FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Automatic Geometric Evaluation of Stator Laminations

LEONARDO GARCIA DE LIMA MONTEIRO

Mestrado em Engenharia Mecânica
Automação

Supervisor: Prof. António Ramos Silva

Co-supervisor: Prof. Filipe de Sousa Pereira

Co-supervisor at WEG: Eng. Jorge Fernando de Sousa

July 25, 2023

Automatic Geometric Evaluation of Stator Laminations

LEONARDO GARCIA DE LIMA MONTEIRO

Mestrado em Engenharia Mecânica
Automação

July 25, 2023

Abstract

Electric motors are the basis of movement in most existing systems. In Europe, they consume around 50% of all electricity produced. Therefore, also in light of increasingly stringent environmental legislation, the energy efficiency of electric motors is an important issue. To mitigate power losses, the stator of electric motors is composed of a stack of metal plates separated by an insulating layer, thus avoiding large losses of energy by eddy currents. These plates are produced by punching, a process that inevitably produces burrs, which due to the natural vibrations of the motor can lead to wear of the insulating layer, and consequent loss of efficiency. In the production of these plates, it is necessary to be very careful with the parameters set in the punching machine, as if it is not well calibrated, it can cause spacing errors between the slots of the motor plates, thus hindering the assembly of the stator and rotor blocks. In order to inspect the possible manufacturing defects mentioned, this project was developed in partnership with WEG, and it proposes the idealization of a computer vision system that performs the dimensional and geometric analysis of stator plates to increase production control, and ensure its quality, with the aim of minimizing defects in the stator. In order to validate this concept, a computer vision software using Python language was developed to measure the bottom slot diameter, the tooth width, and the plastic deformation caused in the punching cutting edges. Complementarily, a conceptual proposal of a mechanical design on Computer-aided Design (CAD) was developed for the implementation of tests of this system, including the definition of its components. The computer vision hardware to be used was also defined, namely the type of illumination, sensor model, and lens. The developed software was tested on images acquired with the proposed hardware, and the results demonstrated the capacity to differentiate stator teeth that have up to 20 micrometres difference between them, as well as a high measurement capability of the bottom slot diameter. In addition, tests of the computer vision hardware demonstrated high definition of the sample under analysis, and the lighting system was able to highlight the different structures to be inspected. With the implementation of this project in the factory shop floor, it is expected that the company will save time and gain precision in the calibration of the stator plate punching machine parameters, as well as greater control of the production quality allowing identification of when the punching machine needs maintenance. Besides, other computer vision applications that could be useful to the company were identified.

Keywords: Computer vision, Geometric inspection, Electric steel, Machine vision.

Resumo

Os motores eléctricos são a base do movimento da maioria dos sistemas existentes. Na Europa, consomem cerca de 50% de toda a eletricidade produzida. Por conseguinte, tendo também em conta a legislação ambiental cada vez mais rigorosa, a eficiência energética dos motores eléctricos é uma questão cada vez mais relevante. Para atenuar as perdas de potência, o estator dos motores eléctricos são compostos por uma pilha de chapas metálicas separadas por uma camada isolante, evitando assim grandes perdas de energia por correntes parasitas. Estas chapas são produzidas por puncionamento, um processo que inevitavelmente produz rebarbas, que devido às vibrações naturais do motor podem levar ao desgaste da camada isolante, e consequente perda de eficiência. Além disso, os parâmetros da puncionadora, se não bem calibrados, podem causar erro de espaçamento entre as ranhuras das chapas dos motores, dificultando a montagem dos blocos de estator e rotor. Com a intenção de inspecionar os possíveis defeitos de fabrico mencionados, este projeto foi desenvolvido em parceria com a WEG, e nele se propõe a idealização de um sistema de visão computacional que faça a análise dimensional e geométrica de chapas de estator para aumentar o controle produtivo, e garantir sua qualidade, com o objetivo de minimizar os defeitos no estator. De forma a validar este conceito, foram desenvolvidos softwares de visão computacional utilizando a linguagem Python, para medição do diâmetro de fundo de ranhura, da largura dos dentes, e da deformação plástica causada nas arestas de corte do puncionamento. Complementarmente se desenvolveu uma proposta conceptual de projeto mecânico em Computer-aided Design (CAD) para implementação de testes deste sistema, incluindo a definição de seus componentes. Foi ainda definido o hardware de visão computacional a ser utilizado, nomeadamente o tipo de iluminação, modelo de sensor e lente. O software desenvolvido foi testado em imagens adquiridas com o hardware proposto, e os resultados demonstraram capacidade de diferenciar dentes do estator que possuem até 20 micrometros de diferença entre si, assim como uma alta capacidade de medição do diâmetro de fundo de ranhura. Além disso, os testes do hardware de visão computacional demonstraram alta definição da amostra em análise, assim como o sistema de iluminação se mostrou capaz de destacar as diferentes estruturas que se pretende inspecionar. Com a implementação deste projeto no chão de fábrica, é expectável que a empresa economize tempo e ganhe precisão na calibração dos parâmetros da puncionadora das chapas de estator, assim como maior controle da qualidade produtiva permitindo identificação de quando a puncionadora necessita de manutenção. Complementarmente, foram ainda identificadas outras aplicações de visão computacional que poderiam ser úteis para a empresa.

Palavras-chave: Visão computacional, Inspeção geométrica, Aço elétrico, Visão de máquinas.

Acknowledgments

Firstly, I thank my family, without them this journey would not be possible. I will be forever grateful for the opportunity they provided me, and for the unconditional support always offered, so that even though I was physically distant, I always felt close to everyone.

I thank all the friends who were part of this journey, both in the studies and in making the academic experience lighter.

I also thank WEG for the receptivity and excellent stay I had in the company. As well as all the colleagues who helped in the development of this project, both in the quality engineering sector of WEG in Brazil, as well as the friends of MVISIA who were also very helpful, and also everyone in the RD sector of WEG Portugal, especially my supervisor in the company Jorge Fernando de Sousa.

I would also like to thank all the teachers who transmitted the knowledge acquired over the years, and especially the supervisor António Ramos Silva.

Contents

List of Figures

List of Tables

Abbreviations

AC	Alternating Current
CAD	Computer-Aided Design
CCD	Charge-coupled Device
CMOS	Complementary Metal-oxide Semiconductor
CPU	Central Processing Unit
DC	Direct Current
ERP	Enterprise Resource Planning
EU	European Union
FEUP	Faculdade de Engenharia da Universidade do Porto
FFD	Flange Focal Distance
HMI	Human Machine Interface
HTTP	Hypertext Transfer Protocol
IoT	Internet of Things
LED	Light-emitting Diode
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
MP	Megapixel
PLC	Programmable Logic Controller
RAM	Random-access Memory
RGB	Red Green Blue
ROI	Region of Interest

Chapter 1

Introduction

Industry 4.0 is already a reality, and its implementation has been growing in the industry, not only as an option but also as a necessity, due to the high competitiveness in the market. Thus, the industry is continuously implementing the concepts such as Internet of Things (IoT), data analysis, machine learning, and sensors for manufacturing control. An up-to-date industry, which implements these concepts in its environment, has greater production quality control, allowing predictive maintenance, process automation and optimization, and support in taking better decisions, while maintaining high productivity and high product quality.

In this industrial revolution, one of the tools that has been widely used due to its proven utility and efficiency is computer vision systems. These systems allow the analysis of manufacturing quality on a large scale, being able to replace employees in repetitive and exhaustive activities such as measuring parts. They also allow the massification of inspections that many times are only possible to be done to small samples in a production line, as well as releasing the employees for more dynamic tasks that have a greater need for human reasoning.

With this in mind, this project was developed as a part of a Master's thesis in Mechanical Engineering, specialized in automation, at Faculdade de Engenharia da Universidade do Porto (FEUP), which took place at WEGeuro Indústria Elétrica S.A.. It aims to be a proof of concept for the use of a computer vision system in the geometric and dimensional evaluation of stator plates of large electric motors, which have very small tolerances.

1.1 WEG Presentation

WEG (figure 1.1) is a multinational Brazilian company, founded in 1961 in the state of Santa Catarina, by three gentlemen, an electrician, Werner Ricardo Voigt, an administrator, Eggon João da Silva, and a mechanic, Geraldo Werninghaus, who joined their knowledge to start the company, and are the initials of their names that give the current name of the company.

Initially, WEG only produced electric motors, however, already in the 80s it began to expand its product portfolio, beginning the production of electromechanical components, power and distribution transformers, liquid and powder paints, and electro-isolating varnishes, entering deeper

into the automation market, establishing itself as a supplier of complete industrial electrical systems. Currently, WEG has further expanded its portfolio, entering the energy production market, with the production of solar panels and hydro and wind generators, as well as the electric mobility market, and the digital solutions market more recently, including the acquisition of a company of solutions based on computer vision combined with artificial intelligence, MVISIA.

WEG stands out as a company with a successful history and sustainable growth, much due to its set of values, beliefs, and ideals established since its foundation, and that govern the mission, and the path that the company follows.

Figure 1.1: WEG logo.

The beginning of WEG's operations in Portugal occurred in 2002, with the acquisition of a manufacturing plant in Maia, used in the production of low, medium, and high voltage motors. With the consolidation of the motor production, it also started to produce automation panels, thus supplying not only the motors, but also the means for their control.

With the success of the operation, the WEG group decided to expand its activity in the country by building a new industrial park with two factories in the city of Santo Tirso, figure 1.3, which will have approximately five times the area of the Maia plant, figure 1.2, with the aim of enabling greater production, and keeping up with the increasing demand for their solutions.

Figure 1.2: WEG's Maia factory plant.

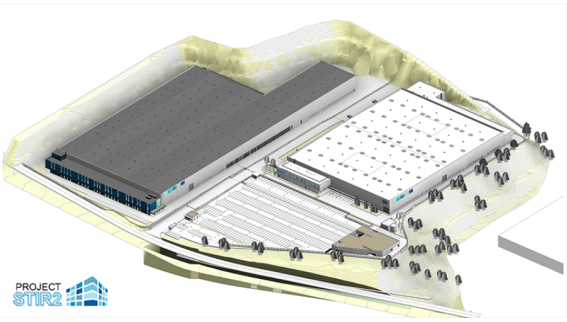

Figure 1.3: WEG's Santo Tirso factory plant.

Currently, the WEG group has over 40,000 employees, manufacturing sites in 15 countries, and commercial operations in 37, totalizing direct presence in 52 countries, producing around 19 million motors per year, and has the largest manufacturing site for the production of low voltage motors, with an area of one million square meters, while having a portfolio of over 1,200 products,

sold to over 135 countries. Despite all this, WEG never stops investing in innovation, with 56.4% of its revenues deriving from products launched in the last 5 years [1].

1.2 Context of the Work

Worldwide, the concerns with the preservation of the natural environment and with preventing the progression of the greenhouse effect have been growing, and at the same time, governments and regulatory entities have created laws and rules to reduce the aggravation of this effect. To help the global environmental goals, it is also part of the guidelines of many industries to try to reduce their carbon footprint, decreasing the polluting potential of their production lines, and also making their products more sustainable.

According to the European Commission's Ecodesign Committee, it is estimated that there are around 8 billion electric motors in use in the European Union (EU), thus consuming approximately 50% of the electricity produced in this region [2]. This way, due to the large share of electricity consumption for which electric motors are responsible, and to increase the sustainability of the use of this mechanism, they must comply with the ecodesign requirements established by the 2009/125/EC directive, and with the (EU) 2019/1781 regulation that became effective on July 1, 2021, which defines what the minimum efficiency class should be for each electric motor taking into account the number of poles, and the nominal power, also considering the use of frequency inverters, and their efficiency as well.

In order to meet the increasing efficiency requirements for the manufacture of electric motors, studies have been made on the different impacts of each production stage [3, 4], where it is shown that the care taken with the quality of the material used, as well as the quality of the production processes, directly influence the efficiency of the motor.

The stators of electric motors are composed of ferrous material, and serve as a magnetically active component of the motor, able to enhance the magnetic field generated by the coils. Since the stator is composed of a conductive material, it needs to be made of a stack of metal laminations insulated from each other, because, due to the variable magnetic field, the stator acts as a conductor of low resistance, allowing the generation of Eddy-current, which dissipates energy in the form of heat, decreasing the efficiency of the motor[5].

However, the stator laminations are manufactured by punching machines, in which the production of burrs on the sheet is inherent to the production method, which can be controlled but can hardly be completely avoided. These burrs, due to the natural vibration of the motor, depending on its size, can degrade the insulating layer of the stator laminations, causing short circuits between them and leading to large efficiency losses due to heat (figure 1.4), which, if the dissipation is concentrated in a single region, can lead to total loss of the motor (figure 1.5) [6].

Among the ways to avoid problems with burrs are to apply a large insulating layer between the laminations, which requires increasing the size of the motor or decreasing the power, and increasing the cost. Another solution is the use of a deburring machine, but its use increases the production time, and also the cost, without adding more value to the final product[6, 7]. So, a good

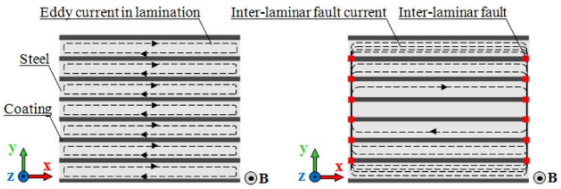

Figure 1.4: Eddy current with and without lamination fault [6].

Figure 1.5: Stator core melting caused by interlaminar insulation failure[6].

strategy to deal with this problem is to have control of the size of the burrs that are being produced and then maintain the cutting machine when necessary, before it starts producing laminations with burrs of unacceptable size.

Out of tolerances burr is not the only defect that is possible to be found in the manufacture of stator lamination. Although the burr problem is a well known defect, as any production process is error prone, it makes sense to try to evaluate other dimensions, in order to assure production quality.

Along with this, automation has been a trend in the factory environment, becoming a necessity in order to increase production quality and quantity, while reducing costs, and making employees available for tasks requiring human rationalization. Furthermore, automation also allows predictive maintenance, and detection of manufacturing errors in real time, which makes possible to the industry the reduction of waste, and better usage of capital.

Inside the automation, a resource that is being increasingly used is computer vision, allowing automatic, non-destructible inspection of parts, checking parts sizes, colors, and many other features that the human vision could inspect, but instead are performed by a computer, with the aid of a camera, and an image processing software. WEG, always following the evolution of the industry, and with its start in the field of digital solutions, in 2020 acquired part of MVISIA, and in 2022, acquired the remaining part of it. MVISIA is a company founded in 2012, and is specialized in computer vision and artificial intelligence, developing turn-key solutions for the industry, installing when needed their own hardware and software solutions for many different types of inspection. With the entry into the field of computer vision, WEG's interest in implementing computer vision solutions in its own industry is even higher [8].

1.3 Problem Definition

The punching process is one of the most common in the industry. Unlike the other plastic forming processes, it does not act only in the elastic and plastic domains of the material, but forces it to its rupture by shear stress. For this process, a punch forces a part of the sheet metal against a die with a shape complementary to that of the punch, thus making a cut in the sheet with the same shape as the tool. For this to occur, it is necessary that there is a clearance between the punch and the die, leading to burr formation, a defect inherent to the punching process. This clearance is a basic parameter of the process and one of the most relevant, because the size of the burr is directly related to the size of the clearance. The smaller the clearance, the smaller the burr, and the better the quality of the product. However, the manufacturer must balance this parameter in order not to increase the production costs too much, because a small clearance also leads to intense wear of the punch, shortening its lifetime and increasing the production cost [9].

The cutting edge of lamination is divided into 4 zones, rollover, burnish, fracture, and burr. The rollover, and burnish are smooth surface zones, the rollover has a curvature, while the burnish has a flat surface. The fracture and burr zones have a rough surface, as their faces are a consequence of the fracture when the limit of the material is reached, with the burr being part of the rough face that exceeds the thickness of the plate (figure 1.6).

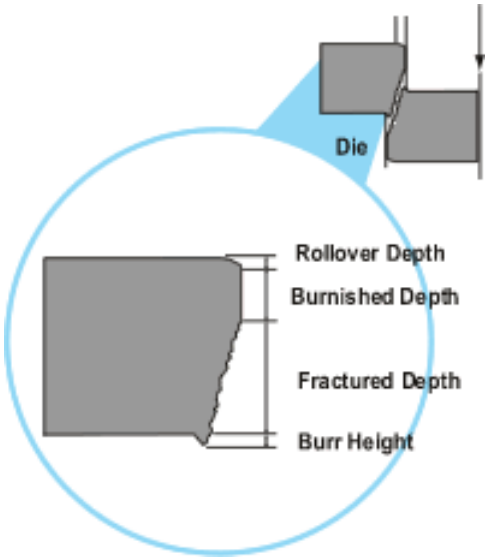

Figure 1.6: Cutting edge profile [10].

The EN 60740-1 standard, entitled "Laminations for transformers and inductors - Part 1: Mechanical and electrical characteristics", indicates that on an edge of up to 10 mm made by punching, the burr should not exceed 0.05 mm, however, punctual burrs of up to 0.1 mm are acceptable [11].

Considering the profile of the cutting edge, initially, two possibilities to approach the problem of production quality analysis were considered. A first approach would be to directly analyze the

burr size, ensuring that the burr is smaller than defined by the respective standard and production requirements. A second possibility would be an indirect analysis of the burr, measuring the rollover, since from this structure it is possible to infer the clearance between the punch and the die, and from this value infer the burr size. The direct burr analysis may have the advantage that, as it is a projection, this property could be used to distinguish it from the lamination with different light positioning. Regarding the rollover analysis, measuring the width of its profile may be facilitated by the reflection of its surface, which is smooth and curved, which would make its structure reflect light in different directions than the lamination planar region, which in turn could be taken advantage of in image segmentation.

However, both the rollover and burr sizes are very small, requiring analysis with precision in the order of tens of micrometers. To make it possible, it is expected that a high resolution camera will be required, in order to build a computer vision setup of high accuracy. At the same time, the illumination part of that system must be carefully chosen to allow good images to be captured by the camera. Despite the setup, even if its hardware is of high quality, it will serve for nothing if the images are processed by low accuracy software, so in order to achieve good results, it is also required the development a high performance software.

One way to artificially increase the resolution of the acquired images is by using image processing algorithms, namely sub-pixel algorithms. This type of algorithm aims to calculate the position of entities in a space between pixels, decreasing the degree of uncertainty that comes from the nature of digital images, which have a discrete representation of the environment. The use of this image processing method can not just improve accuracy, but also reduce the necessity of a camera sensor with extremely high resolution, which would make the system more expensive, and harder to process images.

In addition to the burrs, there are other geometrical features worth being controlled in stator lamination production. The manufacturing process of the slots of stator laminations is based on the rotation of a die, while the punch performs up and down movements, in order to produce all lamination teeth. Usually, the machines for manufacturing slots, work with different lamination diameters, requiring different dies, also to be able to produce teeth of different shapes. As the used dies are not always the same, the machine must have its parameters reconfigured in order to adapt to the different moments of inertia of the dies, and allow proper control of the rotation motion, otherwise, angular manufacturing error can be performed, as can be seen in an exaggerated form, and not representative of WEG's production, in figure 1.7, in this case, a rotor example. That tuning of the machine parameters is sometimes performed by trial and error, mainly when new dies are to be used, therefore it can be a time costly process, that could be improved with the use of an automatic inspection system, improving at the same time the accuracy of the parameters tuning.

Another dimensional error that also can occur during the manufacturing process, although less common, is the production of stator laminations out of circularity tolerances. This manufacturing defect is due to the manner that the laminations are held in place on the punching machine during the process of manufacturing slots. The centering of the lamination on the top of the die is

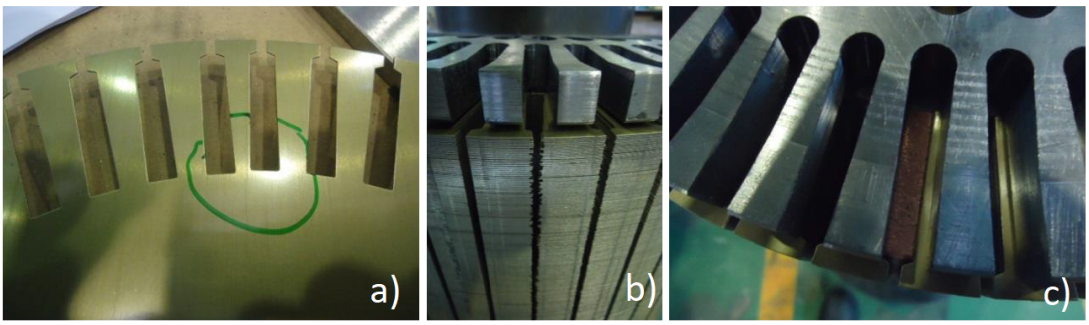

Figure 1.7: a) A rotor lamination with different size teeth, b) Side view of a rotor block with misaligned laminations due to different sizes of teeth, c) Top view of the rotor block on b). These images do not represent WEG's production quality.

performed using rolls that push the lamination in the radial direction. However, if the centering force imposed by the rolls is greater it than should be, it can bend the lamination. As the cut of the internal diameter of the lamination is produced at the same time as the slots, i.e., during the rotation of the die, if the lamination is bent, this diameter acquires an elliptical shape, instead of the intended circular one.

Both the manufacturing problems mentioned make difficult the insertion of the windings of the motor during the assembly of the stator, due to the misalignment of the stator slots. Similarly to the burr defect, the correction of those problems also does not provide added value to the final product. When this misalignment happens, usually it is corrected manually, with the use of a steel file to enlarge slots, a process that takes a long time, and is exhaustive for the worker. Although those defects are not common to occur, checking them up is worth it due to the consequences they imply if they are present in the lamination.

In the market, there are options available for visual inspection of parts, however, most of those solutions are expensive, and it is not possible to couple them to the production line, for example, the ZEISS O-INSPECT (figure 1.8). In order to obtain a solution that can be integrated into the production line, and considering costs, and the specificity of the part features to be recognized and measured, it makes more sense to develop an in-house customized solution.

1.4 Objectives

Regarding the objectives of the proposed work, the main goal is to develop a proof of concept of a system that performs automatic geometric and dimensional evaluation of electric motor stators laminations of different dimensions, in order to verify if the burrs and other dimensions resulting from the lamination cutting process are within the production requirements or not.

This evaluation must happen in a way that does not interfere in the production process, that is, without contact with the part, and without delaying the manufacture. For this reason the inspection through computer vision was chosen, and a low processing time appears as a system requirement.

Figure 1.8: ZEISS O-INSPECT multi-sensor measuring machines [12].

As a computer vision setup is going to be used, the minimum requirements of the hardware must be defined, as well as the most appropriate illumination for it.

To emulate a real usage of the developed system, and better evaluate the feasibility of that solution, it is necessary to develop a mechanical project for positioning the different lamination regions for the computer vision setup, along with a mechanism that can position the setup in order to adapt it to the laminations of different sizes, considering that WEG utilizes laminations of outside diameter from 530 up to 1420 millimeters.

To perform the inspection of the lamination geometry and dimensions, as it is a new and specific use of computer vision, the software for this task is unique and has to be developed almost from the ground, and be capable of performing high performance measuring. For that purpose, Python language will be used, as it is one of the most used programming languages nowadays in computer vision, and has the support of several libraries that help in the development of computer vision software.

1.5 Report Outline

This Master's dissertation was divided into five chapters: The first chapter is an introduction to the developed project, starting with a brief presentation of the company where the work took place, followed by the contextualization of the dissertation topic, soon after defining the problem to be solved, and finally defining the objectives of the project.

The second chapter consists of a literature review, divided into three parts. The first details the component to be evaluated. In the second part, the computer vision hardware is analyzed. The third and last part of this chapter focuses on computer vision software part, analyzing the algorithms and techniques used in image processing.

Regarding the third chapter, its objective is to present and detail the solution developed, both the software and hardware aspects, including the mechanical solution developed.

The fourth chapter is the part where the results obtained by the developed solution are presented and discussed, in terms of software accuracy and feasibility of the mechanical solution.

The fifth and last chapter consists of the conclusions obtained at the end of the project after analyzing the results, and the description of future work to be developed, such as improvements and other applications of computer vision in the WEG plant.

Chapter 2

Literature Review

This chapter is divided into three sections in order to cover the theory and literature review of the main subjects related to the project addressed in this work. The first section contains a brief introduction about electric motors and the components to be analyzed by the vision system to be developed, and also the importance of the analysis of it. The second section focuses on the hardware to be used on vision systems, considering which type of components are used, their properties, and how they work. The last and third section aims to explain how computer vision software works, its theoretical background, which tools there are, and recently developed new solutions.

2.1 Electric steel

Electric motors are one of the most important machinery used in the world. They are responsible for the production of motion through electricity, provide movement to the majority of the non-vehicle machines, and even becoming a trend in the automobile industry as well. The working principle of electric motors is based on Faraday's law of induction, a law of electromagnetism that explains how the interaction of a magnetic field with an electric circuit produces an electromotive force, which is called the induction phenomenon. Considering the motion production in a motor, it has two magnetically active parts, a typically outer one, static, coupled to the motor's frame, called stator (figure 2.1), and a typically inner one, movable which is the rotor. The movement of an electric motor is created by a magnetic field generated in the stator, that interacts with the rotor by Faraday's Law, and makes it rotate. However, this interaction can happen differently depending on the type of electric motor.

There is no absolute consensus about the classification or grouping of electric motor types, however, a common manner of dividing them is into two main groups regarding the type of current that is supplied to the motor, i.e., direct current (DC) motors and alternating current (AC) motors. The working principle of the first one is based on a static magnetic field generated by the stator, while the rotor has windings fed with DC current, then a force is imposed on the rotor winding due to the Lorentz force principle. Regarding AC motors, their main difference is to have

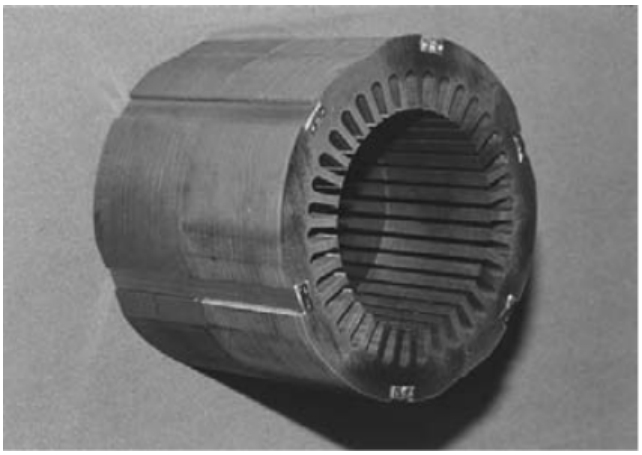

Figure 2.1: Lamination stator stack [5].

a rotating magnetic field in the stator, which causes the rotor to rotate. However, there are two main subdivisions in this group, the synchronous motors, which are the ones that the rotor rotate at the same speed of the varying magnetic field of the stator, e.g. reluctance and permanent magnet motors, and the asynchronous motors, in which there is a slight difference between the speed of rotor and of the stator rotating magnetic field, with this difference being called slip.

Asynchronous motors are one of the most used in the industry and are relatively cheaper than synchronous ones. The working principle of those motors is to have a rotating magnetic field in the stator that induces a current on the rotor winding or bars, depending on its constructive solution, and there are models that are supplied by one and three phases. In WEG Portugal, the production is focused on three phase asynchronous motors of low, medium, and high voltage.

Regarding the construction solution of the stator and rotor of three phase asynchronous motors, they are made with thin plates called laminations, in order to reduce power loss by eddy current. These laminations are the carriers for the magnetic fluxes in the motor, which actively participate in motion production. Therefore, they are composed of ferromagnetic material, in order to magnify the magnetic field that interacts between the components. Considering these aspects, laminations play an important role in guaranteeing motor efficiency and high power density.

2.1.1 Laminations Properties

The lamination needs to have ferromagnetic properties since one of its main functions is to magnify the magnetic field generated in the stator, so they are made of ferrous material. The alloy composition of the lamination usually contains silicon as an additional element, up to 3% of the mass. The addition of silicon is useful in reducing eddy current losses, however it increases the cost of the material. Another property that varies with the addition of silicon is the reduction of the saturation value of the magnetic flux density, decreasing the magnetic permeability. Other expected characteristics of the alloy are low or no carbon, sulfur, and titanium [5].

Regarding the thickness of the laminations, the thinner they are, the more expensive they are to produce, but also the lower the eddy current loss. It should be considered that the thinner the

laminations, the lower the amount of ferrous material in the stator block, because there will be more plates, and therefore more interfaces. Since the plates have insulation between them, so ferrous material is replaced by the presence of insulator material, which also leads to a decrease in the saturation value of the magnetic flux density.

As mentioned before, the constructive solution of using several plates in the stator construction is to reduce the eddy current losses, so the laminations need to be insulated from each other. The insulating material is either organic or inorganic, or even a mixture of both. A good coating also has properties that help in the performance of the cutting process, by lubricating the tools, reducing the wear of the punch and consequently reducing the production of burrs [5]. The insulation layer also prevents contact between one lamination and the burr of another, giving a small spacing between them, and avoiding a short circuit, as can be seen in the figure 2.2.

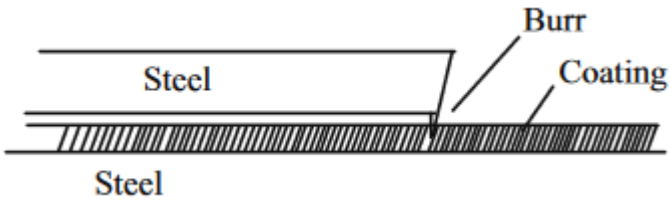

Figure 2.2: Coating layer between two steel plates [5].

2.1.2 Punching

The laminations are produced by the punching cutting process, and this process has the advantages of high repeatability, precision, and speed, and to comply with industry standards [13]. The punching cutting process consists of a punch that forces the material to be formed against a die of complementary form, and is considered a plastic forming process.

A disadvantage of this type of process is the strain hardening event, which in some industrial processes can be desired. However, when the objective is the construction of magnetically active components, the operation by cutting leads to reduced magnetic properties, and consequent losses in the core [13]. Also, the sensitivity of the deterioration of magnetic properties due to the stresses related to the cutting process is also variable according to the material being used. In [3] it is possible to see that the higher the concentration of Si in the alloy, the greater the sensitivity to deterioration. In [14] it is also shown that the closer to the deformed region from the production process, the greater is the loss of magnetic properties, so it is interesting that the deformed region be as small as possible.

In punching cutting processes, the quality of the cut and the size of the burrs are directly related to the clearance between the die and punch, so, the smaller the clearance, the better the quality and the lower the loss of magnetic properties and burr formation. However, in [15] it is analyzed the non feasibility of using very small clearances, which quickly lead to wear of the cutting tool, causing non-coaxiality between the die and punch, generating poor quality cuts, and increasing the cost of the process by the constant need for re-sharpening.

Burrs occur in many manufacturing processes, and are one of the biggest problems of high-precision machining. The burrs are the consequence of plastic deformation of the material in the production process, causing a projection of the material beyond the pre-defined geometry, creating inspection, assembly, and component production automation problems[7].

The possibility of using other production methods has already been analyzed in other studies, such as the wire electrical discharge machine, laser, and abrasive water jet processes. In [16] the efficiency losses of laminations produced by all these methods were compared, concluding that the wire electrical discharge machine method had the least losses, and the abrasive water jet method had the most power losses. Another interesting analysis was the observation of roughness on the cutting surface, since it is also related to the efficiency of the machine, and the method that generated less roughness in this region was the punching method.

2.1.3 Eddy current analysis

Insulation between laminations may be able to prevent eddy currents from flowing across different plates, however, if the plates are produced with poor quality, they will have large burrs as a consequence of the cutting process. Due to their size and the natural vibration of the motor operation, short circuits between laminations can happen at burr points, causing the problem of power loss to reappear throught heat dissipation by eddy currents. The eddy current losses in these situations can be estimated through analysis using the finite element method as done in [17], where it was possible to analyze that there is a linear relationship between the value of the losses and the size of the burrs, as well as a cubic relationship between the losses and the number of teeth that are short-circuited due to the burrs (figure 2.3), this second relationship is due to the skin effect.

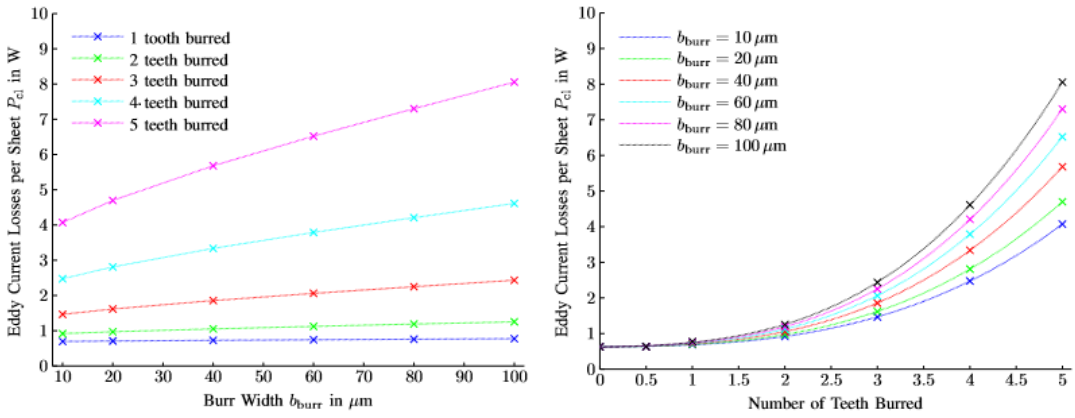

Figure 2.3: Electric motor efficiency loss due to burr. a) Losses as function of burr width for different numbers of teeth, b) Losses as function of the number of burred teeth [17].

2.2 Image Acquisition

2.2.1 Camera Sensors

The camera's sensors are one of its main components, and their function is analogous to the human eye's retina, being the part responsible for detecting the luminous energy of the environment to be processed into an image. The sensors are composed of a matrix of sensors called pixels, which are responsible for generating electrical signals proportional to the intensity of the light that shines on each one of them.

There are two main types of sensors on the market, the charge-coupled device (CCD) and the complementary metal-oxide semiconductor (CMOS). Both sensors are metal-oxide-semiconductor (MOS) based, with the difference that the CCD uses MOS capacitors, while the CMOS is a more recent model that uses Metal Oxide Semiconductor Field Effect Transistor (MOSFET) technology, which is a type of transistor.

Regarding the CCD-type sensor, figure 2.4, its functioning is based on a grid of sensors that are able to maintain an electrical charge due to the capacitors that each pixel owns, an electrical charge that has a value proportional to the light intensity that irradiated on the pixels of the sensor. With the capacitors properly charged, each line transfers its charge sequentially to a shift register. After passing through the shift register, the electrical signals are converted into digital signals to be processed by the camera electronics. This process is repeated several times until all the pixels of the sensor are completely read, and finally, the electronics build the final image. CMOS sensors, on the other hand, have both the options of capturing the image line by line of the sensor (rolling shutter), or globally (global shutter), while the pixels of the CCD capture the image only in global shutter mode. In addition, the CCD creates analog information that is transformed into digital information by using an analog-to-digital converter, while CMOS produces information at each pixel that needs to be converted to analog for further conversion to digital. The active pixel CMOS sensors have one MOSFET transistor and one amplifier per pixel, while the passive pixel type has one amplifier per pixel column, which is the case of the image 2.5, that works as a charge gate for the amplifier of its respective column [18].

By early 2000s CMOS sensors were practically not used, due to their very low quality, as a result of the lack of appropriate technology for their manufacture with better performance. After a few more years their use has been growing, and since around 2010, their quality has been increasing more and more, often surpassing the CCD in current times [20].

Nowadays one of the main advantages of the CMOS sensor over the CCD is the faster image capture, besides having a lower energy consumption, being cheaper, and also having a smaller size. An overall comparison between the two models can be seen in table 2.1.

In order to be able to identify the colors of the images, a filter is applied on the sensors, which only allows the passage of specific wavelength bands, so that each pixel measures a specific color. The most common case of application is the Bayer filter, figure 2.6, in which half of the sensors are responsible for identifying the green color, a quarter have the blue filter, and another quarter have the red filter, thus composing the Red Green Blue (RGB) channels. The reason for having

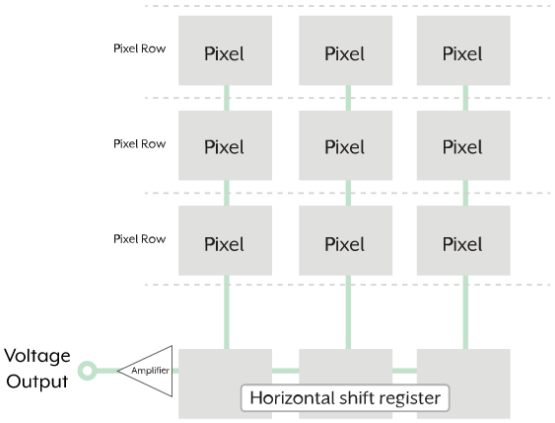

Figure 2.4: Schematic of a CCD sensor [19].

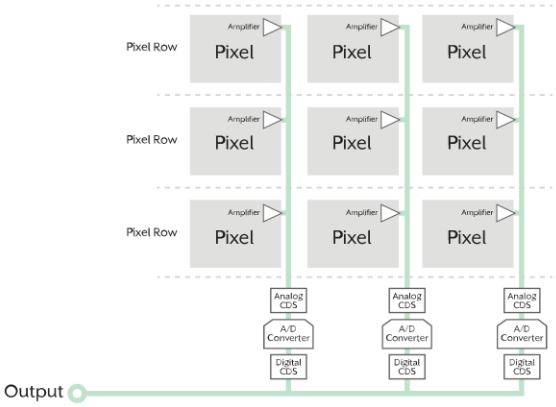

Figure 2.5: Schematic of a CMOS sensor [19].

Table 2.1: CCD and CMOS sensors comparison [18].

CCD	**CMOS**
Low noise	More susceptible to noise
Greater light sensitivity	Lower light sensitivity
Greater power consumption	Lower power consumption
High dynamic range	Moderate dynamic range
Expensive	Cheaper

more green filter pixels than the other colors is to simulate how the human eye perceives colors, since humans are also more sensitive to green.

In color sensors, although the pixels of the sensor capture specific wavelength ranges, in the end all pixels need a light intensity value for the 3 RGB channels, to join the values of each channel and be able to define which specific color should be represented at each point. In this way, pixels of specific colors get values for all channels by interpolating the intensity values obtained in neighboring pixels of different colors. Since then the values of pixels in channels other than their original one are obtained through interpolation and not a direct reading of the light intensity, there is loss of signal quality. In addition, monochromatic sensors, in which all pixels have as their signal value the general light intensity incident on them, also have about twice as much sensitivity, being able to acquire images with greater clarity and richness of detail, and are therefore more recommended for high-precision applications where color identification is not required.

2.2.2 Lens

Lenses are an essential component of image acquisition: Without them it would be impossible to acquire a well-defined image without all the shapes being completely blurred in the image. The purpose of the lens is to converge the light rays reflected/emitted by objects in various directions onto a single point on the camera sensor, so as to obtain well-defined shapes. This conversion

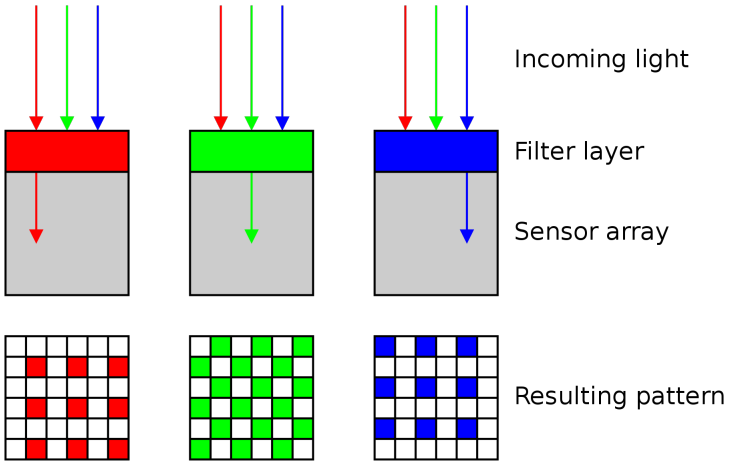

Figure 2.6: Bayer filter [21].

can be achieved by taking advantage of the refraction properties of the material that composes the lens. Glass is usually employed, however, there are other less common material options.

A very important characteristic of lens is the focal length. This characteristic states, usually in millimeters, the distance between the starting point of convergence of the light rays and the point where they meet with the lens focused on the infinity, i.e., in the case of the figure 2.7, it would be the distance between the focal point and the plane of symmetry of the lens. The relevance of this feature is that it is responsible for the area that will be reproduced on the image sensor. The greater the focal length, the smaller the area reproduced on the sensor, so the focal length is inversely proportional to the angle of view, figure 2.8. This way, it is clear that there is a direct relationship between the focal length and the zoom of the lens.

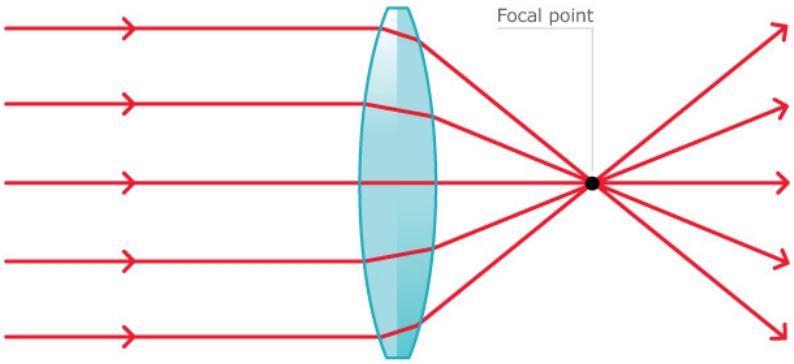

Figure 2.7: Converging lens [22].

Although lenses are an essential component for image formation, they also produce some defects that are impossible to be completely avoided, although they can be reduced in special types of lenses. One of these defects is chromatic aberration, since lenses are a kind of prism, the same effect of color separation due to different refractive indexes for each light spectrum also occurs in

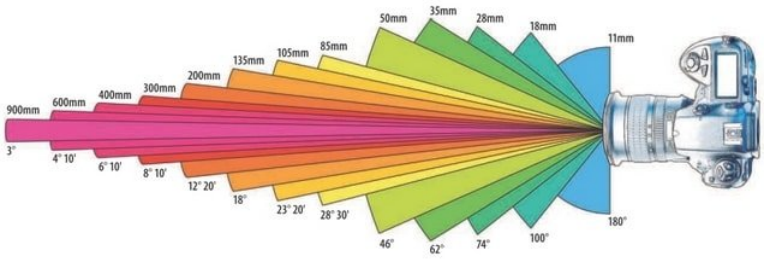

Figure 2.8: Relationship between focal length and angle of view [23].

lenses. However, this phenomenon occurs mainly at the edges of the lens, the region that has a greater difference of angle between the entrance and exit faces, while in the center of the lens this effect occurs at a much smaller scale. Another defect produced by lens is optical distortion, which is caused by lens distortion, an optical aberration that causes objects in a lens-captured image to deviate from their intended shapes. Several elements, including the lens's shape, the angle at which light enters the lens, and the lens's placement in relation to the subject, contribute to this variation. Barricade and pincushion distortion are the most typical types of lens distortion, and they can be fixed through lens design or post-processing software. Understanding lens distortion is important because it affects measurement precision and image quality in computer vision.

A very special and extremely relevant type of lens in the field of computer vision is telecentric lens. This type of lens has the feature of forming the image of objects using only the light rays that arrive with parallel orientation to its axis. As the focus is positioned at infinity, it is possible to obtain a constant magnification of the objects, i.e., regardless of their position or distance, their image is always produced in the same size, creating an orthogonal projection, instead of a perspective projection. This is an extremely useful feature in machine vision applications that aim at high precision measurements, because this way it is possible to avoid inaccuracies associated with variations in the position of the sample to be analyzed, as can be seen in the figure 2.9.

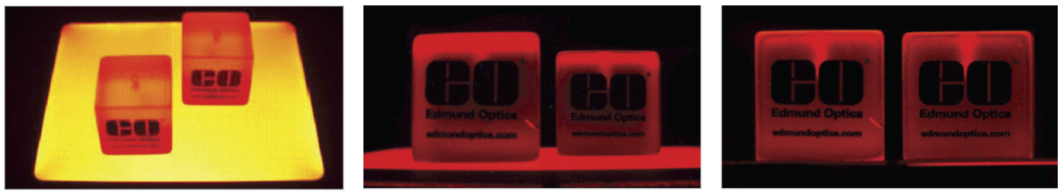

Figure 2.9: Comparison between telecentric and regular lens image acquisition. The left image is the test setup. The center image is the setup captured with regular lens. The right image is the setup captured with telecentric lens [24].

The type of telecentric lens mentioned above consists of the mono-telecentric lens, in which only the plane of the object is at infinity. However, there is also the bi-telecentric lens, in which the image plane is at infinity, and in addition to the magnification immunity to the object placement, the variation of distance between the lens and the camera sensor also does not interfere in the image production, then the bi-telecentric lens turns out to be an afocal lens. The use of these

lenses occurs in cases where even greater precision is required, since distortion is minimized even more, causing the accuracy of image formation to be virtually the same regardless of whether the objects are positioned in the center or on the periphery of the lens.

Another important aspect of the lens that has to be considered when dimensioning a computer vision system, is its mount type, which the camera mount type will restrict. The most common mount types found in computer vision cameras are the C and CS mounts, and both are attached to cameras by a threaded connection. The lenses that have this mount type are very similar in general, differing mainly on their Flange Focal Distance (FFD), which is the distance between the back of the lens and where the image is focused. The FFD of the C-mount is 17.5 mm, while the FFD of the CS-mount is 12.5 mm, making it possible to use a C-mount lens in a CS-mount camera with the use of an adaptor that makes the lens approximately 5 mm more distant from the camera sensor, while it is impossible to make the inverse combination, as there is no way of removing 5 mm out of the camera.

2.2.3 Camera Calibration

As seen in the previous section, with the exception of telecentric lenses, it is common for them to cause distortion when forming the image on the camera sensor. Since it is impossible to avoid distortion in conventional cameras, when one wants to perform high precision measurements and wants to compensate the error caused by distortion, compensation by means of software is performed. This is done by calculating what should be the real position of points in the image taking into account lens and sensor parameters, which are called intrinsic parameters and distortion factor. When working with 3D vision, it is also necessary to calculate the so-called extrinsic parameters, which are important to relate the 2D images acquired with the 3D world, as these parameters are related to the position of the camera in space [25, 26]. The main intrinsic parameters are the focal length, focal center, and the skew coefficient. These parameters are then grouped in a matrix, called intrinsic matrix 2.1, which makes the correction of the pixels considering these parameters.

$$k = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

- k - Intrinsic matrix.
- s - Skew coefficient, non-zero if the image axes are not perpendicular.
- c_x, c_y - Optical center, in pixels.
- f_x, f_y - Focal length, in pixels.

On an excellent camera calibration, it is still necessary to take into account the lens distortion coefficients, which are not considered in the intrinsic matrix. These parameters are important to account for radial and tangential image distortion. Radial distortion refers to the lens distortion

already discussed, while tangential distortion is relative to the non-parallelism between the camera sensor and the lens. The radial distortion parameter are k_1, k_2, k_3, and can be seen in the equation 2.2 that represents the distortion in order to the pixel coordinates. The calculation of the corrected position of the pixel is also shown in 2.2.

$$\begin{aligned} x_{distorted} &= x(1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6) \\ y_{distorted} &= y(1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6) \end{aligned} \tag{2.2}$$

- x, y — Undistorted pixel locations. x and y in normalized image coordinates. Normalized image coordinates are calculated from pixel coordinates by translating to the optical center and dividing by the focal length in pixels, thus, x and y are dimensionless.
- k_1, k_2, and k_3 — Radial distortion coefficients of the lens.
- $r^2 = x^2 + y^2$

The process of acquiring these parameters, intrinsic and distortion, is currently very simple. Most computer vision softwares already have functions to calculate these parameters, for example MATLAB has the `cameraCalibrator` function, and the OpenCV library in Python has the `calibrateCamera`, in which both return the values of the intrinsic matrix and the radial distortion coefficient. The input that image processing software requires in order to compute these values, are images with some known pattern in many different positions, which allows it to be recognized and compared with the original shape of the pattern. This pattern usually has the shape of a checkerboard or something similar, as the one in figure 2.10, and is usually provided by computer vision software and libraries [27, 28].

The last step in calibrating a vision system is to obtain the relationship between how much a pixel of the image is in metric dimensions, so that it is possible to obtain real measurement values from the images. The most common way to proceed with this calibration is acquiring an image of a caliper of known size, and analyze in how many pixels it is represented, and by means of the equation 2.3 it is possible to measure samples in question directly in metric values. It is also possible to do this calibration with an inverse method, as it is possible to see in [29], where the author moves an object by a known distance, and analyses the observed difference in pixels, obtaining the same relation.

$$p_{mm} = \frac{pixel}{d_{milimiter}} \tag{2.3}$$

2.2.4 Illumination

Illumination is a critical part of dimensioning a vision system, since it is the component that will make possible the visualization of what one wants the camera sensor to see. For that purpose there are many different types of illumination techniques, each one with its own purpose, that is ideal for a specific type of application, and also there are different types of lamps to be used.

Figure 2.10: OpenCV calibration checkerboard [27].

Illumination components are divided into categories according to how its light rays are spread, the main three categories being, point, spot, and area light. Point light is one of the most common lighting types, an example of it is light bulbs, in which light rays are spread in all directions with a common source point. Spot light is a very direct light source, in which the light rays are spread in a conical volume, just like a lantern, in which the closer the object from the light source, the brighter it is. In area light, the light rays have a larger source, commonly with a rectangle or a circle shape, from where the light rays are emitted more parallel to each other than in the other types. Usually, spot lights are more simple and cheaper, however, they can create an uneven brightness in objects that are highly reflective, causing what are called hotspots, and interfering in inspection. An area light, in turn, could illuminate a surface more evenly, but with a larger size, what could be a problem in a system in which space is a constraint.

Regarding the object to be analyzed, every surface interacts with light differently. There are three different types of surfaces regarding reflectance: Specular, glossy, and diffuse reflection. Surfaces that have specular reflection, reflect all incident light regarding the law of reflection, which says that the incidence and reflection angle regarding an axis perpendicular to the surface, have the same value. This kind of surface is commonly seen in metallic materials, and the more polished it is, nearly to specular its surface is. On the other hand, surfaces with diffuse reflection reflect light in all directions, which is seen in surfaces with rough finishing such as regular paper. Between specular and diffuse reflection, there is glossy reflection, which is an intermediate level, in which lights rays are reflected in different directions but with an average reflection angle close to the incidence angle, being similar to the diffuse or specular reflection depending on the surface roughness, and the material. The reflectance of the object is a very important factor when deciding which type of lighting technique to use, as these different types of surfaces will appear differently

on the image depending on the illumination, interfering in identifying specific features of an object, or the readability of information on it, due to the hotsposts previously mentioned.

Concerning lighting techniques there are many options, each with advantages and disadvantages. The most common is direct lighting, which can be performed with many different types of lamps. This method consists in direct illuminating the object to be analyzed, so it can reflect light to the camera sensor and then be seen. One of the simplest lighting techniques is bar lighting, in which a lamp with a bar shape, or a light-emitting diode (LED) strip is used to light the sample or a specific edge of it. This method is a direct lighting method, that is cheap and simple, however, depending on camera and lamp positioning, specular reflection can be a problem, but it can also be managed by finding another positioning in which the specular reflection is directed away from the camera sensor. It is also possible to use multiple lamps of this type at the same time to cover larger areas.

Another type of illumination that is widely used in the industry is coaxial illumination, in which the light emitted onto the sample has a direction coaxial to the central axis of the camera lens. This is made possible by positioning a light source at 90° to the camera axis in combination with a mirror, which reflects the light in a direction coaxial to the camera. This way it is possible to obtain images with fewer shadows, and to highlight specular surfaces which are positioned perpendicular to the camera, while surfaces with other angles appear darker in the image. These are therefore useful for inspection and measurement tasks of flat and shiny objects for example.

Ring light illumination is also very common, and consists of using a ring light around the camera lens, being a relatively simple and cheap lighting technique. The advantage of this type of lighting is the great decrease in the effect of shadows on the acquired image. However, as it is a direct lighting method, there is a risk of generating specular reflection, especially if the light source is LEDs without a diffuser over them.

When you want to avoid a surface with a specular characteristic being identified with high brightness and hotspots, a good solution is to use dome-type lighting. This type is known for creating a cloudy day effect, in which the entire surface is uniformly illuminated, without being directly illuminated, thus avoiding the formation of shadows. To obtain this type of lighting, a dome is used, usually with matt white surfaces, which allows indirect and uniform lighting of the sample. This type of illumination is widely used in the inspection of specular surfaces.

When the objective is the identification of small structures, elevations and depressions in the sample, the ideal solution is the use of darkfield illumination, which consists in illuminating the samples with a small angle between the light source and the surface to be analyzed. This way, the light rays are reflected away from the camera sensor, being reflected to the sensor only the light rays that encounter deviations in the surface. As such, these small structures are seen standing out from the rest of the surface, being useful for example in the inspection of texts embossed in a metallic surface.

When in the inspection task what you want to see are the external contours of the sample, the ideal illumination is done using the backlight technique, in which the sample is placed between the light source and the sensor, so that what is represented with more intensity in the image is the

outer region of the object, while it appears dark. With this technique it is possible to obtain an outline of the sample with a high contrast, which is a very interesting solution for measuring the external shape of samples. However, the light rays emitted by a backlight are not all parallel to each other, but instead are emitted diffusely in different directions. So if the sample is too close to the backlight, the so-called stray rays can illuminate the side of the sample instead of just its back, lowering the expected constrast. To improve the contrast produced by backlight illumination it is necessary to place the sample at a distance from the backlight.

When the analysis to be done is of very high precision and the contrast obtained by using an ordinary backlight is not sufficient even with the distance between the sample and the light source, one option is to use a telecentric back light, which consists of the combination of an lighting source with a telecentric lens. This type of lighting has the same working principle as telecentric camera lenses, so all the rays emitted by the light source are parallel to each other, extinguishing the decrease in contrast due to stray rays that happens in the common backlight. However, this solution is much more expensive. When selecting the illumination, depending on the application, it is interesting to select a source that emits a specific color, in order to highlight the sample from its background and then make an easier and more accurate analysis. For sensors that work with color, it is common to use lamps that emit white light, so that objects can be seen in their natural colors. However, when colors are not important to be identified, and a grayscale sensor is used, illuminating an object with light of the same color, will make it appear brighter and stand out in the image, and it is also an option to illuminate it with its complementary color, so that the object will appear dark. The decision of which of the two options to use is dependent on the color of the image background, and which option creates more contrast between the sample and its background.

In order to obtain coloured lights, it is common to resort to the solution of using filters in lamps that emit white light, so as to let through only specific wavelengths, and absorbing the others. However, there is also a special type of filter, called polarizers, which only allow the passage of photons that oscillate in a specific direction. Its most common use in the field of computer vision is using a polarizing filter in the light source, and another in the camera lens, having the 2 filters out of phase by 90º, making it possible to obtain images with reduction of hotspots on specular surfaces.

2.3 Image Processing

Computer vision is a topic that is rapidly evolving in the engineering field, and an increasing number of applications for this technique are being developed, as can be seen in [30, 31, 32, 33, 34], being applied for different purposes, as well as with the use of different techniques. This area of study is intended to enable machines to have an understanding of the physical environment, interpret it, and then be able to produce information about its state, emulating human vision, and assisting in decision-making and process control. Due to the great development of artificial intelligence, which has been increasing in recent years, computer vision applications have become

more and more extensive and complex, allowing computers to extract information from the environment and perform tasks that were previously performed or comprised exclusively by humans. The application of this technology is therefore broad, raging from detection and understanding of facial expressions [35], pedestrian detection for accident prevention against intelligent vehicles [36], and also automatic detection of piston chamber defects in engine blocks [37].

In order to obtain the desired information from a computer vision process, it is necessary to know how to process the images acquired by the system, namely the camera used in the setup. In general, cameras generate an image, which is a matrix, where the values of its elements represent the light intensity measured by each pixel of the camera sensor. The size of those matrices is equivalent to the number of pixels of the camera sensor, with each pixel representing the luminous intensity measured with an 8-bit value. In the case of a monochrome camera, the value 0 would represent the lack of light, i.e., black, while 255 would represent white, while in the case of a color camera, three 8-bit channels are produced simultaneously, each indicating the light intensity of its respective color spectrum. Commonly the channels produced are red, green, and blue, which are the primary colors of the light spectrum, that combined with different intensities are able to produce a large color spectrum.

With the image acquired, it is necessary to process the data obtained, with the aim of obtaining the desired characteristics and information of the object, environment or shape to be analyzed. With this objective, there are several algorithms to be used for different purposes, from options for noise reduction that interfere in the definition of features and measurements to be made, to others that find edges of objects, others for identification of textures, and even for image compression [38, 39].

2.3.1 Binary images

One of the most common operations in image processing is binarization. To do this, it is first necessary to be handling a single-channel image, which can be done by working individually on each color channel of the image, or by merging them to form a grayscale image, and then you can binarize the image in question. Binarization consists in converting all pixels in the image into values of 0 or 1, with the output value being defined by a threshold; if above the threshold a value of 1 is assigned, and if below it a value of 0 is assigned to the respective pixel, where in images 1 is usually represented by the color white, and 0 by black. This technique is very important for segmenting objects and image features, as well as for using other algorithms that work based on binarized images, such as some morphological operations, and edge recognition, which will be described later. In a general manner, considering a threshold of value S, applied to an image $I(i,j)$ binarization can be represented by (2.4).

$$B(i,j) = \begin{cases} 1 & \text{if } I(i,j) \geq S \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

The most common form of binarization is by implementing a global threshold defined by the user, so that values above and below this threshold have as output pixels of value 1 or 0. However, there are other options with more parameters to be defined, and still others in which the threshold has different values throughout the image. A first alternative for image binarizing is to use the Otsu's method [40], which has the objective of automatically selecting the value of the threshold to be applied, based on the histogram of the intensity of the pixels in a grayscale image. The method works in order to set a threshold value to have the lowest possible variance between pixels belonging to the same classes, 1 or 0, while having the highest possible variance between classes, thus achieving a better separation between pixels for segmentation than if an arbitrary threshold value were assigned. Although Otsu's method is an interesting option for optimizing the threshold value, as it tends to set a value that is closer to the class of higher intra-class variance when there is a large inter-class variance. Due to this, there are already improved versions of this algorithm, as seen in [41], where the threshold value is enhanced by analyzing the relationship between pixel grayscale value and the change of cumulative pixel number, for selecting the ratio of pixel gray level value to a certain cumulative pixel number as the adjusted threshold.

In certain cases, applying a global threshold may not be the best option. Considering an example in which an object is not evenly illuminated, it is possible that part of its body is classified as below the global threshold, and part above, so it would not be possible to segment the object completely. In order to overcome this limitation, there are segmentation methods that apply local threshold values, varying according to the gradients found in different sections of the image, thus improving the segmentation of a poorly lit object [42], as can be seen in the figure 2.11. Two different methods of applying adaptive thresholding are to use the average of groups of pixels to define the local threshold, or to use the Gaussian distribution function (2.8) to calculate the influence of neighboring pixels on the threshold's local value [43].

Previously it was discussed the possibility of using changes in the histogram to increase contrast and improve the image processing capacity, more specifically in the binarization of the image. Considering cases of implementation of computer vision systems for the identification of features in objects with specular reflection, and with curved surfaces, it may be difficult to accurately identify the boundaries of this object during binarization, due to the gradual change of illumination on its surface, causing the segmentation to occur on the object, instead of separating it from the environment as expected. An alternative way to improve this segmentation is by normalizing the grayscale image using the k-means algorithm, figure 2.12. In [45] a simple and efficient method to implement a k-means algorithm for image processing was developed. In [46] this algorithm was used to classify the gray gradients in the image, which lead to more precise binarization of a metallic structure with rounded shapes and smooth edges, in which the objective was to precisely dimension the gaps on the analyzed object, and the results of image processing can be seen in figure 2.13. For better understanding, the k-means algorithm performs as follows [47]:

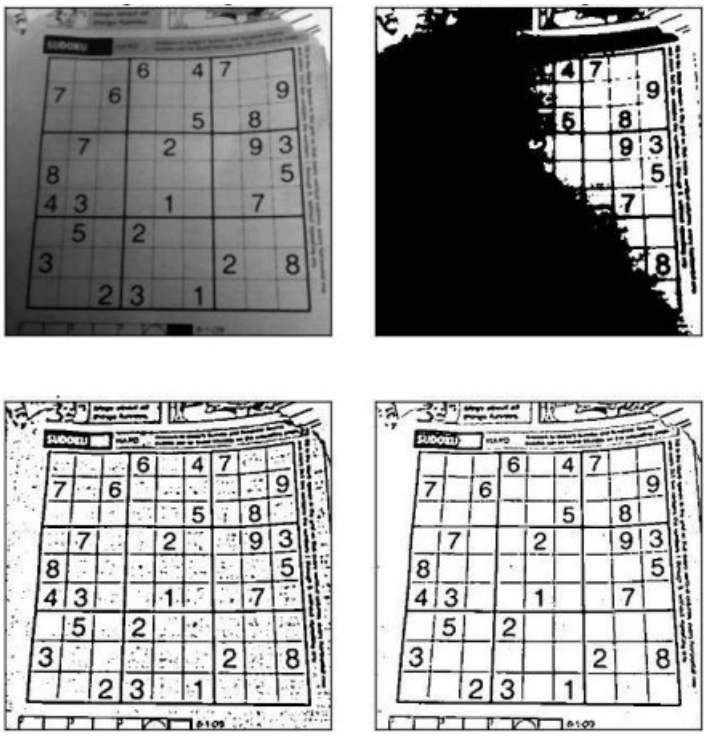

Figure 2.11: The upper left is the original grayscale image. The upper right is the image binarized by a global threshold of value t=127. The lower left is the image binarized by adaptive mean threshold. The lower right is the image binarized by adaptive Gaussian threshold [44].

1. Determine k number of centroids.
2. Randomly positions the k centroids among the dataset.
3. Classify the data as belonging to the group of the nearest centroid.
4. Calculate a new centroid for each group.
5. If the difference between the previous and the new centroid is above the defined threshold, repeat steps 3, and 4. Else, terminate the algorithm.

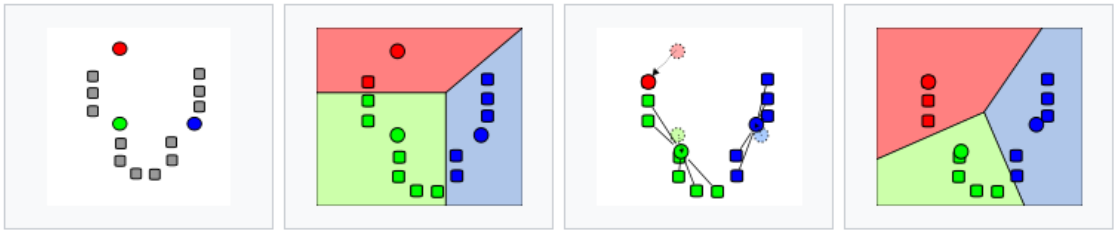

Figure 2.12: Demonstration of K-means algorithm [48].

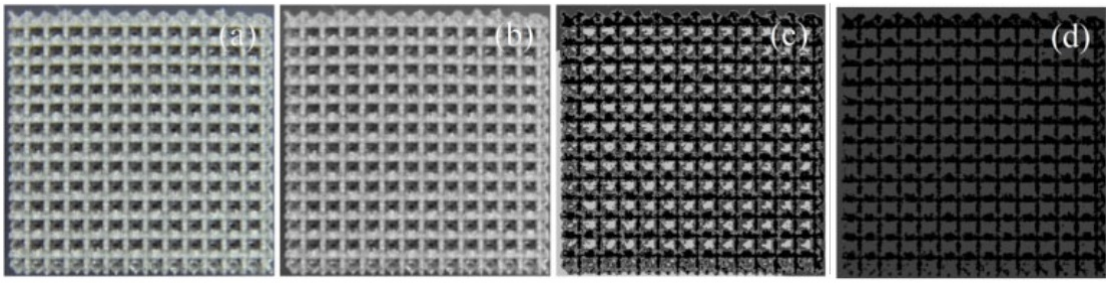

Figure 2.13: Image processing result, binarization after k-means clustering. a) Original image, b) Gray scaling, c) k-means clustering, d) binarization [46].

2.3.2 Morphological operation

Another different group of operations in image processing is morphological operations. This type of operations contains a wide range of options, sharing the characteristic that each pixel of the image is changed according to nearby pixels values, and the number of pixels to be considered in the neighborhood is a parameter controlled by the user. It can be defined whether the pixel in question will be altered considering only the ones in a close neighborhood, or whether it will also consider the more distant ones [49]. This type of operation is also widely used to detect objects and their features because its algorithms are specifically designed to handle shapes, since the formulation of these techniques is based on the theories of morphological mathematics. With its appropriate use it is possible to simplify images, eliminating some irrelevant shapes and maintaining essential forms. In general, these operations are applied to binary images, in which all pixels have a value of 1 or 0, however, it is also possible to apply them to grayscale images [50].

As mentioned previously, morphological operations are based on morphological mathematics, which is based on set theory. In morphological mathematics in image processing, the sets represent the shapes within the image to be analyzed, and in the case of binary image, the sets of pixels of value 0 and the sets of pixels of value 1, together form the complete shape of the image. Morphological mathematics can be applied to sets of any dimension in Euclidean space, however, when analyzing 2D, binary images, its elements are described in a two-dimensional Euclidean space. As the dimensions are increased, the images can turn into binary images that vary with time, or grayscale images, or binary solids [50].

Among the types of morphological operations, the two simplest and most common are dilation and erosion, and both work in an antagonistic manner; while the first promotes the expansion of the existing shapes in the images, the second reduces these shapes. Regarding dilation, its operation is based on the combination of sets, thus performing a vector sum to the values of the elements of the sets, the dimension of the element vectors being equal to the number of the Euclidean dimension of the sets of images. Thus, being A and B sets of a space of dimension N, with elements $a = (a_1, ..., a_n)$ and $b = (b_1, ..., b_n)$ of N coordinates, then the dilation of A by B is defined by the set of all possibilities of vector sum between elements of A and elements of B. Thus, the dilation of A by B is described by $A \oplus B$ and defined as (2.5), with E^N representing the dimension in Euclidean space [50]. The result of a dilation operation can be seen in figure 2.15.

$$A \oplus B = \{c \in E^N | c = a + b \quad \text{for some } a \in A \text{ and } b \in B\} \tag{2.5}$$

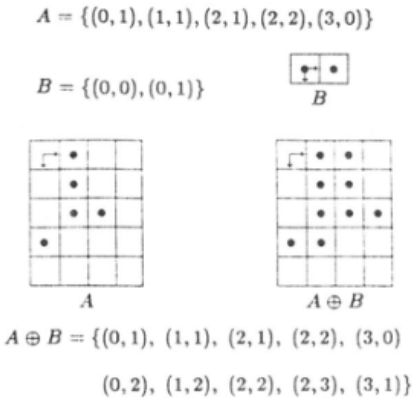

Figure 2.14: Example of the dilation operation functioning [50].

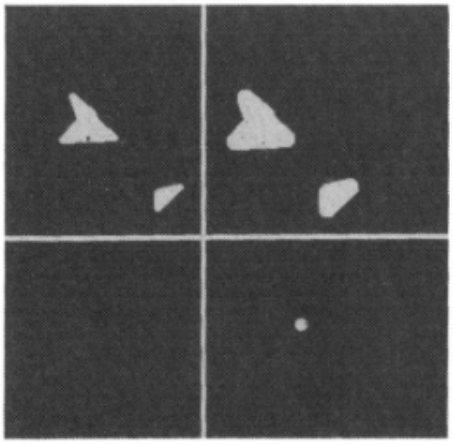

Figure 2.15: The upper left shows the input image consisting of two objects. The lower right shows the octagonal structuring element. The upper right shows the input image dilated by the octagonal structuring element [50].

Concerning the morphological operation erosion, it works in a manner to diminish shapes (figure 2.17, being then an operation contrary to dilate, so instead of performing a sum between elements of the sets, it performs a subtraction. Therefore, being A and B sets of dimension N in Euclidean space, the erosion of A by B is the set of elements x where $x + b \in A$ for every $b \in B$. The operation is then defined by (2.6).

$$A \ominus B = \{x \in E^N | \text{for every } b \in B, \text{ there exists an } a \in A \text{ such that} x = a - b\} \tag{2.6}$$

The mathematics that defines the two operations mentioned previously, were defined by two authors, the first, dilation, was defined by Minkowski in [51], and the erosion operation was defined later by Hadwiger in [52].

Although these operations can be used separately, and are useful by themselves, in practice it is more common to use them together, thus creating two other morphological operations, called opening and closing, figures 2.18 and 2.19, respectively. Concerning the opening operation firstly, its main use is to make small shapes that exist in the image disappear, by using the erosion operation, followed by dilation. This way the small bodies disappear with the first operation, and dilation is done with the purpose of returning the shapes in the image to their original size. Since only small bodies disappear and the large ones retain their original size, the opening is often used for noise reduction, as can be seen in [53, 54], where noise reduction through morphological operations was helpful in increasing the accuracy of edge detection in the images. For the closing

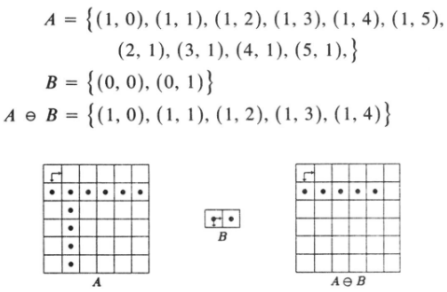

Figure 2.16: Example of the erosion operation functioning [50].

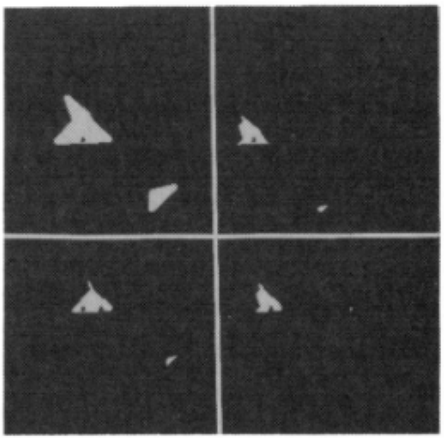

Figure 2.17: The upper left shows the input image consisting of two objects. The lower right shows the octagonal structuring element. The upper right shows the input image dilated by the octagonal structuring element [50].

operation, the goal is to fill small spaces in the images, first using the dilation operation, and then with the gaps no longer existing, the erosion operation is applied, so that the shapes regain their original size [53].

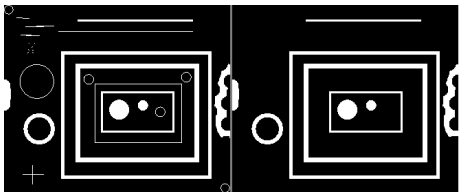

Figure 2.18: Example of the opening operation [49].

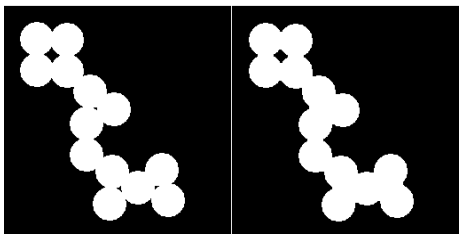

Figure 2.19: Example of the closing operation [49].

There are also other morphological operations that come from the basic operations dilation and erosion, namely skeletonize, which erodes the shapes towards the formation of a central line, without disregarding the existence of branches and gaps between shapes. In addition there are the operations top-hat and bottom-hat, the first of which performs an opening operation and then subtracts the result from the original image, thus increasing the contrast in a grayscale image and helping to separate lighter structures from darker ones. Regarding the bottom-hat, it does the opposite of the top-hat, it first performs a closing operation, and then subtracts this result from the original image, so it is possible to isolate pixels that are darker relative to their periphery [49].

2.3.3 Denoising

Noise is one of the standard problems to be observed in computer vision systems, and to avoid it there are several resources to be used, either hardware or software. There are cameras for image acquisition with different signal to noise ratios, however, naturally the quality of the camera varies proportionally to its price. For this reason it is necessary to try to reduce the noise level via image processing to cheapen the cost of the process, or even do it in conjunction with good hardware for better image quality. But there is also a balance to be made in noise processing, since the better and more complex the algorithm, the greater the time and demand for its processing, requiring more expensive hardware to accomplish the noise reduction task in less time.

For the sake of understanding, linear convolutions are based on masks of rectangular matrices of dimensions $L_1 \times L_2$, and an image of dimensions $M \times N$, which thus can have the convolution defined by the equation (2.7) [39].

$$g(i,j) = \sum_{l=0}^{M-1} \sum_{k=0}^{N-1} f(l,k)h(i-l,j-k) \tag{2.7}$$

One of the most common methods of noise reduction is the use of image smoothing operators, applying linear and non-linear filters, which aim to reduce and eliminate noise in the image by smoothing out the image irregularities. Linear filters are based on applying convolutional masks to the pixels, with the parameters to be changed being the level of influence of each pixel in the mask, which is dependent of the type of filter applied, and the size of the mask. In a linear filter of arithmetic mean, for example, the value of the central pixel of the mask will be equal to the mean value of the pixels present in the mask, while in the median linear filter, the central pixel of the mask will have the value of the median of the pixels present in the mask. One of the most common linear filters is the Gaussian filter, in which the coefficient of the mask that multiplies the pixels has its values calculated analytically considering the Gaussian distribution function (2.8), being a filter particularly effective in reducing Gaussian noise [39].

$$d(i,j) = k \cdot e^{-\frac{i^2+j^2}{2\sigma^2}} \tag{2.8}$$

Although smoothing techniques are very useful and have their proper applicability, they have a drawback, which is the non-preservation of original shapes. At the same time that outlier pixels are neutralized in the convolution process, the values of the noiseless pixels are also changed according to the values of the pixels in their neighborhood. This makes it impossible to precisely define the image characteristics, being necessary to appeal to different forms of denoising for precision scenarios [39, 53].

One of the possibilities of noise suppression without harming the original size of the image shapes has already been mentioned above, which is the use of the morphological operation opening. An example of application can be seen in [53], where the author intends to obtain the edges of agricultural products accurately, thus, common linear filters for denoising are avoided, so the opening operation is used, which in addition to noise reduction, it preserves the original edges of

the objects, and after that, the extraction of the intended features of the image can be well performed. The same technique can be observed being applied similarly in [46], where prior to using an edge recognition algorithm to measure the pores in a metallic structure, the opening operation for denoising was first used.

2.3.4 Histogram Modification

Histogram modification is a very useful and frequently used tool in image processing, which allows manipulation of the intensity distribution of the pixels in an image. Using this tool it is possible to strengthen or suppress features in an image, as well as correct image defects due to overexposure, and also improve contrast. The histogram of an image consists of a vector whose values are the frequency of each pixel intensity in the image, and is a global statistic of the picture, which can be the same for different images, so it is not data capable of defining a single image or its shapes. In figure 2.20 it is possible to see two images and their respective histograms [38].

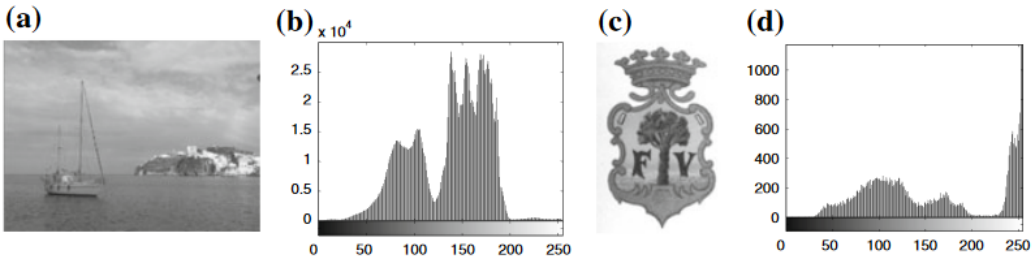

Figure 2.20: Images **a** and **c** with the respective histograms **b** and **d** [38].

There are different methods of making changes to the image histograms, such as equalization, matching, and contrast stretching. The first aims to equalize the frequency of each intensity level, increasing the intensity spectrum and therefore increasing the overall contrast of the image, thereby increasing contrast in areas where it was low. The second has as its main purpose the alteration of a histogram in order to shape it as desired, and is used in some calibration techniques. The last method, on the other hand, consists in just broadening the intensity spectrum to be used, thus increasing the contrast, but without changing the value of the frequencies, For example, on an image I(i,j), its output G(i,j) in a histogram stretching operation that intends to use the full 8-bit pixel intensity spectrum, will be defined by the function (2.9).

$$G(i,j) = \frac{I(i,j) - Min(I(i,j))}{Max(I(i,j)) - Min(I(i,j))} \cdot 255 \tag{2.9}$$

2.3.5 Feature/edge detection

In computer vision, one of the most important tasks to be performed is edge detection, since most applications are intended for body recognition and/or measurements. This task is important to acquire the position of features of the objects to be identified by the vision system, and thus be able to proceed to further analysis, such as measuring the diameter of a milling tool to be

inspected [29], the conformity or not of a plate flatness [55], or even recognition of shapes in magnetic resonance images in the medical field [56].

Since the operation of the definition of edges is a primordial task in computer vision systems, even today many new studies are still being done on the subject, trying not only to develop new methods, but also to improve the existing ones and to combine them with different image processing operations in order to increase the accuracy and efficiency of the detection of edges. Like any other image processing operation, it is also necessary to take into consideration the processing time required in each type of algorithm to be used, considering that real applications, in general, have time constraints to obtain an analysis result. So, a trade-off must be made between the complexity of the method, the obtained result level and the processing time. In light of the time requirement, it is also common to find studies that aim to decrease the processing time of edge detection algorithms.

The edge detection algorithms assign values to pixels, defining them as belonging to edges or not. However the most common methods, which are "local edge detectors", do not provide additional information that allows the pixels to be connected to each other, i.e., the analysis is done individually to the pixels, and it is possible that there are fails along the length of the edges defined. This happens because the local edge detectors classification method is based on calculating local variations of pixel intensity only, and the images are not a perfect representation of reality, since it is susceptible to noise as mentioned previously. This causes intensity gradients between objects and their backgrounds to not have a perfect transition and easy identification, having in reality transitions with oscillating variation values, as well as discontinuities and some abrupt random intensity variations between some of the pixels. In an ideal scenario, one would want the intensity variation between object pixels and their background to be a step or line discontinuity (figure 2.22), with a perfect difference between the intensity values of the pixels belonging to the object to be identified, and those of the background; however, the most commonly encountered variation profile can be seen in the figure 2.21. Two other factors that cause an oscillating profile of intensity variation of the pixels are the low resolution of the image, which causes the mixing of light intensity of different points in space in the same pixel, difficulting the exact perception of the environment, and the possible reflections of light on the shapes of objects, causing these shapes to be more or less bright in the image, making them stand out, and be recognized as edges, interfering in a perfect identification of the objects, which can also be seen in the figure 2.21.

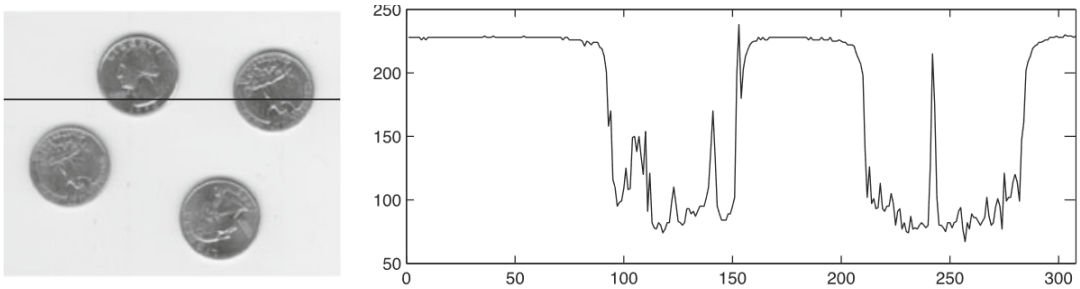

Figure 2.21: Profile of gray levels for an image line [39].

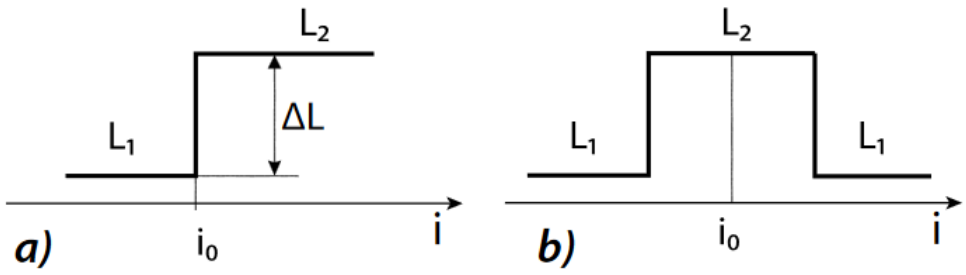

Figure 2.22: Ideal discontinuities types. a) Step b)Line [39].

Local edge detector operators, in general, perform two operations simultaneously. Since their purpose is to detect high-frequency variations in the image, i.e., high pixel intensity gradients, it becomes necessary to reduce the high-frequency noise, to be able to avoid detecting noise as edges. According to this, one of the processing operations consists of image smoothing and denoising, with the second processing operation being the detection of the intensity gradients themselves. However, depending on the expected degree of accuracy of the analysis to be performed, the smoothing effect of the edge detection algorithms should be well analyzed when choosing which one to use for the intended application, so as not to lose accuracy as would happen in an analogous manner to the denoising algorithms cited previously such as the Gaussian filter for instance. In summary, and considering other points already mentioned that are important for computer vision, the choice of an edge detection algorithm should be based on three aspects:

- Reliability, of the physical-mathematical model;
- Efficiency, in terms of computational processing;
- Accuracy, of the achieved result.

One of the simplest ways to detect edges is by means of a symmetric difference filter ${}^{S}h_{R}$. This method is based on the approximate calculation of the symmetric intensity gradient of the pixel value, obtaining the vertical and horizontal gradient separately, thus resulting in one filter for detecting horizontal edges and another for vertical ones. In the case of the horizontal axis, the gradient is calculated based on the partial derivation in order to x, which in this case is done by the following equation (2.10):

$$f_x(x,y) = \frac{\partial f(x,y)}{\partial x} = \lim \Delta x \to 0 \frac{\partial f(x+\Delta x,y) - f(x-\Delta x,y)}{2\Delta x} \approx \frac{1}{2}[f(x+\Delta x,y) - f(x-\Delta x,y)] \tag{2.10}$$

The symmetric difference filter for the vertical gradient is obtained in a similar manner, however, in this case the pixel intensity function is derived in order to Y. Thus, the symmetric difference for rows and columns is defined as follows:

$$f_x(i,j) = \frac{1}{2}[f(i,j+1) - f(i,j-1)] \tag{2.11}$$

$$f_y(i,j) = \frac{1}{2}[f(i+1,j) - f(i-1,j)] \tag{2.12}$$

Once defined, these approximate gradient calculations can be implemented as linear filters by means of the convolution masks (2.13) (2.14), where the indexes R and C identify the mask for rows and columns respectively.

$${}^{s}h_R = \frac{1}{2}\begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \tag{2.13}$$

$${}^{s}h_C = \frac{1}{2}\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \tag{2.14}$$

In order to demonstrate that this filter contains the two previously mentioned operations of smoothing and edge detection, it can be shown that it operates as a low-pass filter combined with an edge-enhancing filter (2.15), with B_R being an elementary smoothing filter, which is an average between two pixels, and ${}^{-}h_R$ highlighting the difference between two pixels by calculating the difference between them, being an asymmetrical backward difference filter. The same demonstration applies similarly to the column's filter. The symbol "•" represents the pixel being processed.

$${}^{s}h_R = {}^{-}h_R \star^1 B_R = \begin{bmatrix} -1_{\bullet} & 1 \end{bmatrix} \star \frac{1}{2}\begin{bmatrix} 1 & 1_{\bullet} \end{bmatrix} = \frac{1}{2}\begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \tag{2.15}$$

And the application of the convolution masks is represented by the equation (2.16), which results in a gradient image 2.23. The column's mask is applied in a similar manner.

$$g_R(i,j) = f(i,j) \star h_R(i,j) = \sum_{l=-1}^{+1} f(i,l) \cdot h_R(l) \tag{2.16}$$

Figure 2.23: Ege detection with filter gradient approximated to finite differences: a) Original image; b) Horizontal differences; c) Vertical differences.

Since each filter only calculates the gradient of the pixels in the direction for which it was developed, i.e., vertical or horizontal, their application results in two images, however, it is intended

to produce a single image of the gradients, which contains all the edges of the image. For this purpose, the magnitude of the pixel gradients is calculated using the equation (2.17), or (2.18), which is a reasonable approximation of the magnitude within image processing, requiring lower computational complexity. The θ direction of this gradient can be calculated by the equation (2.19), which is the direction perpendicular to the local edge, i.e., the direction of maximum intensity variation, relatively the horizontal direction. After obtaining a single gradient image representing existing edges, it is convenient to apply a threshold on the gradient magnitude, to keep only the edges that are of interest, which is one more method of reducing the noise identified as edges.

$$|\nabla f| = \sqrt{g_R^2 + g_C^2} \tag{2.17}$$

$$|\nabla f| \cong |g_R| = |g_C| \tag{2.18}$$

$$\theta(x,y) = arctan\left(\frac{g_C}{g_R}\right) \tag{2.19}$$

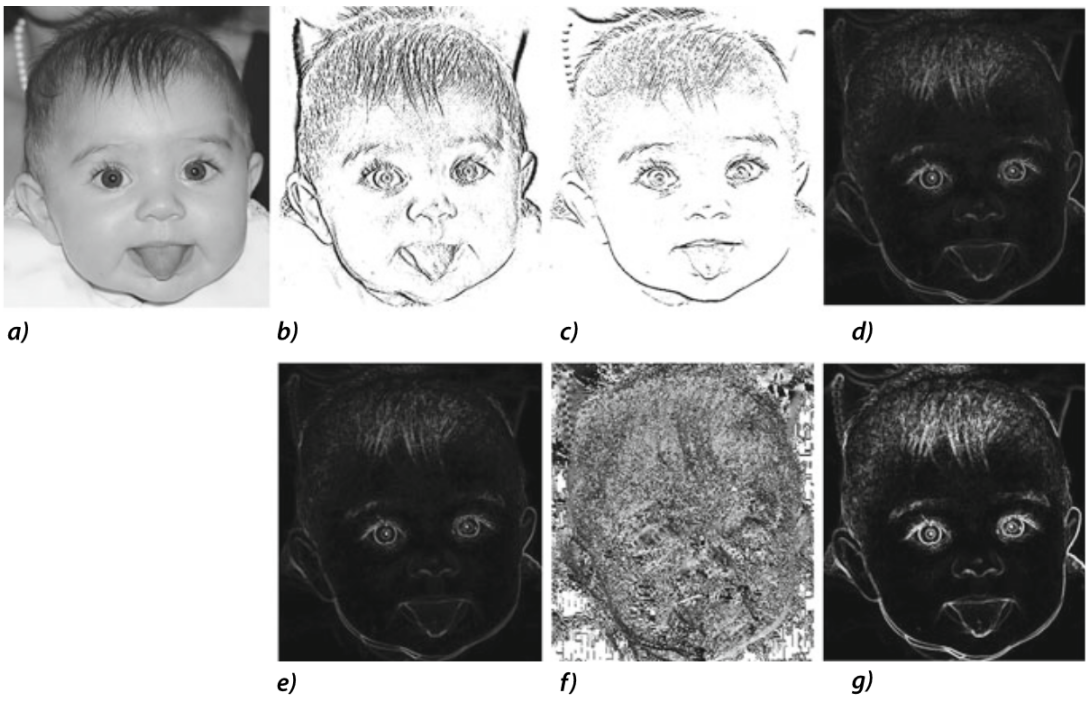

Figure 2.24: Example of an image gradient calculated with symmetric difference filters: a) Original image; b) Image of the horizontal gradient calculated with equation (2.11); c) Image of the vertical gradient calculated with equation (2.12); d) Gradient magnitude calculated with equation (2.17); e) Gradient magnitude calculated with equation (2.18); f) Gradient direction θ calculated with equation (2.19); g) Gradient magnitude of image d after applying an appropriate threshold to highlight the significant edges [39].

A drawback to be observed with this method is that the calculated gradients do not exactly represent the gradient of the pixel under analysis, there is a 0,5 pixel offset between the location of the processed pixel and the gradient real coordinates, thus, the identified edges actually have a positioning error of 0.5 pixels. However, this study provides the theoretical basis for understanding other local edge detectors, which are based on this formulation, although each has its own specific characteristics, as will be seen below.

A first alternative of local edge detectors is the Roberts Operator, whose main differential is the analysis of symmetrical differences on the diagonal, as it is possible to see in the definition (2.20) (2.21). This way the direction of the gradient of greater magnitude with respect to the horizontal axis is defined by (2.22), and the convolution masks by (2.23), which are applied in the same way as the symmetrical difference gradients.

$$f_x(i,j) = f(i,j) - f(i+1,j+1) \tag{2.20}$$

$$f_y(i,j) = f(i+1,j) - f(i,j+1) \tag{2.21}$$

$$\theta(i,j) = \frac{\pi}{4} + arctan\left(\frac{f_y}{f_x}\right) \tag{2.22}$$

$$h_R = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} ; h_C = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{2.23}$$

Robert's operator is one of the simplest local edge detectors, and analyzing the result of its application, it is possible to see a slight qualitative improvement in the figure 2.25, when compared to the result of applying the symmetrical difference gradient method in the figure 2.23. However, in comparison to other common differential edge detectors, this method recognizes fewer edges than the Canny, Laplacian of Gaussian, or Prewitt methods [57], while it has a longer processing time than the Prewitt and Laplacian of Gaussian methods [56].

Figure 2.25: Edge detection with Roberts filter: a) Original image; b) Result applying the h_R mask to extract the horizontal and vertical components c) by applying the h_C convolution mask; d) Final result by applying the gradient magnitude (2.17); e) Possible edge points extracted from the gradient image d) if the magnitude value exceeds a threshold [39].

Another common edge detection operator is the Sobel operator, and it was developed with the object of mitigating two problems of the previously mentioned algorithms, which are the creation of edges at least two pixels wide, and the high sensitivity to noise.

In order to decrease the sensitivity to noise, the Sobel operator has a 3x3 convolution mask (2.24), differently from the smaller masks seen before (2.13), (2.14), (2.23), so there is a greater smoothing effect in the application of the operator. Since it is a 3x3 convolution mask, the method verifies the gradient not only on the analyzed axis, but also on the rows (or columns) above and below the one being processed, and averages the gradients. However, a higher weight K is assigned to the pixels belonging to the row (or column) being processed, in the case of Sobel K = 2. It is important to emphasize the symmetry of the mask, in order to guarantee a gradient of 0 in case all pixels inside the mask contain the same intensity value. The application of the operator is similar to the previous ones, as can be seen in equations (2.25) (2.26).

$$h_R = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} ; h_C = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{2.24}$$

$$g_R(i,j) = f(i,j) \star h_R(i,j) = \sum_{l=-1}^{+1} \sum_{l=-1}^{+1} f(i+l, j+k) h_R(l,k) \tag{2.25}$$

$$g_C(i,j) = f(i,j) \star h_C(i,j) = \sum_{l=-1}^{+1} \sum_{l=-1}^{+1} f(i+l, j+k) h_C(l,k) \tag{2.26}$$

In [58] an improved form of Sobel's operator is observed, in which the gradient is analyzed not only with respect to rows and columns, but also in two more positions, at 45º and 135º. This achieves better noise suppression, as well as better edge detection, by using two more convolution masks (2.27), and the magnitude of the gradient is calculated by (2.28). Due to the increase of calculations to be done, this method comes at the cost of an increase in computational processing requirements.

$$h_45 = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix} ; h_135 = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix} \tag{2.27}$$

$$|\nabla f| = \sqrt{g_R^2 + g_C^2 + g_{45}^2 + g_{145}^2} \tag{2.28}$$

Another very common operator is the Prewitt Operator, which works in a very similar way to the Sobel Operator, the only difference being that it has K = 1 (2.29).

$$h_R = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} ; h_C = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \tag{2.29}$$

As the last two operators mentioned, there are others of similar function, such as Frei & Chen for example, which are other options to be tested and have their results compared in order to find the best operator for each specific application

Currently, the best and most used differential edge detection operator is the Canny operator [59, 60]. Comparing this method with those described previously, it has three distinguishing characteristics, which are good performance in edge detection, single response for the same edge, and high accuracy in edge positioning [39, 61]. This method of edge detection is highly sensitive to noise, and therefore requires the use of denoising prior to the application of its recognition algorithm, as is usual in differential operators. In its standard formulation, this denoising is done by applying the Gaussian smoothing filter [39]. After smoothing the image, derivative masks (2.30) are applied to calculate the gradient of the image.

$$h_R = \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} ; h_C = \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix} \tag{2.30}$$

The following step consists of the calculation of the magnitude and direction of the obtained gradients after the application of the convolution mask. Those calculations are proceeded by means of the same equations utilized on the differential symmetric difference method (2.17) (2.19). After the definition of the magnitude of the gradients, and their respective directions, the next step is to pass the image through the process of non-maximum suppression, which aims to refine the edges found, by maintaining only the gradient peaks. This is achieved by assigning the value 0 to all pixels that are in the the θ direction of the gradient of the processed pixel which are not maximum gradient peaks, thus obtaining edges of only one pixel, according to the feature proposed by the method of unique identification of each edge. After that, the last step of Canny's algorithm consists in applying a threshold, in which only magnitudes above the defined threshold are kept in the final image, being possible to decrease the false edges identified due to the noise that persists in the image even after its initial processing [39].

A limitation of the Canny method is the non-detection of some edges that are very small, due to the application of the Gaussian filter. In [57] a method is presented that consists of transforming the ramps of consecutive change of pixel intensity values, into steps, resulting in detection of 0.95% more edges than the traditional method on average, and further demonstrating that the Canny method is able to detect more edges than the Roberts, Prewitt, and Sobel methods seen earlier.

Also in [53] one can see another adaptation in the use of the Canny method. Although its original process has the application of the Gaussian filter, and thresholding, it remains considerably susceptible to noise interference, as well as interference caused by lighting in the environment from which the image acquisition was done. With these aspects in consideration, the author proposes the pre-treatment of the images with the opening and closing morphological operations for noise reduction, instead of using the Gaussian filter of the original algorithm. This way, at the same time that denoising occurs, the method of morphological operations conserves the edges better than when applying the Gaussian filter, which has the consequence of modifying the value

of pixels that are not noise. Another modification proposed in the same work is the calculation of the gradient of the pixels also in the 45º and 135º directions, in order to improve the characterization of the edge direction. The last proposed change is the application of an adaptive threshold, more specifically dual detection threshold, as compared to the single threshold applied in the traditional Canny operator. As a result of the proposed method, a less sensitive algorithm to noise was obtained, with better edge detection, including better connectivity between the identified edges as can be seen in figure 2.26.

Figure 2.26: Results of the algorithm proposed in [53]. a) Lena artwork b) Traditional Canny algorithm c) Improved algorithm.

With the frequent use of computer vision systems for production inspection, it has been necessary to use more advanced resources to be able to inspect objects of a wide variety of shapes and dimensions. Considering the use of optical inspection for visualization of small dimensions objects, one solution is to approach the camera closer to the object to be inspected. However, this solution may not be enough in all situations, thus being necessary to appeal to sub-pixel processing algorithms, virtually increasing the resolution of the image produced. This increase in resolution can also be useful in the calibration process between the two cameras of a stereo vision system [62], to obtain a better correlation of their images. The use of sub-pixel can be also applied directly to the measurement and feature recognition phase of stereo vision systems as can be seen in [55, 63]. Regarding 2D vision systems, this type of algorithm can also be applied in order to obtain more accurate results, as done in [64], and also in [65], where this feature was used to measure a cutting tool to the order of micrometers, with an error of less than one micrometer.

Among the uses of sub-pixel edge detection algorithms, there are three main methods, fitting, interpolation, and moment methods. Of the sub-pixel methods, the moment method is the most used. Zernike moments method have the characteristic of being less sensitive to noise as the differential methods seen previously, by being based on a integral operation. Among moment methods, the spatial moments was proposed firstly, however, this method has redundancy of information due to lack of orthogonality, which does not occur on Zernike moments method [66].

Zernike moments consist of a set of orthogonal complex polynomials defined under a circle that is useful in image processing and pattern recognition tasks. This is a robust method with respect to noise and geometric transformations, seen as appropriate for object recognition operations, even under varying illumination circumstances. The Zernike step model is presented on

figure 2.27, and the Zernike moment is defined by equation (2.31), where n is a positive integer or zero, m is a positive or negative integer, with $n-|m|$ being even, and $|m| \leq n$, r is equivalent to d from figure 2.27, which is the distance between the center of the circle and the foot point (x,y), thus $r = \sqrt{x^2+y^2}$, with $x,y = [-1,1]$. $V_{nm}(r,\theta)$ is the Zernike complex polynomial, with $V_{nm}(r,\theta) = R_n m(r) e^{jm\theta}$, and $V^*_{nm}(r,\theta)$ being its complex conjugate. $R_{nm}(r)$ is the edge foot point radial polynomial, and is defined by equation (2.32) [67].

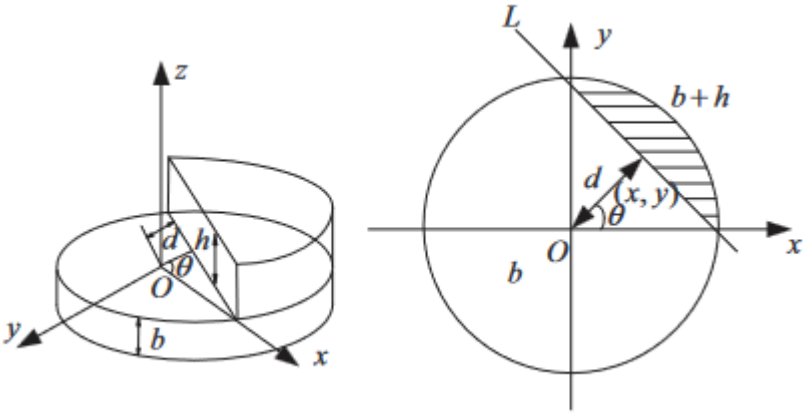

Figure 2.27: Zernike step model.

$$\begin{aligned} B_{z,nm} &= \frac{n+1}{\pi} \int_0^1 \int_0^{2\pi} f(r,\theta) V^*_{nm}(r,\theta) r \, dr \, d\theta \\ &= \frac{n+1}{\pi} \int_0^1 \int_0^{2\pi} f(r,\theta) R_{nm}(r) e^{-jm\theta} r \, dr \, d\theta \end{aligned} \tag{2.31}$$

$$R_{nm}(r) = \sum_{s=0}^{\frac{n-|m|}{2}} (-1)^s \frac{(n-s)!}{s! \left(\frac{n+|m|}{2} - s\right)! \left(\frac{n-|m|}{2} - s\right)!} r^{n-2s} \tag{2.32}$$

An important property of Zernike moments is its rotational invariance. Assuming a moment $B'_{z,nm}$ of order n and repetition m, associated to the function $f'(x,y)$, obtained from the rotation of the image $f(x,y)$ by an angle α with respect to the x-axis, by the equation (2.33), with $\theta' = \theta - \alpha$ it is possible to realize that the module of the Zernike moment remain unchanged, occurring only a change on the phase angle.

$$\begin{aligned} B'_{z,nm} &= \frac{2(n+1)}{\pi} \int_0^1 \int_0^{2\pi} f(r,\theta') R_{nm}(r) e^{-jm(\theta'+\alpha)} r \, dr \, d\theta' \\ &= [\frac{2(n+1)}{\pi} \int_0^1 \int_0^{2\pi} f(r,\theta') R_{nm}(r) e^{-jm\theta'} r \, dr \, d\theta'] \cdot e^{-jm\alpha} \\ &= B_{z,nm} e^{-jm\alpha} \end{aligned} \tag{2.33}$$

In edge detection there are three main Zernike moments that are used, B_{00}, B_{11}, B_{20}, and their

respective integral functions are $V_{00} = 1, V_{11} = x + jy, V_{20} = 2x^2 + 2y^2 - 1$. And the relation between those moments before and after rotation are:

$$B'_{00} = B_{00},\ B'_{11} = B_{11}e^{j\alpha}, B'_{20} = B_{20}$$

With a rotation *alpha* in which the edge to be identified is positioned parallel to the Y-axis, the imaginary part of the moment B11 then becomes zero

$$Im[B'_{11}] = sin(\alpha)\ Re[B_{11}] - cos(\alpha)\ Im[B_{11}] = 0$$

Where the real and imaginary parts of B'_{11} are the same ones of B_{11}, thus the rotation angle of the edge can be calculated by the equation (2.34).

$$\alpha = arctan\left(\frac{Im[B_{11}]}{Re[B_{11}]}\right) \tag{2.34}$$

Calculating the rotated Zernike moments according to equation (2.31), considering the model of the figure 2.27, the following results are obtained:

$$B'_{00} = b\pi + \frac{b\pi}{2} - h\ arcsin(d) - hd\sqrt{1-d^2}\ ;\ B'_{11} = \frac{2h(1-d^2)^{\frac{3}{2}}}{3}\ ;\ B'_{20} = \frac{2hd(1-d^2)^{\frac{3}{2}}}{3}$$

Therefore, according to the equation (2.31), the distance between the center point of the circle, and the edge, can be calculated by equation (2.35). This result also allows calculating the step h using equation (2.36).

$$d = \frac{B_{20}}{B'_{21}} \tag{2.35}$$

$$h = \frac{3B'_{11}}{2(1-d^2)^{\frac{3}{2}}} \tag{2.36}$$

With the value of d defined, the sub-pixel position of the edge on the image can be calculated by equation (2.37).

$$\begin{bmatrix} x_s \\ y_s \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + d \cdot \begin{bmatrix} cos\ \alpha \\ sen\ \alpha \end{bmatrix} \tag{2.37}$$

Although the calculation of Zernike moments is a powerful tool in the precise identification of object characteristics in images, it requires considerable computational power. However, in [68] an algorithm is presented that aims to reduce the required computational power, in order to reduce the mathematical operations to be performed in the calculation of moments

There is yet another type of similar orthogonal polynomials, which are the pseudo Zernike polynomials, which are used in image processing in the same way as Zernike moments. The only

difference being the composition of the polynomials as can be seen in equation (2.38), and not requiring that $n-|m|$ have an even result [69].

$$R_{nm}(r) = \sum_{s=0}^{n-|m|} (-1)^s \frac{(2n+1-s)!}{s!(n+|m|+1-s)!(n-|m|-s)!} \tag{2.38}$$

The advantage of using pseudo Zernike moments is that it is a method that can be even less sensitive to noise in feature recognition, but comes with a higher computational cost than classic Zernike moments [70]. In [71] an algorithm for edge detection at the sub-pixel level using pseudo Zernike moments is proposed, in which the author tested his algorithm on a standard image with straight and curved lines in which he added Gaussian noise to also analyze the performance under noisy image condition. It obtained an accuracy of 0.07 pixels for straight lines, and 0.1 pixels for curved lines, being a higher accuracy than the results obtained by applying on the same image spatial moments algorithms, and classical Zernike moments.

Chapter 3

Proposed Solution

This chapter will describe the proposals and solutions developed throughout the work, as well as the justification of the decisions made, and the development process itself. In addition, initial decisions that underwent changes will also be exposed.

In the first part, the decisions made regarding the hardware of the vision system will be discussed, such as the components selected, and the composition of the setup.

Then, the software solutions for the measurements of the plate structures in question will be presented. Two softwares will be presented, one for measuring the rollover profile on the edges of the plates, and another for measuring the circularity of the slot bottom diameter, which does together the measurement of the width of the stator teeth, for analysis of the accuracy of the slot punching operation. After that, the methods for setting parameters for image processing functions will be shown, such as interpolation types for the geometric operations performed and threshold values of Canny operations. At the end of the chapter, the proposed mechanical solution for inspection automation through the vision system will be presented.

3.1 Computer Vision Setup

3.1.1 Sensor

One of the objectives of this work is not only to evaluate the feasibility of a stator laminations inspection system using artificial vision, but also to assess the feasibility of this system using the company's own products. Thus an initial constraint of the selection of the sensor to be chosen, is the catalogue of sensors from MVISIA.

Among the options that MVISIA has, there are three models available. Between these models, two are similar to each other, being these the MV-SENS-100, and the MV-SENS-300, that consist basically in sensors with availability of coupling of different lenses. The other option is the model MV-ESOS, a different solution from the others because at the same time that it has the sensor, it has a computer processing unit to process the acquired image and communicate with another component of a system.

Specifically about the MV-SENS-100 (figure 3.1), this is a sensor of small dimensions, more specifically 110 mm x 45 mm x 45 mm, and lightweight, of aluminium housing, with a mass of 150g. About the technical specifications regarding image acquisition, it has a colour sensor of 1920 x 1080 (2MP) resolution, with an acquisition rate of up to 30 frames per second. There is also the option of this product with a monochrome sensor, with a resolution of 1280 x 720 (1MP) with an acquisition rate of up to 60 frames per second. The acquisition rate is dependent on the image resolution, so the higher the resolution, the lower the image acquisition rate. About the type of lens coupling, this model has support for S-mount type lenses or M12 thread. Regarding the trigger for image acquisition, it can be triggered via software [72].

Figure 3.1: MV-SENS-100 [72].

Regarding the MV-ESOS vision system (figure 3.2), as said before, this is not only a sensor, but it also has an integrated central processing unit (CPU) which is able to run image processing programs, having been especially designed to have embedded the software product developed by MVISIA itself, having then the advantage of, in a compact manner and in a single place, contain these two components of artificial vision systems. About the technical specifications of this model, it has an antenna for wireless communication, having support for Wi-Fi 2.4 Ghz connection. Due to its greater complexity, its size and weight are higher compared to the model presented previously. The dimensions are 124 mm x 174 mm x 85 mm, and the gross weight is 1.15 kg, with the housing made of aluminium, with some details made of polycarbonate. As for the image acquisition specifications, the sensor has both colour and monochrome options. As for the resolution, this is 1600 x 1200 (2MP), with an image acquisition rate of 40 frames per second. As with the previous model, if a lower resolution is used, it is also possible to increase the acquisition rate, reaching 100 frames per second at 800 x 600 (0.5 MP) resolution. About the type of lens coupling, this model has support for CS-mount type lenses. Regarding the trigger for image acquisition, it can be triggered via software and via 12 24Vcc signal [72].

The newest sensor from MVISIA is the MV-SENS-300 model (figure 3.3), having the best qualities in image acquisition. Similar to the MV-SENS-100, this is a sensor only model, having

Figure 3.2: MV-ESOS [72].

small dimensions and weight. This model is the smallest of all, having the dimensions 48.2 mm x 29 mm x 29 mm, and weighing less than 80 grams. As for the image acquisition specifications, the standard sensor has a resolution of 1440 x 1080 (1.5 MP), which can be changed according to the selection of the area of interest. However, in this model there is the possibility of using different sensors, with options up to 5472 x 3648 (20 MP), this being the CMOS type. As there is a relationship between resolution and image acquisition rate, in the standard model the rate is up to 227 frames per second, however, in the higher resolution option, the acquisition rate is 5 frames per second. About the type of lens coupling, this model has support for C-mount type lenses. Regarding the trigger for image acquisition, it can be triggered via software, hardware or even free-run [72].

Figure 3.3: MV-SENS-300 [72].

Taking into account the intention of validating the project with WEG products, it was decided to use the best product available in terms of image acquisition quality, which was the MV-SENS-

300. Other reasons for the selection of this model was its small size, making it convenient to have as small a component as possible in the system, and the flexibility that it has regarding resolution, since it can receive a wide range of sensor resolutions.

The methodology used to define the resolution of the sensor to be selected consisted firstly in knowing the dimensional tolerance of the structures to be measured, and ideally a pixel should be equivalent, in millimetres, to a dimension smaller than half the tolerance. To know what would be the sensor's pixel-millimeter relation, it is also necessary to define the field of view, being then necessary to acquire this second input for the sensor definition. With these two information, it is possible to define the sensor resolution, calculating if the pixel millimeter ratio is within the desired through the formula presented on chapter 2, on (2.3).

Firstly, the dimensional tolerances were obtained by analysing the company's technical documents on the production of the stator laminations, which contained all the dimensional and geometric information on this component.

The field of view was then defined. The objective would be to create a system that could perform a dimensional and geometric analysis of all the plates produced in the company, since WEG's Portugal unit, until now, produces motors with shaft heights of up to 900 mm. So, for sizing, considering the largest field of view, the plates of the motors with the greatest shaft height were considered, which have an outer diameter of approximately 1500 mm, and an internal diameter variable according to each project, but conservatively an internal diameter of approximately 900 mm was considered. With these dimensions, a field of view of 300 mm was defined in the radial direction of the lamination, with the objective of being able to view both the internal and the external diameter in the same image. In the plate tangential direction, the field of view dimension is 450 mm due to the sensor proportion being 1.5 times larger in one direction than the other, as previously described in the MV-SENS-300 technical specifications. The resulting field of view can be seen in the figure 3.4.

However, for smaller laminations, it will be an exaggerated field of view as can be seen in the figure 3.5, though it becomes possible to move the camera closer to the lamination, decreasing the field of view, and increasing the resolution, according to the formula cited previously (2.3).

After defining the field of view and first selecting the maximum resolution, a pixel per millimeter ratio of 12.16 is obtained, and the inverse, millimeter per pixel of 0.082, which is considerably greater than the dimensional tolerances practiced. Thus, is not possible even by the maximum resolution to make measurements with the desired precision at the pixel level, which will require the use of sub-pixel techniques in the development of computer vision software, using the use of the maximum resolution available. Another factor considered in the choice of maximum resolution, is that there is not a significant price difference between a 20MP setup and others of considerably lower resolution, such as 12 MP [73].

3.1.2 Lens

For the definition of the lens to be used, several parameters were taken into consideration. Since the field of view is large, it is not convenient to use a telecentric lens, and due to its high cost,

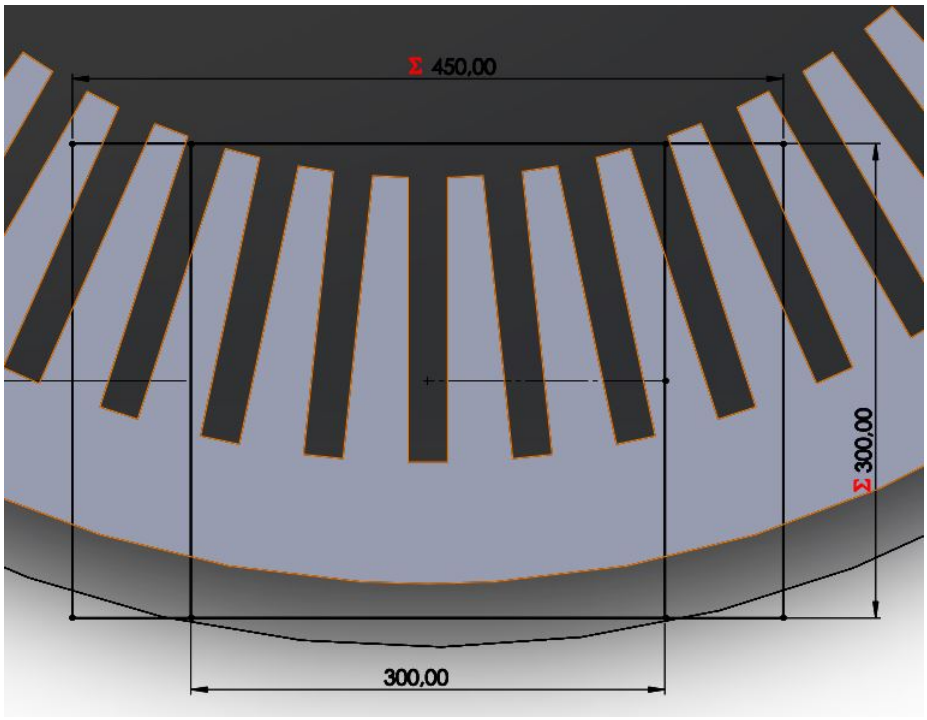

Figure 3.4: Field of view of frame 900 stator lamination.

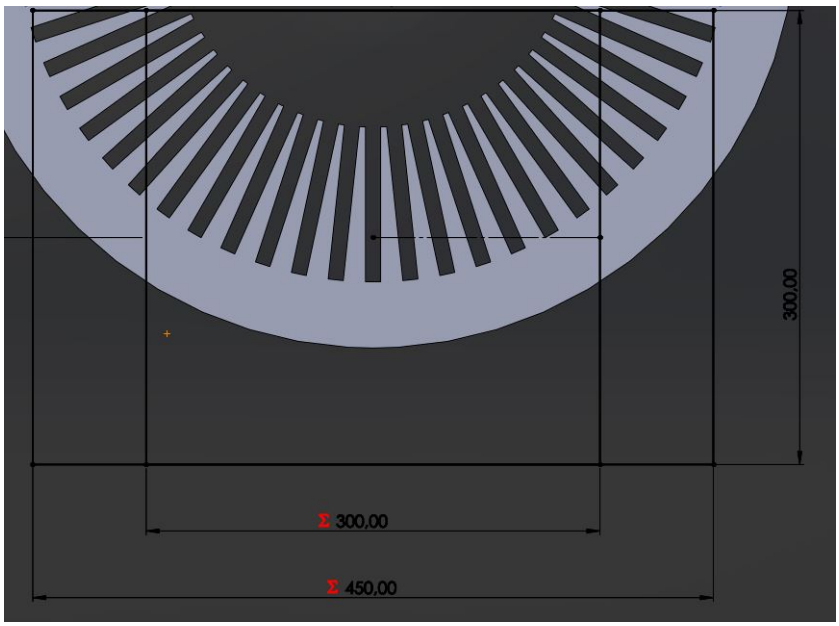

Figure 3.5: Field of view of frame 315 stator lamination.

which also grows a lot with the increase of its diameter, this option was discarded, even though it is the type of lens with the least distortion, which is interesting for precision analysis such as this. Thus, it was sought a lens that had a low distortion, and that had a considerable working range, and a small minimum working distance, being possible to position the camera with different distances from the laminations, to position it so that the field of view is filled in the best possible way

With these parameters in consideration, after researching several options, the model *Fujinon CF25ZA-1S* was chosen (figure 3.6). About the technical specifications of this lens, it has support for sensors up to 23 MP (for pixel sizes up to 2.5 micrometers), and with a C-mount type, so it is compatible with the selected camera. In addition, this camera has a focal length of 25 mm, which will be an important parameter in defining the positioning of the camera as it relates to the field of view by means of equation (3.1), which consists of a similarity of triangles. This lens was also selected because of its wide working range, being able to focus on an object 100 mm away up to infinity, thus managing to work well within the required range that will be defined later in this section. Finally, another feature of this lens that was very relevant in its choice, was its low distortion, this being -0.83% TV distortion [74].

Figure 3.6: Fujinon CF25ZA-1S lens [74].

$$f = \frac{s}{S} \cdot D \tag{3.1}$$

- f - Focal lenght.
- S - Field of view lenght.
- s - Sensor lenght.
- D - Object distance.

The TV distortion index is defined by the formula (3.2), with its parameters represented in the figure 3.7 [75].

$$D_{TV}(\%) = \frac{\Delta H}{H} \cdot 100\% \tag{3.2}$$

At this point it is already possible to define the height of the camera in the maximum and minimum situations described above, of the 900 and 315 frame motor laminations, using the equation (3.1). These distances are 852 mm and 483 mm respectively.

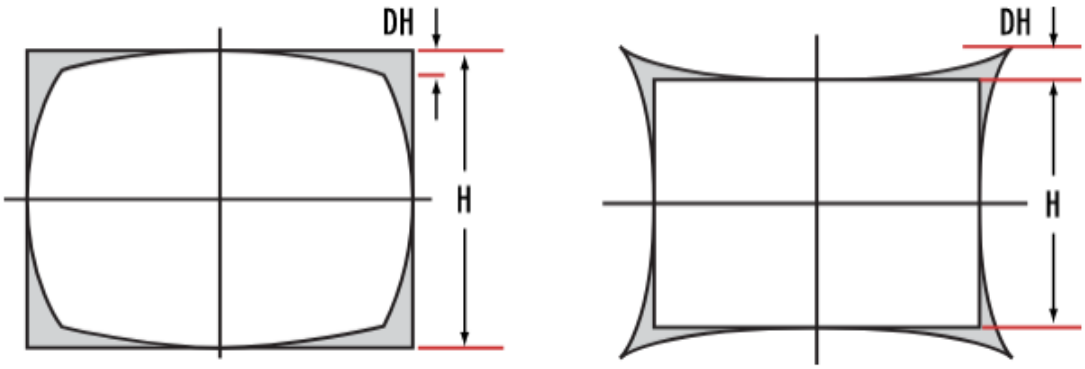

Figure 3.7: TV distortion index [75].

3.1.3 Lighting

Illumination is an extremely critical factor in the sizing of the vision system, since it is the illumination that will make it possible to visualize the structures that one intends to evaluate. Thus, there were three possibilities considered to reach the final lighting decision.

The first option considered was the use of a backlight in conjunction with a direct and collimated light source as done in [54]. With this setup, the burr measurement would be performed indirectly, measuring the rollover that would be isolated in the image as it is possible to see in figure 3.8. Regarding the measurement of the circularity of the bottom slot diameter, and the width of the teeth of the lamination, only the backlight would be used, which would allow obtaining the edges of the lamination with good contrast helping in the accuracy of the analysis.

Figure 3.8: Rollover highlighting through lighting techniques in [54].

However, this possibility was discarded for two reasons, firstly, since the whole sheet does not fit in the field of view, and it will have to be analyzed in sections. And secondly, the lamination would have to move to be completely analyzed, and it was not possible to find a robust solution that, with this movement, the backlight would not be subject to wear due to the relative movement. Another reason for discarding this option was the failure to find a solution in which the backlight would not be in direct contact with the lamination, and at the same time the lamination would

be stable, which is difficult to achieve given that the plate is very thin, and without being fully supported, it bounces easily, which would affect the quality of the analysis.

The second option to be evaluated was to use a dome type lighting, while the background of the plate would be a dark surface to create contrast. The justification of the dome light would be to make the lamination as bright as possible, highlighting it, but in a way that there would be no hotspots. One advantage of using this type of lighting, is that it is possible to construct it from led light sources, and a dome made of some material that is possible to be painted with white matte paint, so it would be a flexible and inexpensive solution.

Experiments were performed to analyse the result that was possible to obtain with the use of this type of lighting, and an excellent highlight of the lamination in relation to its background was achieved, as can be seen in the figure 3.9, using the setup shown in figure 3.10.

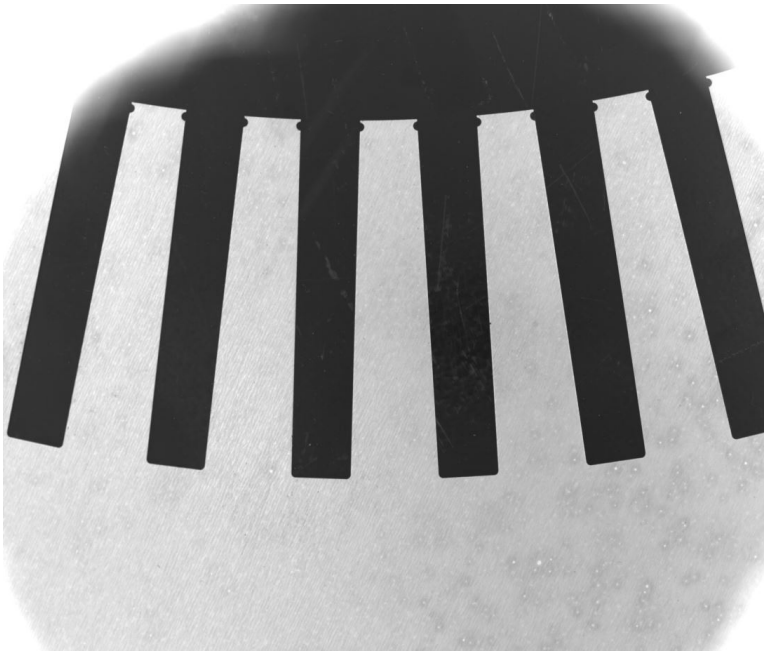

Figure 3.9: Stator lamination with dome light illumination.

This solution proved to be very good for bottom slot diameter and tooth width analysis, as well as economically interesting. However, in order for a single system to be able to evaluate both the burrs and the other two mentioned features, a different solution had to be found, since with this one it was not found a manner to highlight the effects of the punching cut on the lamination.

Finally, the feasibility of using darkfield low angle lighting was studied. With this solution it was possible to highlight both the complete lamination in relation to the background, as well as only the structures resulting from the punching process. The figures 3.11 and 3.12 were obtained using the darkfield low angle lighting setup seen in figure 3.13. The only difference between the two images of the lamination was the exposure time of the sensor, which was 90 milliseconds in 3.11, and 10 milliseconds in 3.12.

The darkfield low angle lighting setup used is a model available at WEG, which is a custom made solution for the company, consisting of a set of four bar lamps of 14.4W each, attached to an

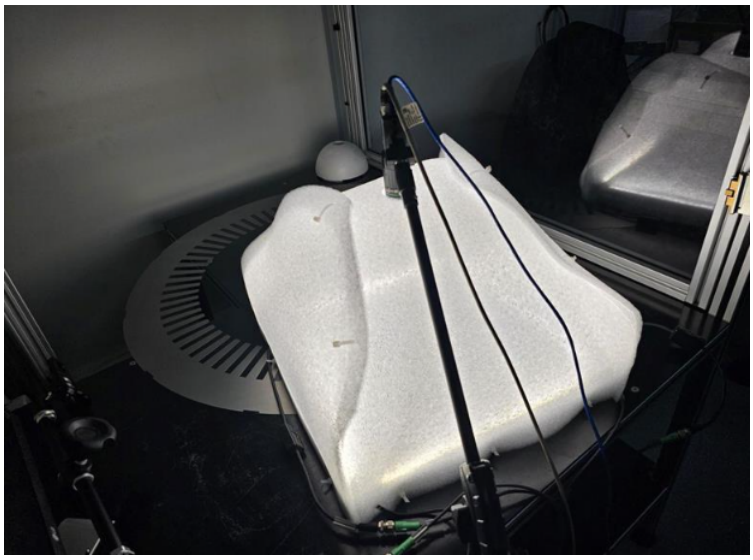

Figure 3.10: Dome light setup at WEG Brazil.

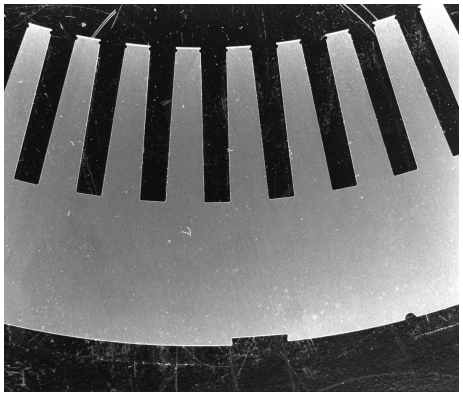

Figure 3.11: Stator lamination with dark field lighting, 90 miliseconds exposure.

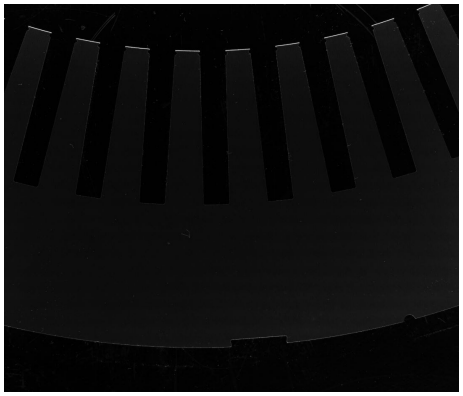

Figure 3.12: Stator lamination with dark field lighting, 10 miliseconds exposure.

aluminium structure that supports the system. This aluminium structure also has the function of blocking the light emission directly to the system sensor, which would prevent the darkfield effect. The structure in total has an approximate dimension of 450 mm x 450 mm, being able to contain the desired field of vision.

With the results obtained by using darkfield low angle lighting, it was decided to choose this method, as it allows the identification of all the structures to be identified through the use of only one setup, being a great advantage since this is a conceptual study, thus to have a single prototype allows the reduction of the complexity of the system as well as a lower cost production of the proposed solutions.

Researching the products available from various suppliers, the biggest difficulty was finding a darkfield lighting model that was capable of containing the defined field of view of 300 mm x 300 mm, as most of the options available were circular and smaller in diameter than necessary.

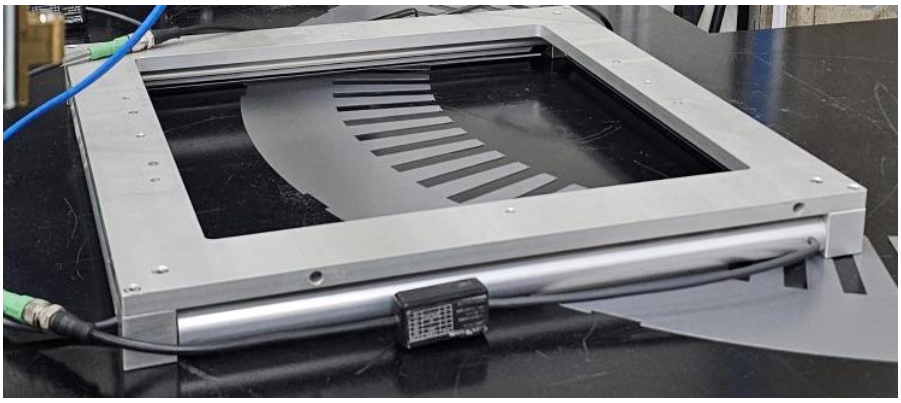

Figure 3.13: Darkfield low angle lighting setup in WEG Brazil.

However, the model DFL-12 (figure 3.14) was found, which proved to be a good option given the similar power of the model used to validate the method, and for having a convenient working height for the constitution of a prototype, as well as a structure made of 30 mm x 30 mm aluminium extrusions that facilitate the development of a modular mechanical prototype [76].

Figure 3.14: DFL-12 darkfield lighting [76].

Although the solution presented was already sufficient, in contact with a local supplier of computer vision equipment, it was offered another model that could be custom-made for the development of the prototype proposed in this work, which would be similar to the one used in the validation of using darkfield lighting, which is interesting in order to ensure similar results after purchasing the components. Another relevant advantage of this new model was the much smaller price than the one previously found. This solution uses four bar shaped lamps (figure 3.15), 492 mm long and 27 mm in diameter, model WLS27XW430DSQ [77], powered by 12-30 V DC, and LED technology. This lamp is ideal for industrial environments, IP66 rated or higher, and would be coupled to aluminium profiles by means of the LMBWLS27U (figure 3.16) [78] connector.

However, as will be seen later, the development of the Computer-Aided Design (CAD) for the

Figure 3.15: WLS27XW430DSQ industrial lamp [77].

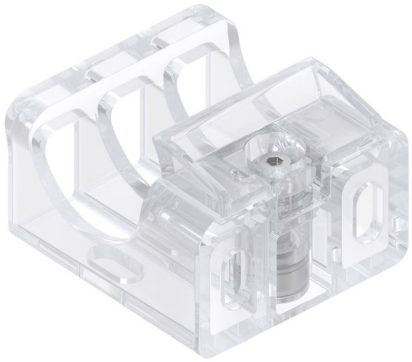

Figure 3.16: LMBWLS27U lamp connector [78].

mechanical project proposal was done with the first darkfield solution found, since when the new option was found, the mechanical project CAD had already been developed, and the modification to the new darkfield lighting model would not cause relevant changes in it.

3.1.4 Computer Processing Unit

Naturally, all software needs to be processed in some processing unit, and in this way MVISIA has two different solutions, one being the evolution of the other. The first model to be released was the MV-CVU, and then the latest model MV-CVU 300, shown in figures 3.17 and 3.18, respectively.

Figure 3.17: MVISIA MV-CVU processing unit [72].

Figure 3.18: MVISIA MV-CVU 300 processing unit [72].

About the MV-CVU, and its technical specifications, its processing has a maximum clock of 2.0 GHz, with 6 processing cores, 2 GB of Random-access Memory (RAM), and 10 GB of storage. This model also has support for various communication protocol options, such as Modbus TCP/IP, Ethernet IP, and OPC UA. Its connection is via a common network cable, RJ45-Cat6e. The MV-CVU also has software configurable I/Os, with both normally open and normally closed contact ports. This model was developed to work in conjunction with the MV-SENS-100 sensor.

Regarding mounting, the MV-CVU is made with the intention of being mounted on a standard DIN rail [72].

As for the MV-CVU 300, it has a quad-core processor, with a clock of 1.5 GHz, with a much more powerful RAM than its predecessor, being 8 GB DDR4 2666 MHz, and an Intel@ UHD Graphics card. It also has two RJ45 ports for I/O interface. For video output the model has HDMI support, with FullHD playback and 60 frames per second. This model was also produced with the intention of being mounted on a standard DIN rail [72].

The CPU selected for the work was then the MV-CVU 300, given its better processing capabilities, but an important factor in the decision is the fact that this model was designed with the purpose of working together with the selected sensor, the MV-SENS-300.

3.2 Rollover Measurement

At the beginning of the project, prior to the identification of more geometries of the lamination that were interesting to be analysed, the goal was the measurement of the size of the lamination burrs, due to the importance of their detection to avoid the eddy current occurrence as explained in chapter 2.

Initially, the profile of the lamination after the punching process was analysed. For this, samples, which were the slug from laminations punching process, were cut, immersed in polymeric resin, and then sanded and polished by materials of different grain sizes. In this process, it was removed approximately 3 millimeters of the resin with sample, with the objective of having the real profile of punching in the surface of the resin, and remove the part of the sample that may have been deformed in the cut of the slug for insertion in resin, resulting in what can be seen in figure 3.19.

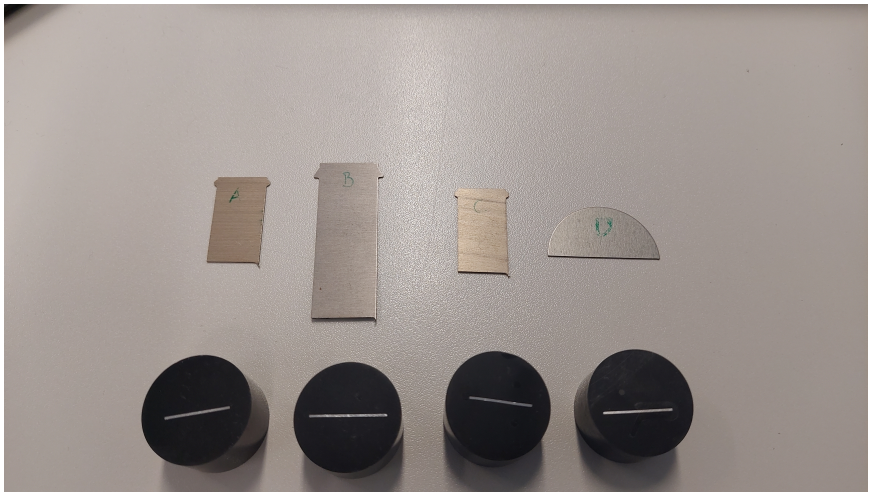

Figure 3.19: WEG samples in resin.

These samples were then analyzed under a microscope and the images 3.20, 3.21 were obtained, in which it is possible to see that the burr has a very small dimension, in the order of

approximately 10 micrometers. Given that, it was considered that the approach made in [54] of analysing the rollover to infer the quality of the cut and burr size, made more sense than directly measuring the size of the burr. In comparison, in the images 3.20, 3.21 it can be seen that the rollover profile is approximately seven times the size of the burr.

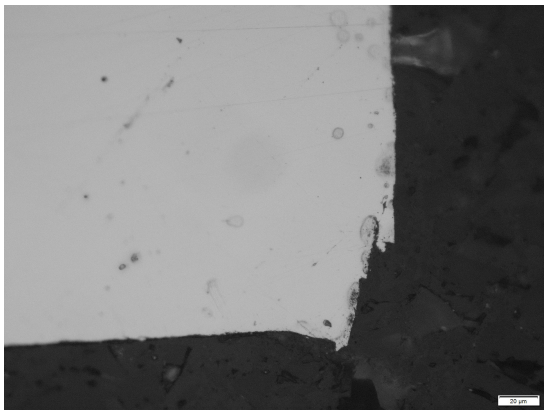

Figure 3.20: Microscopic analysis of WEG sample, burr, magnification 50x.

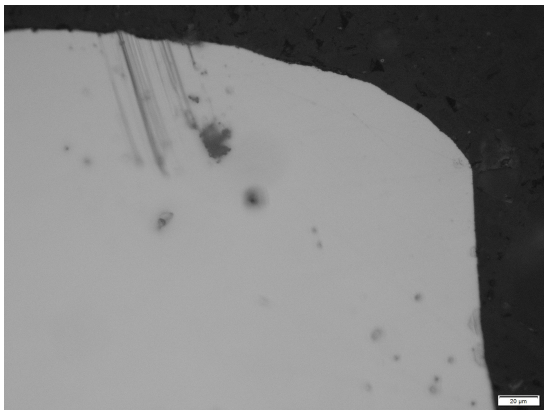

Figure 3.21: Microscopic analysis of WEG sample, rollover, magnification 50x.

As to make possible the beginning of the development of a software of burr measurement without the dependence of the acquisition of professional hardware, an aluminium sample (figure 3.22) was acquired and cut by an industrial guillotine with great clearance in the cut, in order to obtain a sample with great burr and rollover profiles (figure 3.23), which photos were taken with a mobile phone, using an improvised dome light. On top of these photos a software for rollover measurement was developed, which will be described next.

Figure 3.22: Aluminium sample for rollover measure software development.

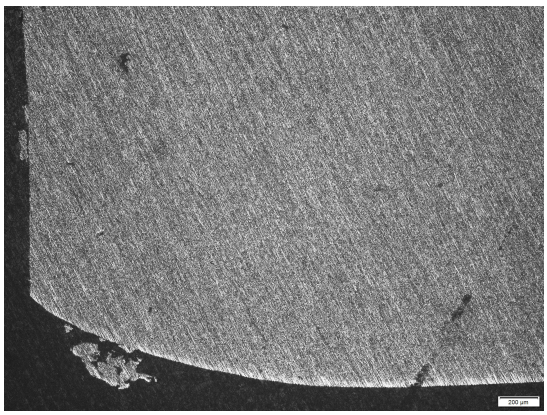

Figure 3.23: Aluminium sample for rollover measure software development in microscope, magnification 5x.

Naturally, like any computer vision software, the image to be analysed is first read in its entirety. In order to reduce the need for high processing, and facilitate the development of a code

that works well in separating the features to be analysed from the image, a crop of the region of interest (ROI) of the sample is performed, cutting only the rollover.

After that the image processing began, having as first operation the conversion of the image into gray scale to make possible the use of smoothing filters, since the sample has many marks and scratches that make difficult the segmentation of the rollover. For smoothing, several filters were tested, such as the Gaussian, median, and bilateral filters. After this noise removal, the image is transformed into a binary image using the technique of Otsu.

As can be seen in the images 3.24, 3.25, 3.26, the most satisfactory result after binarization was the one that was previously smoothed by the bilateral filter. However, it is possible to see that the binarized image still has a lot of noise, so a morphological opening operation is performed, and the result can be seen in 3.27.

Figure 3.24: Binarization of ROI after gaussian filter.

Figure 3.25: Binarization of ROI after median filter.

Figure 3.26: Binarization of ROI after bilateral filter.

Figure 3.27: Binarized ROI after opening operation.

With the objective of removing the remaining noise, and to define the maximum and minimum limits of the rollover profile, a function was developed in the program which has as inputs a threshold and an analysis interval. In this function, the columns of the image will be verified, having the amount of columns specified in the analysis interval input, and within these columns,

the number of white pixels per line will be verified, which are expected to be all representative of the rollover area. After the calculation of the number of white pixels per line, these are compared with a threshold that is given in percentage, and, in case the percentage of white pixels in the line is bigger than the threshold, all the pixels of that line are set to white, otherwise, they are set to black, in a way to delimit the maximum and minimum limit of the rollover suppressing noise and small alterations due to defects of the sample for the development of the software. All the columns out of the analysis interval are suppressed, and the result can be seen in figure 3.28.

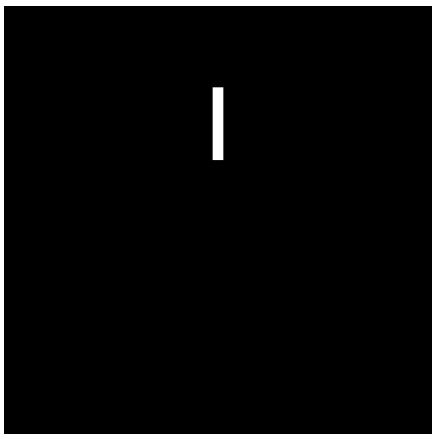

Figure 3.28: Rollover image after horizontal filtering, showing the upper em lower limit of the rollover profile on the horizontal centre of the image.

With the rollover area at the centre of the segmented image, it is possible to proceed with the rollover measurement, by acquiring the index of the highest and lowest lines containing white pixels, and computing the difference between them. After calibration, it is possible the conversion of the length value in pixel to a value in millimetres. Knowing the position of the beginning and end of the rollover profile, it is possible to draw in the original image a double arrow to show the user of the program the position of the rollover, and write in the corner of the image the value of its dimension, as it is possible to see in the figure 3.29.

3.3 Planar Dimensions Measurement

For the measurement of the other two structures relevant to this project, another software was developed. This software will measure two structures at the same time, being the bottom slot circularity, and the stator teeth width. It was possible to make a single program for the measurement of these two shapes due to the interdependence of their measurement, as will be presented further on.

Currently, the verification of the tooth width is done in a small indentation present in the laminations with the use of a caliper. Because of this, the first approach to this problem was through the visual inspection of this same region, having been created a program that generated the output that can be seen in the figure 3.30, trying to replicate through computer vision the inspection that is already done manually.

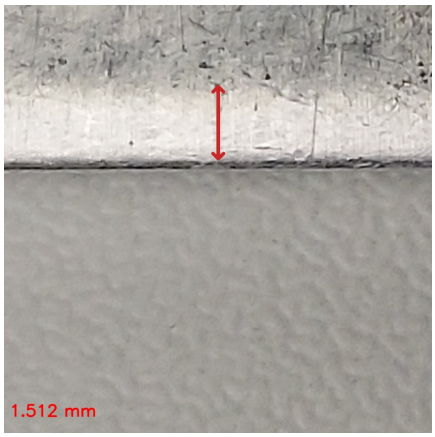

Figure 3.29: Output image of the rollover measure software.

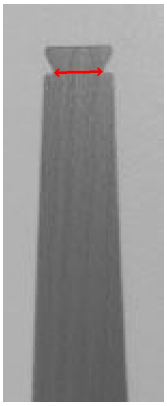

Figure 3.30: Output of the first tooth measurement software.

This software is able to separate any tooth from the stator plate, position it vertically and measure the indentation. Although it is an interesting solution, it was thought that, since the error is in the circumferential direction, given that the punching which produces the teeth is done by rotating the lamination, it was considered that it would be more interesting to analyse the base of the teeth, where the manufacturing error would be magnified, and would be easier and more accurate to detect. Next, the software's logic and operation will be demonstrated on top of figure 3.11

3.3.1 Circularity Measurement

As with the rollover measurement software, the first thing to do is to read the image, and convert it to grey scale. After that, the program starts measuring the circularity of the bottom slot first. The tooth width measurement is only performed after the circularity because the bottom slot diameter geometry is used to find the measurement points of the tooth width as will be seen later.

With the grayscale image, the ROI of this part of the program is cropped (figure 3.31), which is the bottom slot region. With this region separated, an opening operation is performed followed

by a closing operation, so that it is possible to reduce the image noise, as can be seen in the figure 3.32

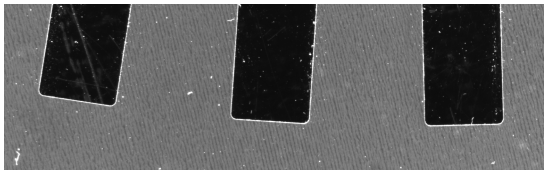

Figure 3.31: ROI for circularity measurement.

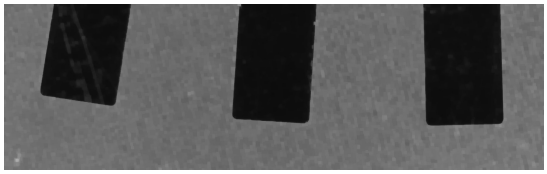

Figure 3.32: ROI for circularity measurement after morphological operations.

With the denoised image, it is already possible to extract the edges using the Canny method with greater precision, obtaining the result that can be seen in the figure 3.35, where it is possible to perceive a good extraction of edges, but still with some noise. However, as the objective is only the extraction of the bottom slot edges, it is necessary to remove the edges that constitute the profile of the lamination teeth. To this end, a function was created that uses Sobel's filter to calculate the image gradient in the X and Y directions (figures 3.33 and 3.34), and then checks which direction has the highest gradient at the position where there are pixels identified as edges after using Canny's method. When the gradient is higher in the Y direction, the pixel classified as an edge remains unchanged, however, if the opposite is true, the pixel classified as an edge is set to black.

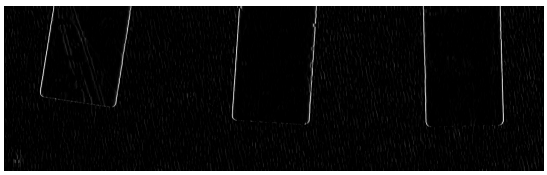

Figure 3.33: Circularity ROI after Sobel filter in X direction.

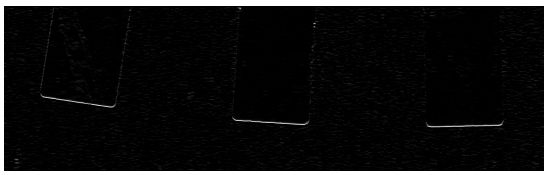

Figure 3.34: Circularity ROI after Sobel filter in Y direction.

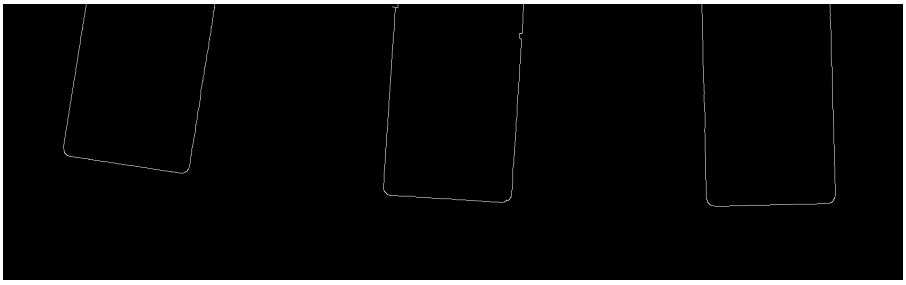

Figure 3.35: Circularity ROI after Canny operation.

After this operation all vertical edges were removed (figure 3.36), however some small noises are still present, which are removed by an opening operation with kernel dimensions (2,1) (figure 3.37). Once this is done, it is noticed that some other parts of the image that had a higher gradient in the Y direction, and that had been classified as edges, and are not part of the bottom slot, are still

in the image, and need to be removed. To do this, another function was created, which identifies the row index of all pixels of value 255, and averages these indices. This average is taken as the average value of the height of the pixels representing the bottom slot edge, since most of the remaining pixels in the image are bottom slots edges. This function takes as input the percentage of the image that should be considered that surely the bottom slot is within it. Knowing the average position of the bottom slot pixels, and having defined the percentage of the image that will be kept unchanged, all the remaining pixels are set to a value of 0, in order to simply eliminate the extra edges that were remaining on the edges image (figure 3.37). Finally, another function is used, which aims to detect the beginning and end of the edges of each bottom slot and trim the tips, in order to cut the concordance radius that exists between the side of the tooth and the bottom slot, since these edges can disturb an accurate measurement of the bottom slot diameter. Thus one gets an image that has as white pixels only the points of the image that represent edges of the bottom slot, as can be seen in figure 3.39

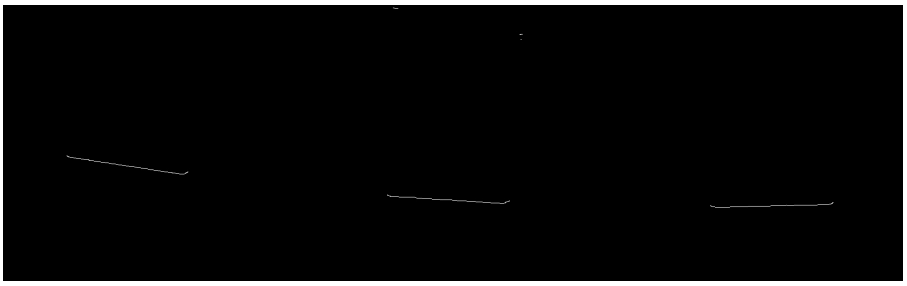

Figure 3.36: Circularity ROI after vertical edges removal.

Figure 3.37: Horizontal edges after opening operation.

Figure 3.38: Horizontal edges after setting black pixels out of threshold.

To measure the diameter of the bottom slot, the next step is to acquire the coordinates of all the white pixels in the image 3.39, and use them in a circle fitting algorithm, using the method of least squares, and it is possible to obtain a circle formula, which has the parameters x_c, y_c, and r (coordinates of the circle center, and radius), and thus it is possible to characterize the circle of the bottom slot at the sub-pixel level, since the circle fitting formula is not restricted to the position of the pixels in the image to calculate the circle that would best have the white pixels as points. For the user of the software to visualize the result of the program, it is possible after calibration to insert the measure of radius (or diameter) into the image, as well as an indication of the shape of the calculated circle representing the diameter of the bottom slot. This output image can be seen in figure 3.40.

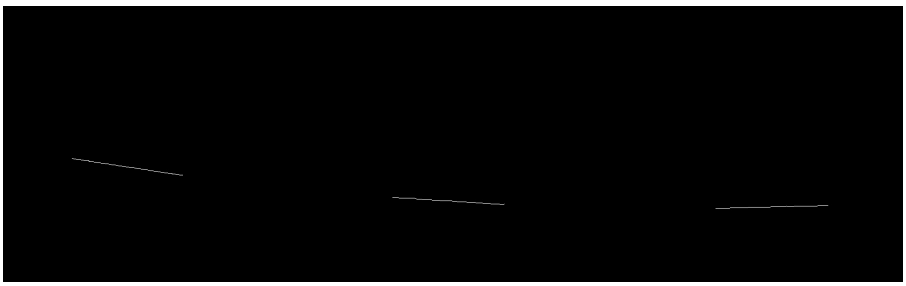

Figure 3.39: Horizontal edges after setting black pixels out of threshold.

3.3.2 Tooth Width Measurement

With the bottom slot diameter defined, it is possible to proceed to the tooth width measurement. With the image already read by the software, the first step is to crop out the teeth to be analysed. In this demonstration, three teeth will be measured, however the program does not impose a limit, being a configurable feature. The ROI's can be seen in the images 3.41, 3.42, 3.43, however throughout the program description only the image processing operations on a single tooth will be demonstrated, in order to simplify the document.

Once the ROI is obtained, image processing operations can begin. First the image goes through a binarization process, figure 3.44, obtaining an image with a significant amount of noise, which are from the defects of the background, which are being highlighted with considerable intensity in the image due to the darkfield lighting technique, thus, it is noticeable that in a final prototype it is relevant to have a background with few defects, as well as it is necessary the development of a software with considerable capacity of noise suppression.

According to what has been analysed, a morphological opening operation is then performed followed by a closing operation. The first aims to eliminate noise from the image and the second to fill the spaces within the tooth region that could not be avoided during binarization, as can be seen in the images 3.45 and 3.46.

With the teeth properly segmented, the next step is to identify the beginning and end of their linear profile, both on the left and right sides. This is possible by means of a function created specifically for this task, which has as input an analysis step, a threshold, and steps to return.

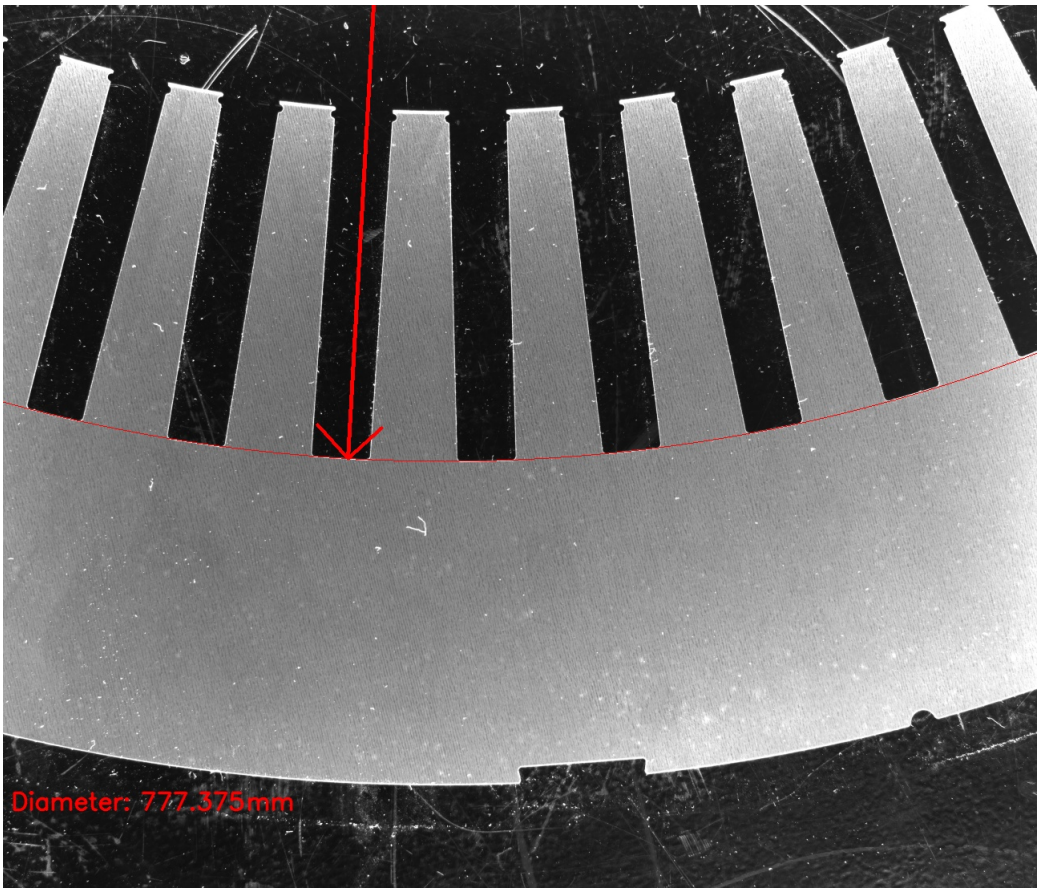

Figure 3.40: Output image of bottom slot diameter measure.

Exemplifying the identification of the linear profile of the tooth on the left side, first the function finds the lowest pixel where the linear profile of the tooth begins, then it analyzes in a number of rows above equivalent to the step input, which is the column of the leftmost white pixel, and saves the difference in columns between these first two pixels. The function repeats this process iteratively, always analyzing the column difference between the current and the previous pixel, saving this value, and averaging the values found.

This is repeated until the difference verified between columns in the current iteration is greater than a certain value times the average of the differences found, this multiplication value being the threshold given as input to the function. When this situation is found, the row considered as the final one is the pixel row of the last iteration minus the step value of the function times the step value to be returned, to have a safety margin in defining the limits of the linear profile of the tooth. Then the same is repeated for the opposite side of the tooth.

The behavior of this algorithm can be observed in image 3.47, where in each iteration the reference pixel is the green point, which is compared with the pixel located approximately at the red point. In this example, the value of steps to be returned was 2, so the blue point, which represents the row considered as the end of the linear profile, is positioned below the last green point a number of rows equivalent to twice the step.

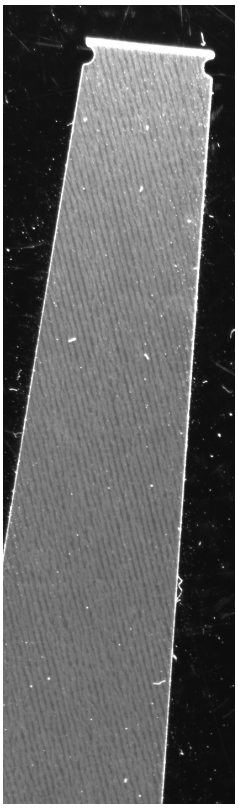

Figure 3.41: Tooth ROI 1.

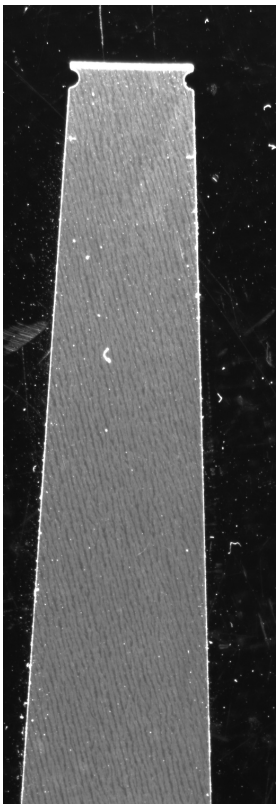

Figure 3.42: Tooth ROI 2.

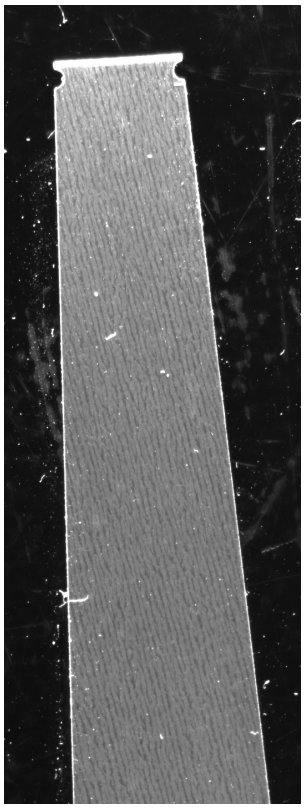

Figure 3.43: Tooth ROI 3.

Once it is known where the linear profile begins and ends on both sides of each tooth in the image, the next step is to characterize this linear profile by defining a linear function equivalent to the edges of the teeth. For this, the strategy is similar to that of defining a circular function as previously done, it is necessary to obtain the coordinates of the pixels which compose the linear profile of the tooth and use them in a linear fitting algorithm.

The coordinates of the pixels that compose the edges of the teeth are found by reading row by row, between the boundaries of the linear profiles, the leftmost and rightmost pixels, which are considered to represent the linear profiles. The coordinates of all these pixels are then stored and used as input for the linear fitting algorithm. Once this is done, the algorithm returns a point on the line and its slope, and the linear profile is fully defined. As a last step in this task, the referential of the equation found, which are the limits of the ROI of the tooth, is transferred to the referential of the original image.

With the linear profiles defined and with the referential of the functions in the original image, it is possible to find the intersection points between the circumference of the bottom slot and the linear profiles of the teeth. The distance between the intersection of the bottom slot and the two sides of the tooth has a dimension specified in the stator lamination electrical designs, figure 3.48, so it is this distance that will be measured by the software.

With this method of finding the measurement points, again these are found in a way that is

Figure 3.44: Tooth ROI binarized.

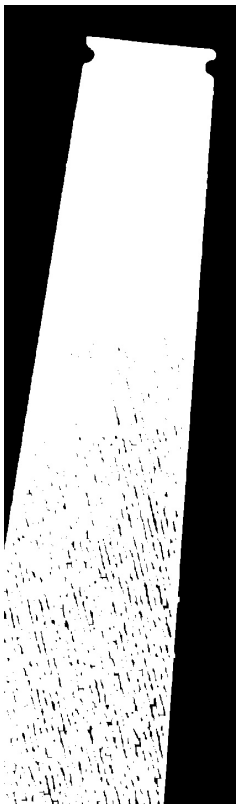

Figure 3.45: Binarized tooth ROI after opening operation.

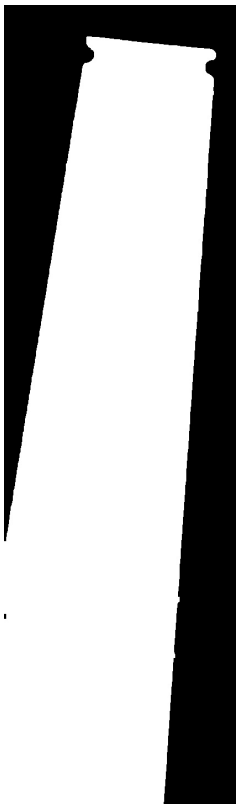

Figure 3.46: Binarized tooth ROI after opening and closing operation.

not limited to pixel locations, as the position of this point is found mathematically rather than by selecting a specific pixel, increasing the accuracy of the program. Once the points are defined, it is possible to calculate their Euclidean distance, and then convert this distance in pixels, to distance in millimetres by means of the calibration already explained earlier in this work.

Also with the objective of expressing the result of the program in a clear way for the user, at the end of the measurement of the width of all the teeth, in the original image double arrows are added between the measurement points of each tooth, and also an identification of the tooth in the middle of this arrow, while in the corner of the image the measurements are added beside the identification of the respective tooth, as can be seen in figure 3.49

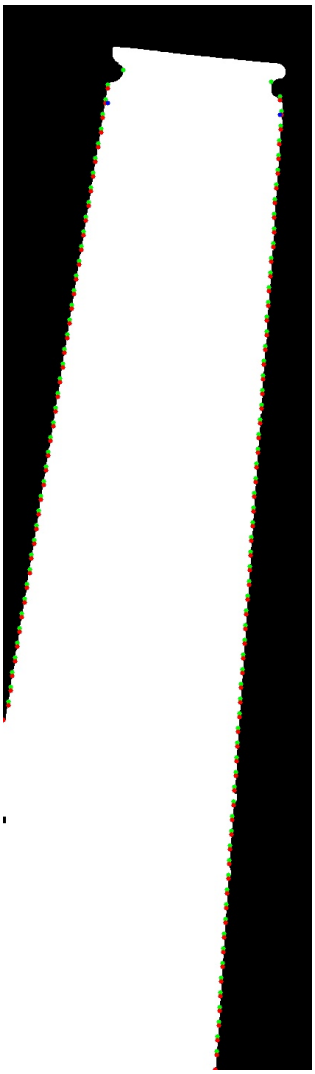

Figure 3.47: Linear tooth limits algorithm illustration.

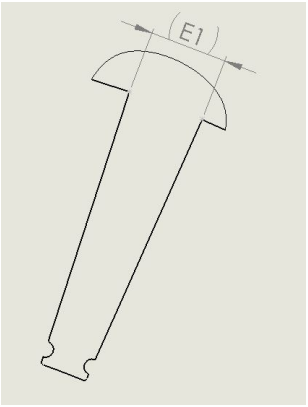

Figure 3.48: Tooth width, measuring points specification.

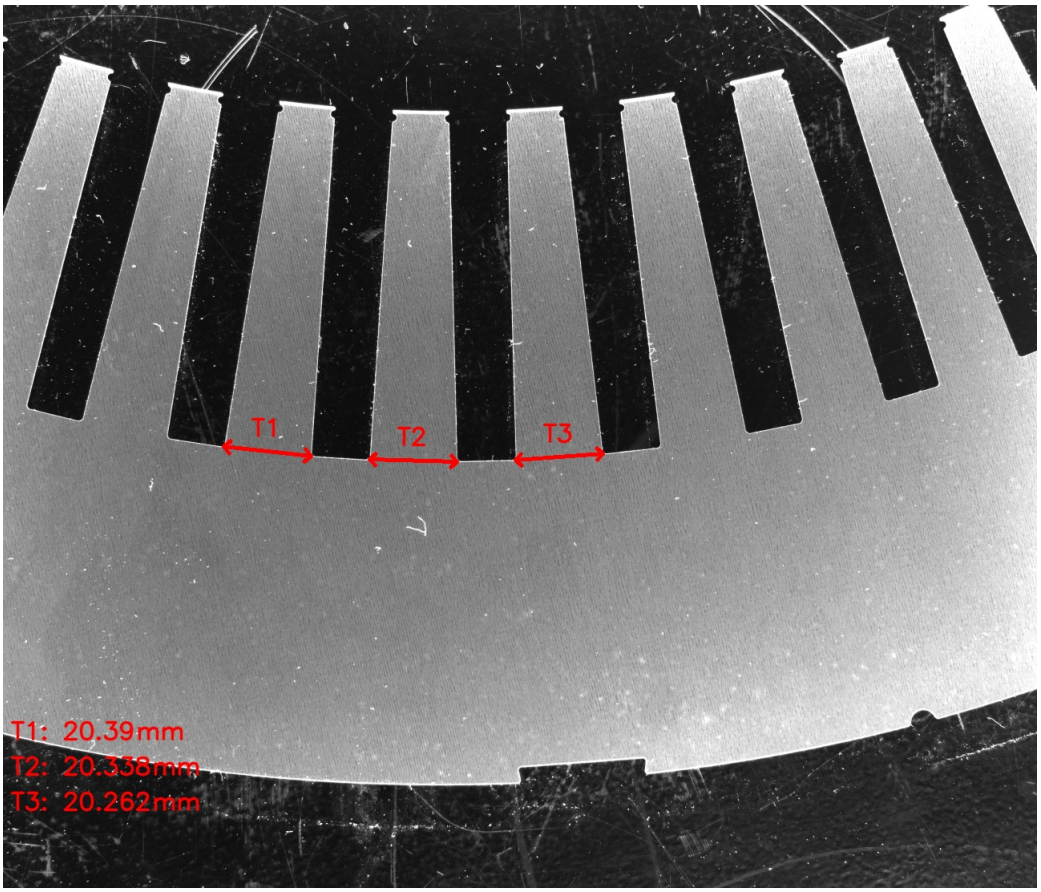

Figure 3.49: Output image of tooth measurement software.

3.4 Image rotation optimization

The universe has a continuous nature, different from digital data, in which information is discrete. Thus, as mentioned before, the images that are captured from the surroundings are not able to represent it with total reliability, so there is always a loss of information in the image acquisition process, which tries to represent the continuous space in a discrete form through the pixels.

When applying geometric operations, as seen in chapter 2, there are different types of interpolations available, which are necessary when changing the position of the pixels, since they are not necessarily transposed exactly to the position of another pixel during the operation most of times, thus using interpolation to define the pixel values after the geometric operation.

The developed solution uses geometric operations, more specifically rotation, which requires the use of interpolation. Since the application of the developed software is precision measurement, it is important to try to maintain the best possible image quality during all the processing phases, therefore it is interesting to analyze the changes and losses that the image processing operations impose when applied.

To this end, an auxiliary program has been developed that tests the rotation of a test image. Since in the presented solution the detection of edges and corners of the sample under analysis is

performed, it is interesting that the test image has high frequency characteristics, such as rough texture, and sharp edges, therefore it was chosen as a test image a brick wall available in an image database from the University of Southern California [79], which has features that match the ones mentioned above, as can be seen in figure 3.50.

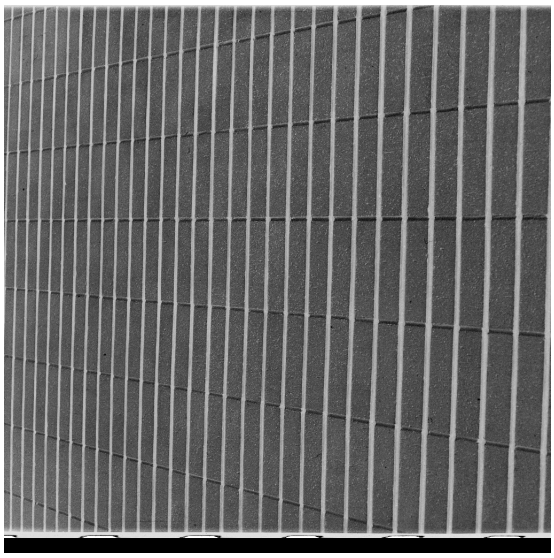

Figure 3.50: Test image for interpolation methods in rotation operation [79].

On the testing program, the software first reads the original image, then applies a 10 degrees counter-clockwise rotation, using a defined interpolation method, and repeats this process 36 times to perform a complete rotation, saving the resulting image at the end. In this manner, it is possible to magnify the error that occurs in each interpolation, rather than analyzing an image that went through only one rotation. Then the program repeats the same process for other interpolations methods, always saving the resulting image for future analysis.

The first interpolation method tested was the Nearest Neighbor. This is a simple interpolation method, that instead of setting the pixel value by means of calculations of near values, the pixel value is set equal to the nearest pixel that after rotation is virtually placed the nearest to the position of the real pixel. This simplicity leads to low processing costs, but low quality results, as can be seen in figure 3.51. Only the center of the image is preserved, while the corners are lost. This occurs as a result of the appearance of black areas when the image is rotated, since the corners are positioned outside the image window, and those missing parts are lost along the rotation operations that follow, while the black areas are rotated as well.

The second method that was evaluated was bilinear interpolation (figure 3.52). This interpolation method is often used in computer vision operations, and is used to interpolate two variables at the same time by performing two linear interpolations, the first in one direction, and the second in a perpendicular direction to the first. When a rotated image pixel cannot have a value assigned to it directly from a pixel of the original image, bilinear interpolation is performed by considering four original pixels that have been repositioned closest to the position of the new real pixel, and taking a weighted average of these, to assign the value of the new pixel. For example, a pixel with

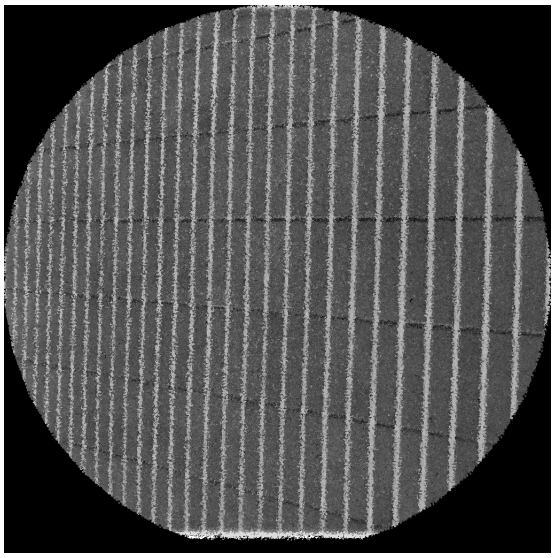

Figure 3.51: Figure 3.50, after several rotations with Nearest Neighbor interpolation method.

coordinates in row 20.2 and column 14.5, would first be interpolated between pixels of columns 14 and 15, in row 20, and then the same would be performed for pixels of the same columns but in row 21. Once this is done, a further interpolation is performed between rows 21 and 20 to obtain the final value of the pixel in cause, as can be seen in the figure 3.53, using equations (3.3), (3.4).

$$\begin{aligned} f(x,y_1) &= \frac{x_2-x}{x_2-x_1}\cdot f(Q_{11}) + \frac{x-x_1}{x_2-x_1}\cdot f(Q_{21}) \\ f(x,y_2) &= \frac{x_2-x}{x_2-x_1}\cdot f(Q_{12}) + \frac{x-x_1}{x_2-x_1}\cdot f(Q_{22}). \end{aligned} \tag{3.3}$$

$$f(x,y) = \frac{y_2-y}{y_2-y_1}\cdot f(x,y_1) + \frac{y-y_1}{y_2-y_1}\cdot f(x,y_2) \tag{3.4}$$

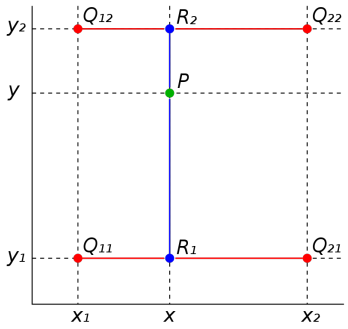

Figure 3.52: Bilinear interpolation, where the red dots are the original data, while the green dot represents the point to be interpolated.

Regarding the result of this interpolation method, it is possible to realize that the shapes and features of the original image are better preserved than using the nearest neighbor method, and are also less noisy, although it is evident that the result is blurred, as seen in figure 3.53. That may difficult edge detection, or at least decrease the accuracy on detecting its location, which is supposed to be avoided.

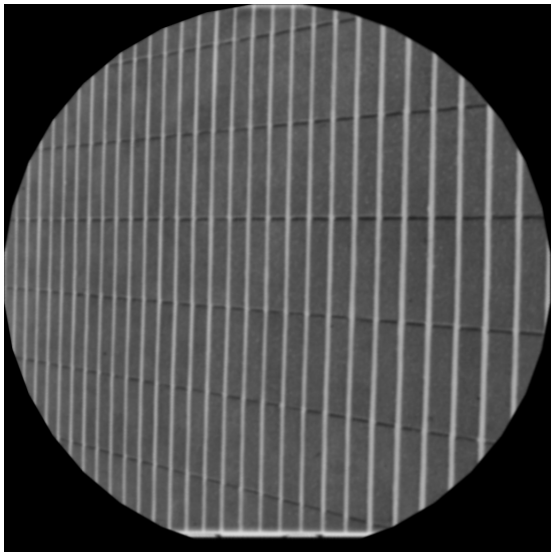

Figure 3.53: Figure 3.50, after several rotations with bilinear interpolation method.

The next evaluated method was the bicubic interpolation. One of its main differences regarding bilinear interpolation, is that the bicubic method considers a 4x4 kernel around the pixel to be interpolated, thus it uses the value of sixteen different pixels in order to compute the new pixel value, instead of the four considered on the previous method. Besides the number of pixels considered, the calculation method is also different, as it performs a convolution kernel stated in equation (3.5), which interpolates the four pixels of each row, and then interpolates the four previously obtained values using the same equation [80]. The a parameter in equation (3.5), can be set by the user, however on this test it was performed using the OpenCV library, which sets $a = -0.75$ as default value, and is related to the third order convergence of the equation. This method demands more processing power, and thus takes longer to be performed than the previous one, as it is more complex.

$$W(x) = \begin{cases} (a+2)|x|^3 - (a+3)|x|^2 + 1, & \text{for } |x| \leq 1 \\ a|x|^3 - 5a|x|^2 + 8a|x| - 4a, & \text{for } 1 < |x| < 2 \\ 0, & \text{otherwise} \end{cases} \tag{3.5}$$

Regarding the obtained result, it clearly provides a better result, with the shapes and features of the original image well preserved, as seen in figure 3.54. Another characteristic easy to notice is an increase in the acutance of the image, which is a consequence of the overshoot that comes from the negative lobes of the applied kernel. Although the result has a good quality when compared to the

previous methods, it still presents some changes relative to the original image, more specifically a noise with a granular aspect, mainly on the edges, which could decrease the precision in the definition of their location.

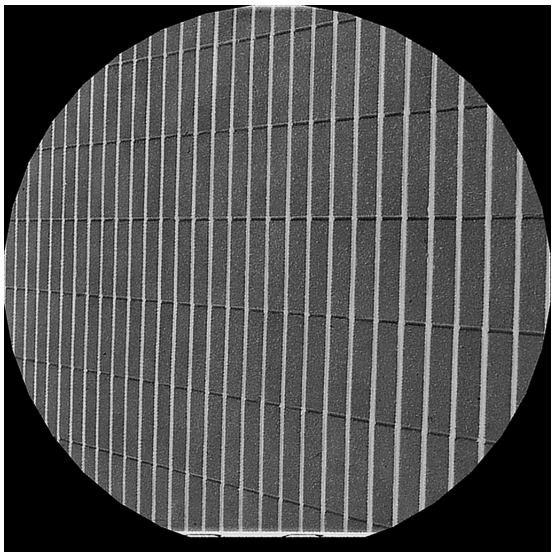

Figure 3.54: Figure 3.50, after several rotations with bicubic interpolation method.

Another tested method was the pixel area interpolation option available in OpenCV. The algorithm of this method is similar to the bilinear interpolation one, with the difference in how the weights of the interpolation are computed. In the bilinear method, the interpolation weights are computed considering the distance between the input and output pixels, while the pixel area method computes the weights based on the overlapping area between the input and output pixels. This manner of computing weights decreases the contribution of input pixels that only partially overlap the output ones, thus producing a better result than the bilinear interpolation method as can be seen in figure 3.55.

The result from the pixel area method, visually, seems very similar to the one obtained by the bilinear, that is, it maintains the original image's overall shape, however with a significant blur on it, which as discussed previously, decreases the precision of edge detection, and other features.

The following interpolation to be tested was the fill outliers method, also available in OpenCV library. The main objective of this method is to better compute the value of outliers pixels, which are the pixels with invalid or missing values, a situation that happens in some cases when the nearest neighbor method is utilized. This phenomenon happens when the output pixel is mapped to a non-integer coordinate, in which the output pixel is normally calculated as the nearest neighbor, however, in some cases this neighbor does not exist in the input image, or is outside boundaries, leading to a missing or undefined pixel, which the fill outliers method is able to correct, by filling those pixels using a nearby pixel value, producing a result more accurate than the nearest neighbor method. However, because both methods are very similar in general, differing only on how they deal with outlying pixels, in a macro scale they provide similar results, as can be seen in figure 3.56, if compared to figure 3.51.

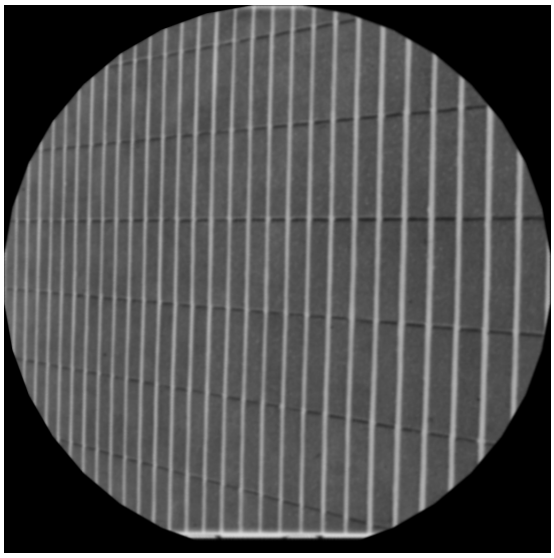

Figure 3.55: Figure 3.50, after several rotations with resampling using pixel area relation.

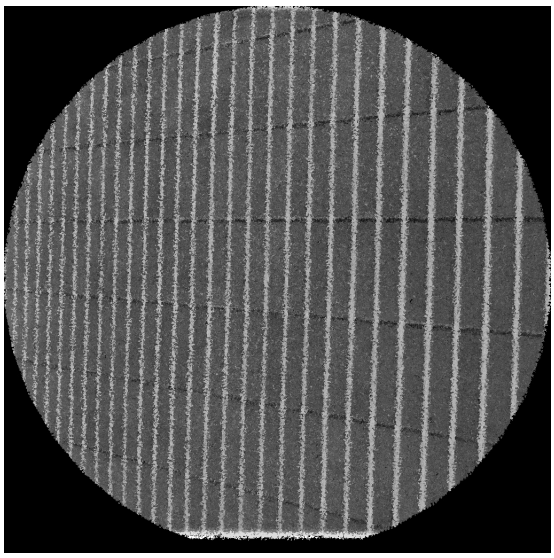

Figure 3.56: Figure 3.50, after several rotations with fill outliers interpolation OpenCV option.

The last tested method was the Lanczos interpolation. The Lanczos resampling is a well know interpolation method, it was created to be used in the telecommunications field, however, with the purpose of performing smooth interpolations in digital signals. It is based on the normalized *sinc* function, defined in (3.6), and represented in image 3.57.

$$sinc(x) = \begin{cases} 1, & x = 0 \\ \frac{sin(\pi x)}{\pi x}, & x \neq 0 \end{cases} \tag{3.6}$$

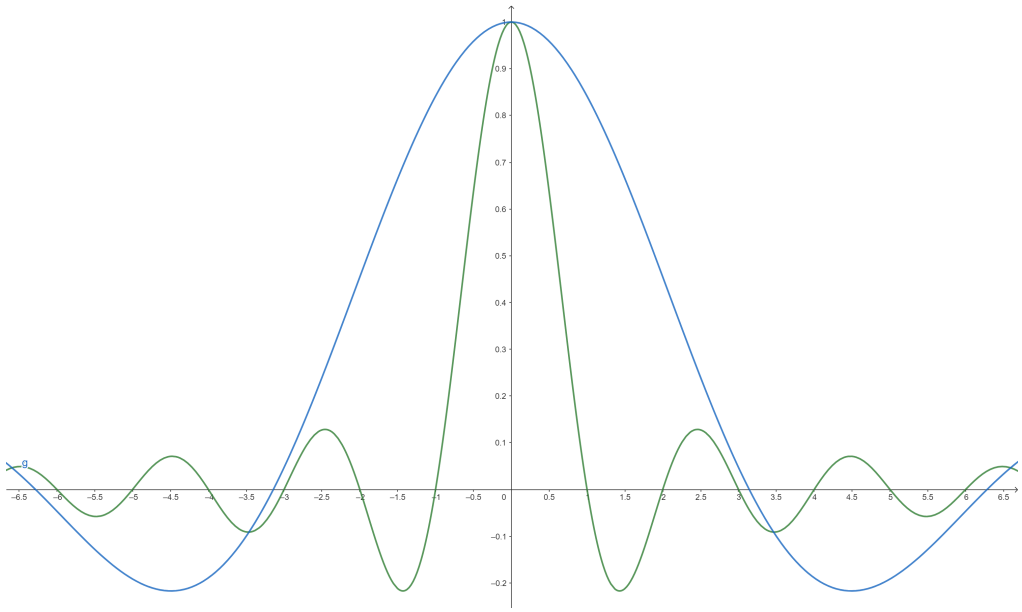

Figure 3.57: In green the *sinc* normalized function (3.6), and in blue the original *sinc* function (3.7)

$$sinc(x) = \begin{cases} 1, & x = 0 \\ \frac{sin(x)}{x}, & x \neq 0 \end{cases} \tag{3.7}$$

In image processing, the application of the Lanczos method consists of interpolating utilizing a kernel, which is convoluted throughout the image to compute the output pixels value considering the nearby input ones. As it is based on the *sinc* function, and it is intended to consider part of the image pixels on the interpolation of each output pixel, the original *sinc* function has to be fitted into a kernel $L(x)$. The Lanczos kernel is the *sinc* function multiplied by the Lanczos window, which consists in the central lobe of the *sinc* function stretched horizontally by the addition of the parameter a (the kernel size is defined by a), as can be seen in figure 3.58, resulting in the kernel shown in equation (3.8), and figure 3.59.

$$L(x) = \begin{cases} 1, & \text{if} x = 0 \\ sinc(\pi x)sinc(\pi x/a), & \text{if} -a < x < a \text{and} x \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.8}$$

With the interpolation kernel defined, the interpolation itself is performed following equation (3.9), for one dimensional data, with s_i as pixel values. However, as images are two dimensional data, the kernel for this case is obtained as stated in the equation (3.10).

$$S(x) = \sum_{i=\lfloor x \rfloor - a + 1}^{\lfloor x \rfloor + a} s_i \cdot L(x - i) \tag{3.9}$$

Figure 3.58: Lanczos window, with $a = 1$ in blue, $a = 2$ in green, and $a = 3$ in red.

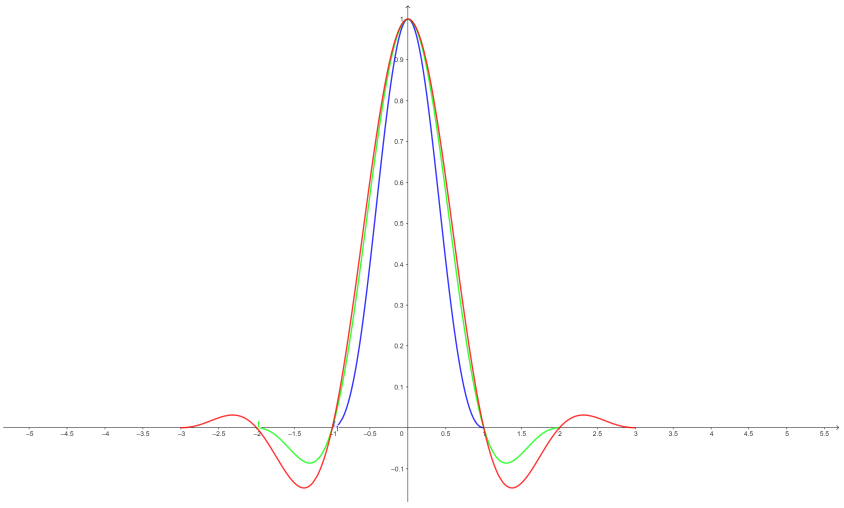

Figure 3.59: Lanczos kernel with $a = 1$ in blue, $a = 2$ in green, and $a = 3$ in red.

$$L(x,y) = L(x) \cdot L(y) \tag{3.10}$$

In OpenCV, the a parameter is set in such a manner that the Lanczos Kernel has an 8x8 dimension, thus taking into consideration 64 pixels on the output pixel calculation, which in the performed test, led to the result that can be seen in figure 3.60. The outcome of the Lanczos interpolation method was the one of higher quality among the obtained results, on a macro scale it almost did not cause any modification on the image. Although the result is good, on a smaller scale, it is possible to realize that the texture of the bricks of the image, a high frequency feature was lost in some regions, or turned into small circles. This disturbance in the interpolation of small details is a consequence of the Gibbs phenomenon, which happens due to the overshoot and undershoot that the Lanczos method performs when interpolating signals of significant difference between them, as exemplified in figure 3.61.

Although the quality of the result of the interpolation method is a important aspect on choosing which method to use, it is also important to consider the computational complexity of each method,

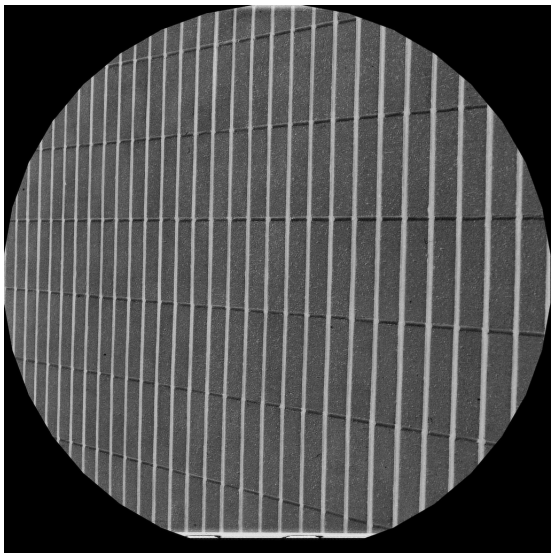

Figure 3.60: Figure 3.50, after several rotations with Lanczos interpolation method over 8x8 neighborhood.

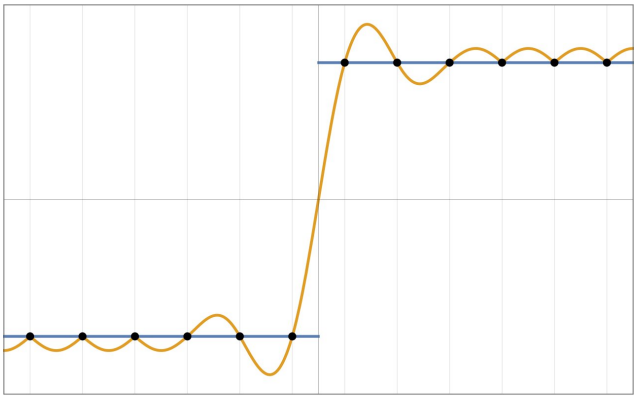

Figure 3.61: Gibbs phenomenon representation on performing a Lanczos interpolation in a step signal.

as the computing time is a relevant matter on the development of a computer vision solution. With that in mind, the rotation test program was altered in order to proceed 1000 rotations operation, and record the time that was necessary to accomplish this task. The interpolation method that took less time to be processed was the fill outliers one, which needed 1.0153 seconds to perform the 1000 rotation, then the other methods processing times were compared to this one and the difference can be seen on table 3.1.

Regarding the difference in the required time for performing each method, although the Lanczos provides the best interpolation results, depending on the processing time of the final solution software, the bicubic interpolation may be a good alternative, as it also produces good results in comparison to other methods that not Lanczos, which needed more than 10 times the time of the fill outliers method to be processed, while the bicubic took around five.

Table 3.1: Processing time of different interpolation methods, compared to fill outliers one.

Interpolation method	Processing time (method/fill outliers)
Nearest neighbor	123%
Bilinear	182%
Bicubic	488%
Pixel area	181%
Lanczos	1057%

3.5 Parameters Tunning

In order for image processing algorithms to have the best possible result, it is necessary to evaluate the result of different parameters used as inputs. For the selection of these parameters, practical and theoretical information was considered

3.5.1 Canny

For the Canny parameters, which consist in the definition of the upper and lower thresholds for edge classification, as described in chapter 2, the initial value of these parameters was the one suggested by the creator of the algorithm, who says that in general the value of the lower threshold, if set as one third of the value of the upper threshold, in general good results of edge extraction are obtained [61].

However, it is still necessary by experience to select the upper threshold value, and then confirm that this difference between the thresholds works well for this specific application. To do this, a software was developed, ans it allows one to change the value of the thresholds in real time and see the result, as can be seen in the figure 3.62. It is therefore possible to obtain experimentally the best value of the parameters for each application.

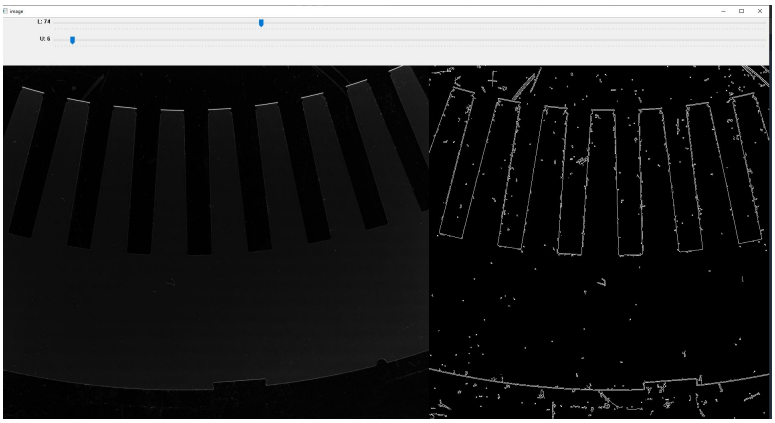

Figure 3.62: Canny parameters tunning software interface.

The parameters of the Canny algorithm set as described above, once defined, can be used to measure laminations of different sizes and with different sensor positions within the limits of

the mechanical design if the same sensor, lens and illumination are used. It is only necessary to re-evaluate the parameters if any of the previous components are replaced by another with an alternative configuration, and thus the contrast or illumination effect on the sample changes.

3.5.2 Morphological operations

For the kernel sizing of the morphological operations a similar strategy was implemented. A software was developed that using the arrow keys, it is possible to alter the size of the kernel both in number of rows and columns, and see in real time the result of the morphological operation with the new kernel size, until the most adequate value for the application is found.

As for the parameters of the morphological operations, they follow the same logic of the parameters defined for the use of the Canny method. As long as the vision system setup remains the same, the same parameters can be used for the inspection of any lamination. But in this case, it can be valid to analyze the best kernel for laminations of different diameters, since if a smaller kernel is sufficient for another case, it can lead to fewer processing requirements

3.6 Mechanical Project

For the use of the software developed in a factory environment, a mechanical design proposal was conceived, which would support the computer vision components, as well as the positioning of the laminations, which have large dimensions, for image acquisition of all its regions. In this first version of the mechanical project, it was designed to insert the laminations manually by the operator.

One aspect that was taken into consideration when developing the project was to select all the components, or as close to it as possible, from the same supplier, in order to simplify the acquisition of the parts of the project, and even making it possible to negotiate values given the number of components.

The project can be divided into three main parts, one being the support of the computer vision system, another is the vision system itself, and lastly there is the rotary table. The first part has the purpose of supporting the vision system above the rotating table, in a way that there is no contact between them, and it also has the function of moving the vision system in a parallel direction to the floor. The vision system is the part that contains the sensor and the lighting, besides being responsible for the movement of the sensor perpendicularly to the floor. The rotating table, on the other hand, has the objective of being the support for the lamination to be analyzed, as well as rotating it.

Starting with the vision system (figure 3.63), this part of the project aims to position, and support all the components of the vision system. Because the structure has medium dimensions, it was decided to use type B aluminium extrusions, 8 mm slot, 30 mm x 30 mm profile, as there are no great stresses on the structure.

Regarding the sensor, it is fixed to an aluminium profile that is in a horizontal position, using an aluminium plate as an intermediary, figure 3.64. This way, the plate is attached directly to

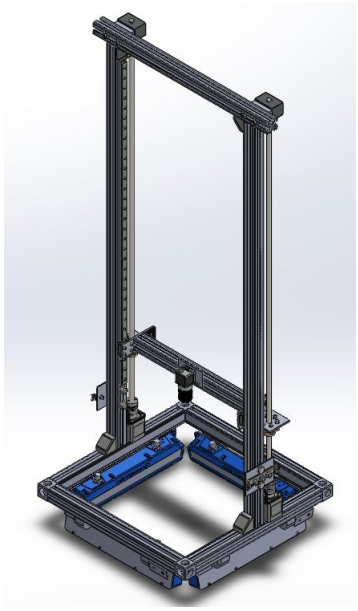

Figure 3.63: Vision system mechanical project.

the profile using two screws that are threaded into T-type nuts. The sensor, on the other hand, is attached to the aluminium plate using the threaded holes provided for M3 screws. The extrusion where the sensor is fixed is supported on each side by another aluminium plate, which has the purpose of connecting this extrusion to two parts, being these a lead screw nut and a linear carriage, as can be seen in figures 3.65, 3.66. The connection of the extrusion with the lead screw nut has the role of moving the extrusion in the vertical direction, while the connection with the linear carriage aims at stability and guidance of the movement. As for the linear carriage, the two carriages are not the same; while one has its bearings fixed, the other has freedom in the Y direction, the extrusion's axial direction, in order to compensate for constructive dimensional errors.

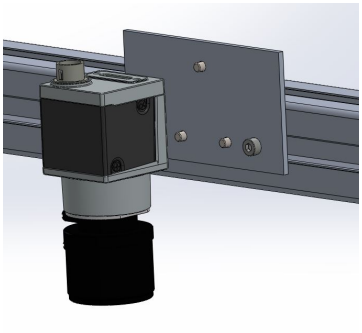

Figure 3.64: MV-SENS-300 fixing.

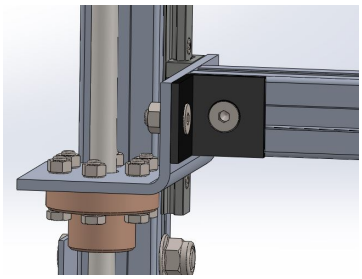

Figure 3.65: Sensor extrusion conection.

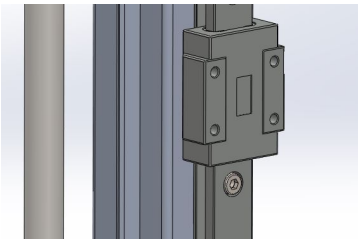

Figure 3.66: Linear carriage of sensor extrusion.

To produce the vertical movement two stepper motor type NEMA 17, 2 poles 4 wires, with 1.8º step angle, are used, figure 3.67. A shaft coupler is used with these motors to couple a trapezoidal lead screw with 10 mm diameter and 2 mm step. Given the motor and spindle step, the sensor

can be positioned with an accuracy of 0.01 mm. The sensor travel length is 725 mm, and can be positioned from 335 mm to 1060 mm away from the lamination, meeting the dimensioned distances for measuring all lamination of different diameters produced in the company.

For movement in the direction parallel to the ground, a simple and economic solution found was to install rollers on the sides of the vision system, as can be seen in the figure 3.68. For this, aluminium plates are also used as intermediates for the connection of the rollers to the extrusions structure. Both this aluminium plate and the one used to fix the sensor's support extrusion, can be produced by laser cutting, using bending as an auxiliary process to achieve the desired shape.

As for the lighting system, being of the darkfield type, it needs to be close to the object of analysis, and then it is located at the bottom of the vision system. In the CAD model presented is represented the DFL-12 model, the first to be chosen for the project, however, the coupling of the final chosen darkfield solution would be attached in a similar manner, not significantly altering the mechanical design.

The final dimension of the vision system part of the mechanical design is 500 mm x 500 mm for the base, and 1100 mm high.

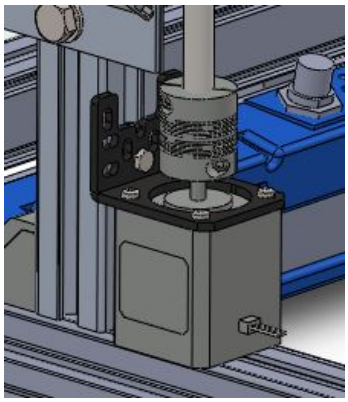

Figure 3.67: Stepper motor for vertical positioning of the sensor.

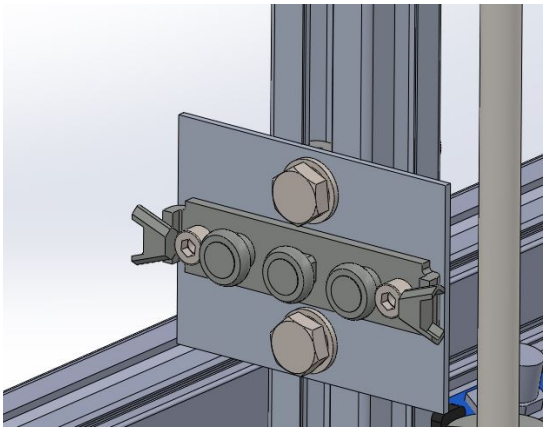

Figure 3.68: Rollers on the side of the vision system.

Regarding the support structure of the vision system (figure 3.71), as it is a larger structure and will have to support a weight in its centre, it was decided to use extrusions with a larger profile than the one used in the vision system. Therefore, this structure is formed by profile extrusions 45 mm x 45 mm, type B with a 10 mm slot. The height of the vision system support is 1200 mm.

This part of the project connects with the vision system at two points. By the roller rail, which will be the part responsible for supporting the weight of the system, and allowing its movement in the direction parallel to the floor, and also by a lead screw nut, which gives movement to the vision system in a similar way as the vertical movement of the sensor is done, as it can be seen in figures 3.69, 3.70.

As for the components of this part of the project related to motion generation, these are the same as those used in the sensor movement, regarding the lead screw, stepper motor, and shaft coupler.

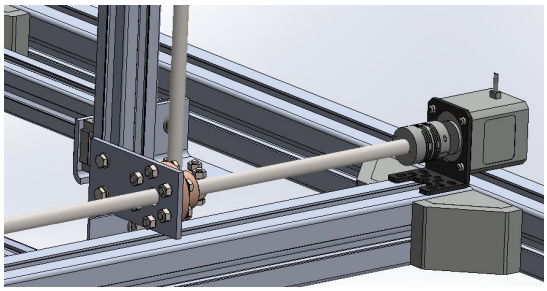

Figure 3.69: Transmission system for horizontal positioning of the sensor.

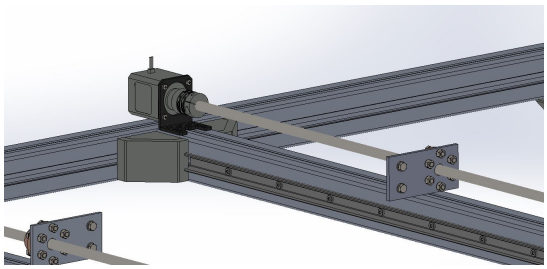

Figure 3.70: Rail for vision system rollers.

Figure 3.71: Vision system support project.

At last, there is the rotating table that will support the laminations to be analysed, and position it. As mentioned earlier in this work, the table should be dark in color to increase the contrast between the lamination and its background to improve the identification of the edges by computer vision software. The proposal is to use a wooden table, painted black, with the objective of using a material that has low specular reflection, and also has low density, in order to have a low moment of inertia, facilitating the start and stop of the table's rotation movement.

For moving the table, a belt pulley system was chosen (figure 3.73), in order to be able to use a small motor to move the table, as well as to multiply its torque, and increase the accuracy of table positioning. Since there is not a very large load to be moved, it was decided to use the same stepper motor that is used to move the sensor, and T10 profile pulleys and belts.

To allow relative movement of the table in relation to the structure that supports it, it is mounted

on a bearing that has the outer ring toothed with the T10 profile, and 34 teeth, thus simplifying the system assembly, and avoiding the use of another component. This bearing can be seen in the figure 3.72. The pulley coupled to the motor has 12 teeth, thus achieving an angular positioning accuracy of the rotary table of up to 0.635º.

Figure 3.72: Bearing with toothed outer ring of T10 profile [81].

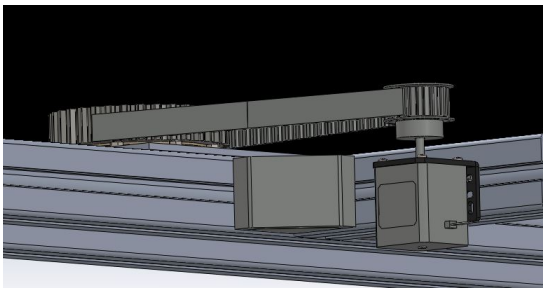

Figure 3.73: Rotary table belt-pulley system.

Chapter 4

Results

4.1 Computer Vision

As shown in the figure 3.4, the largest field of view required for the vision system will be for the 900 frame motor, with a field of view of 300 mm in the vertical direction of the sensor, and in this case the pixel millimeter ratio is 0.0817. However, there is still the best case, which is the 315 frame motor, where the field of view is only 170 mm in the same direction, as can be seen in the picture 4.1.

If one does the same relation as in the previous case, it is possible to verify what the pixel per millimeter relation will be in the best case, for the 315 motor, and in this case a value of 0.046 millimeters per pixel is obtained. This value is still above the production tolerance, and it is not possible to inspect the quality using pixel level analysis within the production specifications even in the case where the best resolution is obtained, reaffirming the need for sub-pixel techniques.

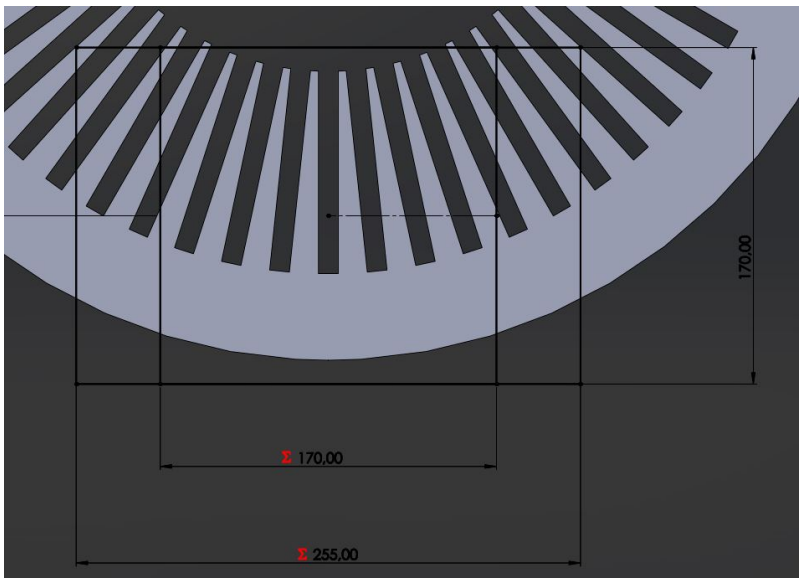

Figure 4.1: Required field of view for 315 frame motor

As commented in chapter 1, one of the important parameters to be analyzed in the proposed solution, is the processing time of the measurement software, given that if one intends to install the inspection system on the production line, the lamination analysis should happen in a time equal or shorter than the production time in order to avoid delays in the manufacturing process.

With this in consideration, and with the support of MVISIA in Brazil, the software for measuring tooth width and bottom slot circularity was tested on two different CPU's, the CVU-100 and the CVU-300. It is important to mention that in this test version of the program, the diameter of the bottom slot was measured, and the width of 3 teeth. It is also worth mentioning that an image acquired using the MV-SENS-300 with a 20MP sensor was used. As for the test on the CVU-100, the result obtained was 1.47 seconds to process the program completely. For the CVU-300 test, the result was drastically better, taking a little less than 0.5 seconds to process the program, that is, almost three times faster than its predecessor model.

As for the selected lighting system, it brought a very interesting highlight on the lamination burrs. The sensor exposure time that best highlighted the burrs in relation to the rest of the lamination was 0.01 second. In the images 4.2, and 4.3, it is possible to see how highlighted the burr is with this configuration, and what is the result of a photo acquired in exactly the same way, but from the opposite side of the plate, which has no burr.

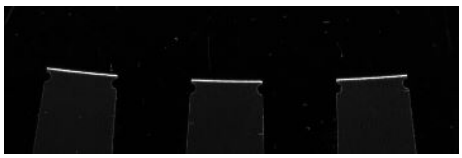

Figure 4.2: Lamination burr side lit with darkfield technique, and sensor exposure time of 0.01 second.

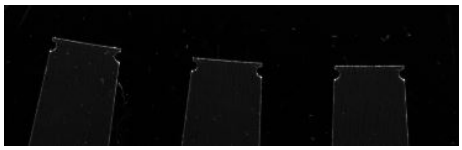

Figure 4.3: Lamination burr-free side illuminated with darkfield technique, and sensor exposure time of 0.01 second.

While for burr highlighting the best exposure time of the sensor was 0.01 second, for measuring bottom slot diameter and tooth width the chosen exposure time was longer. For measurement of these other features, it is necessary that the lamination stands out from the background completely, which is best achieved with an exposure time of 0.09 second.

The exposure time is relevant not only to obtain the image with the desired characteristics, but also because it needs to be added to the processing time, to reach the total time that the system takes to perform the inspection. Thus, adding the longest exposure time with the processing time, we obtain approximately 0.58 seconds of total time for inspecting 3 teeth and the bottom slot diameter.

As for the software's measurement capacity, tests were made, and it was possible to see that the program developed was capable of differentiating the width of teeth with a difference between them of approximately 20 micrometres. Firstly, tests were made with the software developed on an image acquired by a mobile phone that has 64 MP resolution, however, when analysing the image in detail, it exhibited low definition and high distortion compared with images acquired by industrial cameras and lenses. Regardless of that, tests were made with this image, and when

analysing teeth of different dimensions that were measured using a caliper, it was noticed that the program was able to identify which tooth was bigger or smaller. However, the measurement did not obtain the correct value at the level of tenths of millimetres, with an error of approximately 40 micrometres. One of the images resulting from this test can be seen in the figure 4.4.

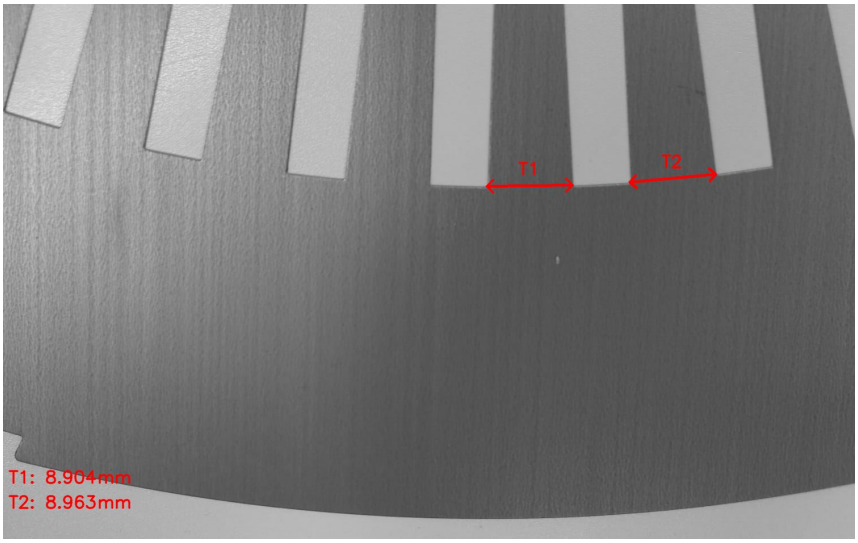

Figure 4.4: Testing measurement software on lamination image acquired by mobile phone.

Later, with the support of the quality engineering department of WEG's headquarters in Brazil, it was possible to test the program on better quality images (figure 4.5), both in terms of image quality and the lighting used in the acquisition. The results were even better, being possible to distinguish teeth with width differences of up to 20 micrometers. The teeth of the plate of this image were also measured with the help of a caliper. In this test, despite the success in identifying the largest and smallest teeth, a difference was still observed between the values measured with the caliper and the measurement made by the software, but smaller differences than those found in the previous test.

As for the measurement of the diameter of the bottom slot, in the sample where the real size was known, and where it was possible to make the measurement by software, the error was approximately 50%. However, the images were acquired using a mobile phone, which is not the most appropriate method, and may contain human error.

4.2 Mechanical Project

The result of assembling all the parts of the system can be seen in the figure 4.6. As can be seen, there is no contact between the rotary table and other parts of the project, in order that possible vibrations from the rotary table movement do not interfere on the quality of the images acquired by the sensor.

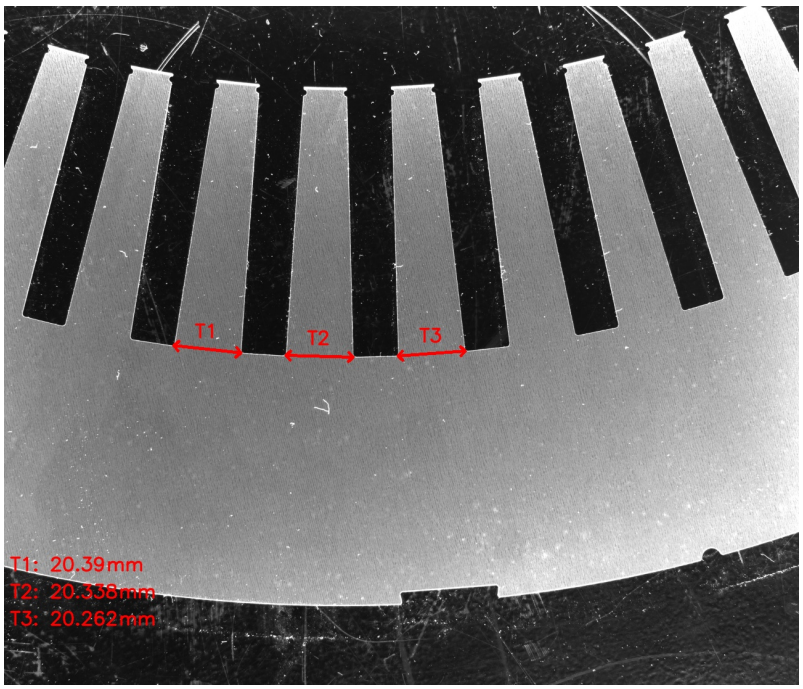

Figure 4.5: Test of measurement software on lamination image acquired by quality engineering department at WEG in Brazil.

Figure 4.6: Complete mechanical project.

Chapter 5

Conclusions and Future Work

5.1 Conclusion

The results of the computer vision program have shown a good level of accuracy in the analysis, allowing differentiation at the level of tens of micrometres, and it is still possible to improve this result with the addition of more techniques and refinement of what has already been done.

As mentioned in the chapter 4, there was a non-negligible error in the absolute values of measurement, but it was still possible to differentiate structures with different dimensions from one another. Among the samples in which it was possible to verify the measurement capacity of the developed software, the smallest size difference measured between teeth using a caliper was 20 micrometers. In this case, the calibration of the relationship millimetre per pixel was made in such a way that the tooth with the largest dimension was correctly measured, and the measured size of the smallest tooth smallest, but with an error of 20 micrometres. In this way, it was possible to verify which of the two was larger, and which was smaller, however with error.

Regarding the error of measurement, there are some hypotheses to justify it. First and most obvious is the possible human error in measuring the teeth, as these are measured using a caliper, and any variation in the positioning of the instrument on the teeth would cause a variation in the measurement result obtained. Furthermore, the caliper is not the ideal instrument for measurements in the order of hundredths of a millimetre, as it may cause a measurement error in this order of magnitude, and unfortunately it was not possible to access the ideal instruments for this type of analysis. An example of equipment that could be used to validate the measurement by software would be a digital profile projector, figure 5.1, which would allow the measurement of the structures of the lamination with micrometer precision, depending on the model to be used.

Another factor that may be influencing the measurement quality is the non-guarantee of a perfectly perpendicular positioning of the lens axis of the setup used in the image acquisition, to the lamination, since it was not possible to build the setup in loco. Deviations in the perpendicularity of the sensor positioning would lead to the acquisition of the bottom slot diameter with an elliptical profile not consistent with reality, due to the manner in which the lamination would be projected

Figure 5.1: Digital profile projector [82].

onto the sensor. This elliptical profile causes errors both in the measurement of the bottom slot diameter and the tooth width, as one measurement is related to the other.

When analyzing the points that the coordinates are stored to define the linear profile of the teeth, another situation was noticed that may probably be interfering with the measurement results. In figure 5.2, it is possible to see that the points do not follow a perfect line, which is a consequence of the characteristic of digital images of being represented by pixels, as well as the modifications caused in the image by its processing. This variation of the points outside the linear profile may be causing inaccuracy in the fitting algorithm, and it is possible that a preprocessing of the coordinates data before being inserted in the fitting, may improve the result of the measurements.

Although it is still not possible to measure with the precision of hundredths of millimetres as desired, as already mentioned, the software developed with the proposed setup, is able to differentiate structures that have only tens of micrometres of difference between them, thus, it was concluded that through computer vision it is possible to perform the proposed analysis of dimensional and geometric inspection of laminations of electric motors.

About the rollover or burr measurement, the conclusion was that the proposed use of darkfield illumination for burr highlighting works well. However, it was not possible to have access to darkfield lighting in situ during the work, which would be needed in order to perform a measurement via software, and subsequently via micrometer or microscopic analysis for validation. So no conclusion was reached as to whether it would be possible to measure this structure using classical vision as was done with the other structures in the lamination.

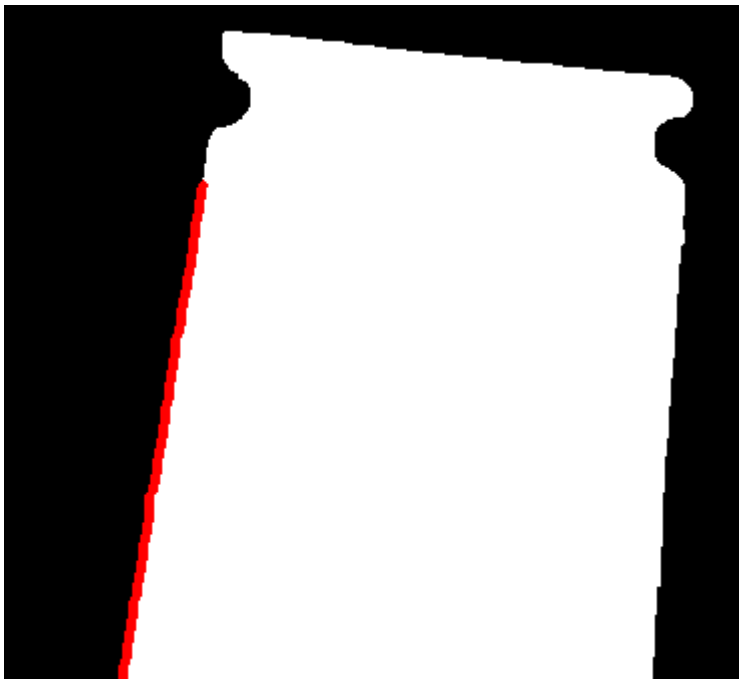

Figure 5.2: Detected points for linear regression at tooth edge definition.

As for the processing time of the software, the result was very positive. It was also proven that the CVU 300 is in fact the best option for this application due to its processing time of approximately half a second, and being approximately three times faster than its predecessor model. The time of half a second for the analysis of three teeth, allows one to conclude that with approximately 10 seconds it would be possible to analyse a lamination of 60 teeth in its entirety, which is approximately the same time as the punching of an entire lamination. Thus, it is possible to use an inspection system by computer vision coupled to the laminations production line without any production delay.

5.2 Future Work

5.2.1 Vision System

One of the improvements that could be implemented in the software is the pre-processing of the data used in the fitting algorithms, such as excluding outliers, and keeping all coordinates within a standard deviation. This would increase the accuracy of the measurement, and would not bias the result, given that the way the slots are manufactured, the linear profile of the tooth edges is in fact rectilinear. Furthermore, the testing and introduction of modified and optimized Canny methods found in the literature and cited in the chapter 2, could also be helpful in improving the results.

In order to improve the results, when acquiring the hardware, it is important to calibrate it, acquiring the intrinsic parameters of the sensor and lens setup presented in chapter 2, and use

them in conjunction with the formula 2.2. Thus, we can find the real positions of the pixels, correct the distortion, and improve the accuracy of the system in a simple manner.

It is also necessary the validation of the rollover measurement software, which requires the use in situ of the inspection system, test it, and then validate it by micrometer or microscopic analysis, which was not possible during the development of this work. After this validation, if the software gives the expected result, it could be implemented. Otherwise, it would be worth using a classifying software to analyse the burrs highlighted by the darkfield technique. This experience would consist in training a neural network with images with burrs within the tolerance as the negative case, and with burrs within the tolerance as the positive case, and then classify if the burrs of the lamination are or not compliant.

It would be also interesting to analyse the possibility of reducing the processing time of the program, optimizing it. Reducing the processing time of what was developed, allows the introduction of new functionalities that can increase the precision of the measurement.

5.2.2 HTTP Server

In order to allow interaction between the user of the system, and the machine, it is interesting to create a Hypertext Transfer Protocol (HTTP) server. With this server, a WEB page would be made where it would be possible to give inputs to the system, as well as see its outputs. With an HTTP server, users of the system could have access to what is being measured without being beside the inspection machine, whenever they wanted updates on the production quality of the laminations.

This same server could be accessed and be permanently displayed in the Human Machine Interface (HMI) of the inspection system, in a way to be useful in the presentation of the inspection results, as well as to serve as input for the type of plate to be analyzed. In an ideal scenario, when implementing the system in the production line, its input, the nominal values of the laminations outer and inner diameter to be analyzed, which is registered in the production order, would be retrieved from the company's Enterprise Resource Management (ERP) system. However, in the proposed prototype, the operator could insert the lamination information in this HTTP server through the HMI installed in the inspection system. With that information, the system could automatically place the sensor in the ideal position regarding the sample.

The use of this server would also be useful for communicating with a company database, making data resulting from lamination inspection accessible, enabling the creation of a report on the quality of the samples analysed, as well as the need for intervention in the production process depending on the results obtained.

5.2.3 Mechanical project

In order to complement the mechanical project, some more components need to be added. It is necessary the definition of the drivers that will control the stepper motors, as well as the Programmable Logic Controller (PLC) for general control of the system, and also an HMI for user interaction.

As for the drivers, it is a common and easy component to acquire, it just needs to be within the specifications of the motor selected for the project, which is also a widely used motor. So the driver selection is simple and it is enough to check with the WEG electronic materials supplier which compliant option they possess and purchase it.

Regarding the PLC, WEG has the PLC500 model (figure 5.3), which has its programming based on CODESYS. This model has the necessary specifications for control of the system, besides having support for various industrial communication protocols, which would allow the sending of information obtained by the inspection system to a database of the company, as well as wireless communication with the CVU 300 if desired [1].

Figure 5.3: WEG - PLC500 [1].

One component that also remained to be defined were the mechanism's feet. There were two options, one to simply place the system on the ground, figure 5.4, and another with a complementary ground fixing option, figure 5.5. Ideally, one would use the option of just landing the structure on the ground, however, due to the vibrations of the structure after movements, the ground fixing option may be required.

Figure 5.4: Foot for aluminium extrusion structure [83].

Figure 5.5: Foot for aluminium extrusion structure, with ground fixing [83].

Also in relation to the effect of movements on the structure, when analysing its behaviour, if necessary it is still possible to install reinforcements in the corners, such as the one that can be seen in figure 5.6.

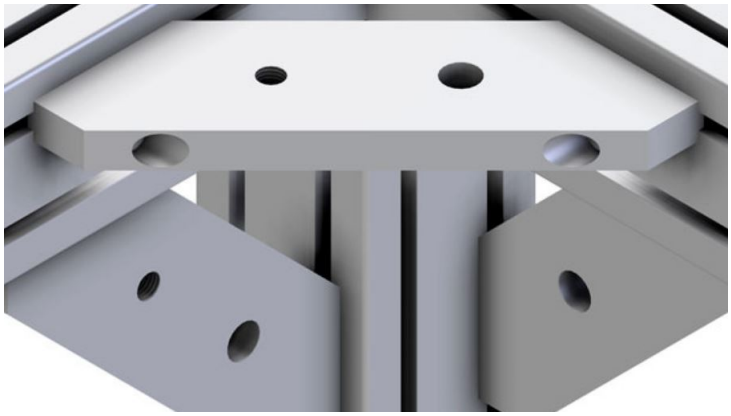

Figure 5.6: Reinforcement for aluminium extrusion structures [83].

In addition, it is also interesting to perform a load analysis of the stepper motors to be used in the system. The loads and inertias of the system are low, however, it is interesting to analyse them in conjunction with the the motor behaviour in order to document and ensure that the model selected is adequate.

Regarding the positioning of the laminations on the rotary table, in the current model, this would be done by means of reference markings. Although the bottom slot diameter is variable according to the electrical design, the outer diameter is standardized according to the frame to be built. With this in consideration, the table would have markings for each outer diameter size, which the operator would use as a positioning reference, and this would already be sufficient, given that the inspection software is immune to small positioning deviations. However, for a greater repeatability of the positioning of the samples, in the future it would be interesting the development of a self-centering system of the lamination, with some type of mechanism similar to the one in the figure 5.7, or with a similar functioning to that of the lathes' chuck.

5.2.4 Others Computer Vision applications in WEG's Plant

Since this is a proof-of-concept study of the use of WEG artificial vision products on the company's production line, the company is also interested in identifying other applications of this type of system in the manufacturing environment, so as to obtain the advantages of using vision systems in production at the company's Portuguese unit.

For this purpose, visits to the production site were carried out. Also regarding laminations, it was noticed that it is interesting to verify the production of dovetails on the lamination's outer edge. This plate characteristic is used to position the clamps that keep the stator block cohesive, and also for plate alignment during block assembly. Therefore, the analysis of this structure is interesting to ensure that the clamping and alignment of the laminations can be performed without difficulty.

Figure 5.7: Self-centering optics mount [84].

By observing how the stator block is assembled, a possible improvement in the pressing process was noticed. After all the laminations of the block are stacked and aligned, they are placed in a hydraulic press, which axially presses the block until it reaches a previously determined dimension. Currently, the checking of the achievement of this dimension is done by manual inspection. One possibility would be the integration of a vision system, which would constantly measure the size of the block. This vision system could be connected to a PLC, in a way that the operator could constantly activate the press, and as soon as the desired dimension was reached, the PLC would be informed, and it could interrupt the electromechanical circuit of activation of the press. With a system like this, the pressing process would be faster, as it would not require successive manual measurements, and would also have greater accuracy.

A possible use of a vision system in the production of rotors made of cast aluminium has also been observed. In casting manufacturing processes, it happens that not all parts of the solid to be formed are well filled with the melted metal, and geometrical deformations occur. At [85] it is possible to see an application of convolutional neural networks and 3D vision, for deformations inspection in a cast part. A similar use of computer vision could be used in the manufacture of cast aluminium rotors, with the aim of identifying how different from the expected shape the rotor has become. With this inspection it would also become possible to parameterise the quality of the casting, something difficult to do since deformations can occur in very different ways from one casting to another.

Concerning the rotors manufactured by casting, the moldings used, as they are subjected to high temperatures repeatedly, with time they deform, and can start to generate deformations in the rotors. To avoid this, it is possible to create an inspection station for these mouldings, so that after their use, and cooling, their main geometries could be examined by a vision system. The dimensions obtained in the inspection could be crossed with the existing ones in the ERP system, and if the molding is outside the dimensional tolerances, it would be informed to the person in

charge of the section to do a deeper evaluation of the need to replace this molding. This type of verification still meets the Critical Tool Control methodology that has been established in the company.

It was also identified a possibility of applying computer vision to diagnose shaft warping. When assembling the stator block on the shaft, heat transfer is involved, since when the block is inserted, it is at a high temperature. This process leads to stress relaxation in the shaft, which naturally causes it to bend. The tolerance of deviation of the shaft from its central axis has different values in each shaft section, varying according to the purpose of each section, for example, there are different tolerances between bearing mounting regions and the section that will be coupled to the load. With a vision system to check the complete shaft, it would be possible to examine the deflection of all sections at the same time, saving time and simplifying the process.

A motor characteristic that may also be worth inspecting is the air gap. This is a critical characteristic of motor construction, and could be inspected without great difficulties after the rotor is assembled in the frame. The advantage of inspecting the air gap by means of computer vision is the possibility of automatically checking it along its entire perimeter, while a manual inspection would assess it by regions.

The last computer vision inspection application identified was the analysis of the production of motor certifications and characteristics identification plates. These plates are engraved by laser, and it would be possible to install a camera above the engraving station, which would verify what was engraved and confirm that the information is in accordance with the description in the company's ERP. This way it would be possible to confirm that the correct information has been recorded, since errors in the engraving of these plates are critical.

Appendix A

Tooth Width and Circularity Measurement Software

The following code was developed using Python 3.10.7, with the libraries OpenCV 4.6.0, and NumPy 1.23.3, and is executed as follows.

```
import math
import cv2 as cv
import numpy as np

from utils.tools import crop, draw_doublearrow, midpoint, measure, resize
from scipy.optimize import least_squares

#----------------CONSTANTS-----------------

# Text font
FONT = cv.FONT_HERSHEY_SIMPLEX


## Coordinates in the order X1, X2, Y1, Y2
# 3 central teeth region crop coordinates
ROI_CIRCULARITY = [115*5, 452*5, 304*5, 407*5]

# Each tooth crop coordinates
ROI_TOOTH = [[200*5, 284*5, 68*5, 358*5],
             [299*5, 402*5, 66*5, 368*5],
             [401*5, 508*5, 67*5, 355*5]]

# Kernels for morphological operations
MORPH_KERNEL = (7, 7)
MORPH_KERNEL_CIRC = (8,8)
DENOISE_KERNEL = (2,1)

# Scale for output image resize
```

```
RESIZE_SCALE = 30

# Output image features properties and measuring parameters
DECIMAL_PLACES = 3
COLOR = (0, 0, 255)
MM_PX = 0.055762
SCALE = 'mm'

# Denoising parameter
BOTTOM_SLOT_IMG_PERCENTAGE = 20

#--------------CONSTANTS_END--------------



#---------------FUNCTIONS-----------------

def compare_equal_zero(arr):
    """
    Compares all the values of an array with 0 and returns True if all the values
↪   are equal to 0, else False.

    Args:
    arr(array): Array with instances to be compared.
    """
    for i in arr:
        if i != 0:
            return False
    return True


def compare_different_zero(arr):
    """
    Compares all the values of an array with 255 and returns True if all the values
↪   are equal to 255, else False.

    Args:
    arr(array): Array with instances to be compared.
    """
    for i in arr:
        if i == 0:
            return False
    return True


def horizontal_edges_extraction(img, img_cannied):
    '''
    Extract just the horizontal edges of a cannied image.
```

```
    Args:
    img(mat): Original image.
    img_cannied(mat): Cannied image.

    Returns:
    hor_edges(mat): The cannied image with only the horizontal edges
    '''

    # Sobel Edge Detection on the X and Y axis
    sobelx = cv.Sobel(src=img, ddepth=cv.CV_64F, dx=1, dy=0, ksize=3)
    sobely = cv.Sobel(src=img, ddepth=cv.CV_64F, dx=0, dy=1, ksize=3)

    # Scale gradient values
    sobelx = cv.convertScaleAbs(sobelx)
    sobely = cv.convertScaleAbs(sobely)

    # Obtain edges points coordinates
    edge_coord = np.nonzero(img_cannied)

    # Verify higher gradient direction of edge points
    for i in range(len(edge_coord[0])):
        row = edge_coord[0][i]
        col = edge_coord[1][i]
        # If the X direction gradient is higher, set the edge point as 0.
        if abs(sobelx[row, col]) > abs(sobely[row, col]):
            img_cannied[row, col] = 0

    return img_cannied


def trim_edges(img, columns_toTrim, sensitivity):
    '''
    Trim edges of the horizontal edges, in order to remove possible curvatures at
 ↪  the end of it.

    Args:
    img(mat): Horizontal edges image.
    columns_toTrim(int): Number of columns to trim.
    sensitivity(int)

    Returns:
    img(mat): Horizontal edges with ends trimmed.
    '''

    row_index = []
    edge_start = []
    edge_end = []
```

```
    # Make an array with the index of edge pixels per column. If none is found,
    ↪  index = 0.
    for i in range(img.shape[1]):
        row_index.append(np.argmax(img[ : , i]) )

    # Find left and right ends of the edges, and save them.
    for i in range(len(row_index)):
        left_part = row_index[i-sensitivity : i]
        right_part = row_index[i : i + sensitivity]

        if compare_equal_zero(left_part) and compare_different_zero(right_part):
            edge_start.append(i)

        if compare_different_zero(left_part) and compare_equal_zero(right_part) and
        ↪  i > sensitivity:
            edge_end.append(i)

    # Trim left side of edges
    for i in edge_start:
        img[:, i:i+columns_toTrim] = 0

    # Trim right side of edges
    for i in edge_end:
        img[:, i-columns_toTrim:i] = 0

    return img


def objective(params, x, y):
    '''
    Define the objective function for fitting a circle to a set of points.

    Args:
    params(array): Array with xc, yc and r parameters.
    x(float): X coordinate of a point.
    y(float): Y coordinate of a point.

    Returns:
    Circle equation with the parameters and point coordinate.
    '''
    xc, yc, r = params
    return (x - xc)**2 + (y - yc)**2 - r**2


def fit_circle(x, y):
    '''
    Fit a circle to a set of points.

    Args:
```

```
    x(array): Set of X coordinate of points
    y(array): Set of y coordinate of points

    Returns:
    xc(float): x coordinate of center of the fitted circle.
    yc(float): y coordinate of center of the fitted circle.
    r(float): Radius of the fitted circle.
    '''
    # Initial estimate of the circle center and radius
    xc0, yc0 = np.mean(x), np.mean(y)
    r0 = np.sqrt((x - xc0)**2 + (y - yc0)**2).mean()

    # Use least squares optimization to find the best-fit circle
    result = least_squares(objective, [xc0, yc0, r0], args=(x, y))
    xc, yc, r = result.x

    return xc, yc, r


def edge_limits(img, edge_limits_identification_parameters):
    '''
    Finds the upper and lower limit of both sides linear profile of tooth edges.

    Args:
    img(mat): Contour or segmented tooth image without noise.
    edge_limits_identification_parameters(array): Step, threshold and steps back
↪   parameters respectively. Step is the number of rows skipped in the evaluation
↪   of the linear profile. Threshold is the sensitivity of evaluating if the
↪   profile is still linear or not. Steps_back is the number of steps to go
↪   backwards and define as the end of the linear profile.

    Returns:
    left_lower_row_index(int): Row index of the lower pixel of left side linear
↪   edge profile
    left_upper_row_index(int): Row index of the approximate upper pixel of left
↪   side linear edge profile
    right_lower_row_index(int): Row index of the lower pixel of right side linear
↪   edge profile
    right_upper_row_index(int): Row index of the approximate upper pixel of right
↪   side linear edge profile
    '''

    # Define edge limits identification parameters
    step = edge_limits_identification_parameters[0]
    threshold = edge_limits_identification_parameters[1]
    steps_back= edge_limits_identification_parameters[2]

    # # Uncomment to visualize the algorithm result. Also need to uncomment the
    ↪  cv.imshow and cv.circle along the function.
```

```
# points_img = img.copy()
# points_img = cv.cvtColor(points_img, cv.COLOR_GRAY2BGR)

###------------------------------LEFT_SIDE------------------------------###
# Begin variables
calculated_differences = []
left_lower_row_index = 0

# Extract first column of pixels
first_column = img[:, 0]

# Find row index of first non-zero pixel in first column, counting from top to
↪  bottom
row_index = np.argmax(first_column != 0)

# Check if any non-zero pixels were found in the first column
if row_index == 0 and first_column[row_index] == 0:
    left_lower_row_index = img.shape[0] - 1
else:
    left_lower_row_index = row_index

# Find last row from down to up that is the last row of linear tooth profile
counter = 0
for i in range(left_lower_row_index, -1, -step):
    counter = counter + 1

    # Edge column in last row
    nonzero_column_index = np.nonzero(img[i])[0][0]

    # Edge column in last row minus step
    nonzero_index_step = np.nonzero(img[i - step])[0][0]
    index_difference = abs(nonzero_index_step - nonzero_column_index)

    # Compute mean difference between edges columns by analyzed rows
    calculated_differences.append(index_difference)
    mean_differences = np.mean(calculated_differences)
    # cv.circle(points_img, (nonzero_column_index, i), 3, (0, 0, 255), -1)
    # cv.circle(points_img, (nonzero_index_step, i-step-5), 3, (0, 255, 0), -1)

    # Verify if edge column computed in this iteration is above the mean (end
    ↪  of linear profile)
    if index_difference > threshold * mean_differences:
        left_upper_row_index = i + steps_back * step
        # cv.circle(points_img, (nonzero_column_index, left_upper_row_index),
        ↪  3, (255, 0, 0), -1)
        break
###-----------------------------LEFT_SIDE_END-----------------------------###
```

```
↪ ###--------------------------------RIGHT_SIDE--------------------------------###
# Begin variables
calculated_differences = []
right_lower_row_index = 0

# Extract last column of pixels
last_column = img[:, img.shape[1]-1]

# Find row index of first non-zero pixel in last column, counting from top to
↪ bottom
row_index = 0
row_index = np.argmax(last_column != 0)

# Check if any non-zero pixels were found in the last column
if row_index == 0 and last_column[row_index] == 0:
    right_lower_row_index = img.shape[0] - 1
else:
    right_lower_row_index = row_index
counter = 0
# Find last row from down to up that is the last row of linear tooth profile
for i in range(right_lower_row_index, -1, -step):
    counter = counter + 1
    # Edge column in last row
    nonzero_column_index = np.nonzero(img[i])[0][len(np.nonzero(img[i])[0])-1]

    # Edge column in last row minus step
    nonzero_index_step = np.nonzero(img[i -
    ↪ step])[0][len(np.nonzero(img[i-step])[0])-1]
    index_difference = abs(nonzero_index_step - nonzero_column_index)

    # Compute mean difference between edges columns by analyzed rows
    calculated_differences.append(index_difference)
    mean_differences = np.mean(calculated_differences)
    # cv.circle(points_img, (nonzero_column_index, i), 3, (0, 0, 255), -1)
    # cv.circle(points_img, (nonzero_index_step, i-step-5), 3, (0, 255, 0), -1)

    # Verify if edge column computed in this iteration is above the mean (end
    ↪ of linear profile)
    if index_difference > threshold * mean_differences:
        right_upper_row_index = i + steps_back * step
        # cv.circle(points_img, (nonzero_column_index, right_upper_row_index),
        ↪ 3, (255, 0, 0), -1)
        break

↪ ###-----------------------------RIGHT_SIDE_END-----------------------------###

# cv.imshow('points_img', points_img)
# cv.waitKey(0)
```

```
    # cv.destroyAllWindows()

    return left_lower_row_index, left_upper_row_index, right_lower_row_index,
    ↪  right_upper_row_index


def tooth_edges(img, tooth_edge_limits):
    '''
    Find linear function that represents the lineard edge profile.

    Args:
    img(mat): Tooth edges image.
    tooth_edge_limits(array): Array with left side linear profile lower limit, left
↪   side linear profile upper limit, right side linear profile lower limit, right
↪   side linear profile upper limit, respectively.

    Returns:
    m_l(float): Left linear edge function slope.
    x0_l(float): x coordinate of a point in the linear function that representes
↪   the left tooth edge.
    y0_l(float): y coordinate of a point in the linear function that representes
↪   the left tooth edge.
    m_r(float): Right linear edge function slope.
    x0_r(float): x coordinate of a point in the linear function that representes
↪   the right tooth edge.
    y0_r(float): y coordinate of a point in the linear function that representes
↪   the right tooth edge.
    '''
    ###------------------------------LEFT_SIDE------------------------------###
    lower_row_left = tooth_edge_limits[0]
    upper_row_left = tooth_edge_limits[1]

    # Define the number of rows that are part of the linear profile
    rows = list(range(upper_row_left, lower_row_left + 1))
    points = []

    # Save the X coordinate of the left edge points in all rows of the linear
    ↪  profile
    for i in rows:
        points.append([i, np.nonzero(img[i])[0][0]])

    points = np.array(points, dtype=np.float32)

    # Fit the points in a line and compute the line slope
    [yv_l, xv_l, y0_l, x0_l] = cv.fitLine(points, cv.DIST_L2, 0, 0.1, 0.1)
    m_l = yv_l / xv_l
    ###---------------------------LEFT_SIDE_END---------------------------###

    ###-----------------------------RIGHT_SIDE-----------------------------###
```

```
    lower_row_right = tooth_edge_limits[2]
    upper_row_right = tooth_edge_limits[3]

    # Define the number of rows that are part of the linear profile
    rows = list(range(upper_row_right, lower_row_right + 1))
    points = []

    # Save the X coordinate of the right edge points in all rows of the linear
    ↪  profile
    for i in rows:
        points.append([i, np.nonzero(img[i])[0][len(np.nonzero(img[i])[0])-1]])

    points = np.array(points, dtype=np.float32)

    # fit the points in a line and compute the line slope
    [yv_r, xv_r, y0_r, x0_r] = cv.fitLine(points, cv.DIST_L2, 0, 0.1, 0.1)
    m_r = yv_r / xv_r
    ###---------------------------RIGHT_SIDE_END--------------------------###

    return m_l, x0_l, y0_l, m_r, x0_r, y0_r


def find_intersection(xc, yc, r, m, x0, y0):
    '''
    Find intersection points between the bottom slot diameter and the a linear.

    Args:
    xc(float): x coordinate of center of the fitted circle.
    yc(float): y coordinate of center of the fitted circle.
    r(float): Radius of the fitted circle.
    m(float): Linear function slope.
    x0(float): x coordinate of a point in the linear function.
    y0(float): y coordinate of a point in the linear function.

    Returns:
    x(float): x coordinate of the intersection point.
    y(float): y coordinate of the intersection point.
    '''

    a = 1 + m ** 2
    b = -2 * xc + 2 * m * (y0 - yc - m * x0)
    c = xc ** 2 + (y0 - yc - m * x0) ** 2 - r ** 2
    discriminant = b ** 2 - 4 * a * c

    if discriminant < 0:
        # No intersection points
        return []

    elif discriminant == 0:
```

```
        # One intersection point
        x = -b / (2 * a)
        y = m * (x - x0) + y0
        return (x, y)

    else:
        # Two intersection points
        x1 = (-b + math.sqrt(discriminant)) / (2 * a)
        y1 = m * (x1 - x0) + y0
        x2 = (-b - math.sqrt(discriminant)) / (2 * a)
        y2 = m * (x2 - x0) + y0
        if x1 < 0 or y1 < 0:
            return (x2, y2)
        else:
            return (x1, y1)


def apply_opening(img, kernel_size):
    '''
    Perform an opening operation.

    Args:
    img(mat): Image to perform the opening operation.
    kernel_size(array): Kernel of the opening operation.

    Returns:
    img(mat): Image after opening operation.
    '''

    kernel = cv.getStructuringElement(cv.MORPH_RECT, kernel_size)
    opened = cv.morphologyEx(img, cv.MORPH_OPEN, kernel)
    return opened


def apply_closing(img, kernel_size):
    '''
    Perform a closing operation.

    Args:
    img(mat): Image to perform the closing operation.
    kernel_size(array): Kernel of the closing operation.

    Returns:
    img(mat): Image after closing operation.
    '''

    kernel = cv.getStructuringElement(cv.MORPH_RECT, kernel_size)
    closed = cv.morphologyEx(img, cv.MORPH_CLOSE, kernel)
    return closed
```

```
def delete_non_bottom_slot(img, area_to_maintain):
    '''
    Set to 0 all pixels of rows that are not considered part of the bottom slot
↪   edges rows.

    Args:
    img(mat): Image to clear non bottom slot edge rows.
    area_to_maintain(int): Percentage of the rows in the image that are not to be
↪   changed.
    '''
    # Find mean row of bottom slot edges
    points_non_zero = np.nonzero(img)
    mean_row = np.mean(points_non_zero[0])

    # Find the upper an lower limits of the rows that are not to be changed
    lower_threshold = mean_row - img.shape[0] * area_to_maintain / 100
    upper_threshold = mean_row + img.shape[0] * area_to_maintain / 100

    # Clear the rows that are not considered part of the bottom slot edges rows
    img[0:int(lower_threshold), :] = 0
    img[int(upper_threshold):img.shape[0]-1, :] = 0

    return img

#--------------FUNCTIONS_END--------------



#----------------PROGRAM-----------------

# Read original image
file_path = 'images\imgs_eng_qual_br\90k lado R.bmp'
img = cv.imread(file_path, cv.IMREAD_GRAYSCALE)


########------------------CIRCULARITY------------------########
# Crop ROI's
img_ROI = crop(img, ROI_CIRCULARITY)

img_ROI = apply_closing(apply_opening(img_ROI, MORPH_KERNEL_CIRC),
↪   MORPH_KERNEL_CIRC)

# Extract edges
cannied = cv.Canny(img_ROI, 117, 205)

# Obtain cannied image with only horizontal edges (bottom slot edge)
horizontal_edges = horizontal_edges_extraction(img_ROI, cannied)
```

```
## Desnoise bottom slot edges image
# Perform opening operation to remove small noise
opened = apply_opening(horizontal_edges, DENOISE_KERNEL)

# Denoise image, turning black all point far from bottom slot edges
opened = delete_non_bottom_slot(opened, BOTTOM_SLOT_IMG_PERCENTAGE)

# Trim edges ends to remove curvature from edges intersection
trimmed = trim_edges(opened.copy(), 10, 10)


## Find circle function that describes the bottom slot radius and position
# Obtain bottom slot edge points coordinates
coords = np.nonzero(trimmed)
x = np.array(coords[1])
y = np.array(coords[0])

# Fit a circle to the data
xc, yc, r = fit_circle(x, y) # "r" is the measure in pixels of the lamination
↪  bottom slot radius

# Transfer circle center coordinates in ROI to coordinates on original image
xc += ROI_CIRCULARITY[0]
yc += ROI_CIRCULARITY[2]


########------------------CIRCULARITY_END------------------########



########------------------LINEAR_PROFILE------------------########

n = len(ROI_TOOTH) # Number of teeth

# initiate images array variables
tooth = []
tooth_cannied = []
for i in range(n):

    # Crop teeth ROI's
    tooth.append(crop(img, ROI_TOOTH[i]))

    #Detect edges
    tooth_cannied.append(apply_closing(apply_opening(cv.threshold(tooth[i], 98,
    ↪  255, cv.THRESH_BINARY)[1], MORPH_KERNEL), MORPH_KERNEL))
```

```
linearEdge_limits_identification_parameters = [[20, 2, 1], # step, threshold,
↪  steps_back
                                               [20, 3, 1],
                                               [20, 3, 1]]

tooth_edges_limits = [] # left_bottom_row, left_upper_row, right_bottom_row,
↪  right_upper_row
tooth_edge_profile_parameters = [] # l_m, l_x0, l_y0, r_m, r_x0, r_y0

for i in range(n):
    # Find limits of linear edge profile and the linear function that describe the
    ↪  edges
    tooth_edges_limits.append(edge_limits(tooth_cannied[i],
    ↪  linearEdge_limits_identification_parameters[i]))
    tooth_edge_profile_parameters.append(list(tooth_edges(tooth_cannied[i],
    ↪  tooth_edges_limits[i])))

    # Convert the tuple to a list before modifying
    tooth_edge_profile_parameters[i] = list(tooth_edge_profile_parameters[i])

    # Transfer line reference from tooth ROI to original image
    x_offset = ROI_TOOTH[i][0]
    y_offset = ROI_TOOTH[i][2]
    tooth_edge_profile_parameters[i][1] += x_offset
    tooth_edge_profile_parameters[i][4] += x_offset
    tooth_edge_profile_parameters[i][2] += y_offset
    tooth_edge_profile_parameters[i][5] += y_offset

########----------------LINEAR_PROFILE_END----------------########




########----------TOOTH_WIDTH_POINTS_&_MEASUREMENT----------########

# Begin variable of tooth intersection points coordinates, and measurement values
tooth_intersection_points = []
tooth_width = []

for i in range(n):
    # Get linear edge profile parameters
    left_m, left_x0, left_y0 = tooth_edge_profile_parameters[i][0],
    ↪  tooth_edge_profile_parameters[i][1], tooth_edge_profile_parameters[i][2]
    right_m, right_x0, right_y0 = tooth_edge_profile_parameters[i][3],
    ↪  tooth_edge_profile_parameters[i][4], tooth_edge_profile_parameters[i][5]

    # Find intersection points between linear profile and bottom slot diameter
    left_point = find_intersection(xc, yc, r, left_m, left_x0, left_y0)
    right_point = find_intersection(xc, yc, r, right_m, right_x0, right_y0)
```

```
    # Save intersection points for image features
    tooth_intersection_points.append([left_point, right_point])

    # Measure tooth width and save it
    tooth_width.append(measure(left_point, right_point, MM_PX))

# "Measure" bottom slot diameter
diameter = round(2*r * MM_PX, DECIMAL_PLACES)

########---------TOOTH_WIDTH_POINTS_&_MEASUREMENT_END---------########




########---------------DISPLAY_IMAGE_FEATURES---------------########

#####---------TOOTH_WIDTH---------#####
tooth_width_img = img.copy()
tooth_width_img = cv.cvtColor(img, cv.COLOR_GRAY2BGR)
tooth_width_img = resize(tooth_width_img, RESIZE_SCALE)


for i in range(n):

    # Get measuring points
    left_point = list(tooth_intersection_points[i][0])
    right_point = list(tooth_intersection_points[i][1])

    # Get measuring points of resized image
    left_point[0] = left_point[0] * RESIZE_SCALE / 100
    left_point[1] = left_point[1] * RESIZE_SCALE / 100
    right_point[0] = right_point[0] * RESIZE_SCALE / 100
    right_point[1] = right_point[1] * RESIZE_SCALE / 100

    # Draw doublearrow
    draw_doublearrow(tooth_width_img, (int(left_point[0]), int(left_point[1])),
     ↪  (int(right_point[0]), int(right_point[1])), COLOR, 3)

    # Find double arrow midpoint
    mid_x, mid_y = midpoint((int(left_point[0]), int(left_point[1])),
     ↪  (int(right_point[0]), int(right_point[1])))

    # Write tooth ID above double arrow
    label = 'T{}'.format(i+1)
    label_size, _ = cv.getTextSize(str(label), FONT, 1, 2)
    x_offset = label_size[0] / 2
    y_offset = label_size[0] / 2
```

```
    cv.putText(tooth_width_img, label, (int(mid_x - x_offset), int(mid_y -
    ↪  y_offset)), FONT, 1, (0, 0, 255), 2, cv.LINE_AA)

    # Get the size of dimensions text
    text_size_label, _ = cv.getTextSize('{}: '.format(label), FONT, 1, 2)
    text_size_dimension, _ = cv.getTextSize(str(round(tooth_width[i], 3)), FONT, 1,
    ↪  2)
    text_size_scale, _ = cv.getTextSize(SCALE, FONT, 1, 2)

    # Define the dimensions text position
    text_x = 10
    text_y = tooth_width_img.shape[0] - 10 - (2 * text_size_dimension[1] * (1 + n -
    ↪  i))

    # Write dimensions text
    cv.putText(tooth_width_img, '{}: '.format(label), (text_x, text_y), FONT, 1,
    ↪  (0, 0, 255), 2, cv.LINE_AA)
    cv.putText(tooth_width_img, str(round(tooth_width[i], 3)), (text_x +
    ↪  text_size_label[0], text_y), FONT, 1, (0, 0, 255), 2, cv.LINE_AA)
    cv.putText(tooth_width_img, SCALE, (text_x + text_size_label[0] +
    ↪  text_size_dimension[0] , text_y), FONT, 1, (0, 0, 255), 2, cv.LINE_AA)

#####-------TOOTH_WIDTH_END-------#####


#####-----BOTTOM_SLOT_DIAMETER-----#####
diam_measure_img = img.copy()
diam_measure_img = cv.cvtColor(diam_measure_img, cv.COLOR_GRAY2BGR)
diam_measure_img = resize(diam_measure_img, RESIZE_SCALE)

# Get the size of the text
text_size_diam_label, _ = cv.getTextSize('Diameter:', FONT, 1, 2)
text_size_number, _ = cv.getTextSize(str(diameter), FONT, 1, 2)
text_size_scale, _ = cv.getTextSize(SCALE, FONT, 1, 2)

# Define the position of the text
text_x = 10
text_y = diam_measure_img.shape[0] - text_size_number[1] - 76

# Write text
cv.putText(diam_measure_img, 'Diameter:', (text_x, text_y), FONT, 1, (0, 0, 255),
↪  2, cv.LINE_AA)
cv.putText(diam_measure_img, str(diameter), (text_x + text_size_diam_label[0] + 10,
↪  text_y), FONT, 1, (0, 0, 255), 2, cv.LINE_AA)
cv.putText(diam_measure_img, str(SCALE), (text_x + text_size_diam_label[0] +
↪  text_size_number[0] + 10, text_y), FONT, 1, (0, 0, 255), 2, cv.LINE_AA)


# Draw arrow from center of image to a circle point
```

```
center = (int(xc * RESIZE_SCALE / 100), int(yc * RESIZE_SCALE / 100))

# Define a point in the bottom slot perimeter to point the radius arrow
x_circle = (img.shape[1] / 3)
y_circle = math.sqrt(r**2 - (x_circle - xc)**2) + yc

# Find where the arrow to be draw begins (if it begins at the circle center, the
↪  arrow is displayed too big)
m = (y_circle - yc)/(x_circle - xc)
# y_arrow_begin = diam_measure_img.shape[0]/2
y_arrow_begin = 0
x_arrow_begin = -(y_circle - y_arrow_begin) / m + x_circle

# Change points coordinates to resized image
x_circle = x_circle * RESIZE_SCALE / 100
y_circle = y_circle * RESIZE_SCALE / 100
y_arrow_begin = y_arrow_begin * RESIZE_SCALE / 100
x_arrow_begin = x_arrow_begin * RESIZE_SCALE / 100
r = r * RESIZE_SCALE / 100

# Define radius arrow points
pt1 = (int(x_arrow_begin), int(y_arrow_begin))
pt2 = (int(x_circle), int(y_circle))

# Draw arrow and circle features
cv.arrowedLine(diam_measure_img, pt1, pt2, COLOR, 3)
cv.circle(diam_measure_img, center, int(r), COLOR, 1)

#####---BOTTOM_SLOT_DIAMETER_END---#####

############################ UNCOMMENT TO SEE RESULT IMAGES
# cv.imshow('tooth_width_img', tooth_width_img)
# cv.imshow('diam_measure_img', diam_measure_img)
# cv.waitKey(0)
# cv.destroyAllWindows()

########--------------DISPLAY_IMAGE_FEATURES_END--------------########

#--------------PROGRAM_END--------------
```

Appendix B

Rollover Measure

The following code was developed using Python 3.10.7, with the libraries OpenCV 4.6.0, and NumPy 1.23.3, and is executed as follows.

```
import cv2 as cv
import numpy as np
from utils.tools import crop, measure, draw_doublearrow

#----------------CONSTANTS-----------------

ROI_ROLLOVER = [2300, 3100, 4700, 5500] #alB_dome
FONT = cv.FONT_HERSHEY_SIMPLEX

KERNEL = np.ones((5,5), np.uint8)

ARROW_COLOR = (40, 40, 200)
ARROW_THICKNESS = 5

MM_PX = 21 / 1875
SCALE = "mm"

#--------------CONSTANTS_END--------------


#----------------FUNCTIONS-----------------

def horizontal_denoising(img, threshold_percent, range_of_analysis):
    '''
    Horizontally classify if the amount of white pixel are above or below a
↪   threshold.

    Args:
    img(mat): Image to perform the classification.
    threshold_percent(int): Percentage of white pixels threshold for
↪   classification.
```

```
    range_of_analysis(int): Number of columns to be considered on the
↪   classification.

    Returns:
    img(mat): Image with rows classified.
    '''
    first_column = int(img.shape[1] / 2 - range_of_analysis / 2)
    last_column = int(img.shape[1] / 2 + range_of_analysis / 2)

    # Compute percentage of white pixels in each row
    row_sums = np.sum(img[:, first_column:last_column] == 255, axis=1)
    percent_white_pixels = 100 * row_sums / range_of_analysis

    # Select rows that need to be denoised
    denoise_rows = percent_white_pixels >= threshold_percent

    # In-place modification of denoised array
    denoised = np.zeros_like(img)
    denoised[denoise_rows, first_column:last_column] = 255

    return denoised


def upper_and_lower_limits(img):
    '''
    Find the upper and lower row on the centre of the image that contains a white
↪   pixel.

    Args:
    img(mat): Image to find the limits.

    Returns:
    first_all_white_row_index(int): Index of the first row on the column at the
↪   centre of the image that contains a white pixel.
    last_all_white_row_index(int): Index of the last row on the column at the
↪   centre of the image that contains a white pixel.
    '''
    # Get index of all white pixel on the middle of the image
    nonzero_rows = np.nonzero(img[: , int(img.shape[1]/2)])

    # Get index of first white pixel row
    first_all_white_row_index = np.min(nonzero_rows)

    # Get index of last white pixel row
    last_all_white_row_index = np.max(nonzero_rows)

    return first_all_white_row_index, last_all_white_row_index

#--------------FUNCTIONS_END--------------
```

```
#----------------PROGRAM-----------------

# Read original image
file_path = './images/pictures_al_B_20-03/alB_dome.jpg'
img = cv.imread(file_path)

# Crop ROI's
img = crop(img, ROI_ROLLOVER)

# Convert to grayscale
gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)

# Denoise using bilateral filter
bilateral = cv.bilateralFilter(gray,9,75,75)

# Binarize using Otsu's method
_, bilateral_bin = cv.threshold(bilateral, 0, 255, cv.THRESH_BINARY +
↪  cv.THRESH_OTSU)

# Perform opening operation
opening = cv.morphologyEx(bilateral_bin, cv.MORPH_OPEN, KERNEL, iterations=1)

# Classify rows as part of the rollover
horizontal_filtered = horizontal_denoising(opening, 50, 10)

# Find limits of the rollover
upper_limit, lower_limit = upper_and_lower_limits(horizontal_filtered)
pt1 = (int(img.shape[1]/2), upper_limit)
pt2 = (int(img.shape[1]/2), lower_limit)

# Draw doublearrow on output image
draw_doublearrow(img, pt1, pt2, ARROW_COLOR, ARROW_THICKNESS)

# Measure rollover size
size = measure(pt1, pt2, MM_PX)

# Get the size of the text
text_size_number, _ = cv.getTextSize(str(size), FONT, 1, 2)
text_size_scale, _ = cv.getTextSize(SCALE, FONT, 1, 2)

# Define the position of the text
text_x = 10
text_y = img.shape[0] - text_size_number[1] - 10

# Write texts on output image
cv.putText(img, str(size), (text_x, text_y), FONT, 1, (0, 0, 255), 2, cv.LINE_AA)
```

```
cv.putText(img, SCALE, (text_x + text_size_number[0] + 10, text_y), FONT, 1, (0, 0,
↪  255), 2, cv.LINE_AA)

#--------------PROGRAM_END--------------
```

Appendix C

Auxiliary Tools

The following code was developed using Python 3.10.7, with the libraries OpenCV 4.6.0, and NumPy 1.23.3, and is executed as follows.

```
import math
import cv2 as cv
import numpy as np

def angular_cropping(img, coordinates):
    '''
    Crop a rectangle of any orientation and rotate it to make sides parallel to the
↪   vertical and horizontal axis.

    Args:
        img(mat): Image from where a part will be cropped.
        coordinates(array): Vector of the 2D points of the corners of the cropping.

    Returns:
        img(mat): Cropped and reoriented region.
    '''

    rect = cv.minAreaRect(coordinates)
    box = cv.boxPoints(rect)
    box = np.intp(box)

    if rect[2] > 45:
        width = int(rect[1][1])
        height = int(rect[1][0])
        src_pts = box.astype("float32")
        dst_pts = np.array([
            [0, 0],
            [width-1, 0], #colocar quarto ponto
            [width-1, height-1],
            [0, height-1]
        ], dtype="float32")
```

```
    else:
        width = int(rect[1][0])
        height = int(rect[1][1])
        src_pts = box.astype("float32")
        dst_pts = np.array([
            [0, height-1],
            [0, 0],
            [width-1, 0],
            [width-1, height-1]
        ], dtype="float32")

    M = cv.getAffineTransform(src_pts[:3], dst_pts[:3])

    warped = cv.warpAffine(img, M, (width, height), flags= cv.INTER_LANCZOS4)

    return warped


def resize(img, scale):
    '''
    Resize image.

    Args:
        img(mat): Image to resize.
        scale(float): Scale to resize the original image, in percentage.

    Returns:
        img(mat): Resized image.
    '''

    width = int(img.shape[1] * scale / 100)
    height = int(img.shape[0] * scale / 100)
    dim = (width, height)

    resized = cv.resize(img, dim, interpolation = cv.INTER_AREA)

    return resized


def coordinates_after_rotation(row, col, rot_mat):
    '''
    Get the destination coordinate of a rotated pixel by means of a rotation matrix

    Args:
        row(int): Original row index of the pixel to be followed
        col(int): Original column index of the pixel to be followed
        rot_mat(mat): Rotation matrix in format proceeding from
↪   cv.getRotationMatrix2D
```

```
    Returns:
        dst_row(int): Destination row index of the followed pixel.
        dst_col(int): Destination column index of the followed pixel.
    '''

    dst_row = int(rot_mat[1,0] * col + rot_mat[1,1] * row + rot_mat[1,2])
    dst_col = int(rot_mat[0,0] * col + rot_mat[0,1] * row + rot_mat[0,2])

    return dst_row, dst_col


def rotate(img, angle):

    '''
    Counter colckwise rotation around center of the image.

    Args:
    img(mat): Image to rotate.
    angle(float): Angle to rotate in degrees.

    Returns:
    img(numpy.ndarray): The rotated image.
    M(numpy.ndarray): The rotation matrix.
    '''

    rows,cols = img.shape
    M = cv.getRotationMatrix2D(((cols-1)/2.0,(rows-1)/2.0), angle, 1)
    rotated = cv.warpAffine(img, M, (cols,rows), flags= cv.INTER_LANCZOS4)

    return rotated, M


def draw_doublearrow(img, point_1, point_2, color, thickness):
    '''
    Draw a doubled arrow between two points.

    Args:
    img(mat): Image where to draw the arrow.
    point_1(int): First reference for doubled arrow limits .
    point_2(int): Second reference for doubled arrow limits.
    color(array): Color of the doubled arrow in BGR.
    Thickness(int): Thickness in pixels of the arrow to be draw.
    '''

    cv.arrowedLine(img, point_1, point_2, color, thickness)
    cv.arrowedLine(img, point_2, point_1, color, thickness)


def crop(img, coordinates):
```

```
    '''
    Crop a rectangle of an image.

    Args:
    img(mat): Image from where a part will be cropped.
    cooridnates(array): Array with upper left and lower right points coordinates of
↪   crop rectangle, in order X1, X2, Y1, Y2

    Returns:
    img(mat): Cropped image.
    '''

    cropped = img[coordinates[2]:coordinates[3], coordinates[0]:coordinates[1]]

    return cropped


def measure(pt1, pt2, mm_px = 1):
    '''
    Measure the distance between two points.

    Args:
    pt1(array): Coordinates of the first point.
    pt2(array): Coordinates of the second point.
    mm_px(float): Millimeter per pixel relation. If not provided, is equal to 1,
↪   and the distance is measured in pixels.

    Returns:
    distance(float): The distance between two points regarding the provided pixel
↪   millimeter relation.
    '''
    d_px = math.sqrt((pt1[0]-pt2[0])**2 + (pt1[1]-pt2[1])**2)

    d_mm = mm_px * d_px

    return d_mm


def midpoint(pt1, pt2):
    '''
    Find the midpoint between two points.

    Args:
    pt1(array): Coordinates of the first point.
    pt2(array): Coordinates of the second point.

    Returns:
    mid_x(float): X coordinate of the midpoint.
    mid_y(float): Y coordinate of the midpoint.
```

```
    '''
    mid_y = (pt1[1] + pt2[1]) / 2
    mid_x = (pt1[0] + pt2[0]) / 2

    return abs(mid_x), abs(mid_y)


def display_image(window_name, img):
    '''
    Display an image until a key is pressed.

    Args:
    window_name(string): Name of the window where the image will be displayed.
    img(mat): Image to be displayed.
    '''
    cv.imshow(window_name, img)
    cv.waitKey(0)
    cv.destroyAllWindows()
```

References

[1] Weg | homepage. https://www.weg.net/institutional/PT/pt/, accessed at 15/06/2023.

[2] European Comission. Electric motors and variable speed drives. https://commission.europa.eu/energy-climate-change-environment/standards-tools-and-labels/products-labelling-rules-and-requirements/energy-label-and-ecodesign/energy-efficient-products/electric-motors-and-variable-speed-drives_en, accessed at 09/01/2023.

[3] A Schoppa, J Schneider, and C.-D Wuppermann. Influence of the manufacturing process on the magnetic properties of non-oriented electrical steels. *Journal of Magnetism and Magnetic Materials*, 215-216:74–78, 2000.

[4] Osamu Nakazaki, Yuichiro Kai, Takashi Todaka, and Masato Enokizono. Iron loss properties of a practical rotating machine stator core at each manufacturing stage. *International Journal of Applied Electromagnetics and Mechanics*, 33(1):79–86, 2010. Publisher: IOS Press.

[5] P Beckley. *Electrical Steel for rotating machines*. The Institution of Engineering and Technology, 2002.

[6] H. Hamzehbahmani, P. Anderson, J. Hall, and D. Fox. Eddy current loss estimation of edge burr-affected magnetic laminations based on equivalent electrical network - part i: Fundamental concepts and FEM modeling. *IEEE Transactions on Power Delivery*, 29(2):642–650, 2014.

[7] Wen Jun Deng, Zi Chun Xie, Ping Lin, and Tong Kui Xu. Study on burr formation at the top edge in rectangular groove cutting. *Advances in Materials Science and Engineering*, 2012:956208, 2012-02-19. Publisher: Hindawi Publishing Corporation.

[8] MVISIA - sistemas de visão industrial e leitores de códigos. https://mvisia.com.br/, accessed at 09/06/2023.

[9] O. Cavusoglu and H. Gürün. The relationship of burr height and blanking force with clearance in the blanking process of AA5754 aluminium alloy. *Transactions of Famena*, 41(1):55–62, 2017.

[10] Improving blank edge conditions. https://www.thefabricator.com/stampingjournal/article/stamping/improving-blank-edge-conditions, accessed at 09/01/2023.

[11] EN 60740-1. Laminations for transformers and inductors - part 1: Mechanical and electrical characteristics, 2005.

[12] ZEISS o-INSPECT: CMM multissensor. https://www.zeiss.pt/metrologia/produtos/sistemas/optica/o-inspect.html, accessed at 17/05/2023.

[13] Z. Fang, K. Nagato, T. Shimura, M. Murakami, and M. Nakao. Micro-level clearance punching on NGO electrical steel. *Journal of Manufacturing Science and Engineering, Transactions of the ASME*, 142(8), 2020.

[14] Kunihiro Senda, Masayoshi Ishida, Youichi Nakasu, and Masaaki Yagi. Influence of shearing process on domain structure and magnetic properties of non-oriented electrical steel. *Journal of Magnetism and Magnetic Materials*, 304(2):e513–e515, 2006. Proceedings of the 17th International Symposium on Soft Magnetic Materials.

[15] K. Lenik. Tool wear and part quality in blanking of electrical steel. *Soviet Forging and Sheet Metal Stamping Technology (English Translation of Kuznechno-Shtampovochnoe Proizvodstvo)*, (3):22–25, 1988. Cited By :3.

[16] Ş. Bayraktar and Y. Turgut. Effects of different cutting methods for electrical steel sheets on performance of induction motors. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 232(7):1287–1294, 2018.

[17] P. Handgruber, A. Stermecki, O. Biro, and G. Ofnery. Evaluation of interlaminar eddy currents in induction machines. In *IECON Proceedings (Industrial Electronics Conference)*, pages 2792–2797, 2013.

[18] S. Mehta, A. Patel, and J. Mehta. Ccd or cmos image sensor for photography. In *2015 International Conference on Communication and Signal Processing, ICCSP 2015*, pages 291–294, 2015. Cited By :28.

[19] Introduction to image sensors | LUCID vision labs. https://thinklucid.com/tech-briefs/understanding-digital-image-sensors/, accessed at 10/03/2023.

[20] Z. Lihua, L. Jijun, L. Lin, D. Yungang, and J. Yongjun. The key technology and research progress of cmos image sensor. In *Proceedings of SPIE - The International Society for Optical Engineering*, volume 7157, 2009.

[21] Bayer filter, 2023-02-19. Page Version ID: 1140206983.

[22] Converging lens. https://www.sciencelearn.org.nz/images/50-converging-lens, accessed at 13/03/2023.

[23] Claudio Castanhola. Distância focal - sem sofrimento para entender, 2017-07-20. https://www.fotografia-dg.com/o-que-e-distancia-focal/, accessed at 13/03/2023.

[24] The advantages of telecentricity | edmund optics. https://www.edmundoptics.com/knowledge-center/application-notes/imaging/advantages-of-telecentricity/, accessed at 16/03/2023.

[25] T.A. Clarke and J.G. Fryer. The development of camera calibration methods and models. *Photogrammetric Record*, 16(91):51–66, 1998.

[26] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

[27] OpenCV: Camera calibration. https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html, accessed at 05/06/2023.

[28] Estimate geometric parameters of a single camera - MATLAB. https://www.mathworks.com/help/vision/ref/cameracalibrator-app.html, accessed at 05/02/2023.

[29] Guo Shuxia, Zhang Jiancheng, Jiang Xiaofeng, Peng Yin, and Wang Lei. Mini milling cutter measurement based on machine vision. *Procedia Engineering*, 15:1807–1811, 2011-01-01.

[30] H. Hao, L. Jin, L. Jianwei, X. Zhenzhong, and T. Zhengzong. Three-dimensional deformation measuring system based on vision method. In *ICMET 2010 - 2010 International Conference on Mechanical and Electrical Technology, Proceedings*, pages 748–753, 2010.

[31] J. Molleda, R. Usamentiaga, D.F. García, F.G. Bulnes, and L. Ema. Shape measurement of steel strips using a laser-based three-dimensional reconstruction technique. *IEEE Transactions on Industry Applications*, 47(4):1536–1544, 2011.

[32] S. Okada, M. Imade, H. Miyauchi, T. Miyoshi, T. Sumimoto, and H. Yamamoto. Non-contact 3-d shape inspection based on optical ring imaging system. In *Proceedings of SPIE - The International Society for Optical Engineering*, volume 2909, pages 58–65, 1997.

[33] R. Wang, A.C. Law, D. Garcia, S. Yang, and Z. Kong. Development of structured light 3d-scanner with high spatial resolution and its applications for additive manufacturing quality assurance. *International Journal of Advanced Manufacturing Technology*, 117(3):845–862, 2021.

[34] A.A. Gorbachev, V.V. Korotaev, and D.V. Apehtin. The system of blade's shape measuring. In *Proceedings of SPIE - The International Society for Optical Engineering*, volume 9446, 2015.

[35] Muhammad Sajjad, Fath U Min Ullah, Mohib Ullah, Georgia Christodoulou, Faouzi Alaya Cheikh, Mohammad Hijji, Khan Muhammad, and Joel J.P.C. Rodrigues. A comprehensive survey on deep facial expression recognition: challenges, applications, and future guidelines. *Alexandria Engineering Journal*, 68:817 – 840, 2023. Type: Review.

[36] S. Kumar, S. C. Sharma, and R. Kumar. Wireless sensor network based real-time pedestrian detection and classification for intelligent transportation system. *International Journal of Mathematical, Engineering and Management Sciences*, 8(2):194–212, 2023.

[37] Marian Marcel Abagiu, Dorian Cojocaru, Florin Manta, and Alexandru Mariniuc. Detecting machining defects inside engine piston chamber with computer vision and machine learning. *Sensors*, 23(2), 2023. Type: Article.

[38] A. Distante and C. Distante. *Handbook of image processing and computer vision: Volume 1: From energy to image*, pages 1–491. Handbook of Image Processing and Computer Vision: Volume 1: From Energy to Image. 2020. Cited By :3.

[39] A. Distante and C. Distante. *Handbook of image processing and computer vision: Volume 2: From image to pattern*, pages 1–431. Handbook of Image Processing and Computer Vision: Volume 2: From Image to Pattern. 2020. Cited By :4.

[40] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979-01. Conference Name: IEEE Transactions on Systems, Man, and Cybernetics.

[41] Pei Yang, Wei Song, Xiaobing Zhao, Rui Zheng, and Letu Qingge. An improved otsu threshold segmentation algorithm. *International Journal of Computational Science and Engineering*, 2020-05-04. Publisher: Inderscience Publishers (IEL).

[42] S.D. Yanowitz and A.M. Bruckstein. A new method for image segmentation. *Computer Vision, Graphics, & Image Processing*, 46(1):82–95, 1989.

[43] M. Ghaziasgar, A. Bagula, and C. Thron. Computer vision algorithms for image segmentation, motion detection, and classification. *Studies in Computational Intelligence*, 782:119–138, 2020.

[44] OpenCV: Image thresholding. https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html, accessed at 28/02/2023.

[45] T. Kanungo, D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, and A.Y. Wu. An efficient k-means clustering algorithm: analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):881–892, 2002.

[46] C.-J. Hung, Y.-R. Tsao, C.-L. Lin, and C.-Y. Liu. Real-time detection and classification of porous bone structures using image segmentation and opening operation techniques. In *Proceedings of the 3rd IEEE Eurasia Conference on IOT, Communication and Engineering 2021, ECICE 2021*, pages 328–332, 2021.

[47] Aristidis Likas, Nikos Vlassis, and Jakob J. Verbeek. The global k-means clustering algorithm. *Pattern Recognition*, 36(2):451–461, 2003. Biometrics.

[48] *k*-means clustering. https://en.wikipedia.org/w/index.php?title=K-means_clustering&oldid=1138258715, accessed at 08/02/2023.

[49] Morphological operations - MATLAB & simulink. https://www.mathworks.com/help/images/morphological-filtering.html?s_tid=CRUX_lftnav, accessed at 23/02/2023.

[50] R. M. Haralick, S. R. Sternberg, and X. Zhuang. Image analysis using mathematical morphology. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(4):532–550, 1987. Cited By :2014.

[51] H. Minkowski. Volumen und oberfläche. *Mathematische Annalen*, 57(4):447–495, 1903. Cited By :422.

[52] H. Hadwiger. *Vorlesungen Über Inhalt, Oberfläche und Isoperimetrie*. Springer, 1957.

[53] X. Yu, Z. Wang, Y. Wang, and C. Zhang. Edge detection of agricultural products based on morphologically improved canny algorithm. *Mathematical Problems in Engineering*, 2021, 2021.

[54] C. Jasiński and Ł. Morawiński. Vision assisted clearance evaluation in cutting process. In *METAL 2018 - 27th International Conference on Metallurgy and Materials, Conference Proceedings*, pages 390–395, 2018.

[55] L. Xie, X. Li, and X. Wei. An efficient algorithm for detecting flatness of small metal parts based on stereo matching of multi-similarity fusion. *Computer-Aided Design and Applications*, 18(3):571–583, 2021.

[56] S. Yunhong, Y. Shilei, Z. Xiaojing, and Y. Jinhua. Edge detection algorithm of mri medical image based on artificial neural network. In *Procedia Computer Science*, volume 208, pages 136–144, 2022.

[57] M.A. Rahman, M.F.I. Amin, and M. Hamada. Edge detection technique by histogram processing with canny edge detector. In *Proceedings of the 3rd IEEE International Conference on Knowledge Innovation and Invention 2020, ICKII 2020*, pages 128–131, 2020.

[58] Enzeng Dong, Yao Zhao, Xiao Yu, Junchao Zhu, and Chao Chen. An improved NMS-based adaptive edge detection method and its FPGA implementation. *Journal of Sensors*, 2016:e1470312, 2015-12-22. Publisher: Hindawi.

[59] M. F. V. Ruslau, R. A. Pratama, and E. Meirista. Edge detection of digital image with different edge types. *Journal of Physics: Conference Series*, 1569(4):042069, 2020-07. Publisher: IOP Publishing.

[60] Giuseppe Papari and Nicolai Petkov. Edge and line oriented contour detection: State of the art. *Image and Vision Computing*, 29(2):79–103, 2011.

[61] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986-11. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[62] P. Ekberg, B. Daemi, and L. Mattsson. 3d precision measurements of meter sized surfaces using low cost illumination and camera techniques. *Measurement Science and Technology*, 28(4), 2017.

[63] J. Mitchell, A. Spence, D. Capson, H. Chan, and M. Goldstein. Stereo vision based close-up dimensional inspection. In *Proceedings of SPIE - The International Society for Optical Engineering*, volume 5679, pages 166–177, 2005.

[64] J. W. Cui, J. B. Tan, Y. Zhou, and H. Zhang. Improvement of vision measurement accuracy using zernike moment based edge location error compensation model. *Journal of Physics: Conference Series*, 48(1):1353, 2007-07.

[65] Z. Qiu, F.Z. Fang, L. Ding, and Q. Zhao. Investigation of diamond cutting tool lapping system based on on-machine image measurement. *International Journal of Advanced Manufacturing Technology*, 56(1):79–86, 2011.

[66] Jiang Mai and Ma Ning. Sub-pixel edge detection method based on zernike moment. In *The 27th Chinese Control and Decision Conference (2015 CCDC)*, pages 3673–3676, 2015.

[67] Appendix vii - the circle polynomials of zernike (§ 9.2.1). In MAX BORN and EMIL WOLF, editors, *Principles of Optics (Sixth Edition)*, pages 767–772. Pergamon, sixth edition edition, 1980.

[68] S. O. Belkasim, M. Ahmadi, and M. Shridhar. Efficient algorithm for fast computation of zernike moments. *Journal of the Franklin Institute*, 333(4):577–581, 1996-07-01.

[69] Michael Reed Teague. Image analysis via the general theory of moments*. *JOSA*, 70(8):920–930, 1980-08-01. Publisher: Optica Publishing Group.

[70] Satish Kumar. Comparative results of zernike moments and pseudo-zernike moments. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 1254–1259, 2016.

[71] Kun Zhang, Haiqing Chen, Qingwen Liang, Chong Huang, and Jiakun Xu. Subpixel edge detection algorithm based on pseudo-zernike moments. *Proceedings of SPIE - The International Society for Optical Engineering*, 7658, 2010-05.

[72] Sistemas de visão industriais com inteligência artificial MVISIA. https://mvisia.com.br/produtos/sistemas-de-visao-industriais/, accessed at 17/05/2023.

[73] Basler AG. Basler ace - area scan cameras. https://www.baslerweb.com/en/products/cameras/area-scan-cameras/ace/, accessed at 13/06/2023.

[74] Fujinon CF25za-1s 23mp 16mm lens | machine vision direct. https://machinevisiondirect.com/products/fujinon-cf25za-1s, accessed at 10/06/2023.

[75] Camera lens distortion - image engineering. https://www.image-engineering.de/library/image-quality/factors/1062-distortion, accessed at 15/06/2023.

[76] DFL-12 dark field lights. https://smartvisionlights.com/products/dfl-12/, accessed at 11/06/2023.

[77] WLS27xw430dsq | WLS27 shatterproof LED light for harsh environments. https://www.bannerengineering.com/us/en/products/part.94276.html, accessed at 11/06/2023.

[78] LMBWLS27u | products for industrial automation. https://www.bannerengineering.com/us/en/products/part.94775.html, accessed at 11/06/2023.

[79] SIPI image database. https://sipi.usc.edu/database/, accessed at 09/04/2023.

[80] R. Keys. Cubic convolution interpolation for digital image processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 29(6):1153–1160, 1981.

[81] Anel rotativo iglidur® PRT-01, anel exterior dentado fabricado em alumínio, elementos deslizantes fabricados em iglidur® j. https://www.igus.pt/product/656?artNr=PRT-01-20-TO-HTD5M, accessed at 12/06/2023.

[82] 5 - 100x, ø 400 mm | HOC400 - digital profile projector by leader precision instrument co. ltd | DirectIndustry. https://www.directindustry.com/prod/leader-precision-instrument-co-ltd/product-99497-926963.html, accessed at 14/06/2023.

[83] Pés, rodas, fixações ao solo, suportes. https://www.fluidotronica.com/pt/produtos/minitec/sistemas-de-perfis/pes-rodas-fixacoes-ao-solo-suportes/, accessed at 14/06/2023.

[84] Large self-centering optics mount - optical mounts - catalog - opto-mechanical products - standa. https://www.standa.lt/products/catalog/optical_mounts?item=390, accessed at 14/06/2023.

[85] A. Mohammed, J. Kvam, I.F. Onstein, M. Bakken, and H. Schulerud. Automated 3d burr detection in cast manufacturing using sparse convolutional neural networks. *Journal of Intelligent Manufacturing*, 2022.