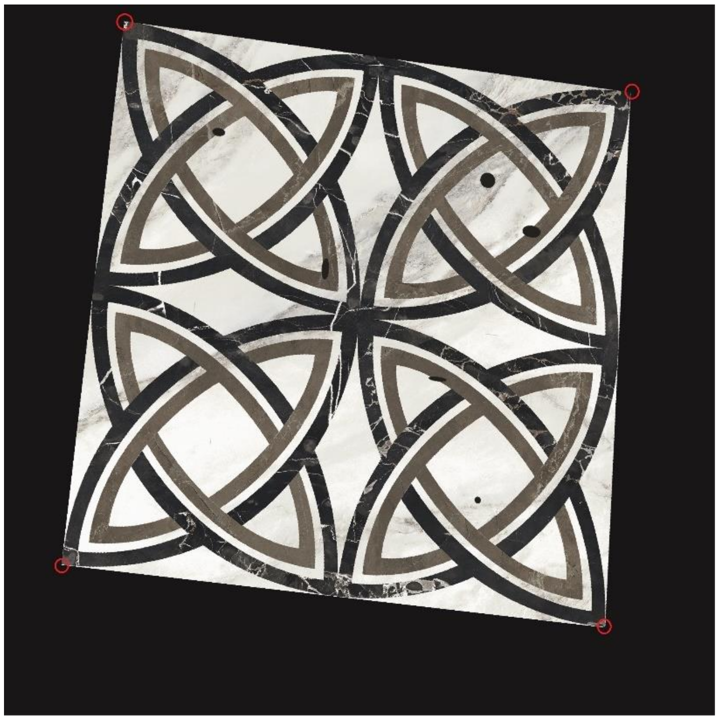


Figure 4.12: Drawing circles on the corners of the tile

Then the corners are rearranged (top left, then top right, bottom left, then bottom right) and the image is wrapped based on the coordinates of the new points:

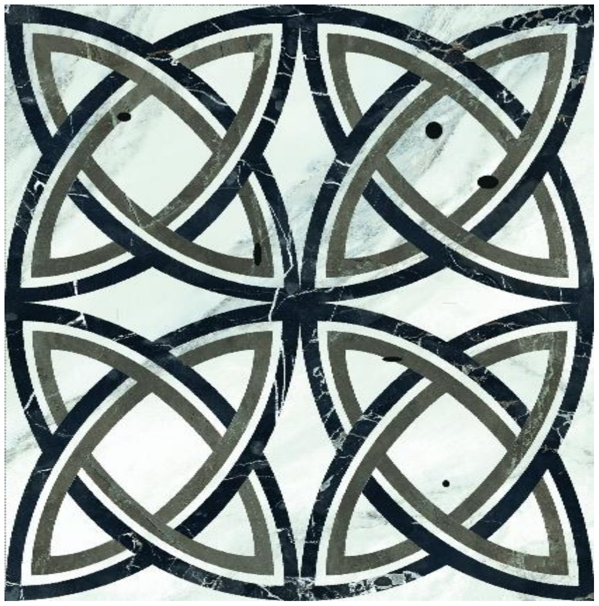


Figure 4.13: Wrapped tile image.

This image is compared to the original one. After giving the value of 0.5% in the difference allowed, the values of each pixel will be reduced from the value corresponding pixel (same coordinates) in the standard image that is already saved in the system.

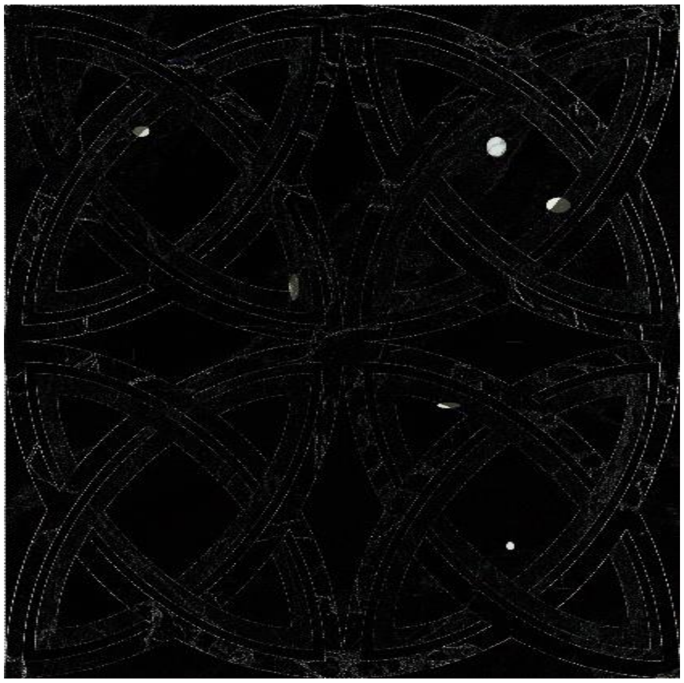


Figure 4.14: Resulting image from reducing the taken image from the standard one.

We can easily notice some white lines that appears on the result, this is resulting from the difference in the dimensions and can be neglected by applying a filter that remove these white lines.

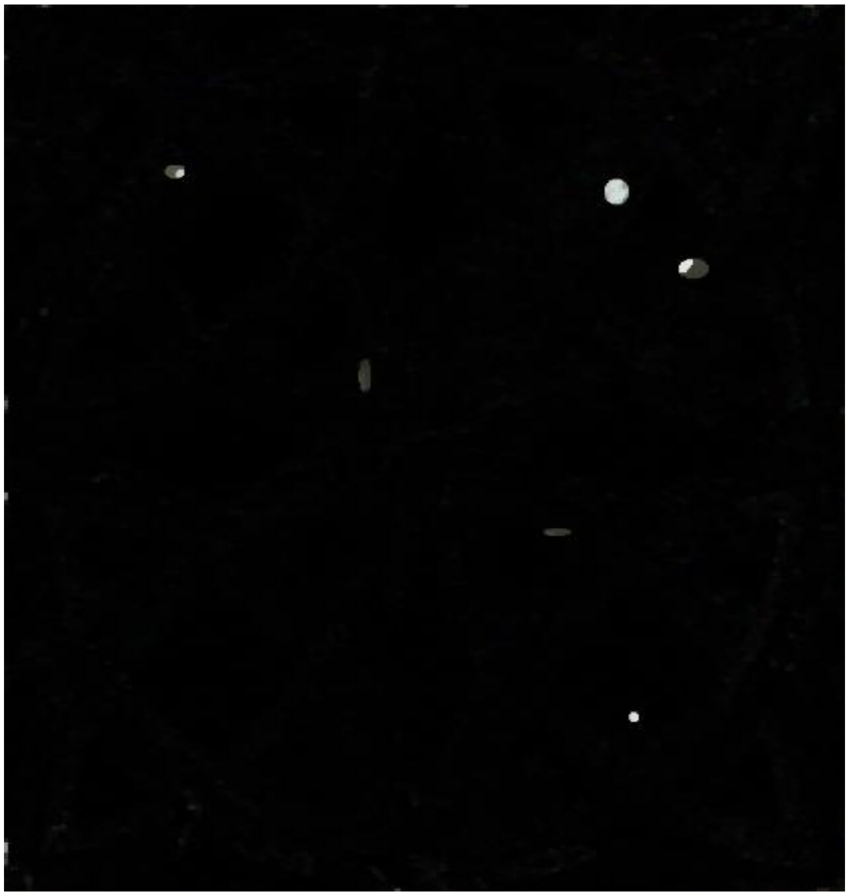


Figure 4.15: Defect detected on the image.

Then the system will apply the exact same filters mentioned before; grayscale filter, gaussian blur filter, and the canny filter to get the edges of the defects. It will get the coordinates of these defects and draw these edges with a pink shiny color in the already wrapped image of the tile to show the places of the infection.

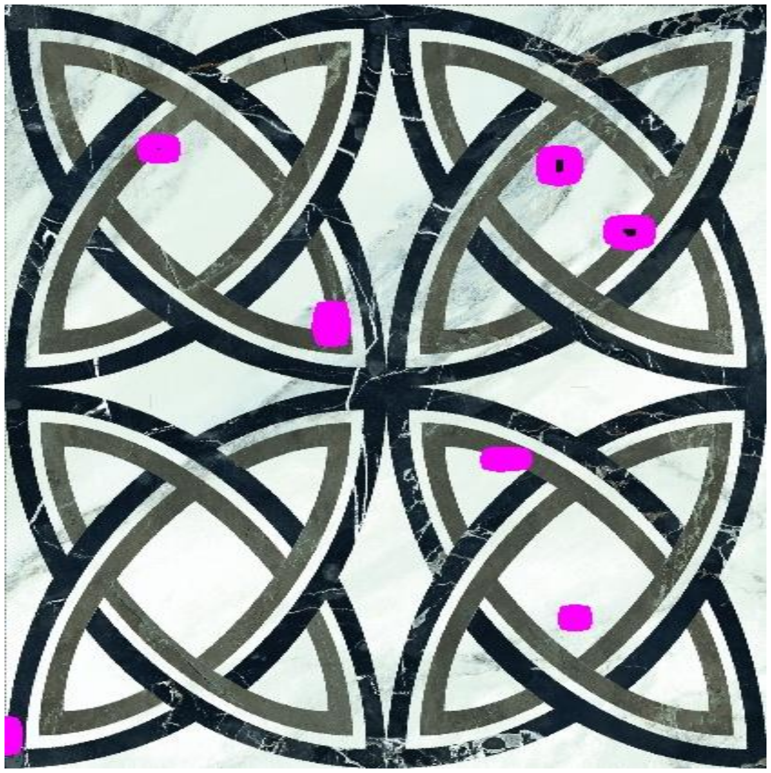


Figure 4.16: Image showing defects on the tile

4.6 Adding new tile model

Tile models are saved in a special file called "addingnewmodel" and it will be displayed on the monitor. When the operator wants to add a new tile model to be compared with, he just needs to enter the file adding and execute the instruction "./addingnew" on the terminal. The system will detect by itself the tile and will save the image of the standard tile of a specific model under the name given by the operator as describer on the flowchart shown in figure 3.5.

The two figures bellow shows the functionality of this program, and the files before and after adding the model standard image.

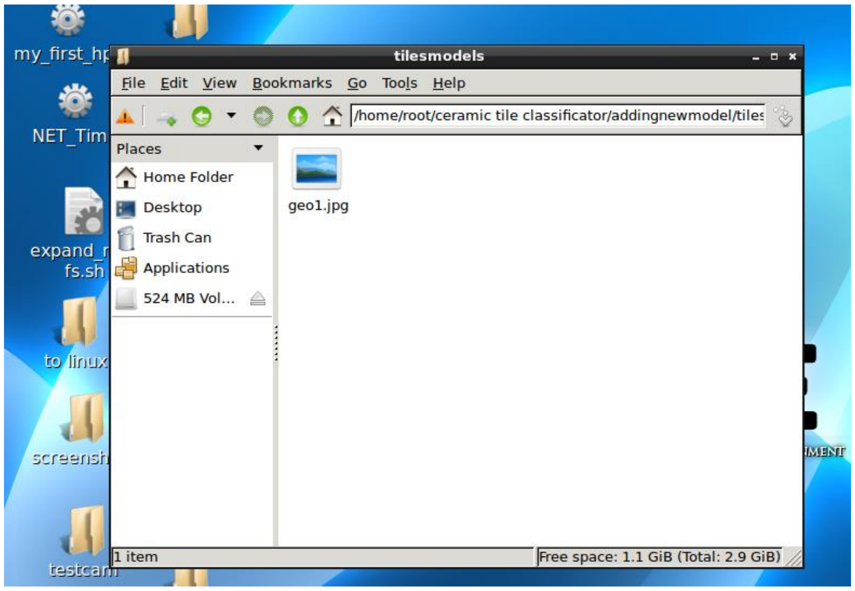
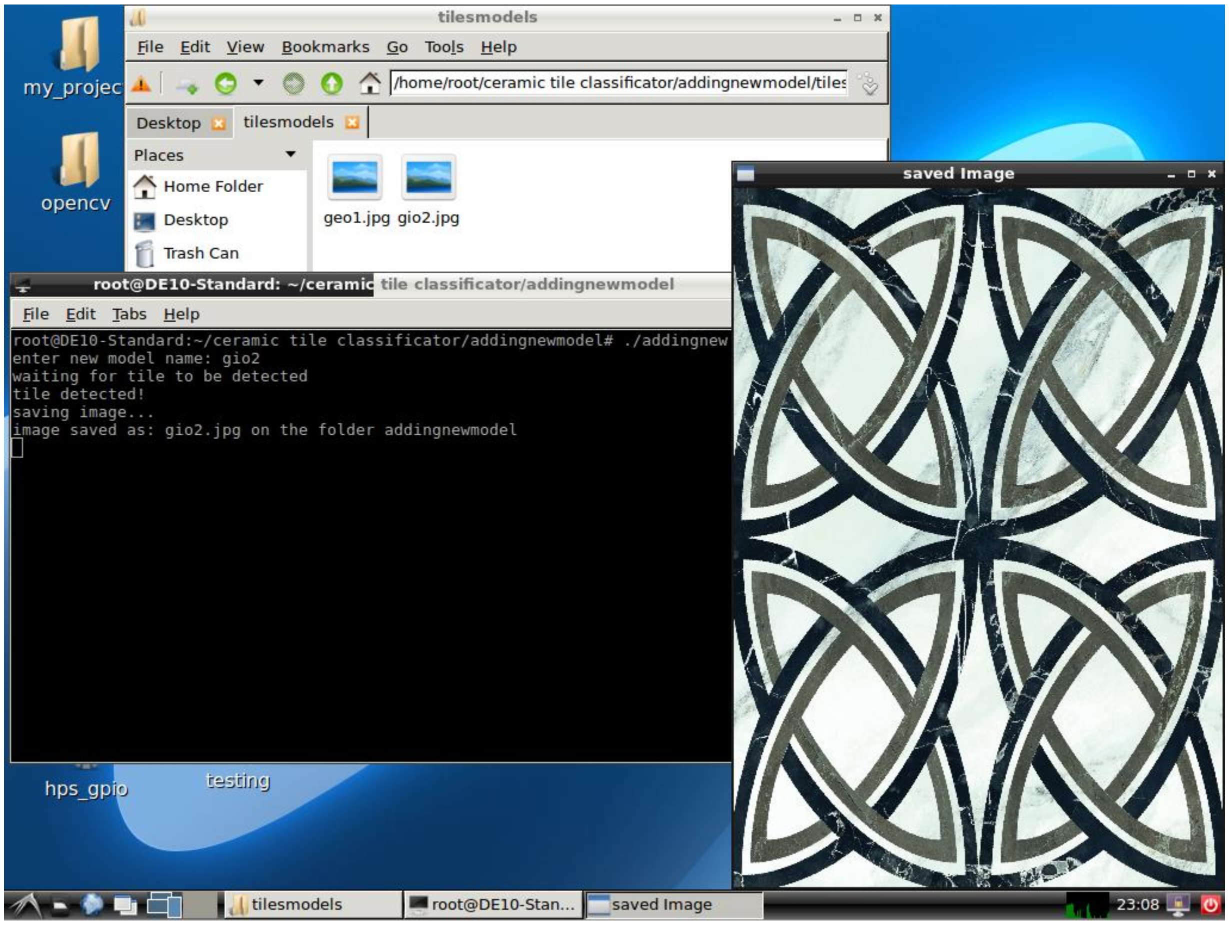

tilesmodels
File Edit View Bookmarks Go Tools Help
/home/root/ceramic tile classificator/addingnewmodel/tiles
Places
Home Folder
Desktop
Trash Can
Applications
524 MB Vol...
geo1.jpg
1 item
Free space: 1.1 GiB (Total: 2.9 GiB)

Figure 4.17: The folder before adding new tile model

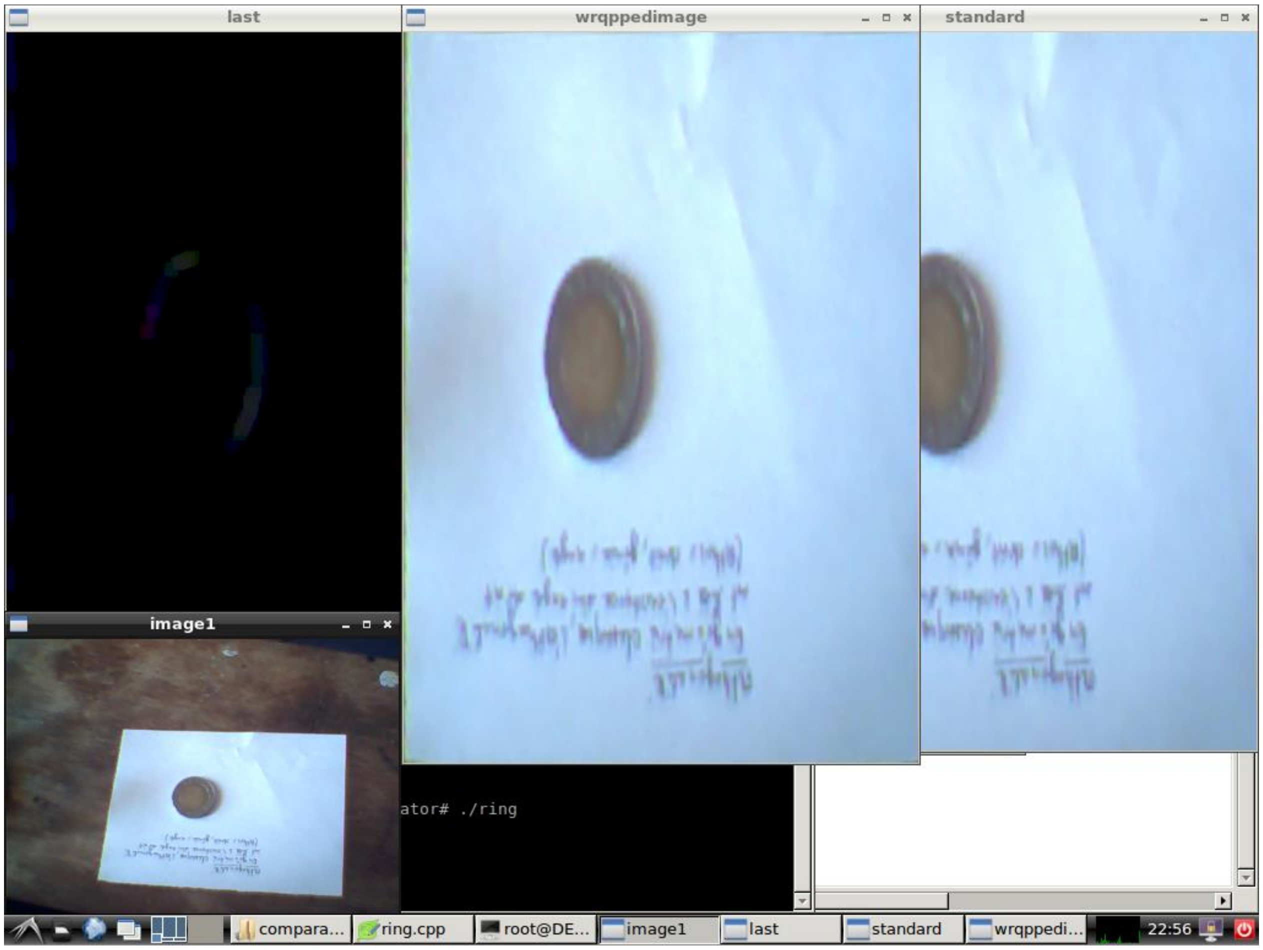

tilesmodels
File Edit View Bookmarks Go Tools Help
/home/root/ceramic tile classificator/addingnewmodel/tiles
Desktop
tilesmodels
Places
Home Folder
Desktop
Trash Can
geo1.jpg
gio2.jpg
root@DE10-Standard: ~/ceramic tile classificator/addingnewmodel
File Edit Tabs Help
saved Image
23:08

```
root@DE10-Standard:~/ceramic tile classificator/addingnewmodel# ./addingnew
enter new model name: gio2
waiting for tile to be detected
tile detected!
saving image...
image saved as: gio2.jpg on the folder addingnewmodel
```

Figure 4.18: After adding new tile model

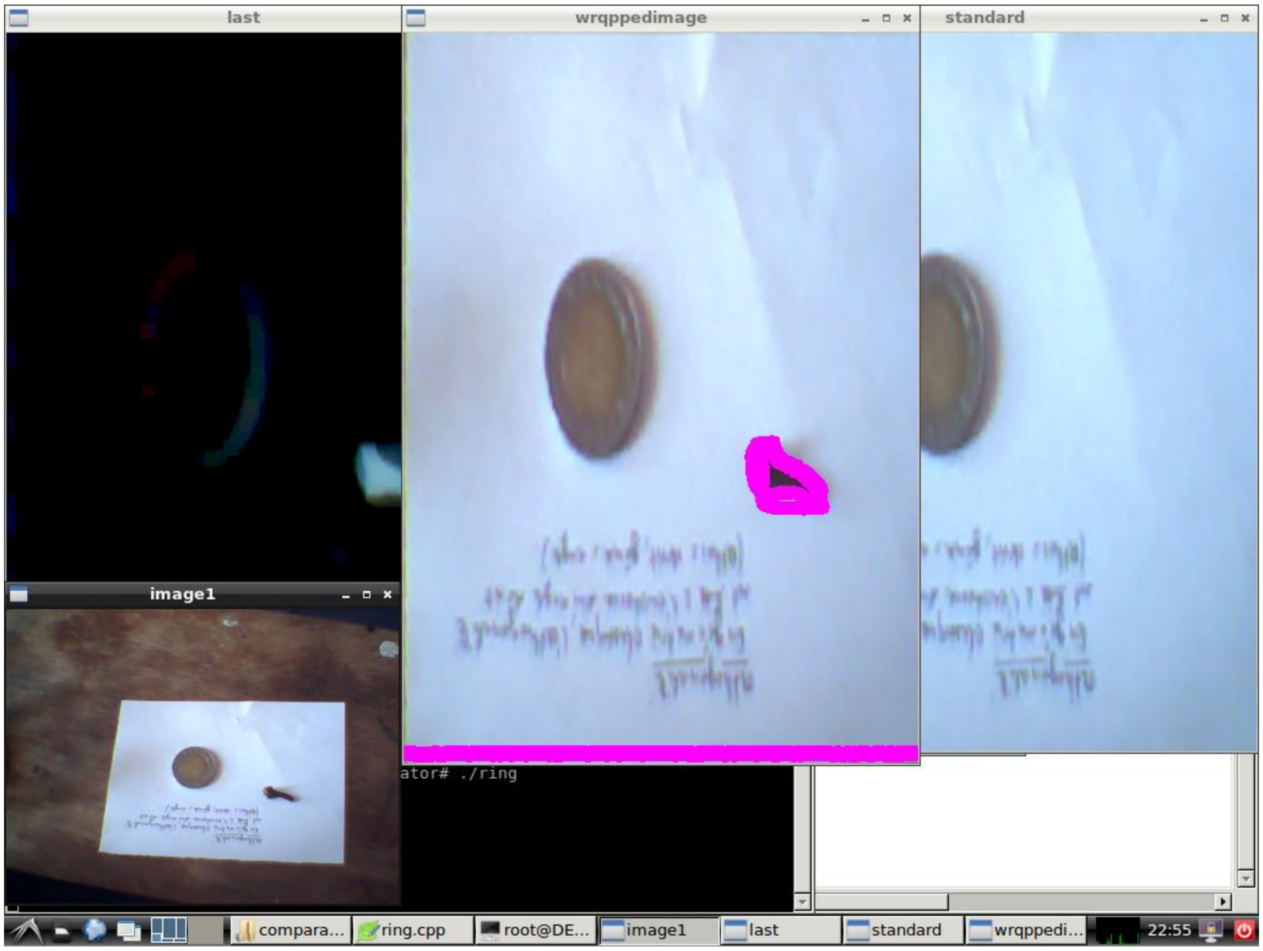
4.7 Using comparator system

To use the classification method correctly, the operator has to open the terminal in the folder named “ceramic tile classification”, then he needs to execute the instruction “./comparator”. The system will ask you to give the name of the model that is already saved. If the model was not saved before, the program will not execute. Next, the program will ask you to enter the height/width of the model that to be classified. Then, each time a tile is detected the monitor will display on the terminal the class of that tile, its dimensions, and the percentage of each choice (premium, commercial and third choice). It will also display the defected area(s) on the tile if there exist any, and it will keep classifying until the system is shut down.



```
ator# ./ring
```

Figure 4.19: Tile classification system when no defect is detected



```
ator# ./ring
```

Figure 4.20: Tile classification system when a defect is detected

The screenshots in this part are just some examples of a rectangular piece of paper that replaces a tile in our project.

4.7 Results and discussion

The purpose of this project was to implement a comparator system that helps to automatically classify tiles with different drawings automatically. Due to the lack of resources and time the project was not completely done. This system is only effective when working with the geometric and unicolor drawing on the tile (these types are mentioned in the first chapter section 1.5). It took 2.5 seconds for each tile to be inspected and that is more than enough since the industrial belt conveyer passes one tile each 4 seconds in its maximum speed and when the production is in its prime. But for random drawings, the system took more than 30 minutes to compile and execute the code for a single tile. That is because of the 1GB RAM which is too low for such searches. The second problem found is the camera. It was too difficult to find a camera that is compatible with the light Linux, and that has a quite acceptable resolution and quality that helps in the inspection of the. This led to decrease the resolution to 320 x 240, where the edge detection became difficult at some point, and camera was detecting defects even that they didn’t exist, and that is because of the low quality of the camera and the bad edge detection due to the low resolution.

This system would have been perfect if it worked in a better CPU and a greater RAM with a high-quality Camera with a higher resolution a higher frame captured per second.

When working just with images with high resolution the system took 3.5 seconds to classify the tile. This is a good example to conclude that if the resources were provided the system would have worked much better and the accuracy would have been higher. And for the Linux LXDE, the usage is a little difficult for the operator of the company. If we had more time, we could have made a Linux application that eases usage for the operator and that executes the program automatically and that has a better more user-friendly graphical interface.

Finally, the system successfully worked after a huge when solving OpenCV programming problems and when interacting the program with the camera. The results are really promising especially that the company managers were really interested in this presented system and they want to integrate it with their current classification system.

Conclusion

As a conclusion we can say that technology right now became an important part of architecture as it depends on it, and in this project image processing was used based on FPGA and HPS for ceramic tile classification. Through the previous chapters we have seen that image processing can be developed not only on Desktop PCs but also on FPGA chip and HPS microprocessor and can give excellent results in finding solutions for industrial problems. The proposed system delivers an appropriate result compared to other systems when it comes to classifying the ceramic tile.

The system can be developed more and is flexible to modify to inspect tiles for other companies based on their criteria of classification. It can also be applied to products that need visual inspection on the final product.

As a future work I will try to implement the system with a better CPU and a bigger RAM with a better camera, we will try to use machine learning techniques to classify random model tiles, we also need to add a data base of each day and print it, finally we will work on integrating an alarm when the system classifies many tiles as non-usable.

References

[1] Specification Sheet SAFCER, SETIF 2019.

[2] iso 10545-2:2010 Determination of dimensions and surface quality.

[3] FPGA Wikipedia, access date: may 2021, Online: https://en.wikipedia.org/wiki/Field-programmable_gate_array.

[4] DE-10 standard user manuel, access date: april 2021, Online: https://www.intel.com/content/dam/altera-www/global/en_US/portal/dsn/42/doc-us-dsnbk-42-5505271707235-de10-standard-user-manual-sm.pdf

[5] Intel, AN 796: Cyclone V and Arria V SoC Device Design Guidelines, 20 february 2017. access date: april 2021, Online: https://www.intel.com/content/www/us/en/programmable/documentation/doq1481305867183.html#sxr1481303255441.

[6] Cyclone V Hard Processor System technical reference, access date: april 2021, Online: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/cyclone-v/cv_54001.pdf

[7] intel, “Cyclone V hard processor System technical Reference Manual”, july 2018.

[8] Introduction to image processing in “digital image processing” course by the University of TARTU, 2019.

[9] Gonzalez, Rafael, “Digital image processing. New York, NY: Pearson”, 2018.

[10] Real-Time Image and Video Processing Concepts, access date: may 2021, Online: http://mariobasededatos.blogspot.com/2011/09/c-h-p-t-er-1-real-time-image-and-video.html

[11] BBC-Bitsize-encoding images, access date: may 2021, Online: https://www.bbc.co.uk/bitesize/guides/zqyrq6f/revision/1

[12] Patrick Hébert, “introduction to artificial vision”, master course, university of Laval, 2017.

[13] Canny, J., A Computational Approach to Edge Detection, IEEE Transactions on Pattern Analysis and Machine Intelligence, 8(6):679–698, 1986.

[14] R. Deriche, Using Canny's criteria to derive a recursively implemented optimal edge detector, Int. J. Computer Vision, Vol. 1, pp. 167–187, April 1987.

[15] Lindeberg, Tony "Edge detection and ridge detection with automatic scale selection", International Journal of Computer Vision, 30, 2, pp 117—154, 1998.

[16] Shaonan Zhang "Real Time Image Processing on FPGAs ", master thesis, university of Liverpool 2018.

[17] Gary Bradsk and Andrian Kaehler, "Learning OpenCV third edition", September 24, 2008

[18] wiki LXDE, access date: June 2021, Online:
https://wiki.lxde.org/en/Main_Page

[19] Fierce electronics, ultrasonic sensors, access date: June 2021, Online:
https://www.fierceelectronics.com/sensors/what-ultrasonic-sensor

[20] Fierce electronics, ir sensors, access date: June 2021, Online:
https://www.fierceelectronics.com/sensors/what-ir-sensor

[21] EEpower, photo resistor, access date: June 2021, Online:
https://eepower.com/resistor-guide/resistor-types/photo-resistor/#

[22] home studio experts, how many lumens you need for a video lighting, access date: June 2021, Online:https://homestudioexpert.com/how-many-lumens-do-you-need-for-video-lighting/

[23] CHATZIGEORGIOU GEORGIOS, "design and implementation of an autonomous SoC FPGA based UAV (QUADCOPTER) for data image and acquisition and processing", master thesis, university of Patras, April 2021.