Master Thesis

Czech
Technical
University
in Prague

F2

Faculty of Mechanical Engineering
Department of Automotive,
Combustion Engine and Railway Engineering

Development of an inspection system for the fast detection of topographic defects on bipolar plates surfaces by means of laser grid projection

Sara Menetrey

Supervisor: Prof. Ing. DrSc. Jan Macek,
M.Eng. Rico Löser,
M.Sc. Leutrim Gjakova,
Dipl.-Ing. Alexander Pierer.
Field of study: Mechanical Engineering
Subfield: Automotive Engineering
August 2022

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Menetrey Sara**	Personal ID number:	**498260**
Faculty / Institute:	**Faculty of Mechanical Engineering**		
Department / Institute:	**Department of Automotive, Combustion Engine and Railway Engineering**		
Study program:	**Master of Automotive Engineering**		
Branch of study:	**Advanced Powertrains**		

II. Master’s thesis details

Master’s thesis title in English:

Development of an inspection system for the fast detection of topographic defects on bipolar plates surfaces by means of laser grid projection

Master’s thesis title in Czech:

Vývoj kontrolního systému pro rychlé nalezení tvarových odchylek na povrchu bipolárních desek pro palivové články pomocí laserem promítané mřížky

Guidelines:

The aim of this work is to develop a fast working optical inspection system for topographical defects on component surfaces. The principle of operation is based on the projection of a laser grid, the image of which is captured on the component surface by a camera. By evaluating local distortions of this image, fine topographic defects shall be detected. For this purpose a suitable experimental setup has to be developed and put into operation. For the development of the corresponding evaluation software, the software framework XEIDANA® can be used and, if necessary, extended by required additional functions within the scope of the work. The validation will be carried out by means of test series on a laboratory scale.

Focus of the work

- State of the art and scientific research (automatic quality monitoring, camera technology, crack detection, classification of surface defects, standards research)
- Requirement analysis under consideration of the boundary conditions in industrial production
- Development of the mechanical and electrical test setup (frame construction, brackets, electrical/control setup)
- Development of a software prototype for image acquisition and defect detection based on
- Evaluation and documentation of the results

Bibliography / sources:

Internal sources of Fraunhofer IWU, Chemnitz, Germany
Instructions for use of software XEIDANA®

Name and workplace of master’s thesis supervisor:

prof. Ing. Jan Macek, DrSc.

Name and workplace of second master’s thesis supervisor or consultant:

Diplôme d’Ingénieur / M.Eng. Rico Löser (Fraunhofer IWU, Chemnitz, Germany)
MSc. Leutrim Gjakova (Fraunhofer IWU, Chemnitz, Germany)
Dipl.-Ing. Alexander Pierer (Frauenhofer IWU, Chemnitz, Germany)Chalu, Renault

Date of master’s thesis assignment: **17.02.2022** Deadline for master's thesis submission: **12.08.2022**

Assignment valid until: ______________

prof. Ing. Jan Macek, DrSc. Supervisor’s signature	doc. Ing. Oldřich Vítek, Ph.D. Head of department’s signature	prof. Ing. Michael Valášek, DrSc. Dean’s signature

Acknowledgements

First of all, I would like to thank my thesis advisors from Fraunhofer IWU, Löser Rico, Leutrim Gjakova and Alexander Pierer for their patience, their availability and especially their judicious advice, which have contributed to my reflection. I would also like to thank my university mentor from CTU Jan Macek for his supervision throughout my research. I would like to express my gratitude to the department automation and monitoring at Fraunhofer IWU and its head of department Michael Hoffmann for having me included in the team. Finally I would like to thank all the teachers and professors from the Master of Automotive Engineering from CTU and ENSTA Bretagne, who provided me with the necessary tools for the success of my university studies.

Declaration

I hereby declare that I have submitted this thesis that I have prepared this submission independently and that I have indicated all the literature used.

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

Prague, 12. August 2022

Abstract

Fuel cells are a promising new alternative to internal combustion engines in the automotive industry. However, the production and examination costs are still too high, which makes this technology less attractive to manufacturers. Particularly with the bipolar plates, care must be taken to ensure that the quality is right. The goal of this work is to develop a new inexpensive and fast inspection system to detect topographic defects on bipolar plates surfaces by means of laser grid projection. This work analyzes the role of a bipolar plate within a fuel cell. It identifies the different production processes used in the industry today to point out the associated defects, reviews the published work of current surface inspection methods and compares their image acquisition and processing units. It derives a list of requirements for the inspection system and gives a proposition for a first prototype set-up for the system. Python was used for the control of the image acquisition and for the image processing. Several algorithms based on statistical properties, the Gray-Level Co-occurence Matrix and the Fast Fourier transform of the image have been tested with metallic discs and metallic bipolr plates. Finally, the system has been validated for the inspection of flat metallic components and needs to be improved for the inspection of metallic bipolar plates. Ideas on how to modify the set-up and the program are given in order to ameliorate the system.

Keywords: fuel cell, surface inspection, laser grid projection, image processing

Supervisor: Prof. Ing. DrSc. Jan Macek,
M.Eng. Rico Löser,
M.Sc. Leutrim Gjakova,
Dipl.-Ing. Alexander Pierer.
Praha, Technická 1902/4,
místnost: B1-720

Contents

Figures

[3] https://www.opto-e.com/products/pattern-PTGR050450-P(10.04.2022)

Tables

List of equations

List of symbols

k_1, k_2, k_3, p_1, p_2	Coefficients for camera calibration	
(H, W)	dimensions of an image	pixel^2
(h, w)	dimensions of one analyzed block	pixel^2
(t, sp)	line thickness and spacing of the grid pattern	μm
(t_p, sp_p)	line thickness and spacing of the projected grid	mm
(x, y)	location of one pixel	
$(x_{correct}, y_{correct})$	corrected pixel coordinates	
α	observation angle of the camera to the vertical axis	°
α_{BPP}	bipolar plate draft angle	°
Δu	height different of a point to its optimal position	mm
Δx	shift depending on Δu along the x-axis	mm
Δy	shift depending on Δu along the y-axis	mm
κ	threshold value	
ϕ	threshold function	
θ	diffraction angle	°
$C_{x,y,q}$	contrast of the pixel (x, y) of the q^{th} picture of a HDR image	
d	diameter of the projection area	mm

d'	diameter of the grid pattern	mm
DoF	depth of field	mm
$E_{x,y,q}$	well-exposedness of the pixel (x, y) of the q^{th} picture of a HDR image	
f, f'	focal lengths	mm
$F/\#$	aparture stop	
g	grayscale function of a picture	
g_{HDR}	grayscale function of a high dynamic range image	
h_{BPP}	bipolar plate channel depth	mm
k	given coefficient for defect inspection [opta]	0.015
$K_{(x,y)}$	Kernel set, set of pixel coordinates surrounding the pixel (x, y)	
k_{block}	block-size of the adaptive threshold neighborhood	pixel
k_{size}	kernel size defining $K_{(x,y)}$	
M	magnitude of the camera	
M_{pers}	perspective correction matrix	
Mtx	camera matrix for calibration	
n	number of lines contained in the grid pattern (along one direction)	
p	width of the observed area of one pixel	mm
p'	pixel width	μm
R	ratio between the length and the width of a picture	
R_{BPP}	bipolar plate outer radius	mm
r_{BPP}	bipolar plate inner radius	mm
s	distance between lens system and observation surface	mm
s'	distance between lens system and grid pattern	mm
s_{BPP}	bipolar plate rib width	mm
$S_{x,y,q}$	saturation of the pixel (x, y) of the q^{th} picture of a HDR image	
t_{BPP}	bipolar plate sheet thickness	µm
u	working distance	
w_{BPP}	bipolar plate channel width	mm

$W_{x,y,q}$	weight of the pixel (x, y) of the q^{th} picture of a HDR image	
x-axis	vertical axis of the camera's pictures	
y-axis	horizontal axis of the camera's pictures	
BFL	back focal length	mm

Acronyms

ANN Artificial Neural Network. 17

BFL Back Focal Length. 21, 22, 24

BPP Bipolar plate. vii–x, 4–9, 11, 12, 19–21, 25, 26, 31–33, 35, 38, 49, 62–65, 71, 74, 76

CAD Computer-Aided Design. 8, 64, 113

CCD Charge coupled devices. x, 13, 14

CMOS Complementary metal oxide semiconductor. x, 13, 14

CTU Czech Technical University. v

DoF Depth of Field. x, 28, 30, 31, 35, 85, 95, 145

ENSTA Bretagne Ecole Nationale Superieure de Techniques Avancees Bretagne. v

FFT Fast Fourier transform. vi, vii, ix–xi, 16, 41, 45–48, 50–55, 63, 65–69, 76, 85, 110, 112, 118, 146–149

GDL Gas Diffusion Layer. 4, 12, 20

GLCM Gray-Level Co-occurence Matrix. vi, vii, ix–xi, 40, 41, 44, 46–55, 63, 65–71, 73–76, 85, 98, 108, 110–112, 118, 145–149

HDR High Dynamic Range. vii–ix, 32, 37, 38, 43, 49, 62–64, 67, 118, 119

IWU Institut für Werkzeugmaschinen und Umformtechnik (Institute for Machine Tools and Forming Technology). v, 2, 73

KNN K-nearest neighbor. 17

LCD Liquid Crystal Display. 18

LED Light-Emitting Diode. 14, 18, 21, 85

MEA Membrane Electrode Assembly. 4, 6, 12, 20

PEM Polymer Electrolyte Membrane. 4

PEMFC Proton Exchange Membrane Fuel Cell. viii, 6, 20

PLA Polylactic Acid. 64, 65

PMD phase measuring deflectometry. 18

PMP phase measuring profilometry. 18

PVC Polyvinyl chloride. 64, 74

STD Standard deviation. 67

STL Standard Triangle Language. 64

SVM Support Vector Machine. 17

WDR Wide Dynamic Range. 32

Xeidana eXtensible Environment for Industrial Data ANAlysis. 73

Chapter 1

Introduction

Climate change and emerging regulations on carbon dioxide emissions for automobiles are driving the automotive industry to find alternatives for the internal combustion engine. The most popular technology is the hybrid electric vehicle. The passenger car sales in the European Union in 2021 were 59.6 % petrol or diesel, 9.1 % battery electric vehicles, 8.9 % plug-in hybrid electric vehicles and 19.6 % hybrid vehicles, the remaining 2.7 % being other technologies [ace]. The biggest disadvantages of these types of drive is that it eather still emits pollutant gaz (petrol, diesel and hybrid vehicles) or it has a low energy density (batteries). According to the Climate Change 2014: Mitigation of Climate Change report [Ede15], 14 % of the 2010 global greenhouse gas emissions are from the transportation sector[1]. This figure increased in the past ten years. A lot of countries have or are planning to introduce new policies to limit those emissions. For example, the European commission intends to reduce emissions from transportation by 90 % by 2050 with the European green deal [Sid20]. An alternative that seeks to avoid this disadvantages are fuel cells, electrochemical cells that convert energy released in chemical reactions between anode and cathode to electricity. There are several types of fuel cells: alkaline fuel cells, proton exchange fuel cells, phosphoric acid fuel cells, molten carbonate fuel cells, solid oxide fuel cells, direct methanol fuel cells ... One component they all have in common is the bipolar plate. Today it is still one of the cost-drivers and any defects on its surface can cause large energy efficiency losses of the fuel cell. Trustworthy and fast inspection systems in production lines are crucial to guarantee high quality and to make this technology competitive in the automotive market. The examination with the human eye, which is the current test method due to the complex manufacturing, is often not accurate enough, repetitive and

[1] The considered gases are carbon dioxide, methan, nitrous oxide and fluorinated gases.

time consuming. According to Pierer et al. [Pie20], human visual inspection has a 10 % to 20 % pseudo reject or slip rate. Sometimes, the defects are too small to be recognized by the human eye. Therefore, high precision cameras could solve this problem.

Fraunhofer Institute for Machine Tools and Forming Technology IWU is one of the leading institute in developing intelligent production processes and control systems. In this work, a new method for inline surface defect detection on bipolar plates will be implemented, tested and compared to existing systems. The goal is to get a more resource-efficient production in order to make the fuel cell technology more affordable and attractive to the producer and the consumer of the automotive industry.

To this purpose, the state of the art of bipolar plates and surface inspection systems will be introduced in Chapter 2. Then, the set-up and the software will be implemented in Chapter 3, and Chapter 4. After that and the system will be tested in Chapter 5. Finally, further adaptations of this inspection system will be discussed in the conclusion in Chapter 6.

Chapter 2

State of the art

2.1 Bipolar Plates

2.1.1 Fuel cells in general

The history of the fuel cell begins in the 19th century with the first concepts. In 1933, Thomas Francis Bacon, developed the first hydrogen fuel cell with practical use [AS09]. In the 21st century, many different manufacturers are working with it. While fuel cells have their place in stationary and mobile applications (mostly for research development or testing), no official standards for the design of it have been established yet. Shaigan et al. [SYG+21] published an article in 2021 that presents an example of what could be standardized testing of the different properties of a bipolar plate, but it does not give any numerical constraints to these properties.

A fuel cell is a electrochemical cell that produces electricity from chemical reactions. The most frequent type for mobile applications is the hydrogen proton exchange fuel cell which oxidizes hydrogen (2.1a) on the anode and reducing dioxygen (2.1b) on the cathode. That leads to the equation 2.1 of the total reaction:

$$H_2 \longrightarrow 2e^- + 2H^+ \tag{2.1a}$$

$$\frac{3}{2}O_2 + 6e^- + 6H^+ \longrightarrow 3H_2O \tag{2.1b}$$

$$H_2 + \frac{1}{2}O_2 \longrightarrow H_2O \tag{2.1c}$$

Its components are the following and are schematized in figure 2.1 [Pop14]:

- A Membrane Electrode Assembly (MEA); which is composed of:
    - a Polymer Electrolyte Membrane (PEM) to transfer the protons from the anode to the cathode, to separate spatially the anode and the cathode and to prevent hydrogen and oxygen diffusion through the membrane,
    - two catalyst layers to enable the reaction on both sides,
    - two Gas Diffusion Layer (GDL) to disperse the gas and prevent bubbles from blocking the electrons the access to the bipolar plates,
- Bipolar plate (BPP); graphite or metallic structure composed of two half plates with channels used to guide the fuel and to collect the current,
- Current collectors; to gather the electrons and bring them in the electric circuit that will charge the batteries or directly power the engine.

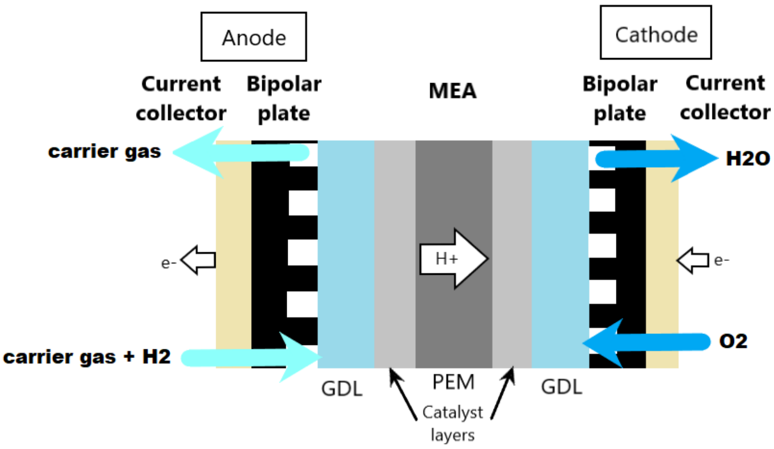


Figure 2.1: Schematic of a cell and the reaction happening inside.

The most important characteristics of a BPP that influences the performance of the fuel cell stack are [PWD20]:

- electrical conductivity
- thermal conductivity
- structural strength
- ensuring the reaction gases supply (hydrogen/anode and air/cathode)
- corrosion resistance (removal of water liquid and vapor)

Those characteristics are influenced by the design of the plate. Each material has its own electrical and thermal conductivity, corrosion resistance and structural strength. The cell's electrical conductivity depends on the contact surface with the membrane. A smaller surface leads to lower electrical conductivity. Thin channels lead to fewer reaction gas supply. Sharp edges lead to a lower structural strength. Due to the number of parameters that influence the functioning of the fuel cell, there is no standardized production and with this no standard inspection.

2.1.2 BPP design

BPPs consist of two half plates, one for the anode side the other for the cathode side. On each side is a flow field. If the cell is liquid cooled, there is a third flow field between the two half plates, as shown in figure 2.2. The design is characterized by the choice of material and the geometrical design.

Material choice. To match the characteristics listed in section 2.1.1, two basic materials are available on the market: graphite-based polymer compounds and metals.

Graphite has a very high corrosion resistance and electrical conductivity (10^5 S/cm, [YYZ$^+$17]) which give it a very high lifetime. It is also very expensive to manufacture. The bulk material is difficult to process and requires a minimum wall thickness to avoid that the fuel is leaking through it [PPW19]. The most common manufacturing process of graphite is milling, which is not suitable for high production rates and does not fit the growing demand.

The preferred metals are stainless steels because of their low cost and good formability. This allows to use other production processes such as rolling which have shorter production cycles and higher production rates and therefore require inline quality control. A presentation of the different processes is in section 2.1.3. The main disadvantage of metallic BPPs opposed to graphite BPPs is the lower corrosion resistance, except titanium alloys. However, this material is more expensive and has a lower formability. To limit the risk of corrosion, different coatings can be applied on the plate. Further elaborations on this topic can be found in [THM07]. Table 2.1 summarizes the performance relevant properties of the BPP materials.

Property	Graphite [azo]	Stainless steel [steb]	Titanium alloys [tita]
Electric conductivity	10^5 S/cm	10^4 S/cm	6,000 S/cm
Thermal conductivity	8.7 - 114 W/m.K	10 - 30 W/m.K	7.1 - 7.3 W/m.K
Tensile strength [1]	4.8 - 76 MPa	400 - 1,100 MPa	862 - 1,200 MPa
Corrosion resistance	Excellent resistance to fresh water	Good resistance to fresh water	Excellent resistance to fresh water
Cost of row material	359 - 6,175 US$/t [Hyk14]	3,200 - 4,450 US$/t[stea]	4,800 US$/t (in 2018) [titb]

Table 2.1: Comparison of relevant properties of graphite, stainless steel and titanium alloys.

Geometrical design. The dimensions and the flow field pattern of the BPP have a big impact on the characteristics listed in section 2.1.1. For instance, the thicker the plate, the higher the resistance and the structural strength; the more curves in the flow fields, the higher the pressure drop created in the cell and the better the water evacuation. Figure 2.2 shows the geometrical difference between a typical graphitic and a typical metallic BPP.

In the case of metallic BPPs, the cooling flow field results from the negative geometry of the anode and cathode plate [PWD20], which saves space in comparison to graphite BPPs where the structure must be large enough to contain the cooling circuit without jeopardizing the plates robustness. Nevertheless, this brings a restriction for the design of metallic BPPs: they must have the same pattern on each side (cathode and anode). In the case of hydrogen PEMFC, all fuels are in gaseous form. Their dynamic properties will be close, and the same BPP pattern will work for both.

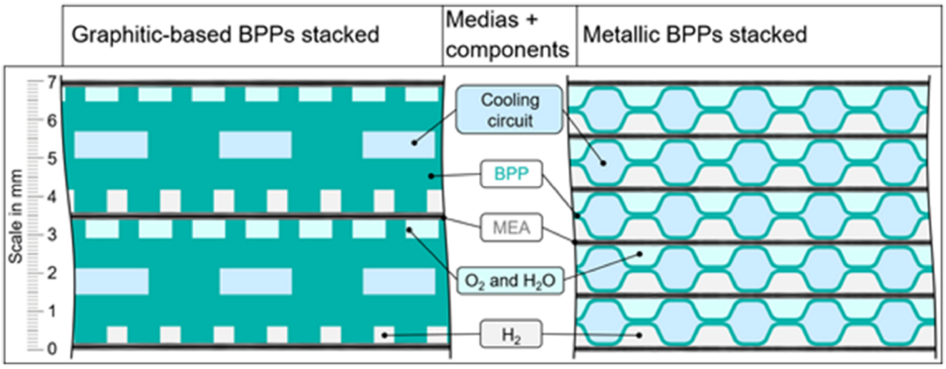


Figure 2.2: Schematic partial cross-sectional view of a graphitic and metallic PEMFC stack [PWD20].

To summarize, the metallic plates are thinner than the graphite plates, according to figure 2.2 even 2.5 times thinner. Neglecting the thickness of the other components (MEA, current collectors) for the same stack volume, a stack with metallic BPPs will contain 2.5 times more cells than a graphite stack. The more cells are stacked, the higher is the power of the stack. Assuming that one metallic cell produces approximately the same power as one graphite cell, the overall power density of a metallic stack is higher than the power density of a graphite stack. Consequently, metallic stacks have a higher power density and are characterized by a more compact design, which is especially interesting in the automotive sector because it is space-saving. For automotive application the goal is to have a very high power density, so the preferred material for this application is metal [PWD20]. This work

[0] Taking into account the products of Cognex, Omron, Keyence and Teledyne

[1] The tensile strength is the maximum stress a material can support during plastic deformation. It therefore influences the formability of the material. The higher the tensile strength, the better the formability.

will be focusing on metallic BPPs only. The graphite BPP are not further mentioned.

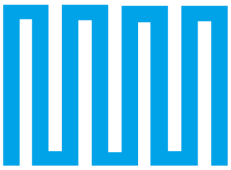
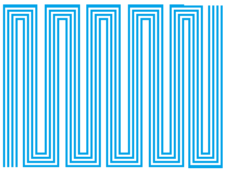
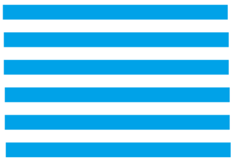
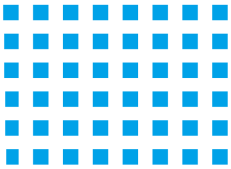
Single serpentine	4-Channel serpentine	Parallel	Pin-type
			
+ reduced risk of flooding[2]due to high pressure drop + good temperature distribution + suitable for small and large active areas − reactants depletion along the channel[3] − high flow resistivity	+ lower pressure drop than with single serpentine + good water removal + uniform gas distribution − lower reactants depletion compared to the single serpentine, but higher compared to the other design − high flow resistivity but lower than with the single serpentine	+ decreased hydraulic resistivity − bad water evacuation − risk of channel blockage through water drops − non-uniform distribution of reactant gases − non-uniform distribution of temperatures	+ lowest pressure drop + less water accumulation than in the parallel design − uneven flow distribution − appearance of stagnant areas − inhomogeneous current density distribution

Table 2.2: Typical flow field patterns in top view [SKPK20].

Different patterns exist for the flow field. Table 2.2 shows the four most typical patterns with their advantages and disadvantages. The more obstacles or bends in the pattern, the more pressure drop there is, the better the water removal and fuel distribution and the more energy is needed to get the fuel through. The fewer restrictions in the fuel path result in the opposite effect. In each case it can be seen that the geometry is mostly divided on two parallel planes less than 2 mm apart and that the plate alternates in mostly two directions repeatedly between these two levels. In addition, every flow field is characterized by the following parameters [ZWZ22]:

[2] flooding is the phenomenon where there is so much water in the circuit, that it clogs the pores of the GDL and restricts access to the catalyst layer. The methanol does not react. For more details on this topic read [IKL$^+$11].

[3] reactants depletion means that the reactants concentration in the flow is much lower at the end of the channel than at the beginning because most reactants already reacted when the flow reaches the end of the channel. For more information [SJDH11].

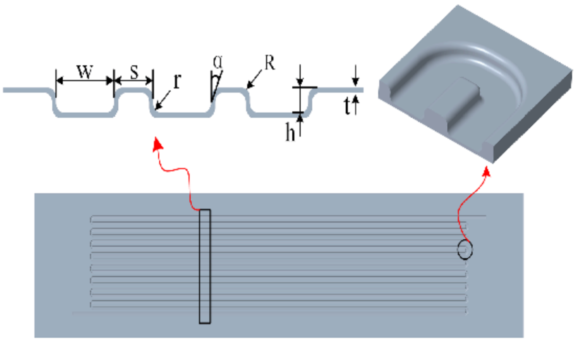


Figure 2.3: The geometric parameters of bipolar plate [ZWZ22].

- the channel width (w_{BPP}),
- the rib width (s_{BPP}),
- the channel depth (h_{BPP}),
- the inner radius (r_{BPP}),
- the outer radius (R_{BPP}),
- the draft angle (α_{BPP}),
- the sheet thickness (t_{BPP}).

For the tests samples of multi-paths serpentine flow fields shown in figure 2.4 will be used. The real dimensions or the CAD were not given, but from measurements the following can be assumed:

- $w_{BPP} = 1\,\text{mm}$
- $s_{BPP} = 1\,\text{mm}$
- $h_{BPP} = 0.6\,\text{mm}$
- $r_{BPP} < 0.4\,\text{mm}$
- $R_{BPP} < 0.4\,\text{mm}$
- $\alpha_{BPP} < 80\,°$
- $t_{BPP} = 50\,\mu\text{m}$

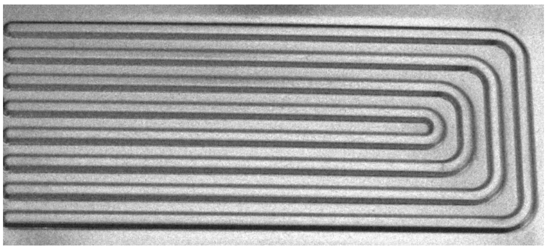


Figure 2.4: Picture of a BPP test sample.

2.1.3 Manufacturing and defects

In manufacturing the first step is to form the two half plates independently and to join them in a second step. The quality of the joining depends on the quality and manufacturing accuracy of the half plates. An inline inspection system is placed just after the forming, thus the minimum speed of the inspection is orchestrated by the maximum production speed. Each production method is mentioned below with its potential defects and its maximum production rate.

Hollow embossing (rubber pad forming). The principle of this process is to position the blank on a rubber pad and to imprint the flow field pattern on the blank via a rigid die. As the rigid die moves down, the rubber deforms elastically and provides a counter-pressure which cause the rubber and the blank to flow into the cavity of the rigid die together until all the blanks are filled [LH10]. The rubber goes elastically back to its original shape and the blank is deformed plastically into the shape of the die. Figure 2.5 shows a sketch of the needed equipment. The main defects caused by this process are cracks, uneven thickness distribution and lack of precision in the pattern due to sharp corner and radii. According to [JJHK14], the attained speed for this production process is 10 s/plate. This result in a production rate of 0.1 plates/s (6 plates/min). But according to [PPW19], the production by hollow embossing can reach a rate of 60 plates/min.

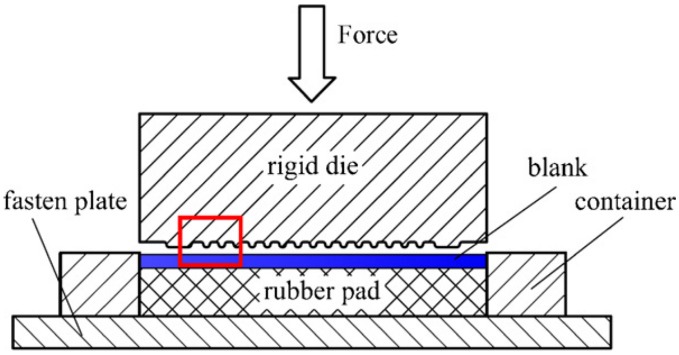


Figure 2.5: Sketch of the equipment used for bipolar plate fabrication by rubber pad forming [LH10]

Hydroforming. The principle of this process is to put the blank on a die and to compress it with hydraulic pressure. The company GRABENER Bipolar Plate Technologies is using this process among others to produce bipolar plates. On their official website[4] they explain the principle with figure 2.6 where the red line is the metal sheet, the upper grey form is the die and the blue surface is the water that applies pressure on the sheet.

The advantage, compared to hollow embossing, is that the pressure is the same at every point of the blank's surface. This enables higher drawing ratios, better surface quality and less spring back [MBJHG16]. The possible defects are the same as for hollow embossing. According to [BCZ20], the hydroforming process takes from 15 seconds for simple geometries up to several minutes for more complicated designs (like the BPP). This results in an overestimated production rate of maximum 0.067 plates/s (4 plates/min). According to [PPW19], it can reach 6.7 plates/min. A more recent and efficient variant is the laser shock hydroforming where the pressure is created by a shock wave [JMK+17]. The sheet can reach a speed of 137 m/s according to He et al. [HGL+21]. There is no concrete application on BPP-like objects

[4] https://www.graebener.com/de/hydroforming

in the scientific literature which makes it difficult to estimate the production time.

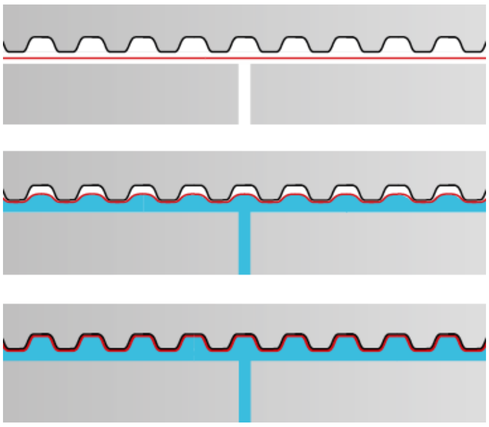


Figure 2.6: Schematic of the hydroforming of a bipolar plate.

Hollow embossing rolling. The principle of this process is to put a metal sheet between two rolls with inserts of the desired structure. The two rolls exert pressure on the sheet that will force it into the structure. The dominated deformation mode in this process is bending, which allows the channel depth to be larger without cracks [AZP$^+$19]. It also leads to a more uniform wall thickness. Figure 2.7 shows an example of a roll that was used at Chemnitz University of Technology in the work [BHA19].

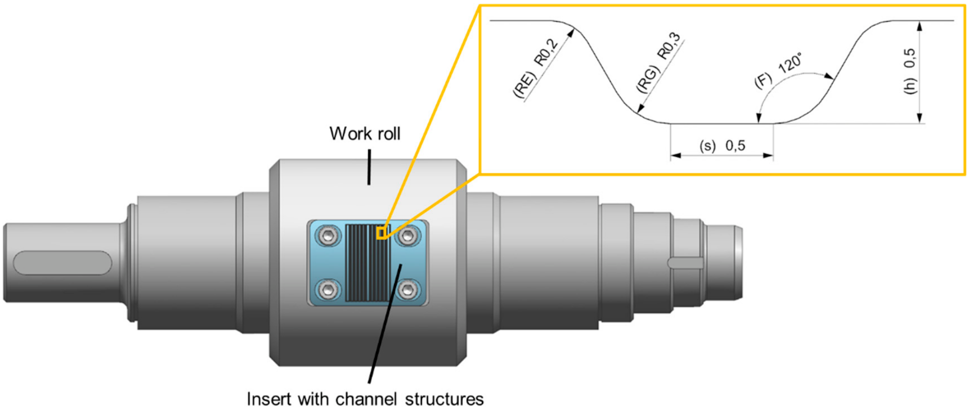


Figure 2.7: Work roll with insert for roll forming of bipolar plate channel geometries (die) and enlarged drawing of the channel cross section (die) [BHA19].

The potential defects with this process are wrinkles and distortion of the original design due to the bending. According to [BHA19] and [AZP$^+$19], the average speed of rolling is between 8 and 9 mm s^{-1}. Considering a plate size of 100 to 300 mm, it results in a production rate of maximum 0.09 plates/s (5.4 plates/min). According to [PWD20], the current developments pursue an up-scaling of the production rates up to 120 plates/min.

Electromagnetic forming. The principle of this process is to accelerate a metal sheet to high velocity via magnetic forces and to compress it against a die. The force is created by a magnetic field generated by coils through which an electric current pulse passes. This magnetic field induces a eddy currents in the metal sheet that will automatically create an opposite force to the magnetic field of the coils [SWHD10]. The system is illustrated in figure 2.8. The main advantages of this process are the single-sided die that reduces significantly the tooling costs and complexity, and improves the formability. The main disadvantages are the need of an electric conductive material, the non-uniformity of the force over the whole surface of the sheet and the need of high currents. The common defects that result from electromagnetic forming are warping, distortion and cracking [KFRF12]. According to [AOP19], the production rate of this process is from 350 to 400 parts per hour, so maximum 0.11 plates/s (6.67 plates/min).

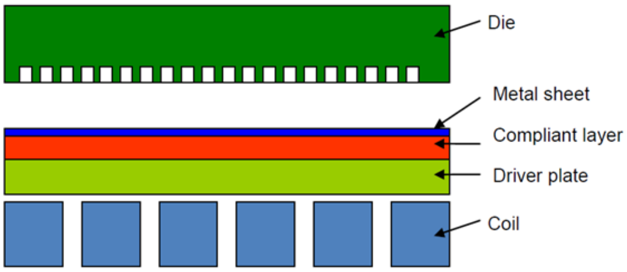


Figure 2.8: Schematic of the equipment needed for electromagnetic forming of BPPs [5][6] [SWHD10].

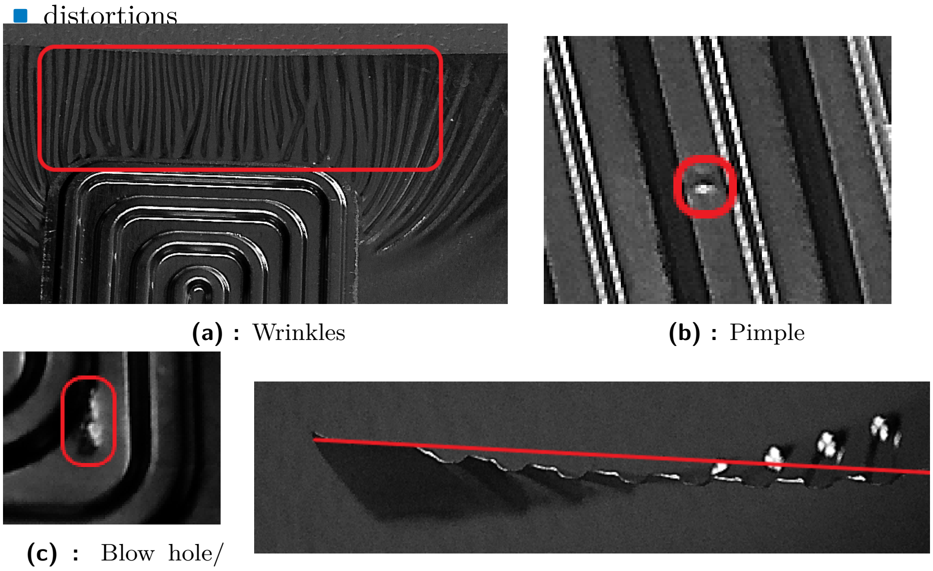
According to the literature [SGTL18] [NMD14], there is no agreed standard for defects, so there is no fixed amount of identified defects. Nonetheless, the following defects have been listed [NMD14] for stainless steel plates:

- cracks/scratches
- holes
- inclusions
- blow holes
- pimples
- wrinkles
- roll marks

[6] The Driver plate is of higher electric conductivity than the metal sheet in order to increase the force exerted on the sheet.

[6] The compliant layer aims at uniforming the force over the whole surface.

- distortions



(a) : Wrinkles

(b) : Pimple

(c) : Blow hole/ role mark

(d) : Distortion

Figure 2.9: Images of different defects observed on the BPP samples.

Added to this and to sum up the Section 2.1.3, the main defects that are supposed to be found due to the forming of the stainless steel sheet are: cracks, wrinkles, distortions, warping, uneven thickness distributions and a lack of precision in the pattern of the BPP.

Risk assessment. These topological defects can affect the fuel cell's performance. The plates are further processed with coatings before the final BPP is produced. Cracks and scratches can lead to corrosion of the stainless steel. Holes may cause leakage of cooling liquids into the cathode or anode circuit, which slows down the chemical reaction or even stops it completely. It is even worse when the hole causes a leak between the anode and cathode circuit. The reaction then takes place completely at the level of the BPP and the hydrogen ions do not cross the MEA. The fuel is then consumed but no energy is recuperated.

The pressure distribution in a cell is an important factor in the cell's operation [PL$^+$09]. A topological defect can alter this distribution and make it inconsistent. When the pressure between the BPP and the GDL is low, the electric conductivity of the cell is reduced. When the pressure is too high, the flow resistance is increased and the MEA could be deformed leading to leakage. In general, all topological defects can deteriorate the flow distribution. As seen previously, if the fuel is not evenly distributed among the cells and channels then the active area of the fuel cell stack is reduced. A reduced active area causes a loss of current and by that a loss of power.

2.2 Surface inspection systems

2.2.1 Inspection systems used nowadays

Different technologies exist to detect defects on materials. They can be separated in two categories: destructive and non-destructive [Cam13]. Some common non-destructive methods are:

- visual inspection [SGTL18]
- eddy current testing [TSTR05]
- x-ray or radiographic testing [ZDC$^+$15]
- magnetic particle inspection [BB02]
- ultrasonic testing [GMR04]

This work is focusing on automated visual surface inspection systems [NMD14][SGTL18]. The key constituents of an automated visual surface inspection system are an image acquisition unit 2.2.1 and an image processing unit 2.2.1.

Image acquisition unit

The image acquisition unit consists of the basic hardware of the system responsible for capturing the image of the examined part. It is composed of light sources and cameras. There are different types of image sensors 2.2.1, light sources 2.2.1, image processing algorithms, etc., that can be used.

Image sensors. There are two, commonly used, types of industrial cameras: charge coupled devices (CCD) and complementary metal oxide semiconductor devices (CMOS). Both technologies are based on the semiconductor material silicon and are thus sensitive in the same spectral range from approximately 300 to 1000 nm. The CCD converts the charge into voltage outside the sensor in the camera electronics and the CMOS converts it in each pixel of the sensor [IMA]. This leads to different advantages and disadvantages of each type listed in table 2.3.

Beside the type of the sensor, the most important parameter of a camera is the resolution P. The resolution must be higher than the smallest detectable defect depending on the overall size of the part to analyze. If the size of the smallest defect to detect is $a \cdot b$ mm and the size of the recorded part is $W \cdot H$ mm, then the minimum resolution formula of the camera can be determined

[7] undesired signal which appears as a brighter vertical stripe coming from a bright part of the image. [pCOG05]

[8] undesirable phenomenon where the well (pixel) capacity is exceeded and the excess charge overflows on the neighboring wells. [pi05]

CCD	CMOS
+ High image quality (High homogeneity of pixels, very uniform signal with low spatial noise) + High sensitivity even at low light intensities + Perfect global shutter (simultaneous exposure of all pixels)	+ No blooming, no smearing (can deal with higher light intensities → suitable for use with reflecting materials) + Higher frame rates at comparable resolutions (parallel image processing) + Low-cost
− Limited readout speed of the serial data stream − Smearing [7] − Blooming effect [8]	− Inequalities in uniformity between pixels in each column − Increased spatial noise

Table 2.3: Advantages and disadvantages of CCD and CMOS image sensors [IMA].

as:

$$P = \frac{W}{min(a,b)} \cdot \frac{H}{min(a,b)}$$

Looking at the product lists of different camera brands in the field of surface inspection (Cognex, Omron, Keyence and Teledyne), ranges for the parameters of nowadays cameras can be given. These ranges are summarized in table 2.4.

Resolution	0.3 MP - 42 MP
Pixel size	1.62 µm × 1.62 µm - 7.4 µm × 7.4 µm
Electronic illumination time	0.017 ms - several seconds
Frame rate	9 fps - 527 fps

Table 2.4: Summary of capabilities from nowadays cameras[9].

Light sources. The right lighting depends on the environment. The most important parameters of a light source are:

- high strength (it creates higher contrast which is a key element in image processing)
- life span
- heat radiation (some materials could be affected by the heat)
- response speed (the light source needs to be synchronized with the camera(s))

Some typical light sources are incandescent lamps (glass bulbs field with gas and a metal wire), fluorescent lamps (vacuum tubes, interaction between mercury vapor and electrodes), LED lights (light-emitting diode, semi-conductors PNP or NPN) lights. Information about LED lights and the use for different colors for the inspection is explained on the website of Metaphase Technologies [led].

[9] Taking into account the products of Cognex, Omron, Keyence and Teledyne

For highly reflecting materials, such as metal, there are two main lighting modes: bright and dark field lighting [FFF+17]. In the bright field lighting mode, the light emitted by the source is reflected by the metallic surface and enters the camera. When there is a 3-dimensional defect on the surface, the reflection will either get diffuse or get a different angle, and the emitted light will not enter the camera completely. The image at this position will be less bright than it should be. If the defect is two-dimensional, there can be a color change of the material at the position. It will absorb a slightly different part of the light than the rest of the part. The system will recognize the defect by a different color/grey tone.

In the dark field lighting mode, the emitted light is also reflected by the metallic surface. Usually, it does not enter the camera because the latter is not placed at the reflection angle, which is equal to the incident angle. This leads to a picture, whichl is darker, hence the name of the mode. A three-dimensional defect will diffuse the reflection at this position and the camera will record a higher luminous intensity. As opposed to the bright mode, the placement of the light source in relation to the camera does not need to be accurate. The recorded reflection does not change even when the incidence angle changes a few degrees.

Image processing unit

The image processing unit is responsible for detecting the defects, marking, classifying and displaying them. As the marking and the displaying are not key challenges for the programming, this report will focus on the existing detection algorithms 2.2.1 and the classification methods 2.2.1.

Detection methods. In 2018, Sun et al. [SGTL18] reviewed 90 papers of studies of software and hardware for visual detection from over the past 30 years. In 2020 was published the survey of Czimmermann et al. [CCM+20] where over 220 scientific articles about defect detection methodologies were reviewed. The combination of these works gives a picture of the different detection methods. The table C.1 in appendix C summarizes the strength and the weaknesses of every method mentioned in the reviews. According to [SGTL18] [CCM+20], the detection methods can be ranged in five basic theories:

- statistical (co-occurence matrix, histogram properties, local binary model)
- structural (primitive measurement, edge features, skeleton representation, morphological operations)
- filtering (spatial filtering, frequency filtering, Gabor transform, Wavelet transform, multiscale geometric analysis)

- models (fractal model, Markov random field model)
- machine learning (artificial neural network, convolutional neural network, sparse coding, moving center hypersphere)

The most appropriate image processing techniques for this work are listed in table 2.5 with their strengths and weaknesses. Choosing factors were: the invariance to translation, rotation and expansion, the computational complexity, and the ability to detect small defects ($< 1\,\mathrm{mm}$). In 2019, Wiener et al. [WPG+19] developed an inline quality monitoring system of coated metallic bipolar plates after the forming process. For its application they used an algorithm that compares the ideal contours of a part with its actual contours. The steps for the image processing were:

	Strengths	Weaknesses
Histogram properties	- Translation and rotation invariance - Simple calculation	- Low detection rate (50 – 70 %) for irregular textures - Sensitive to noise - Requires the assumption that the intensity of defective regions are separable
Other gray level statistics	- Suitable for low resolution images	- Low timeliness - No automatic threshold selection
Primitive measurement	- Simple - Easy to understand and implement	- Sensitive to non-linear noises
Morphological operations	- Computationally simple - Highly suitable for random or natural textures	- Only suitable for aperiodic defects
Frequency domain analysis	- Invariant to translation, expansion and rotation - Suitable for the detection of global and local defects - FFT (Fast Fourier transform) calculation time is short (600 pixels with 2.2 ms)	- Difficult to realize non-interference when dealing with frequency-domain components related to background or defect - Not suitable for random texture detection - Lack ability of spatial orientation
Wavelet transform	- Suitable for multi-scale image analysis - High detection rate (83-97 %) - Efficient image compression with less information loss	- Easily to be affected by feature correlation between the scales

Table 2.5: Strengths and weaknesses of appropriate image processing techniques [SGTL18][CCM+20].

- correction of lens curve

- correction of position (via translation and rotation matrices)
- recognition of contours (outer contours via Canny algorithm)
- comparison of contours with the ideal set

In result they were able to analyze a 50 mm × 50 mm surface in less than 5 seconds with an accuracy of 1 µm.

Classification methods. The typical classifiers can be split in three categories:

- supervised
- unsupervised
- semi-supervised

Supervised classifiers received labeled examples from which they produce an inferred function to make predictions about the output values. Than they compare their outputs to the correct output (also given) in order to find errors and modify the model accordingly. Among them, there are: K-nearest neighbor (KNN), artificial neural network (ANN), support vector machine (SVM), discriminant function and fuzzy logic.

Unsupervised classifiers did not receive labeled examples and correct output. They study how systems can infer a function to describe a hidden structure from unlabeled data. Semi-supervised classifiers are between supervised and unsupervised algorithms. They usually have been given a small amount of labeled and a large amount of unlabeled data. Among the unsupervised/semi-supervised classifiers, there are self-organizing mapping, statistical/Novelty detection and Gaussian mixture model.

2.2.2 Theory of defect detection by laser grid projection

The disclosure document from A. Pierer and T. Wiener [pat] is fundamental for further investigations. It states a principle to detect topographical defects on surfaces of a component, in which at least one pattern is projected onto a surface area of a component. At least one camera is used to produce an image of at least a partial region of the surface on which the respective pattern is projected. An electronic evaluation unit then detects if there are local distortions or local changes of distances or angular orientations of the projected pattern on picture elements and classifies them as defect areas [pat]. In order to take few images and still be able to inspect an entire area quickly, a grid pattern is used as a projection as sketched in figure 2.10. The result is the same as if a single stripe had been used and spread, by taking different pictures, over the whole surface of the part. The working principle of the electronic detection and classification unit is not clarified. Any method

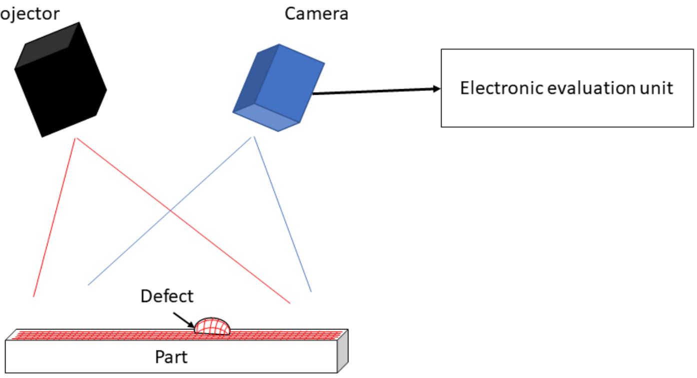


Figure 2.10: Sketch of the theoretical set-up of the laser grid detection system.

mentioned in Section 2.2.1 could be used. The algorithm can also be inspired by similar detection systems working with structured light. According to the website [optb]: "The term structured light or structured illumination refers to the projection of light with a known shading pattern. The result is the projection of a known light pattern on the captured scene. The main purpose of structured light projection is to detect and measure the deformation of the expected pattern on the scene. As an immediate consequence, structured light is used in 3D reconstruction of objects."

Examples of applications. Forte et al. developed an inspection system based on the combination of bright field and dark field lighting using structured diffuse LCD light [FFF$^+$17]. According to their results, the defect contrast went from 40 % to 90 % using a structured light source instead of a uniform light source. Han et al. [HFY20] used structured light (a laser stripe) to do weld quality inspection. Xu et al. [XXZ$^+$11] used it with the triangulation model (explanation in Annexe D.3) to do a real-time 3D shape inspection system of automotive parts. This measuring technique is also called phase measuring profilometry [Gen11]. A similar measuring method is the phase measuring deflectometry [HIZA18]. The difference is that PMP works with the diffused reflection and PMD uses the specular reflection. A product that is similar to this is proposed by KEYENCE, the CV-X/XG-X series[10]. The principle is also to project a light pattern on the part with LEDs and reconstitute it with triangulation of the image the 3D model of the part. Then, the calculated 3D model is compared to the ideal model and deviations are marked as defects.

The goal of this work is only to recognize the defects without measuring them. Calculating the 3D model is not required.

[10] https://www.keyence.de/ss/products/vision/pattern-projection/

Chapter 3

Development of the set-up

3.1 Requirements analysis of the inspection system

Nr.	Description	Imp[1]	Characteristics	Method of verification
1.	Max testing area	3	Min 373 cm^2	Measuring the recorded area
2.	Inspection rate	2	Min 1 Hz	Measuring time between start of projection and showing of results
3.	Acquisition time	3	Max 26 µs	Camera settings
4.	Accuracy	1	Min 5 µm	Measuring the smallest detectable surface defect via microscope or by zooming on the region of interest
5.	Detection rate	1	90 %	Number of right defect detections divided by the total number of defect detections
6.	False defect rate	1	Max 10 %	Number of false defect detections divided by the total number of defect detections
7.	Risk group according to CEI EN 62471:2010	1	exempt	Datasheet of light source
8.	Invariance to the part position	2	rotation translation	

Table 3.1: Requirements table of the inspection system of one BPP.

[1] Importance factor: 1 most important - 5 least important.

Table 3.1 presents the different requirements of the inspection system for a BPP. The origin of these requirements is explained in the paragraphs below.

1. The two models of fuel cells powered cars currently in production are:

- Toyota Mirai: the cell thickness is 1.34 mm, the number of cells is 370 and the total volume 37 L which gives an estimation of $0.1\,\mathrm{L} = 100\,\mathrm{cm}^3$ per cell → $746\,\mathrm{cm}^2$.
- Hyundai NEXO: the number of cells is 442, with an active area of $270\,\mathrm{cm}^2$. The stack's power density is $3.1\,\mathrm{kW\,L}^{-1}$, the stack power is 95 kW, thus the volume of the stack is 30.65 L. So the volume of one cell is $70\,\mathrm{cm}^3$ and assuming the cell thickness is between 1 mm and 2 mm, the area of one cell is between $350\,\mathrm{cm}^2$ and $700\,\mathrm{cm}^2$.

Every cell needs to be clamped and sealed. So there is a part of the surface dedicated to this function. The sensible part of the area is the active area (the surface that is in contact with the MEA and the gas). Assuming the active area is half of the total surface, the minimum area to be inspected is $\frac{746}{2} = 373\,\mathrm{cm}^2$.

2. The highest production rate found in section 2.1.3 is 1 plate/s (by rolling). That means, the inspection timing needs to be at least as fast as 1 plates/s in order not to slow down the production process and lower the efficiency of the manufacturing. Thus, the inspection time of one plate must be under 1 s.

3. When the highest production rate by rolling is 1 plate/s, and the area is $373\,\mathrm{cm}^2$, by approximating the geometry of the plate to a square ($19\,\mathrm{cm} \times 19\,\mathrm{cm}$), the speed of the part is around $19\,\mathrm{cm\,s}^{-1}$. During the acquisition time, the part is not allowed to move more than the smallest defect to recognize to avoid blurring. According to requirement Nr. 5, this distance is 5 µm. So, the maximum acquisition time is $\frac{5\cdot 10^{-3}}{190} \simeq 2.6 \cdot 10^{-5}\mathrm{s} = 26\,\mu\mathrm{s}$. This is evidently only applicable for the rolling process. During the other processes, the part is not moving at a constant speed but rather displaced by regular impulses.

4. According to [PL+09], Liu et al. studied the effect of dimensional error of metallic BPP on the GDL pressure distribution in the PEMFC and found out that "the allowed maximum standard deviation of the dimensional error is calculated as 5 µm".

5. and 6. Both requirements depend on the material and financial loss the company is willing to make. If the detection rate is 90 %, the risk of the component being defective is 10 %. This can be a lot if the defects cause major accidents. But as seen before, these defects will mainly reduce the power and energy output of the fuel cell. A rate of 90 % is therefore realistic. On the other hand the false defect rate does not affect the final product but the production costs. A rate of 10 % means that 10 % of the produced BPPs are taken out of the production line although they may not have been defective.

7. The operator working on the line must not be under any risk of blinding or burning. According to CEI EN 62471:2010 there are different risk groups giving the photobiological safety of lamps and lamp systems. For the operator to be safe, the system must be exempt from every risk group.

3.2 Equipment

3.2.1 Laser grid projector

The laser is not actually a laser source but an LED strobe pattern projector from Opto Engineering (product designation: LTPRUP-R). This is due to several advantages over lasers. LED enables thinner lines, sharper edges and a more homogeneous illumination. A laser beam doesn't have the same intensity at every radii, the center is brighter than the borders. The strobe mode gives a higher light intensity for a short amount of time at a given frequency. It helps to get sharper images of moving objects, while preserving the lifetime of the LED.

The projector emits a red light with a wavelength of 618 nm and a spectral half-width of 20 nm. It is controlled by the controller LTDV1CH-17V and is used with a lens (EN2MP3514) and a grid pattern (PTGR020035P).

The lens. From its focal length f' and its back focal length BFL can be deduced the working distance u (distance between front lens and projection) and the diameter d of the projection area. Assuming that the optical axis is perpendicular to the projection surface, the system can be represented as in figure 3.1.Its minimum working distance is $u = 30\,\text{cm}$. As the accuracy requirement is more important than the testing area requirement, the projection will be chosen as small as possible in order to have the thinnest possible lines projected on the part. This is achieved by using the shortest working distance. From this, the distance between lens system and projection s can be deduced 3.1:

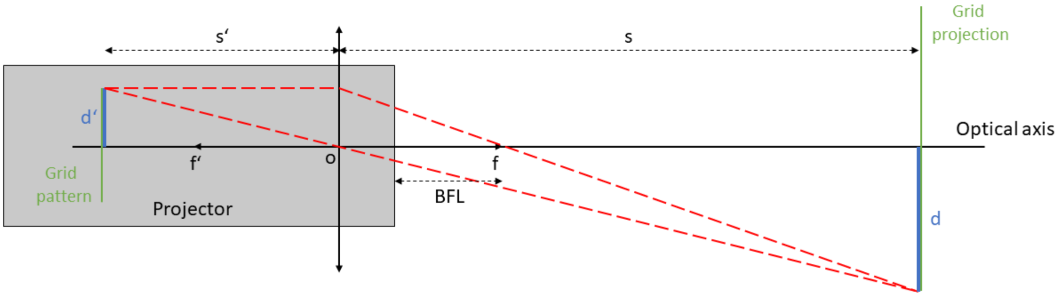


Figure 3.1: Schematic of the optical system of the projector

$$s = u - BFL + f \tag{3.1}$$

According to the datasheet of the lens, the BFL is 14.7 mm and the focal length is 35 mm, so $s = 320.3$ mm. The distance between lens system and pattern s can be found from equation 3.2 (demonstration of this equation in appendix D.2).

$$\frac{1}{s} + \frac{1}{s'} = \frac{1}{f} \tag{3.2}$$

The projection area can be computed by using trigonometrical laws and the size of the pattern that is 11 mm × 11 mm.

$$\begin{aligned} \frac{d'}{d} &= \frac{s'}{s} \\ \iff d &= d' \cdot s \cdot \left(\frac{1}{f} - \frac{1}{s}\right) \\ &= d' \cdot \left(\frac{(u - BFL + f)}{f} - 1\right) \\ &= \frac{d'}{f} \cdot (u - BFL) \end{aligned} \tag{3.3}$$

The minimum size of the projection is a circle of $d \simeq 89.7$ mm diameter ($63.1\,\text{cm}^2$) area; which is 6 times smaller than the requirement. The largest rectangle $H \cdot W$ fitting in this circle is a rectangle with a diagonal being the diameter of the circle. This leads to equation 3.4.

$$d^2 = W^2 + H^2 \tag{3.4}$$

If the ratio is $R = \frac{H}{W}$ it results in the following system.

$$\begin{aligned} W &= \frac{d}{\sqrt{1+R^2}} \\ H &= W \cdot R \end{aligned} \tag{3.5}$$

To be able to test a larger surface in the same amount of time, a larger amount of hardware is needed. The general set up is repeated as many times as the tested surface requires it.

The grid pattern. As mentioned above, the grid pattern was custom made and delivered with the projector. Its product designation is PTGR020035P. Its line thickness t is 20 µm and the line spacing sp is 35 µm (see figure 3.2). According to Opto Engineering, this projection pattern has a geometrical accuracy of 2 µm .

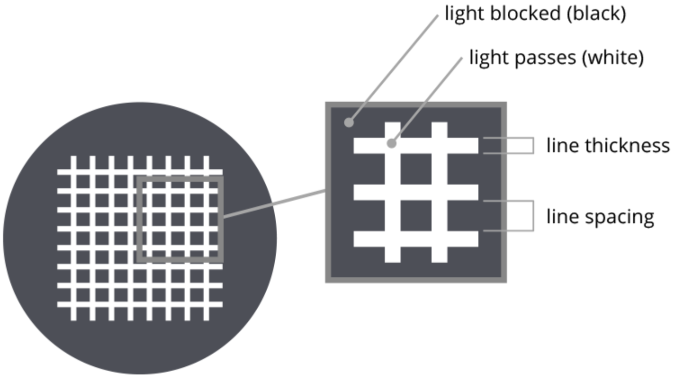


Figure 3.2: Schematic of a grid pattern from Opto Engineering [a]

[a] `https://www.opto-e.com/products/pattern-PTGR050450-P`(10.04.2022)

The line length is 11 mm (d'). From this can be deduced the number of lines (n):

$$n = \frac{d' - t}{t + sp} \tag{3.6}$$

Which gives about 200. The grid is composed of 200 x 200 lines of 20 µm thickness and 35 µm spacing. As seen previously, the 11 mm diameter pattern becomes a 89.7 mm diameter projection, so the projected line thickness is $\frac{89.7}{11} \cdot 0.02 \simeq 0.163\,\text{mm}$ and the projected line spacing is $\frac{89.7}{11} \cdot 0.035 \simeq 0.285\,\text{mm} \pm 0.016\,\text{mm}$ (taking into account the manufacturing pattern accuracy).

3.2.2 Camera

The camera works like the projector but instead of emitting the light it is capturing it. The biggest issue is to find a camera that is able to record and focus the area $H \cdot W$ while being at an acceptable distance from the part[2] and being able to recognize the smallest defect of 5 µm. The manufacturers do not give the BFL, but the pixel size p'. Knowing the camera's resolution, it is possible to find the size of the observed area by one pixel p. Using the definition of the magnitude in equation 3.7 and equation 3.2, the working distance of the camera can be computed.

$$M = \frac{d'}{d} = \frac{s'}{s} = \frac{p'}{p} \tag{3.7}$$

$$3.2 + 3.7 \rightarrow s = f \cdot \frac{M+1}{M} \tag{3.8}$$

Three different cameras from The Imaging Source Europe GmbH[3] were available on time at the laboratory:

- a 1MP GigE monochrome zoom camera with CCD sensor (DMK Z12G445),
- a 1MP GigE color zoom camera with CCD sensor (DFK Z12G445),
- a 5MP GigE monochrome camera with CMOS sensor (DMK 33GX264) and its lens (TCL 1216).

Table 3.2 is obtained using the previous equations.

To be able to focus the part at a magnitude of 0.067, the camera must be at a distance between 76 mm and 917 mm from the part. According to its datasheet, the maximum operating distance is 700 mm. So the camera must be at a distance between 76 mm and 700 mm from the object. The focal length of the 5MP camera is fixed at 12 mm by its lens with a minimum working distance of 0.1 m. The obtained working distance $s \simeq 101.6$ mm is above the minimum working distance. To reach the requirement of 5 µm accuracy, a camera of at least 151 MP must be used. However in chapter 4 about the software development, it will be found that the inspection accuracy

[2] If the camera needs to be very close to the part (< 5 cm), the camera itself is very likely to create a shadow on the projections. On the opposite, if the camera is too far from the part (>100 cm), the overall size of the inspection system would be unreasonable.

[3] `https://www.theimagingsource.com/products/zoom-autofocus-cameras/gige-monochrome-zoom-cameras/dmkz12g445/` (10.04.2022)

[4] The needed camera settings for p to be lower than the smallest defect to identify which is of 5 µm length. The existing camera resolutions were taken from `https://www.visionsystech.com/series/100-mp-200-mp-ultra-high-resolution-cameras`

	1MP	5MP monochrome	Ideal camera[4]
Resolution (pixels)	1280 x 960	2448 × 2048	14192 x 10656 (151 MP)
R	1.33	1.195	1.33
W (mm)	53.9	57.6	53.9
L (mm)	71.7	68.8	71.7
p (μm)	56	28	5
p' (μm)	3.75	3.45	3.76
M	0.067	0.123	0.744
f_{min}	4.8	12	45
f_{max}	57.6	12	45
s_{min}	76.6	110	105
s_{max}	919	110	105

Table 3.2: Geometrical characteristics of the cameras' optics.

actually depends on the grid size and not on the projected pixel size, so as long as the projected pixel size allows to recognize the grid, the camera can be considered for the tests which is the case for all three cameras.

The cameras have a global shutter which enables a short exposure time, because every pixel is exposed at the same time for a same duration. Their maximal sensitivity is for wavelength from 550 to 620 nm. According to the spectral sensitivity diagram in appendix E.1,E.2 and E.3 the sensitivity for a wavelength of 618 nm like the one from the projector is over 90 %. So the three cameras are adapted to the projector.

To determine which one to use, pictures from the same BPP test sample have been taken trying to find the best camera settings to get a sharp picture. The grid projection does not change during those tests. The corresponding camera set-ups can be found in table 3.3.

	1MP monochrome	1MP color	5MP monochrome
Camera inclination	22 °	22 °	22 °
Working distance (mm)	80	80	110
Exposure (ms)	50	50	50
Zoom	60 %	60 %	na.[5]
Focus	by OnePush[6]	by OnePush	na.
Aperture stop	na.	na.	8

Table 3.3: Camera set-ups for figure 3.3.

At first sight, 1MP cameras presents less noise then the 5MP camera. The

[5] Not applicable.

[6] Option on IC Capture, the software from The imaging source, that enables to find the best focal length automatically.

5MP camera works with CMOS sensors. That means it is less sensitive to light intensity. The 1MP cameras are with CCD sensors which have a higher light sensitivity. This could be a reason why the resulting image with the 1MP camera looks sharper and brighter than the image taken with the 5MP camera as the intensity of the projector is not very high. By zooming into the center of the pictures, the figure 3.3 is obtained.

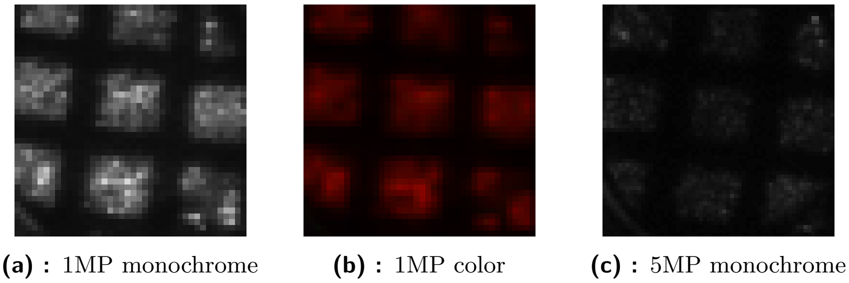

(a) : 1MP monochrome **(b) :** 1MP color **(c) :** 5MP monochrome

Figure 3.3: Zoomed pictures of a BPP test sample.

When looking at the gray value distributions of the three pictures, the color camera presents a standard deviation of 56.7. The monochrome 1MP camera's distribution has the highest standard deviation of 66.2. Because of it and because the pictures from the color camera use three times more storage space saving the intensity levels for green, blue and red, there is no reason to privilege the color camera to the monochrome camera. Especially since in this case there is no defined color defect.

The 5MP camera has the lowest standard deviation of 23.7.The 1MP monochrome camera also uses less data (four times less than the 5MP camera[7]). In addition, it is more sensitive to low light levels and thus allows the capture of images with shorter exposure times, which are necessary for inline inspection. This is why in the following the 1MP monochrome camera (DMK Z12G445) will be used. However, the 5MP will be kept in mind in case a higher resolution is required to obtain suitable results.

3.3 Electrical set-up

For the electrical installation, the projector is delivered with a controller (series number: LTDV1CH). This strobe controller has a single, fixed constant-current pulsed output with a nominal current of 17.0 A. It is driven by a 24

[7] The 1MP cameras have actually around 1.2 Megapixel but is called 1MP for simplification.

V DC power supply (NDR-120-24 from MEAN WELL[8]). It can be used in pulsed mode where the lighting intensity is triggered by digital input (camera or other sources). It also has a trigger output to synchronize and trigger a camera.

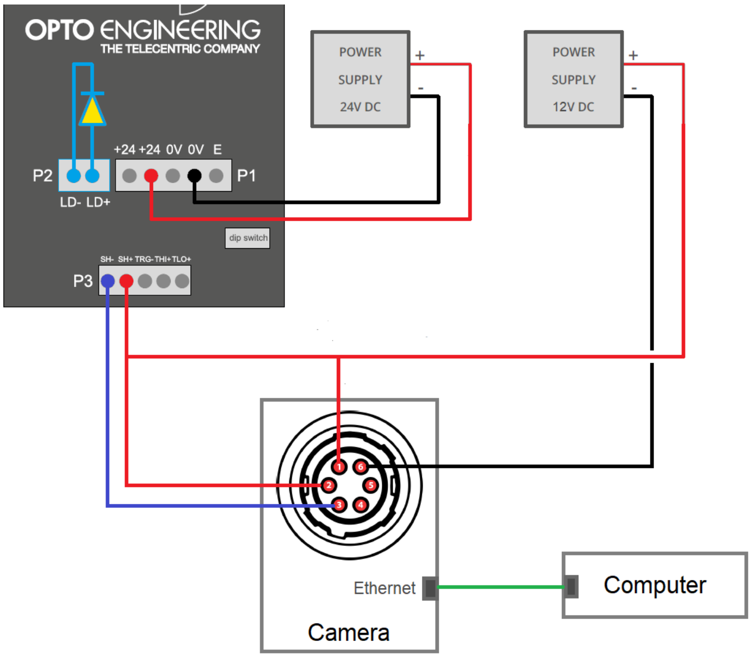


Figure 3.4: Scheme of the electrical set up adapted from illustration 9. in [OPT22].

For the first tests, the test mode was switched on using the free running oscillator, an autonomous asynchronous source as trigger[9]. It has a frequency of 5 Hz and generates short pulses with a width of 500 µs. The camera is connected via Ethernet network cable to the computer and via 6-pin I/O connector to a 12 V DC power supply (NDR-120-12 from MEAN WELL[10]) and the trigger outputs of the projector's controller, as pictured in figure 3.4. The characteristics of the camera are controlled with the supplier's software called IC Capture[11].

[8] `https://www.meanwell.com/` (11.06.2022)

[9] All switches are turned OFF except for the RNG1 and the TEST switch.

[10] `https://www.meanwell.com/` (11.06.2022)

[11] `https://www.theimagingsource.com/products/zoom-autofocus-cameras/gige-monochrome-zoom-cameras/dmkz12g445/` (05.06.2022)

3.4 Frame construction

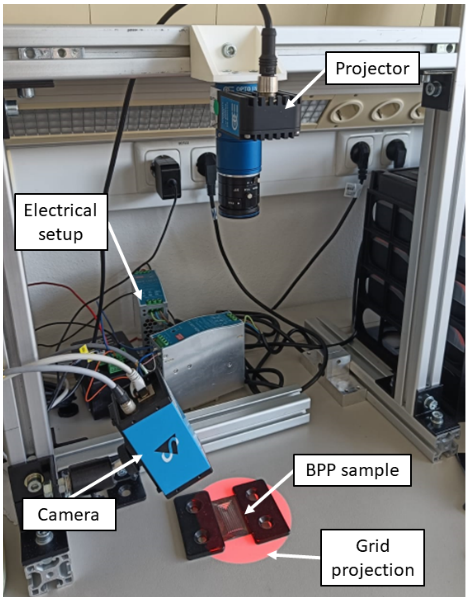


Figure 3.5: Picture of the set-up.

Figure 3.5 shows a picture of the actual set-up. The structure is made of aluminum profiles and corner brackets. When positioning the equipment, three important points must be considered:

- the camera position in relation to the part,
- the projector position in relation to the part,
- and their projector and camera position in relation to each other.

The camera position

As seen previously, an important factor is the distance between camera and object. This distance must be between 76 mm and 700 mm to be able to focus the object. To ensure a more stable image, the working distance must be kept as small as possible. Another important factor is the angle α between the optical and the vertical axis of the camera. If the object is not on that plane, it will appear blurred because the focal plane of a lens is always perpendicular to the optical axis. There is an acceptable depth around this plane in which the object is still in focus, without being exactly on it. This depth is called the Depth of Field (DoF). In the case of the camera the DoF can be defined by equation 3.9 from [opta], where the DoF in mm, the aparture stop $F/\#$, the sensor pixel size p' in μm, a given coefficient k for defect inspection (= 0.015) and the magnitude M.

$$DoF = F/\# \cdot (1 + M) \cdot p' \cdot \frac{k}{M^2} \tag{3.9}$$

$M = 0.067$ and according to the datasheet, $p' = 3.75$ and $F/\# \in [2.2, 2.3]$, so the $DoF \in [27.6, 28.8]mm$. As the object is 71.7 mm · 53.9 mm to have it completely in focus at the same time, the observation angle α must be smaller than 30.8°. Demonstration can be found in appendix D.4.

To confirm the theoretical values, the following test was performed. Pictures of the perpendicular projection on a horizontal white paper were taken with different camera angles. For each picture, the standard deviation of the gray level distribution from the 100 upper and 100 lower lines was computed and compared. The higher the deviation, the better the contrast and the focus.

The projector's optic axis was always vertical at 350 mm from the paper and the camera is fixed on a ball joint at a vertical camera to table distance of 120 mm. The angle is changed by using the joint. A point placed in the middle of the picture (point on the camera's optic axis) is also placed in the middle of the projection (point on the projector's optic axis). The projector was horizontally relocated for both optical axis to cross. All pictures were taken under the same camera properties, except for the focus[12] and the orientation of the optical axis (same zoom: 60 %[13], brightness, etc.). The results are shown in table 3.4.

Angle (in °)	Relative standard deviation of first 100 lines in %	Relative standard deviation of last 100 lines in %	Sum
16	10.5	11.9	22.4
19	10.3	11.5	21.8
21.5	9.5	11.7	21.2
22	10.0	11.3	21.3
24	9.7	11.3	21.0
27	8.3	11.6	19.9
31	9.2	10.0	19.2
34	7.6	8.3	15.9

Table 3.4: Standard deviation of pixel intensities from pictures taken with different camera angles.

Under these conditions, the sharpness seems to decrease slowly for angles over 24°, and more drastically after 31° which was the expected limit. So with a zoom of 60 % and a vertical camera to table distance of 120 mm, the camera angle must not exceed 31°. To make sure that there is no loss is accuracy, the applied limit in this work will be 24°.

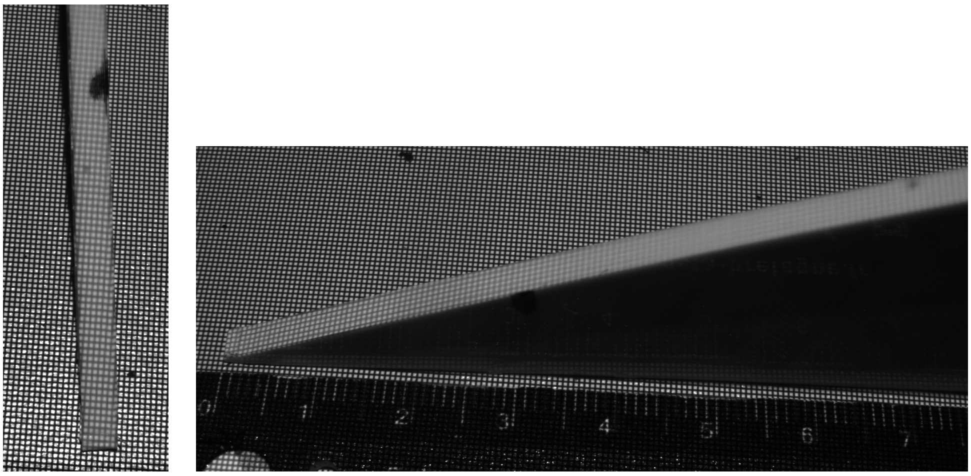
[12] As the ball joint was fixed, the distance between camera and object varied a little and the focus had to be readjusted

[13] The zoom was chosen in order to have a picture of the largest surface of the projection only.

The projector position

As seen previously, the minimum working distance of the projector is 300 mm. To have the best resolution, the grid projector must be as small as possible. So the projector is positioned at 310 mm (±10 mm). As the system of a projector is basically the same as the system of a camera, it is equally affected by the DoF.

In order to get the DoF of the projector, a 3D printed right triangular prism of depth 4 mm, long side 100 mm and short side 50 mm, is placed on the inspection surface. So by placing the surface of the long side on the inspection plane, the angle between the surface of the hypotenuse and the inspection plane is 22.5°. The projector always projects vertically on the object and is focused on the table (manually) and the camera is set to a zoom of 27%, a focus of 138, an exposure of 1 ms and an angle of 22°. This means that the interesting plane of the part is almost parallel to the focusing plane of the camera, so if a point on this plane does not seem focused, it is not because of the camera's but because of the projector's DoF. The hight for which the pattern does not focus anymore is half the DoF. The obtained pictures are shown in figure 3.6.



(a) : Front

(b) : Side

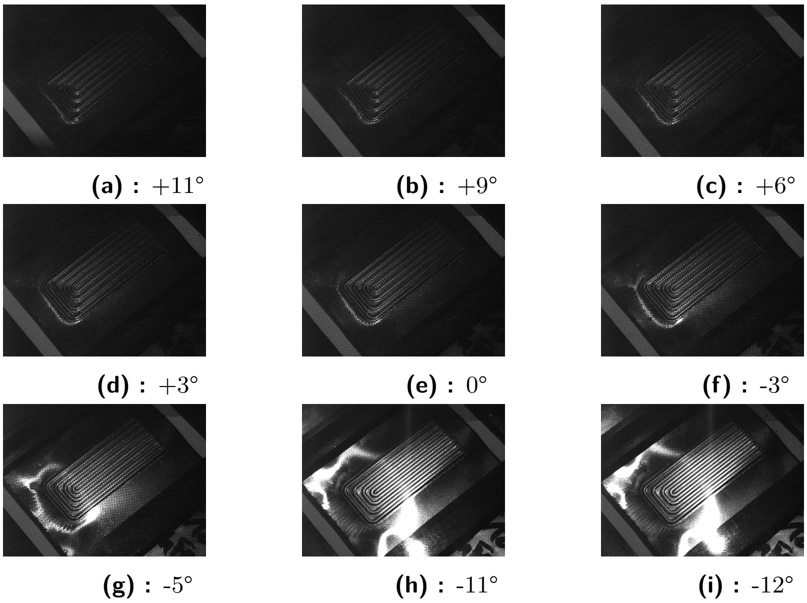
Figure 3.6: Pictures of the 3D printed right triangular prism.

This experiment is repeated two more times removing and replacing the components (camera, projector, triangle) each time. The resulting picutres can be found in appendix I.2. On figure 3.6a and every other picture in the appendix, the grid is no longer distinguishable from the marked point. On figure 3.6b, this marked point is placed at 3 cm from the bottom of the prism.

So the height of this point is $\frac{50}{100} \cdot 30 = 15\,\text{mm}$. In conclusion the DoF of the projector is 30 mm. Using the equation D.8, the maximum acceptable angle for the projector is $\sin^{-1}\left(\frac{30}{71.7}\right) \simeq 25°$.

Camera to projector position

As mentioned in section 2.2.1, there is bright and dark lighting. The projector's light spreads from the lens into a conical manner. This causes a different incident angle for each light ray and makes it difficult to achieve completely bright or dark mode as some rays will likely enter the camera and other will not.


(a) : +11°

(b) : +9°

(c) : +6°

(d) : +3°

(e) : 0°

(f) : -3°

(g) : -5°

(h) : -11°

(i) : -12°

Figure 3.7: Set of images of a BPP test sample under different lighting angles. The camera settings were set on: zoom 60 %, focus 260, brightness 8, Exposure 0.5 ms. All other camera parameters were default parameters.

To illustrate this, some images were taken to see how the angle between camera and projector affects the final image. The camera was set to an angle of 25° to the vertical centered on the object. Both, camera and object, did not move during the shooting. The projector's optical axis is moving on a plane and always crossing the object. The images are shown in figure 3.7 The name of the pictures corresponds to the angle between this axis and the vertical: + when the projector is on the side of the camera (more dark mode), - when the projector is on the opposite side of the camera (more bright mode). All pictures are taken with obstruction of other illumination sources and the diaphragm of the projector fully opened.

To compare these pictures, a threshold at the gray level 255 was used showing only the brightest points. The code can be found in appendix H. Any white surface on the picture is also entirely white on the original picture. The bigger the angle between optical axis of camera and projector is, the bigger is the white surface. Logically, when the angle is large enough, the white surfaces should get smaller, but it makes no sense to place the projector at this angle as the projection would not be focused as explained in section 3.4. On the other hand, when the angle is too small, the picture would get too dark. The results are shown in figure 3.8.

The best position for the projector is projecting vertically. When the projector shines from the same side as the camera records, the picture is too dark and the pattern is not recognizable. When the projector shines from the opposite side, the pictures get too bright at some points and make the pattern once again indistinguishable. Positioning the projector vertically to the BPP also simplifies the assignment in relation to its symmetry. When using multiple cameras to catch every angle, as explained in section 3.4, if the projector is leaning on the same side as one of the cameras, it will likely be leaning on the opposite side for another camera. But, if the camera is perpendicular to the BPP, this does not happen. The geometry of the BPP can cause overexposed areas even with this projector position. Some examples can be seen on figure 3.9. To avoid this, sensors with Wide Dynamic Range (WDR) technology can be used [YP99] or post processing techniques like High Dynamic Range (HDR) [BQO$^+$10] that consist into taking several pictures with different exposure times and combining them. A detailed explanation

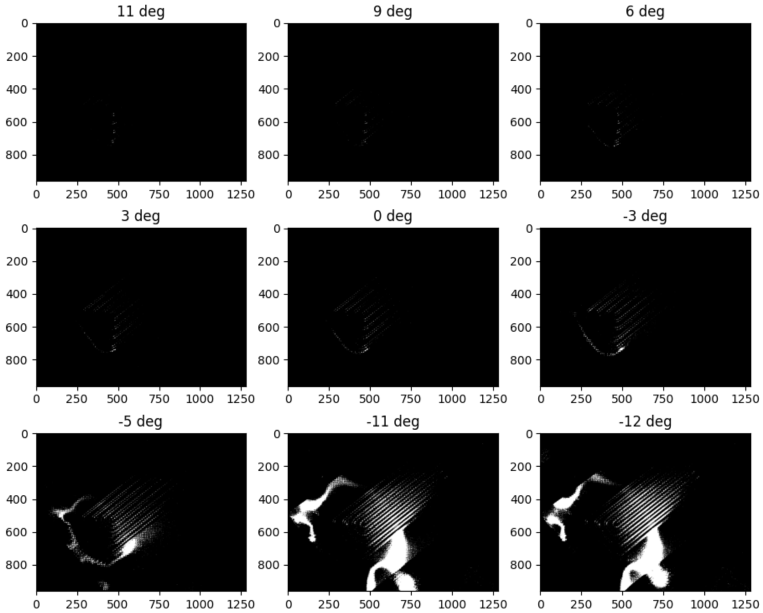


Figure 3.8: Set of images from figure 3.7 after threshold.

with the results can be found in section 4.1.1.

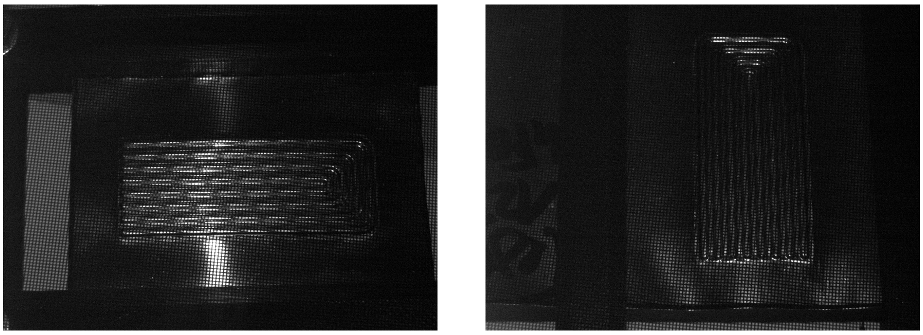


Figure 3.9: Pictures of a BPP test sample showing overexposure. Projector angle: 0°, variation in the BPP position, all other settings are the same as explained in the beginning of the section.

Number of cameras

The geometry of the BPPs with its channels/recesses does not allow to analyze the whole surface from a single shot. In order not to lengthen the acquisition time, it is necessary to add cameras that will simultaneously capture an image from different point of views in order to inspect the entire surface. The restrictions are, that the angle of the observation axis of each camera to the vertical needs to be smaller than 24° and their part to camera distance must be minimum 76 mm.

To simplify the problematic, the following assumption is made. If the surface of an empty, top open cube can be observed (the dimensions of the cube being the largest depth $h_{BPP} = 0.6\,\text{mm}$ and the shortest width $w_{BPP} = 0.6\,\text{mm}$ of a BPP channel), then the system is suitable for capturing the surface of a BPP. The dimensions are based on the typical dimensions for a metallic BPP, overestimating the channel's depth and underestimating the channel's bottom width 2.1.2 [RNJ^{+}21] (worst case scenario to observe the entirety of the channels surface). Positioning the camera at the maximum of $\alpha = 24°$ from the vertical, the biggest "shadow" (not recorded distance of the cube's bottom) is equal to $h_{BPP} \cdot tan(\alpha) \simeq 0.27\,\text{mm}$, which is less than the half of w_{BPP}. So the center point of the bottom surface is observable. By using one camera, as the observation axis can not be vertical because the projector is already vertically oriented, there will always be parts of the cube's inside and outside that will not be observable but it never will be larger than 0.27 mm from the border of the bottom face to the center of this face[14].

[14] this statement assumes that all light rays coming from the object reach the lens parallel to the observation axis which is not true in real application, but this variation is negligible.

For symmetrical reasons pictured in figure 3.11, there must be minimum four cameras to have 100 % certitude to capture the complete surface. Taking only three of the four pictures there will always be one corner that is not recorded. The dimensions of figure 3.11 are not real. The drawings should just make it easier to visualize.

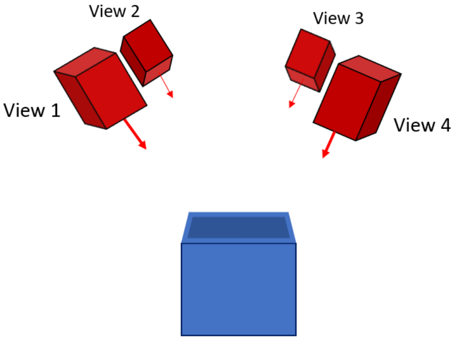
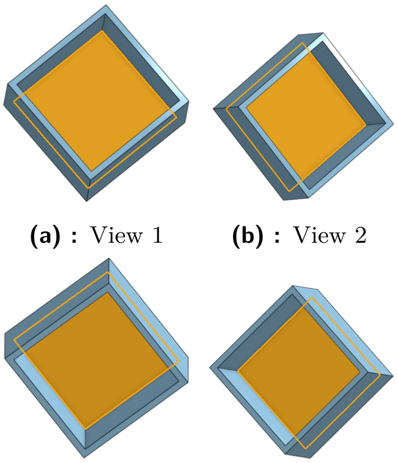


Figure 3.10: Description of the views from figure 3.11



(a) : View 1 **(b) :** View 2

(c) : View 3 **(d) :** View 4

Figure 3.11: Exemple for four point of views of the same empty cube. The orange surface is the bottom of the cube (no real dimensions.

3.5 Partial conclusion

Finally, by applying all the methods mentioned above and being close to the ideal positions, the images before processing look like the images on figure 3.12. The camera and projector settings are summarized in table 3.5 and will stay the same for all the future tests if not mentioned.

Unlike in the calculations the grid is larger than expected because the projector is at a distance of 35 cm instead of 30 cm (to give space for right focusing) and the observed pixel area is smaller because the camera is more zoomed in to better distinguish a line (60 % zoom[15]). The final grid properties are summarized in table 3.6.

[15] Maximum zoom at a working distance of 80 mm for which a focusing on the projection is still possible.

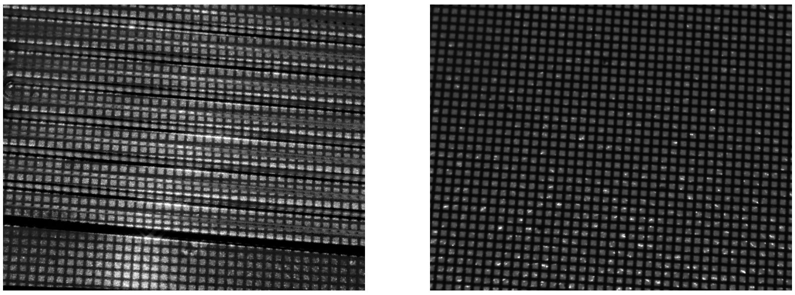


Figure 3.12: Final images before processing with 3.12a and without 3.12b BPP test sample.

Projector	working distance (mm)	350
	focus	set manually
	inclination	0
	F/#	1.4
Camera	working distance (mm)	80
	focus	138[16]
	zoom	60 %
	inclination	22°
	iris	800

Table 3.5: Final camera and projector settings.

According to equation 3.3 the new projector working distance means that the projection circle should have a diameter of $d' \simeq 106\,\text{mm}$ and line thickness plus line spacing should be $\frac{106}{11} \cdot 0.055 \simeq 0.53\,\text{mm}$. Looking at figure 3.12a, it can be seen that there are 75 lines on the length of a 4 cm channel. So the line thickness plus spacing is actually $\frac{40}{75} \simeq 0.533\,\text{mm}$ as expected. In addition, the camera is not recording the largest area of grid but is positioned closer to the part in order to have a better resolution of the lines and because the sample was still fully visible. The first tests will be held in this position. In this case, one line is 6 pixels wide instead of 3. The line spacing is 10 pixels. So, 16 pixels is 0.533 mm which means that one pixel is 33 µm wide instead of 56 µm, and $H \simeq 43\,\text{mm}$ instead of 71.7 mm. According to equation D.8, the maximum acceptable angle is 37°, so there is actually no more reason to worry about the DoF. When those first tests show positiv results, the camera distance will be increased to get to the maximum distance for which the whole image is covered in the grid projection. At this point it will be necessary to pay attention to the angle of the camera. In order to minimize potential errors, the angle limit of 24° will be respected even for the first tests.

[16] on IC Capture.

	Ideal case	Real case
$t_p + sp_p$ (mm)	0.447	0.53
t_p (pixels)	3	6
p (μm)	56	33
H (mm)	71.7	43
W (mm)	53.9	32.3
M	0.067	0.114
α	24°	22°

Table 3.6: Actual grid and image properties opposed to the ideal values.

Chapter 4

Software development

4.1 Image preprocessing

Before finding the defects it is necessary to process the images to obtain a regular and straight grid. For this purpose the following corrections are applied:

- High Dynamic Range (HDR) correction (section 4.1.1),
- Camera lens bias correction (section 4.1.2),
- Perspective correction (section 4.1.3).

Thereafter, every pixel is defined by its location (x, y) and gray level value coded in 8 bits. For a picture with a $H \cdot W$ resolution, the gray value is defined by the following function g.

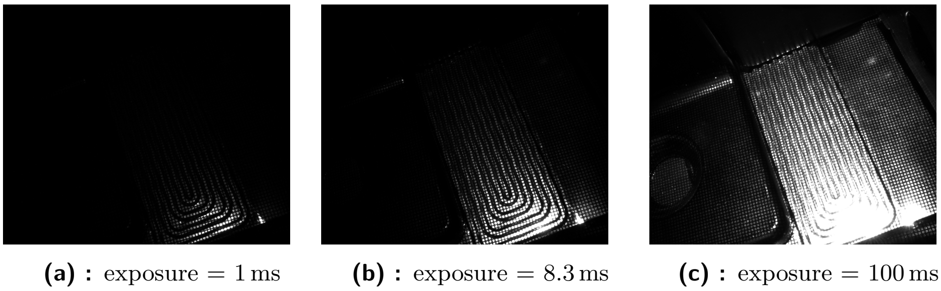
$$g : \left| \begin{array}{rcl} [\![0..H-1]\!] \cdot [\![0..W-1]\!] & \longrightarrow & [\![0..255]\!] \\ (x, y) & \longmapsto & g((x, y)) \end{array} \right. \tag{4.1}$$

4.1.1 HDR correction

This correction is used to eliminate overexposed and underexposed areas on the pictures. Three different OpenCV functions[1] (code in appendix H) were applied on a set of pictures taken with exposures of 1 ms, 8.3 ms and 100 ms:

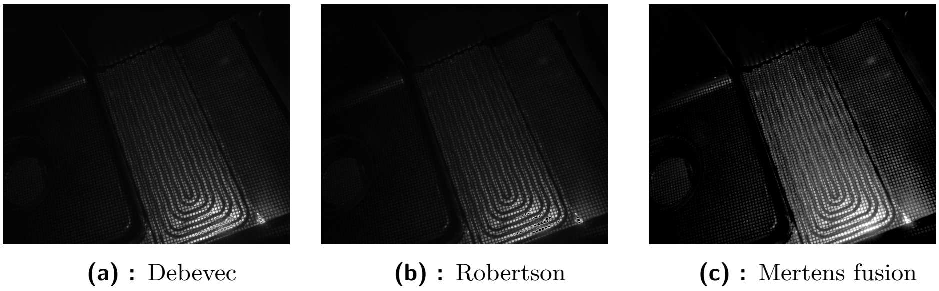
[1] `https://docs.opencv.org/4.x/d2/df0/tutorial_py_hdr.html` functions have been testest (05.06.2022)

Debevec [DM08], Robertson [RBS99] and Mertens fusion [MKVR07]. All three algorithms calculate a weighted average of the pictures considering the exposure values and the gray value of each point. The set is shown in figure 4.1 and the results after combining the three pictures with the three different OpenCV algorithms are shown in figure 4.2.



(a) : exposure = 1 ms **(b) :** exposure = 8.3 ms **(c) :** exposure = 100 ms

Figure 4.1: Pictures sequence of the same BPP test sample.



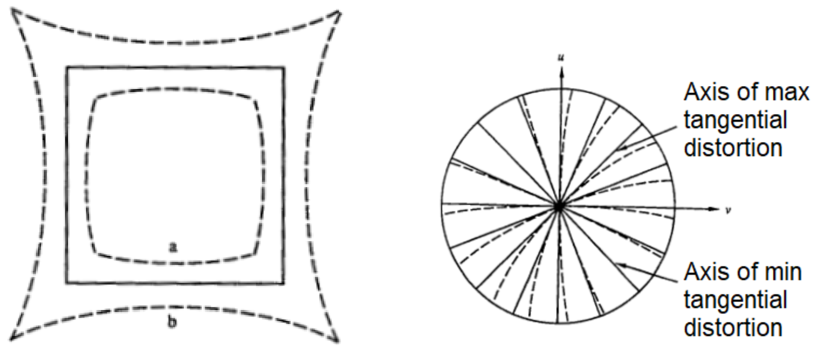
(a) : Debevec **(b) :** Robertson **(c) :** Mertens fusion

Figure 4.2: Results after using three different HDR algorithms.

After applying the algorithms, the darker areas are brighter and the brighter areas are darker. Overall, the grid pattern is more distinguishable using HDR. Scientific papers [PC18] compared the three algorithms and came to the conclusion that the Mertens algorithm works better than the Debevec and Robertson algorithm. In this case, the test has been repeated 12 times with different BPP test samples and position of the samples. The resulting pictures from Debevec and Robertson algorithm showed dark areas in the middle of bright square, where it should not be. The Mertens fusion did not show these dark areas. According to this and to the previous research, all following HDR improved pictures will be created by Mertens fusion (detailed working principle in appendix D.5).

4.1.2 Camera lens bias correction

Due to the curve of the lens all cameras presents distortions that are mostly radial and tangential [WCH^{+}92]. The radial distortion makes the lines look curved the further away they are from the center like on the left of figure 4.3. The tangential distortion displaces the points along a circle centered in middle of the image like on the right of figure 4.3.


a
b
u
v
Axis of max tangential distortion
Axis of min tangential distortion

(a) : Barrel distortion (a), pincushion distortion (b)

(b) : Tangential distortion

Figure 4.3: Sketches explaining the effect of the different distortions. [WCH+92]

Several distortion models exist in order to undistort an image [TVGMM17]. In this work, the OpenCV function is used[2]. Its working principle is explained in [cal] and rewritten in appendix D.6.

Figure 4.4 shows an example of a picture before and after correction. As the camera is very close to the object (100 mm), the distortion is not very large. It is still recognizable by tracing the straight lines on the grid lines. On the uncorrected image, the grid diverges from the straight line by 3 to 4 pixels. It seems to be negligible, but depending on the algorithm chosen, the shift of a few pixels can distort the results. Afterwards, this correction will always be applied.

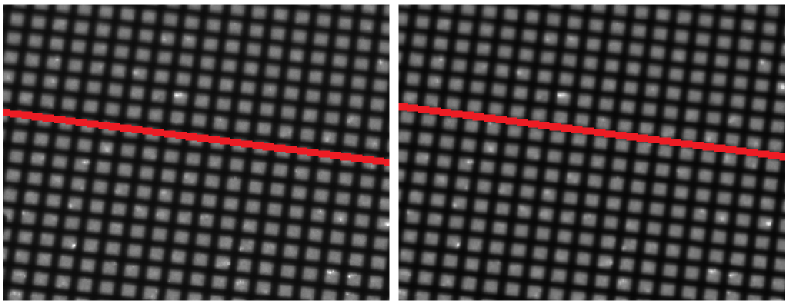


Figure 4.4: Images of the grid before 4.4a and after 4.4b the distortion correction. The red lines are straight and at the same location for both pictures.

[2] `https://opencv24-python-tutorials.readthedocs.io/en/stable/py_tutorials/py_calib3d/py_calibration/py_calibration.html`

4.1.3 Perspective correction

In order to have a straight grid pattern on the picture, the perspective has to be corrected. Therefore, a transformation matrix M_{pers} is applied on the pixels coordinates (x, y) to determine the new coordinates $(x_{correct}, y_{correct})$. equation 4.2 [OBM15] shows the tranformation to apply, where z is a scaling factor by which the result must be divided to obtain the final result.

$$\begin{bmatrix} zx_{correct} \\ zy_{correct} \\ z \end{bmatrix} = \overbrace{\begin{bmatrix} a_1 1 & a_1 2 & a_1 3 \\ a_2 1 & a_2 2 & a_2 3 \\ a_3 1 & a_3 2 & a_3 3 \end{bmatrix}}^{M_{pers}} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{4.2}$$

To find the eight unknown values of M_{pers}, the coordinates of four different pixels on the source picture and their corresponding coordinates in the final picture must be given. In this way, a system of eight equations is obtained that returns the unknown values of M_{pers} (more explanation in [JJ05] and [OBM15]). The points were chosen by taking the four corners of a rectangle formed by the projector, respectively reading their coordinates and changing them so that they form a right rectangle in the output image. The start and end vectors must be defined again every time the camera changes its position. The resulting transformation is depicted in figure 4.5 with a before after comparison.

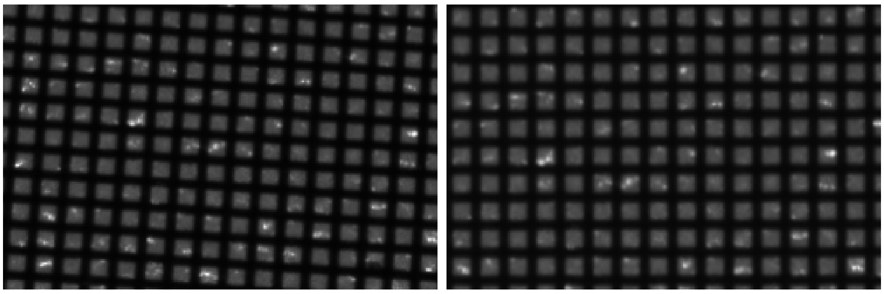


Figure 4.5: Images of the table before 4.5a and after 4.5b perspective correction.

4.2 Main algorithms

Four different algorithms are tested in order to find the defects. They have been chosen by using the state of the art information in paragraph 2.2.1. The first one is based on the computation of several statistical properties. The second one is founded on the calculation of the Gray-Level Co-occurence

Matrix (GLCM). The third algorithm is established on the Fast Fourier transform (FFT). Finally, the fourth algorithm uses both GLCM and FFT.

4.2.1 Statistical comparison

For the statistical comparison, the image and the reference image are compared block-wise. A block is defined by its dimensions $(h \cdot w)$ and its top left pixel coordinates (x, y). The set of pixels in a block will be called $B = [(x + i, y + j)$ for $0 \leq i < h$ and $0 \leq j < w]$. The block frequency is defined by the overlap. When the overlap equals 1, the blocks are following each other without overlapping; the top left pixel coordinates of the next block will be $(x + h, y + w)$. When the overlap value is bigger than 1, the blocks are overlapping and the top left pixel coordinates of the next block will be $(x + \frac{h}{\text{overlap}}, y + \frac{w}{\text{overlap}})$.

From the pixels of one block the mean, the variance, the median, the standard deviation, the skewness and the kurtosis are calculated. The definitions of these function can be found in appendix D.7. In order to simplify the code, the functions from the Numpy library have been used: "mean", "var", "median", "std", "skew" and "kurto". For every property, the absolute value of the difference between the reference value and the actual value is calculated and put into a new matrix; the block-size being $(\frac{h}{\text{overlap}}, \frac{w}{\text{overlap}})$ and its top left pixel's location being the same coordinates as the top left pixel of the analyzed block.

After repeating this for the block-sizes $(16 \cdot 16)$, $(32 \cdot 32)$, $(64 \cdot 64)$ and $(128 \cdot 128)$ (multiple of the grid pattern projection size), only the properties with the most visible defect results were kept for the final program. For each remaining block-size and property, thresholds have been determined experimentally by trial and error. Finally, these thresholds are used on every obtained feature matrix as in equation 4.6, summed together and divided by the number of used matrices.

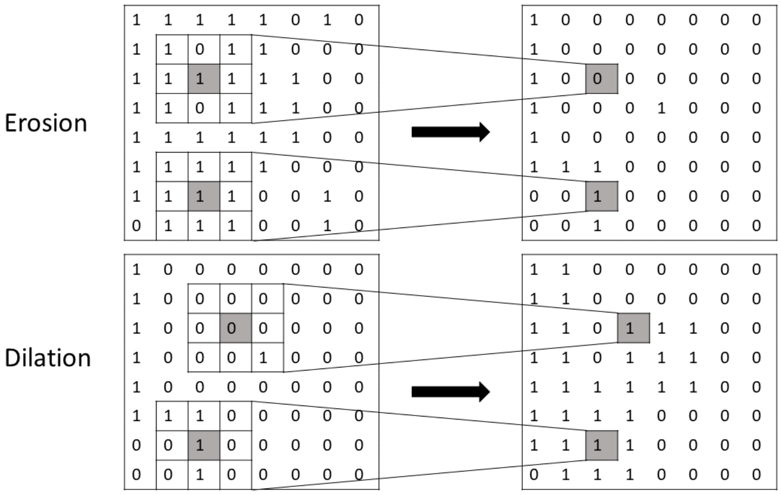
The other algorithms provide better results when a morphological transformation (opening) and a binarization (adaptive threshold) is applied in advance.

4.2.2 Opening and Adaptive threshold

The opening allows the elimination of noise and emphasizes the errors (it also causes loss of some details) and the adaptive threshold allows to obtain binarized images. It enhances the grid and ensures that it is used for defect detection. Without opening the image before the binarization, the defects are less distinguishable. Example of resulting images with and without opening can be found in appendix I.1.

Opening. The opening is a series of morphological transformation; first an erosion and then a dilation is applied, using the same kernel. When applied on a gray-scale image, the opening is working as followed [ope]:

During the erosion, the value of every pixel (x, y) becomes the minimum of the set $K_{(x,y)}$ of the surrounding pixels. This set is defined by the kernel. In this case the kernel is a square of side $k_{size} \in 2\mathbb{N}+1$ with (x, y) being the center point. So $K_{(x,y)}$ can be described by equation 4.3. An example is illustrated in figure 4.6.



1	1	1	1	1	0	1	0
1	1	0	1	1	0	0	0
1	1	1	1	1	1	0	0
1	1	0	1	1	1	0	0
1	1	1	1	1	1	0	0
1	1	1	1	1	0	0	0
1	1	1	1	0	0	1	0
0	1	1	1	0	0	1	0

1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	0	0	0	0	0

1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	0	0	0	0	0

1	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0
1	1	0	1	1	1	0	0
1	1	0	1	1	1	0	0
1	1	1	1	1	1	0	0
1	1	1	1	0	0	0	0
1	1	1	1	0	0	0	0
0	1	1	1	0	0	0	0

Figure 4.6: Illustration of the opening operation with a kernel size of 3.

$$\forall(x,y) \in [\![0..H-1]\!]\cdot[\![0..W-1]\!], K_{(x,y)} = [(x+i, y+j), \forall i,j \in [\![-\frac{k_{size}-1}{2}..\frac{k_{size}-1}{2}]\!]^2] \tag{4.3}$$

The erosion operation is resumed in equation 4.4.

$$\forall(x,y) \in [\![0..H-1]\!]\cdot[\![0..W-1]\!], g((x,y)) = \min_{\forall p \in K_{(x,y)}} g(p) \tag{4.4}$$

The dilation is the same but instead of taking the minimum it takes the maximum value as in equation 4.5.

$$\forall(x,y) \in [\![0..H-1]\!]\cdot[\![0..W-1]\!], g((x,y)) = \max_{\forall p \in K_{(x,y)}} g(p) \tag{4.5}$$

After executing the code with kernel sizes from 1 to 18, it was found that a *kSize* of 10 gave the results with the most distinguishable defects.

Adaptive threshold. This enables to transform a gray-scale in a binary picture by choosing a value κ for which all pixels with a gray value above this κ are granted the value of 1 and all pixels with a gray value underneath κ are granted the value of 0. The threshold function ϕ is defined in equation 4.6.

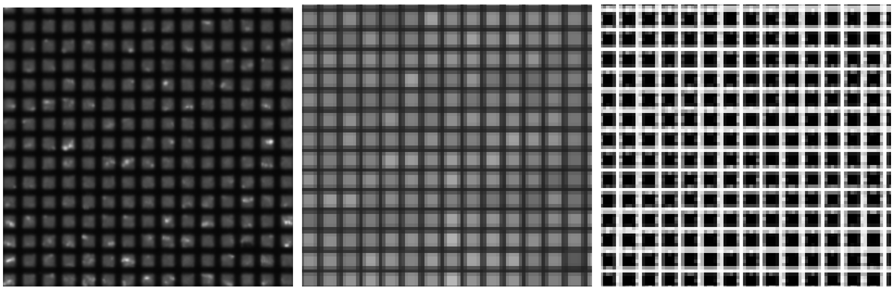
$$\forall(x,y) \in [\![0..H-1]\!] \cdot [\![0..W-1]\!], \phi((x,y)) = \begin{cases} 1 & \text{for} \quad g((x,y)) > \kappa \\ 0 & \text{for} \quad g((x,y)) \leq \kappa \end{cases} \tag{4.6}$$

As seen previously, some areas are overexposed and even when using HDR, these areas can be brighter than others which can cause a loss of information after thresholding. This is why an adaptive threshold is used where the threshold κ is not fixed for the whole picture but varies depending on the location. In this case, the inverted mean adaptive threshold is used, where the threshold κ, used for one pixel (x,y), is the mean value of the pixel's neighborhood, defined by the block-size k_{block} minus a constant value C. So $\kappa(x,y)$ can be written like this:

$$\kappa(x,y) = \frac{\sum_{i=-\frac{k_{block}-1}{2}}^{\frac{k_{block}-1}{2}} \sum_{j=-\frac{k_{block}-1}{2}}^{\frac{k_{block}-1}{2}} f((x+i,y+j))}{k_{block}^2} - C \tag{4.7}$$

$$\forall(x,y) \in [\![0..H-1]\!] \cdot [\![0..W-1]\!], \phi((x,y)) = \begin{cases} 1 & \text{for} \quad g((x,y)) \leq \kappa(x,y) \\ 0 & \text{for} \quad g((x,y)) > \kappa(x,y) \end{cases} \tag{4.8}$$

The block-size k_{block}, the constant value C as well as the kernel size k_{size} were chosen by running the final program several times and selecting the best combination ($k_{block} = 11$, $C = 2$ and $k_{size} = 10$). The different parts of the program are interdependent, it is therefore necessary to repeat the search for the best values for each modification made in the other parts. Figure 4.7 shows the picture of the table with grid projection after the different transofrmations: distortion and perspective correction, opening and adaptive threshold.



(a) : After perspective correction

(b) : After opening

(c) : After binarization

Figure 4.7: Picture of the table after lens and perspective correction, opening and thresholding.

4.2.3 GLCM comparison

For the GLCM comparison, the image and the reference image are compared block-wise as explained in section 4.2.1. From each block is calculated the Gray-Level Co-occurence Matrix. This matrix enables to recognize patterns by counting the amount of times a pair of gray values is positioned at a given angle (more explanation in [SVUB12]). A theoretical example follows in order to illustrate operating of the algorithm. Matrix A is the gray-level matrix block to be analyzed.

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.9}$$

For a given angle and a given distance, the GLCM is a two dimensional matrix. The number of columns and rows is defined by the number of different gray values. The number of rows always equals the number of columns. In the case of matrix A, the set of gray values is $\{0, 1\}$, so the table has two rows and two columns. In each box ($i = \text{row}, j = \text{column}$) is the number of times that a pixel of gray value i is at the given angle (zero being the zero of the trigonometric circle) and the given distance of a pixel of gray value j. When the distance between the two pixels of a pair is one, then the GLCM for the angles $\{0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3}{4}\pi\}$ is formed from the data of the tables 4.1 to 4.4.

	0	1
0	3	0
1	3	6

Table 4.1: Angle $= 0$

	0	1
0	1	2
1	2	4

Table 4.2: Angle $= \frac{\pi}{4}$

	0	1
0	3	3
1	0	6

Table 4.3: Angle $= \frac{\pi}{2}$

	0	1
0	0	5
1	0	4

Table 4.4: Angle $= \frac{3}{4}\pi$

It is obvious that the calculation of this matrix increases exponentially with the number of different gray values. That is why the images are first binarized by adaptive threshold before the GLCM is computed. Several features are computed from these four matrices 4.10 [HSL19]: cluster tendency, contrast, correlation, dissimilarity, entropy, homogeneity, inverse difference moment, maximum probability, and uniformity of energy. The definitions of every function can be found in appendix D.8.

$$G_1 = \begin{bmatrix} 3 & 0 \\ 3 & 6 \end{bmatrix}, G_2 = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}, G_3 = \begin{bmatrix} 3 & 3 \\ 0 & 6 \end{bmatrix}, G_4 = \begin{bmatrix} 0 & 5 \\ 0 & 4 \end{bmatrix} \tag{4.10}$$

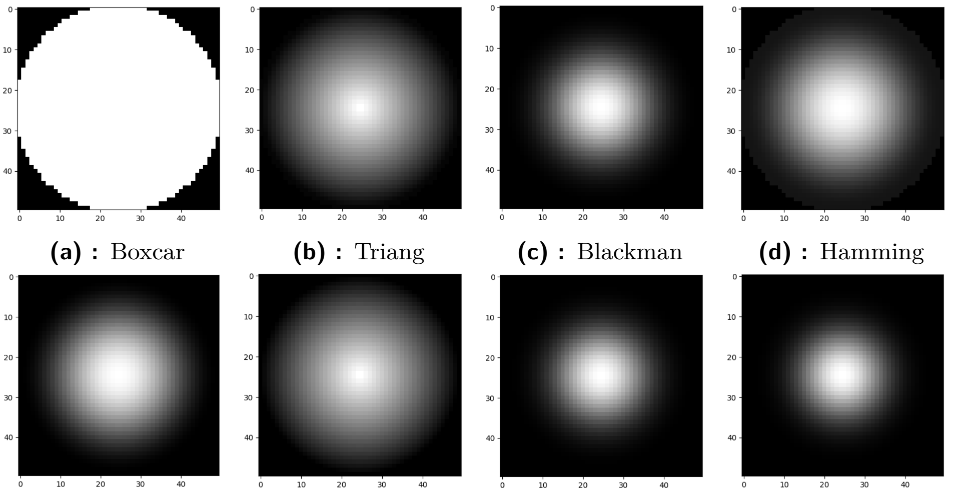
For each block four vectors of attributes are obtained (one for every angle) that vary according to the image pattern. For each feature, the mean value from the results of the angles is stored in a final feature vector. In a first step, the vector is compared to an ideal vector in order to find the defects. In a second step, the variation of the attributes can also be used to classify these defects. But at first, the focus will be on the defect recognition. A new matrix is created as explained at the end of section 4.2.1.

4.2.4 FFT comparison

For the FFT comparison, the image and the reference image are compared block-wise as explained in section 4.2.1. On each block a filter, a so-called passing window, is applied in order not to distort the Fourier transform. When applying the discret Fourier transform calculations on a sampled interval, which is the case for every block, it causes spectral leakage [RH07]. New frequency components appear without existing on the original sample. Applying a window can reduces or accentuate the spectral leakage. The windows are characterized by their dynamic range and resolution. The first is the ability to identify signals with highly different frequencies and amplitudes and the second is the ability to detect signals with similar frequencies and amplitudes [Har78a]. For this, different windows are tested:

- low dynamic range, high resolution: ’boxcar’
- moderate dynamic range and resolution: ’triang’,’hamming’,’hann’,’bartlett’
- high dynamic range: ’blackman’,’parzen’,’blackmanharris’

The 2D representation of these passing windows is shown in figure 4.8. The exact formulas for each window can be found in appendix D.9.



(a) : Boxcar **(b) :** Triang **(c) :** Blackman **(d) :** Hamming

(e) : Hann **(f) :** Bartlett **(g) :** Parzen **(h) :** Blackmanharris

Figure 4.8: 2D plot of the passing windows, size = 50 x 50.

After being filtered, the Fast Fourier transform of the block is calculated for each window. The FFT of the block from the actual image is compared to the FFT of the block from the reference image by computing the mean value of the absolute difference. A new matrix is again created as explained at the end of section 4.2.1.

The functioning of the statistical, GLCM and FFT comparison algorithms are summarized in the diagramm 4.9

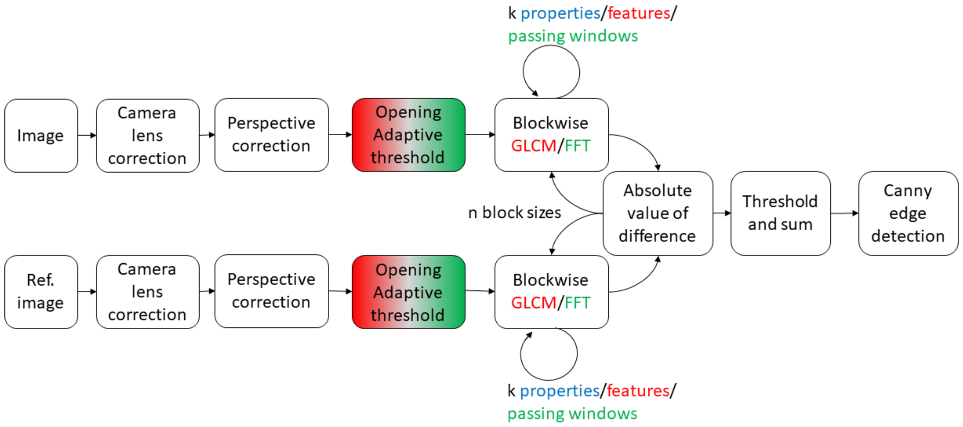


Figure 4.9: Scheme of the statistical (blue), the GLCM (red) and the FFT (green) comparison algorithms.

4.2.5 GLCM comparison with generated template

The GLCM of the image is computed as in the method from section 4.2.3 but instead of comparing the obtained feature matrix to the feature matrix of a reference image, it is compared to the feature matrix of a generated template. To create this template, only the tested image is used. The program computes the Fourier transform of the image to inspect in order to get the frequency of the grid pattern (Here, the frequencies are not expressed in time but in pixels.). For this, the absolute value of the Fourier transform is calculated. To illustrate this, the graph in figure 4.10 shows the result for the horizontal middle line of picture 4.7.

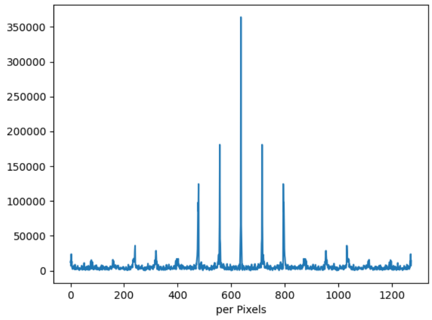


Figure 4.10: Diagram of the absolute value of the FFT of the horizontal middle line of the image 4.7.

The highest peak corresponds to the average value. The other peaks are harmonics. The frequency of the pattern is obtained by taking the difference between the frequency of the highest peak and the second highest peak. The width in pixels of the pattern is then equal to the width of the image divided by the frequency obtained. By assuming that the proportions of the projection are approximately the same as the proportions of the grid plate pattern (line width 20/55 times the hole width) and repeating this procedure observing the vertical middle line, the width and length of a dark square as well as the width of a horizontal and vertical line in pixels can be estimated.

This projection pattern is reconstructed and represented by a array with ones (for white) and zeros (for black). A block is formed which will be compared by GLCM to all blocks of the inspected image (in this case without using overlap). Equation 4.11 demonstrates an example of the template block when the results of the FFT analysis give a square side of 6, a vertical stripe thickness of 4 and a horizontal strip thickness of 3.

$$\text{Template} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{4.11}$$

The operating of this algorithm is summarized in the diagram 4.11.

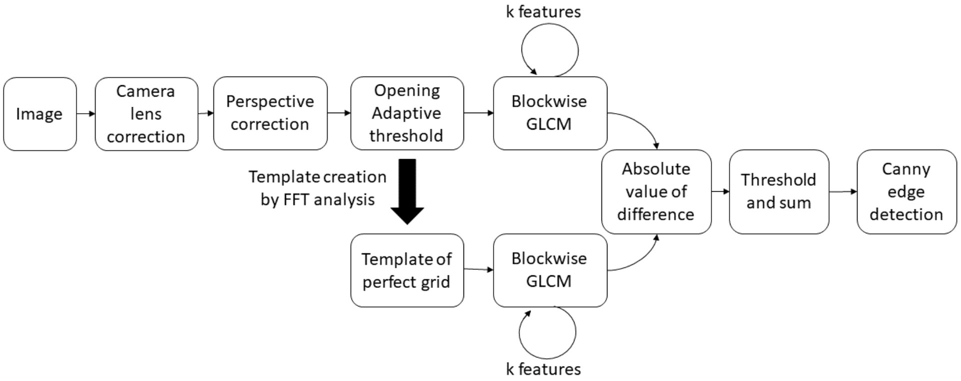


Figure 4.11: Scheme of the GLCM comparison with generated template algorithm.

This algorithm has some major drawbacks. The calculation of the width of a line or a square does not give an integer number. To create the template, an integer is needed because a pixel cannot be divided. The results are therefore rounded to the nearest integer. The starting point of the pattern is also a problem. The chance that the first pixel of the image (top left pixel) is the upper left corner of a dark square is of 1 % when the pattern is bigger than 10 x 10 pixels. To find the starting point, all possible starting points (number of points in the template) must be tested. That can be very time consuming. So, the real grid is different from the template which can cause a mismatch between the image and the template and distort the results. On the other hand, this template corresponds to a perfect grid. When comparing it with the image, any point not belonging to this grid plane will be recognized as an error. Only flat surfaces can be inspected with this algorithm. To control machined or deformed parts, it will be necessary to implement a program that can predict the geometry of the 2D grid projection using a 3D representation of the desired final part. This program has not been coded but it could be interesting to look at this problem in the future. A first attempt of it is exposed in appendix H.

Nonetheless, the code using the simple perfect grid template has been tested with the other programs in chapter 5.

Defects highlighting. At the end of every comparing algorithm (statistical, GLCM, FFT), the defects are highlighted by using the Canny edge detection on the final matrix. First, the resulting matrix is recorrected to the original perspective. To do so, the same function is used as in 4.1.3 except that the start and end points are switched. That way, the highlighting of the defects will be positioned on the original image.

Then, a simple threshold ϕ_{128} is applied. When a pixel's gray value is above 128, it means that it was recognized as a defect on more than half of the feature matrices on which it has been tested. It is therefore considered being a real defect. Finally the Canny edge detection is used to get the edges and plot them on the original, uncorrected picture [can].

The entire program for all four algorithms can be found in Appendix H. The main used prefabricated functions are listed in tableH.1 with their corresponding Python libraries.

Chapter 5

Experiments and results

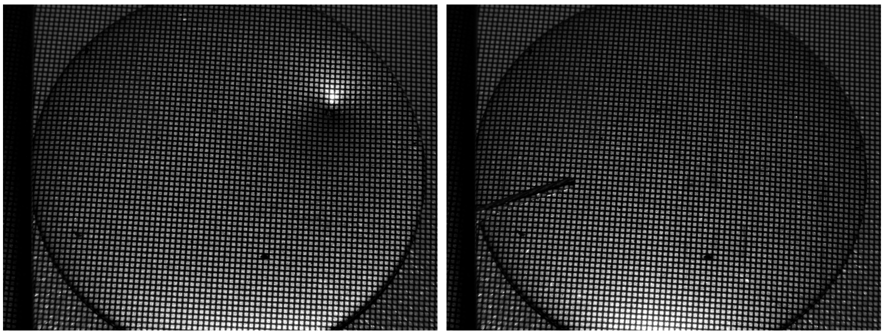
5.1 First experiment

5.1.1 Finding the threshold values

The first idea was to find a sample without defects and to take the image with the grid projection to have the reference image. Then, finding samples of BPP with the same design but with recognizable defects and to take the image with the grid projection at exactly the same position.

The comparison matrices for every feature of every method for different test pictures have to be computed in order to obtain enough matrices to determine the threshold values. It turns out, that the defects on the samples were not recognizable with this set-up and all the samples presented a random curvature which made the comparison with the reference sample impossible without considering the entire curvature a defect[1], so simplified sample parts were developed to determine the thresholds and then transfer it to the BPP. Die-cut metallic discs with a diameter of 40 mm and a thickness of 0.85 mm were used like depicted in figure 5.1. For every disc and position, three images with different exposures have been taken (1 ms, 0.25 ms and 0.1 ms) to improve the image quality by using HDR as explained in section 3.4. For the statistical comparison, the purposeful modes were: the mean, the variance, the median and the standard deviation. For the GLCM comparison

[1] A deviation of the size of recorded pixel area (<0.1 mm in any case) is enough to be noticed on the image.



(a) : with seam

(b) : with cut

Figure 5.1: Pictures of two metallic disc samples with defects and grid projection.

the goal-oriented features were: the cluster tendency, the correlation, the entropy, the maximum probability and the uniformity of energy. For the FFT comparison the only passing windows giving useful results were: "boxcar", "triang", "hamming" and "bartlett".

All four algorithms were applied and the matrices for each remaining mode, feature or window (in general property) and each block-size were plotted. From each of these matrices, the threshold of the defects were identified and inserted in a table (appendix F.1). The thresholds are determined by averaging the thresholds obtained on the different disc images, in this case, five in total (Pictures in appendix I.4). Only the features/modes with recognizable thresholds were kept. The final values used in the program are shown in tables 5.1, 5.2, 5.3 and 5.4.

Block-size	Mean	Median
(16, 16)	46.25	63.3
(32, 32)	43.33	60
(64, 64)	38.7	46

Table 5.1: Thresholds for the statistical comparison algorithm.

Block-size	Cluster tendency	Entropy	Maximum Probability	Uniformity of Energy
(16, 16)	0.54	0.5	0.368	0.37
(32, 32)	0.338	0.272	0.2	0.158
(64, 64)	0.192	0.04	0.098	0.029
(128, 128)	0.098	0.0193	0.044	0.01

Table 5.2: Thresholds for the GLCM comparison algorithm.

Block-size	Boxcar	Triang	Hamming	Bartlett
(16, 16)	6.06	2.52	2.8	2.49
(32, 32)	9.4	3.92	4.12	3.9
(64, 64)	17.6	7.2	8	7
(128, 128)	30.8	12.8	13.6	12.8

Table 5.3: Thresholds for the FFT comparison algorithm.

Cluster tendency	Entropy	Maximum Probability	Uniformity of Energy
0.4	0.8	0.4	0.4

Table 5.4: Thresholds for the GLCM template comparison algorithm.

It can be noticed that in the table of values in appendix F.1, the results obtained for the FFT comparison are the most stable. The relative standard deviation of its data is under 30 %, whereas for the GLCM comparison the maximum relative standard deviation is 54 % and for the statistical comparison it is even 69 %.

5.1.2 First results

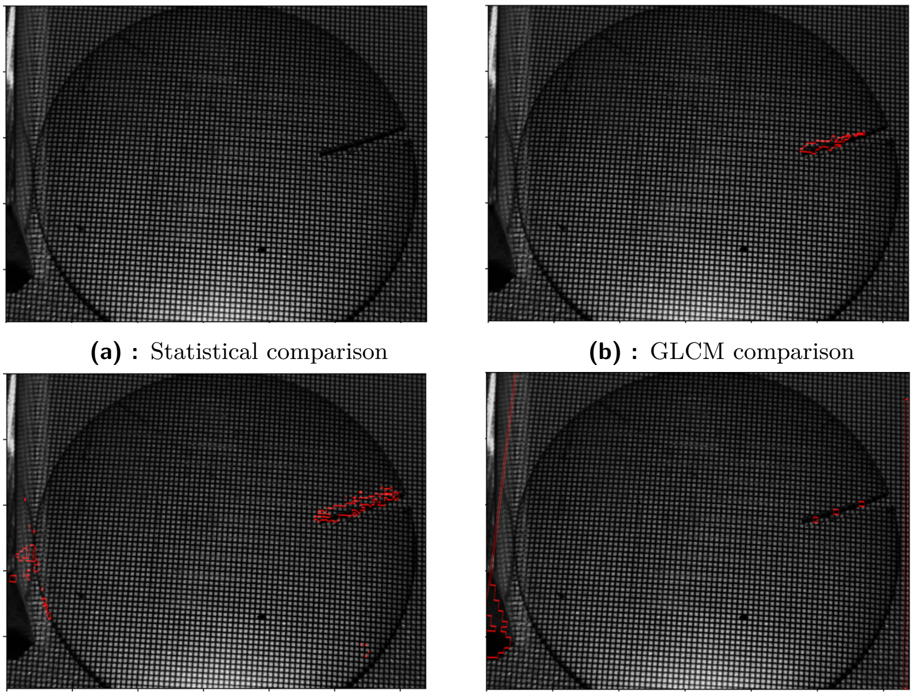

(a) : Statistical comparison
(b) : GLCM comparison
(c) : FFT comparison
(d) : GLCM template comparison

Figure 5.2: Results of the first test.

For the test, the five disc pictures mentioned in the previous section, were evaluated with the proposed methods to see the outcome. Figure 5.2 displays the result of one of the five tested pictures.

The results are positive for the both GLCM algorithms as they recognize all defects without false defects. The FFT program recognizes all five defects while presenting large area of false defect detections and the statistical method only recognizes one of the defects with no false defects (resulting pictures in appendix I.5).

To test the program on a larger scale 36 pictures of different discs in different positions have been taken in the same way as the discs from the previous section. After processing with the thresholds from section 5.1.1, the results from table 5.5 have been found.

	Number of defects	36	100 %
GLCM	Detected defects	20	56 %
	False detected defects	2	6 %
Statistical	Detected defects	18	50 %
	False detected defects	7	19 %
FFT	Detected defects	26	72 %
	False detected defects	36	100 %
GLCM with template	Detected defects	10	28 %
	False detected defects	0	0 %

Table 5.5: Results from the first test.

What can be noticed is that there are no false detections and a low amount of detected defects for the GLCM with template algorithm which can be related to a too severe threshold. On the other hand, the FFT algorithm marks too many surfaces which are not considered as errors. This could be caused by a too low threshold. Exemples for these problems are pictured in figure 5.3. A correction of the thresholds is presented in section 5.2.

The algorithm based on the statistical data of the image seems to find the errors, however it does not locate them correctly. The defects recognized by this algorithm are actually only the overexposed areas. The disadvantage is that only the defects creating these overexposures are recognized (mostly bumps) and the light intensity can vary as the system is not protected from daylight. This method is therefore not reliable and does not use the projected grid which should be the main key of the system. Instead, it compares the brightness of the the squares of the projection pattern which would also work using simple daylight or another light source. During the next tests, this method will no longer be taken into account.

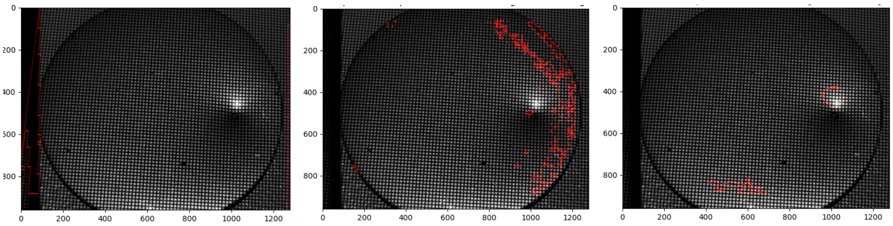
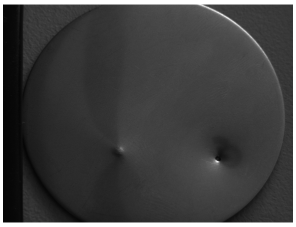


Figure 5.3: Pictures demonstrating the problematic results for the different algorithms. Left:GLCM with template result showing no defect. Middle: FFT result showing false defects. Right: Statistical method result showing defects on bright zones.

5.2 Tests with defined defects

To refine the value of the thresholds and to be able to recognize a pattern between the thresholds and the defects characterized by these thresholds (classification), the test is repeated with discs whose defects are measured and documented. In the descriptions hereunder, δ is the diameter, L is the length, e is the elevation/deepening (depending on the defect type) and l is the width of the defect.



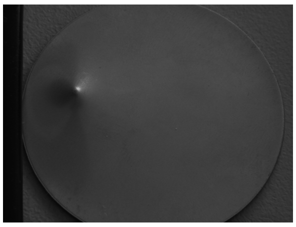
A deepening/hole:

- $\delta = 0.6\,\text{mm}$
- $\text{e} = 0.8\,\text{mm}$

An elevation:

- $\delta = 0.9\,\text{mm}$
- $\text{e} = 0.6\,\text{mm}$

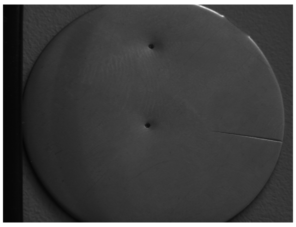
Figure 5.4: Disc 1



An elevation:

- $\delta = 0.8\,\text{mm}$
- $\text{e} = 0.8\,\text{mm}$

Figure 5.5: Disc 2



A hole:

- $\delta = 0.7\,\text{mm}$
- $\text{e} = 0.2\,\text{mm}$

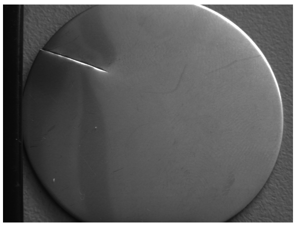
A more centered hole:

- $\delta = 0.7\,\text{mm}$
- $\text{e} = 0.1\,\text{mm}$

A scratch:

- $\text{L} = 11\,\text{mm}$
- $\text{e} = 0.1\,\text{mm}$
- $\text{l} = 0.2\,\text{mm}$

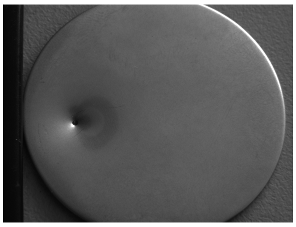
Figure 5.6: Disc 3



A slit:

- $\text{L} = 11\,\text{mm}$
- $\text{e} = 0.8\,\text{mm}$
- $\text{l} = 0$

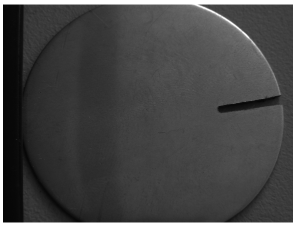
Figure 5.7: Disc 4



A hole:

- $\delta = 0.8\,\text{mm}$
- $\text{e} = 0.8\,\text{mm}$

Figure 5.8: Disc 5



A large hole:

- $\text{L} = 10.3\,\text{mm}$
- $\text{e} = 0.85\,\text{mm}$
- $\text{l} = 1.35\,\text{mm}$

Figure 5.9: Disc 6

The same procedure is performed on these discs as in section 5.1.1 to find the new thresholds. The table with all the readings can be found in appendix F.2. This time, only the blocksizes and the features for which no disc threshold was considered "bad"[2] are kept for the final threshold values. This leads to table 5.6. It is noticeable that a pattern for hole recognition with the GLCM method stands out during the data collection of appendix F.2 because these are the only defects that are recognizable on the entropy and uniformity of energy graphs.

Block-size	Cluster tendency	Maximum Probability
(16, 16)	0.47	0.18
(32, 32)	0.2	0.147

Table 5.6: Improved thresholds for the GLCM comparison algorithm.

Block-size	Boxcar	Triang	Hamming	Bartlett
(32, 32)	10.5	4.6	4.8	4.6
(64, 64)	18.2	7.7	8.2	7.5
(128, 128)	32	12.9	13.9	13

Table 5.7: Improved thresholds for the FFT comparison algorithm.

Cluster tendency	Entropy	Maximum Probability	Uniformity of Energy
0.52	0.8	0.43	0.49

Table 5.8: Improved thresholds for the GLCM template comparison algorithm.

Using the same 36 pictures as in section 5.1.2 by using the new thresholds, the results summarized in table 5.9 are obtained.

		Past thresholds		New thresholds	
	Number of defects	36			
GLCM	Detected defects	20	56 %	22	61 %
	False detected defects	2	6 %	5	14 %
FFT	Detected defects	18	50 %	20	56 %
	False detected defects	7	19 %	17	47 %
GLCM with template	Detected defects	10	28 %	25	69 %
	False detected defects	0	0 %	3	8 %

Table 5.9: Results for the test with refined thresholds.

[2] The threshold is considered "bad" when there is no measurable difference between the defective and the intact zone.

The GLCM with and without template are able to recognize more defects but they also have a higher false detection rate. Unlike the method without template, the method with template presents on every picture defects located on the border of the discs which is logical as it compares the grid of the image with a perfect grid i.e. a flat surface. As the disc is not flat there is an irregularity at the disc's borders which is recognized as defect. This is why these defects are not counted as false defects. The FFT algorithm detects less defects but it also detects less false defects. By increasing the threshold, the false detection rate is reduced but the number of detected real defects is too. Thus, the severity of the threshold does not seem to be the issue.

5.3 Theoretical minimum size of detected defects

To change the size of the projected grid, the defect part must have a height difference with the reference part that causes the grid pattern to be compressed or strained (at least one pixel larger or smaller). For this, equation 3.3needs to be reused (demonstration in appendix D.10). The difference of the projection diameter Δd (mm) for a difference of working distance Δu (mm) is the factor that will influence the detectability of a defect. It is defined by equation 5.1.

$$\Delta d = \Delta u \cdot \frac{d'}{f} \tag{5.1}$$

By taking the theoretical smallest size of the grid pattern for $d' = 0.02\,\text{mm}$ (line thickness), the goal is to find for which Δu Δd is bigger than theoretical observed pixel width of 56 µm calculated in section 3.2.2. This results in the final equation 5.2 to find the minimum recognizable height difference.

$$\begin{aligned} \Delta d &> 56 \\ \Leftrightarrow \Delta u \cdot \frac{d'}{f} &> 56 \\ \Leftrightarrow \Delta u &> \frac{56}{d'} \cdot f \end{aligned} \tag{5.2}$$

To remind, $f = 35\,\text{mm}$ so $\Delta u > 98\,\text{mm}$. Even under the conditions stated in section 3.5 with an observed pixel width of 33 µm, $\Delta u > 58\,\text{mm}$. This means that a defect must present a height difference of minimum 58 mm in order to create a noticeable compression or extension of the grid pattern. These measurements are above the expected defect measurements. So the grid pattern size will not change, but this difference in height could cause a shift on the camera's image in the pattern due to the perspective angle of the camera. Assuming that all incident light enters with the same angle α, as shown in figure 5.10.

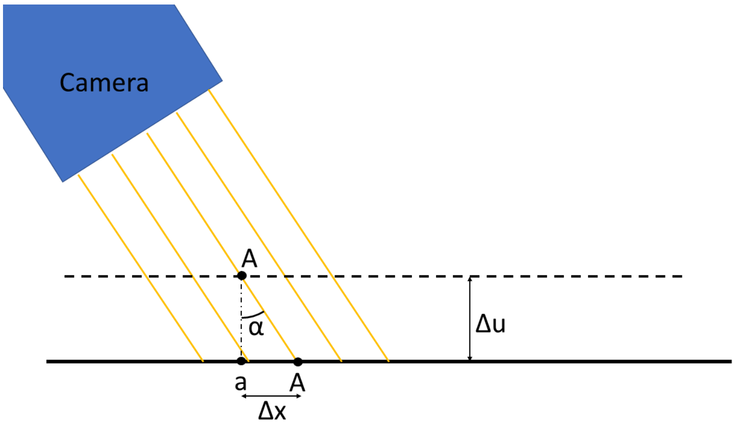


Figure 5.10: Scheme of the shifting problem if all rays were parallel.

When a point on the part is placed at Δu from its ideal height, on the camera it will look like it is shifted of Δx on the perspective corrected picture.

$$\Delta x = \Delta u \cdot \tan(\alpha) \tag{5.3}$$

Unfortunately, this is only true for the middle point of the camera $(\frac{W}{2}, \frac{H}{2})$. For all the other points $(\frac{W}{2} \pm x, \frac{H}{2} \pm y)$, with $(x, y) \in [\![1 \,..\, \frac{W}{2}]\!] \cdot [\![1 \,..\, \frac{H}{2}]\!]$, the incident light has a different angle which creates a different shift for every point as depicted in figure 5.11. This shift is split into two components, along the x axis and the y axis. The demonstration for the final equations 5.4, with

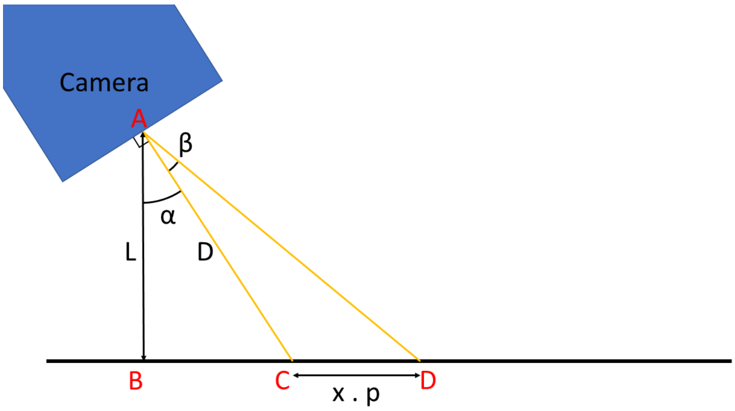


Figure 5.11: More accurate scheme of the shifting problem.

p the size of a pixel in mm and D the distance illustrated in figure 5.11 in mm, is in appendix D.11.

$$
\begin{aligned}
\Delta x &= \Delta u \frac{D\cos(\alpha+\beta_x)}{D\cos(\alpha+\beta_x)+\Delta u}\left(\frac{xp}{D\cos(\alpha)}+\tan(\alpha)\right) \\
\Delta y &= \Delta u \frac{yp\cos(\beta_y)}{\frac{D\cos(\alpha)}{\cos(\alpha+\beta_x)}\cos(\beta_y)+\Delta u}
\end{aligned}
\qquad (5.4)
$$

It is important to notice that in this case, x and y are pixel coordinates (in pixels) so the width recording by a pixel "p" (in mm) needs to be multiplied in order to have the coordinates in mm. Here, p is considered constant for the entire image even though it is not true because it is a function of the pixel coordinates$p \longrightarrow p(x,y)$. It is as if the picture's perspective had already been corrected. For more accurate results, (xp, yp) must be replaced with (X, Y) the point coordinates in mm.

Ideal case (from chapter 3: table 3.6). When computing these equations for different values of Δu and plotting the resulting shifted position of points, figure 5.12 is obtained. The camera is positioned on the left of the graph. What the camera perceives as the bottom is the left side of the graph. The Python code can be found in appendix H. As expected, the closer the points are to the camera, the smaller the shift.

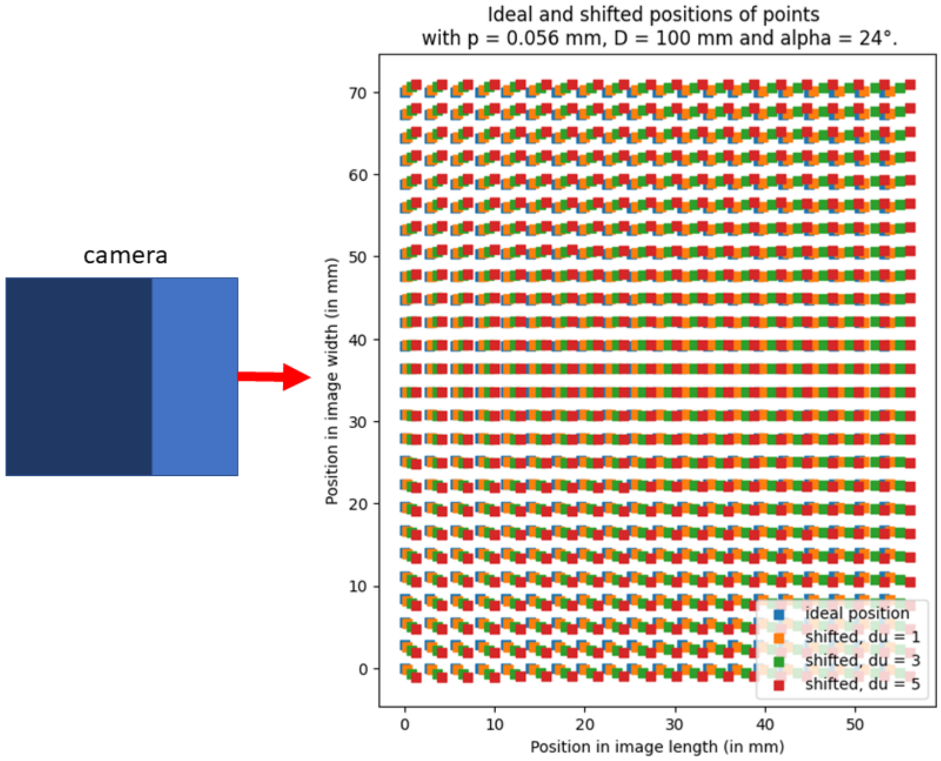


Figure 5.12: Graph of the ideal and shifted positions of points for different height differences (in mm), with $p = 0.056\,\mathrm{mm}$, $D = 100\,\mathrm{mm}$ and $\alpha = 24°$.

The deformation will be distinguishable only when the distances Δx or Δy are larger than the size of one pixel. Knowing the position of the camera, a map can be established to see what is the minimum height difference at each point of the picture to have a notable shift. The resulting graph is shown in figure 5.13. It has to be interpreted for the camera to be looking from the left rear side of the graph. The code can be found in appendix H.

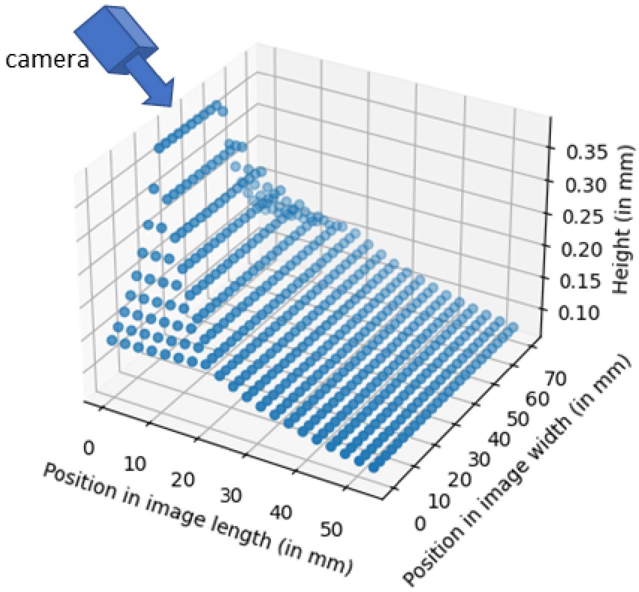


Figure 5.13: Graph of the minimum detectable height differences Δu for every position, with $p = 0.056\,\text{mm}$, $D = 100\,\text{mm}$, $\alpha = 24°$ and the minimum needed shift $= p$.

In congruence with the previous result, the closer the points are to the camera, the larger the minimum detectable height difference because it was seen in figure 5.12 that a larger Δu is needed to get the minimum shift.

If the shift matches the width of the pattern, then it is no longer observable. The offset on the x-axis Δx modulo the length of the pattern must therefore be greater than the length of a pixel for the point to be traceable. Similarly, the shift on the y-axis Δy modulo the width of the pattern must be greater than the width of a pixel.

$$\begin{aligned} \Delta x &\equiv p \pmod{sp} \\ \Delta y &\equiv p \pmod{sp} \end{aligned} \tag{5.5}$$

This last point can lead to detection failure, but in reality, in order for points of the component to reach this altitude, other points are needed to connect it to the original structure of the part. These points will then present a shift that will not respect the equation 5.5. These can be detected. In this case, it is the basis of the defect that are recognized as defects whereas the contours of the defect are not visible. This point is therefore not taken into account in the following. The smallest topological detectable deformation can be estimated from the resulting graph. In the best case scenario, $D = 100\,\mathrm{mm}$ (approximately minimum working distance of the camera[3]) and $p = 56\,\mu\mathrm{m}$ (calculated in section 3.2.2). According to the results, figuring in the graph 5.13, no matter the location of the defect, the topographic defect is noticeable if its height difference with the ideal geometry is larger than 0.4 mm.

This still does not mean that every defect of at least 0.4 mm height difference will be detected, because if the width and length of this defect is smaller than one square of the grid, it can happen that the entire defect is located on a bright square, its shift can be bigger than the size of one pixel but still small enough for the defect to stay in the bright square as illustrated on figure 5.14.

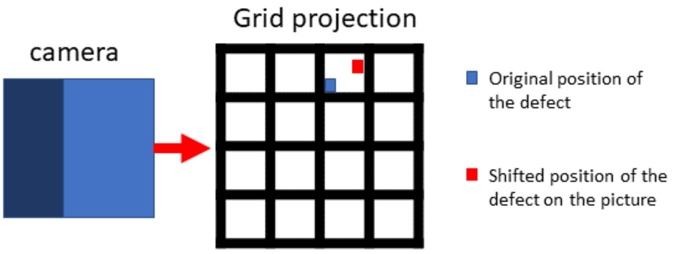

Grid projection
camera
Original position of the defect
Shifted position of the defect on the picture

Figure 5.14: Illustration of a shift that woul lead to no alteration in the projection's shape.

To solve this, either the defect must be large enough: Δu must be large enough to cause a shift that is wider than one square of the grid; or the area of the defect must be larger than the area of one square; or the part must move in relation to the grid. The last option is a constraint which is applicable on a conveyer belt and that will be mentioned again in the discussion section 6.1 of this work. In the following, the limits of the first constraint are computed.

In the ideal case, calculated in section 3.2.1, the dimensions of a bright square (sp) are $0.285 \cdot 0.285\,\mathrm{mm}^2$. So by repeating the last program but replacing the minimum shift value to reach at 0.285 mm, the resulting graph 5.15 shows a minimum height difference of 2 mm. For the topographic defect to be sure of being detected must either present a height difference with the original of at least 2 mm, or have a connected surface of at least $0.285 \cdot 0.285\,\mathrm{mm}^2$ with a height difference of at least 0.4 mm.

[3] D is not exactly the working distance, it is a little larger and and estimated at 100 mm instead of 80 mm

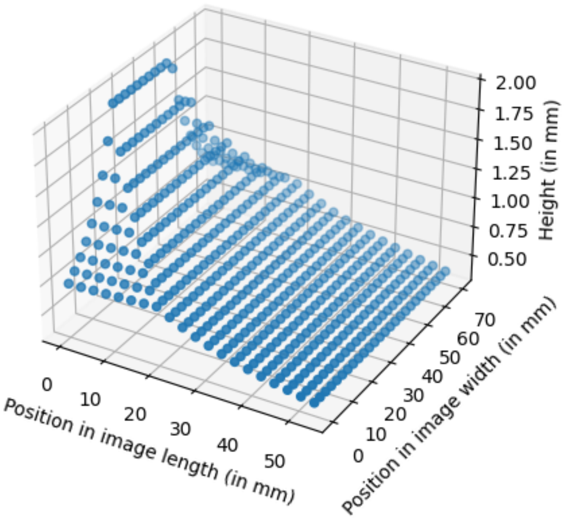
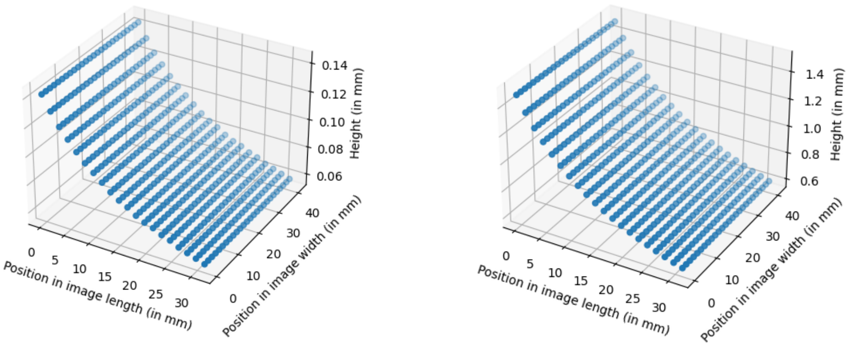


Figure 5.15: Graph of the minimum detectable height differences Δu for every position, with $p = 0.056\,\text{mm}$, $D = 100\,\text{mm}$, $\alpha = 24°$ and the minimum needed shift $= 0.285\,\text{mm}$.

Real case (from chapter 3: table 3.6). For this case the bright square (sp) dimensions are $0.34 \cdot 0.34\,\text{mm}^2$. According to figure 5.16 topographic defect must either present a height difference with the original of at least 1.6 mm, or have a connected surface of at least $0.34 \cdot 0.34\,\text{mm}^2$ with a height difference of at least 0.16 mm.



(a) : With minimum needed shift $= p$.

(b) : With minimum needed shift $= 0.34\,\text{mm}$.

Figure 5.16: Graph of the minimum detectable height differences Δu for every position, with $p = 0.033\,\text{mm}$, $D = 100\,\text{mm}$ and $\alpha = 22°$.

	(a)	(b)
Ideal case	0.4	2
Real case	0.16	1.6

Table 5.10: Results of the theoretical smallest detectable defect with this system. (a): considering the size of a pixel. (b): considering the size grid square.

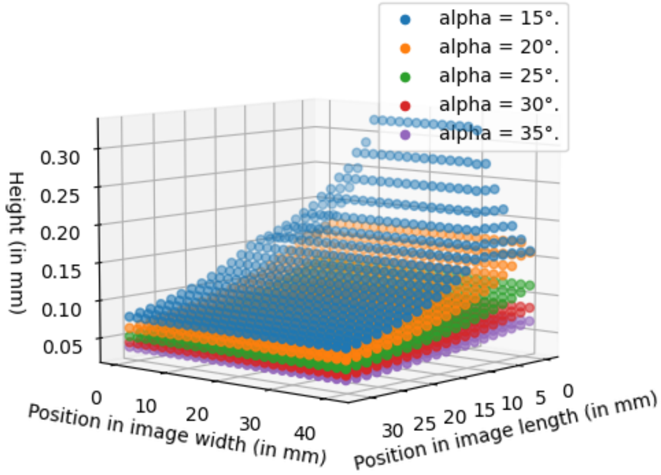


Figure 5.17: Graph of the minimum detectable height differences Δu for different values of α, with $p = 0.033$ mm and $D = 100$ mm.

The limit seems to be lower in the real case although the inclination of the camera is smaller. Graph 5.17 confirms that the higher the inclination angle α, the more accurate the detection. This is due to the fact that all points on the image are considered and the short working distance of $D = 100$ mm causes the bottom centered defects to be very difficult to detect.

Using a higher working distance would result in more flattened graphs around the middle point and lower the overall limit for the defect to be detected as depicted in graph 5.18.

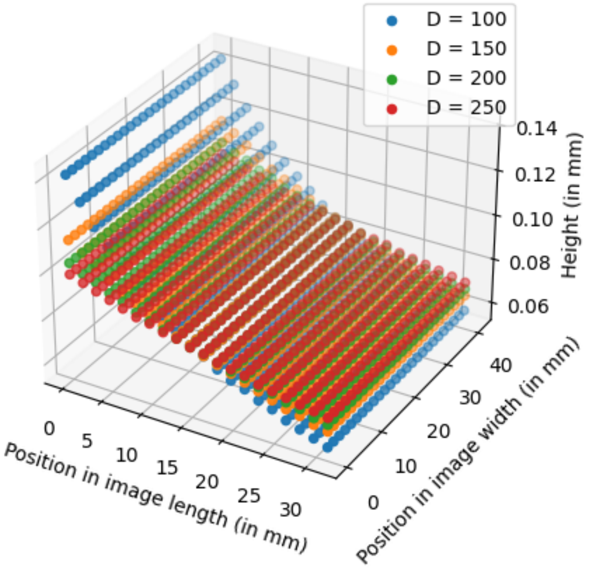


Figure 5.18: Graph of the minimum detectable height differences Δu for different values of D, with $p = 0.033$ mm and $\alpha = 22°$.

The camera resolution also influences the detectable defect size. Graph 5.19 demonstrates the theoretically minimum detectable height difference when observing a same surface with three different resolutions. The projected pixel sizes p correspond to the resolutions calculated in table 3.2 of the 1MP, 5MP and the ideal 151MP camera[4].

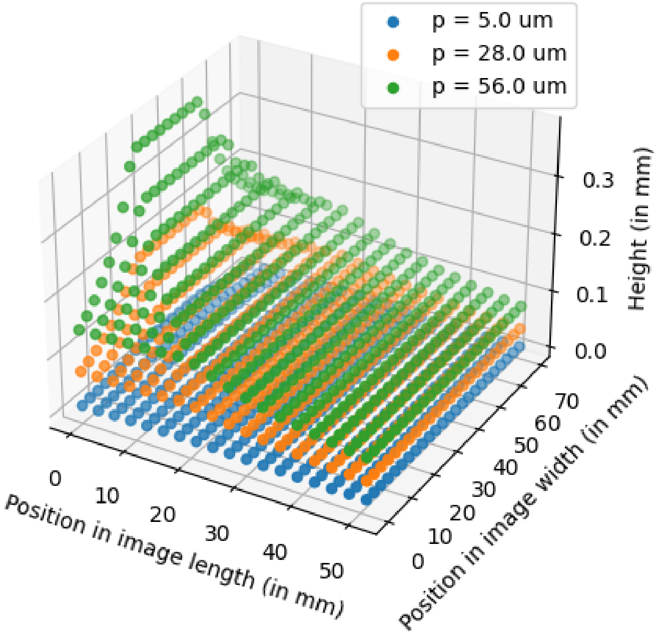


Figure 5.19: Minimum detectable height difference using different resolutions, with $D = 100\,\text{mm}$ and $\alpha = 24°$.

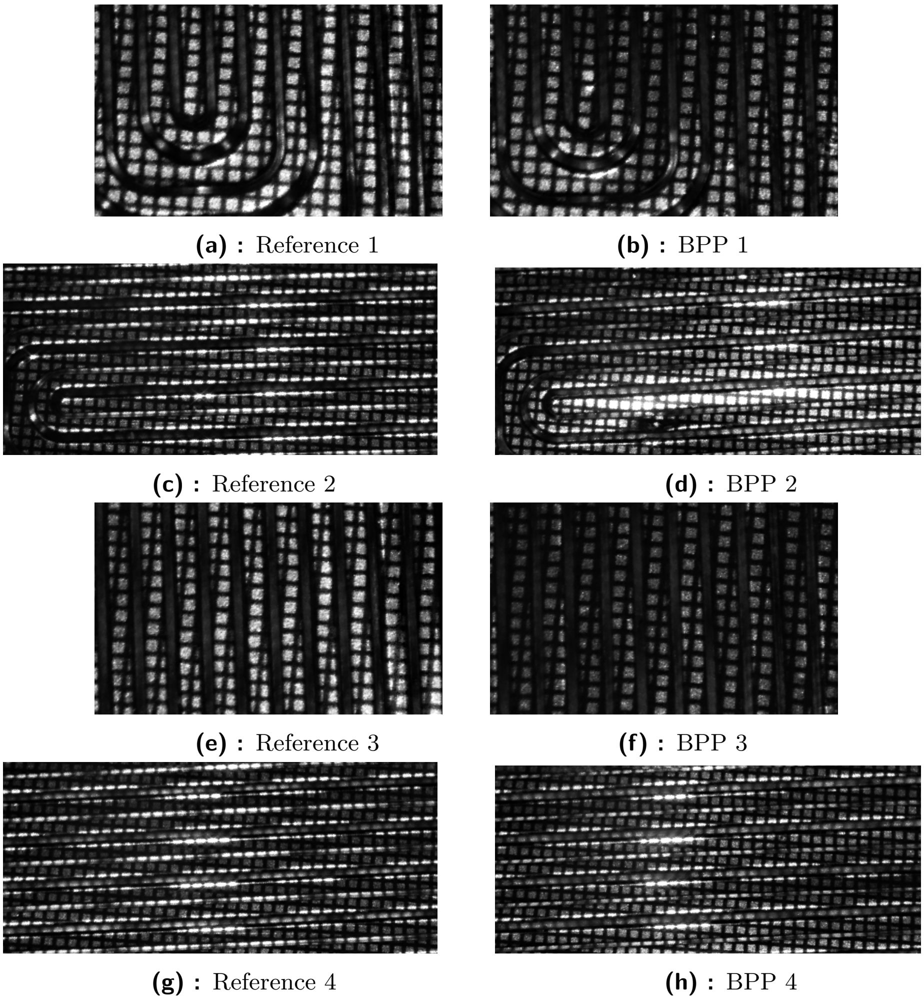
The corresponding maximum values for the 1MP, 5MP and 151MP camera graphs are 0.373 mm, 0.186 mm and 0.034 mm. To obtain a better accuracy it is possible to use a camera with a higher resolution.

5.4 Test with metallic bipolar plates

The program is applied on pictures of the BPP sample. With the theoretical part of the minimum vertical displacement to observe a deformation from section 5.3, it is possible to know at what point the plate must be flat so that its curvature does not influence the measurements. According to graph 5.13 the overall height variation of the plate compared to the reference plate must not exceed 0.05 mm. Figure 5.20 presents the pictures from four orientations of a reference BPP sample and a defect one after HDR correction[5].

[4] The observed area is the one that was computed for the ideal case in section 3.5.

[5] Like in the previous sections when HDR correction was applied, three pictures were used with exposures of 1 ms, 0.25 ms and 0.1 ms.



(a) : Reference 1 **(b) :** BPP 1

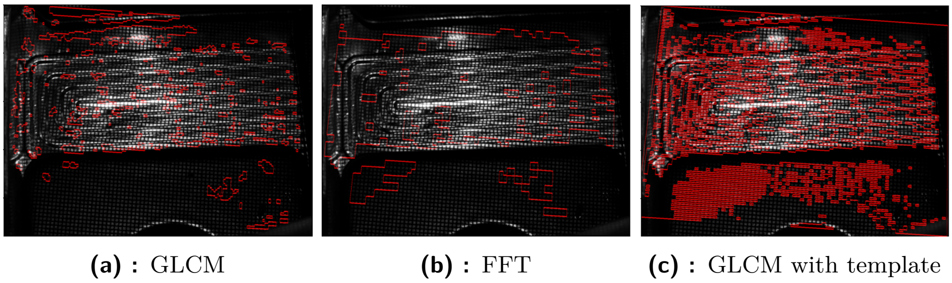
(c) : Reference 2 **(d) :** BPP 2

(e) : Reference 3 **(f) :** BPP 3

(g) : Reference 4 **(h) :** BPP 4

Figure 5.20: Images of reference and test BPP sample in four orientations after HDR correction.

A shift in the grid can be observed and this shift causes an error in the measurements. The results are correspondingly poor as displayed in figure 5.21. The placement of the plates causes problems, because the material is so thin (0.05 mm) that it randomly curves. If the samples are layed on the

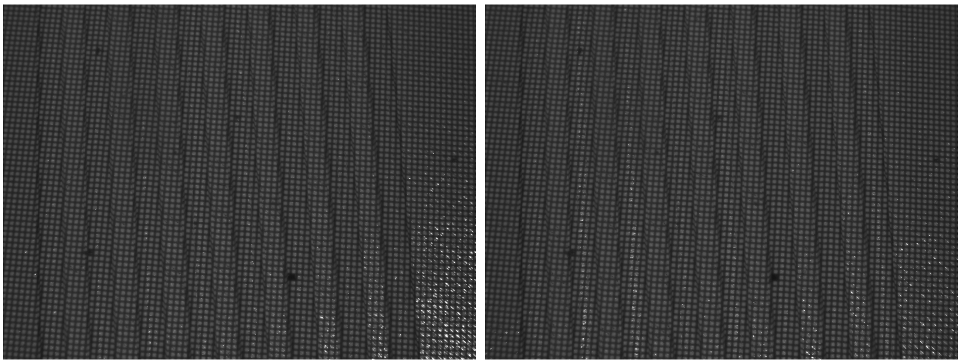


(a) : GLCM **(b) :** FFT **(c) :** GLCM with template

Figure 5.21: Results of all three algorithms with orientation number two.

table (like they would on a production convoyer belt), none of them looks like the other even though it is the same design. A structure to rectify this defect was designed with CAD on PTC-Creo, exported in STL file and 3D printed with an Ultimaker in tough PLA. The drawings of this structure can be found in appendix G. Unfortunately, it did not work; the plate was still randomly curved. Another idea would be to flatten the plate by putting equally distributed pressure on its surface by clamping it between two plates of a transparent material like glass or PVC. However, this would complicate the optical system because the rays of the projector would have to pass through the PVC plate and the angle of refraction caused by it would have to be taken into account.

Another issue is still the size of the defects that are very small (under 0.1 mm so even theoretically not distinguishable). Still, in order to be able to test the program on objects with a geometry close to the one of BPPs, plates drawn with CAD on PTC-Creo at the model of the metallic samples at the scale of 2:1 were printed in 3D with white PLA, because it is the color with the highest reflectance. Thus, several samples of 3D printed BPP with manually added defects of different sizes, all above the theoretical minimum, and no random curbation are used to make the next tests. The two orientations from figure 5.22 have been used.



(a) : 1

(b) : 2

Figure 5.22: Reference pictures for the 3D printed BPP.

A set of 13 times two pictures is obtained. This set is analyzed with the none optimized algorithms and the final thresholds from section 5.2[6] with the exception that no HDR correction is applied because even though the white PLA is reflective, its surface is rougher and therefore scatters the light more, so the overexposure preventing the distinction of the grid is negligible. The pictures from figure 5.22 confirm that there is very little overexposed zones.

[6] As in all the final used methods the images are transformed into binary pictures, the thresholds should be the same no matter what the surface structure of the part.

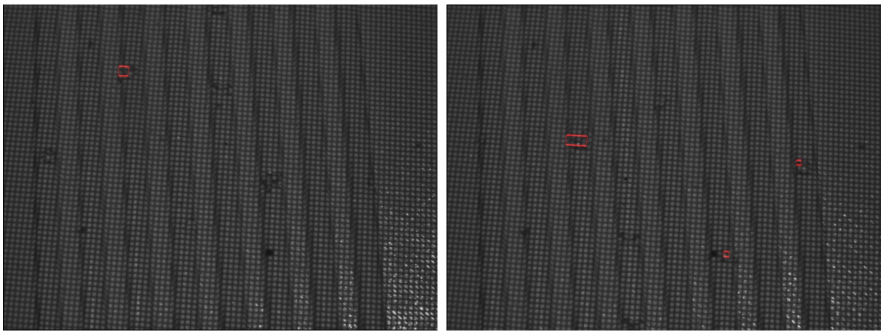


Figure 5.23: First set: only acceptable results for the FFT comparison.

The pictures were taken with an exposure of 0.25 ms. Examples of acceptable and none acceptable resulting pictures are shown in appendix I.3.

The exact positioning seems still to be a problem even when using blocks against which the samples are wedged, because some pictures mark all the channel borders as defects. The FFT method is too sensible to this problem, which makes the results of it unreadable. Only the first set of images give acceptable defect inspections (figure 5.23).This method is therefore considered unreliable. As explained in the previous section, the GLCM with template method is not adapted to none planar objects. Only the GLCM comparison method gives readable results in this case.

A total of 43 defects were created on the surface of the 3D printed BPPs. From these, 23 were detected by the GLCM algorithm with 4 false detections which gives a detection rate of 53 % and a false detection rate of 9 %. The decrease in the detection rate compared to the rate found from the metal discs from 61 % to 53 % may be due to the difference in materials; the PLA is less reflective which causes a less sharp delineation of the grid lines and therefore a loss in detection accuracy[7].

5.5 Testing with more disc samples

To have a better idea of the smallest detectable defects on metallic surfaces, the tests are continued with metallic discs. The same program was also executed on other pictures from different discs with smaller defects. Summing these pictures with the last 36 pictures, there is a total of 64 images and 100 defects. The results are shown in table 5.11.

[7] Although this issue should be solved by the opening function.

Types of defects	Number of defects	GLCM		FFT		GLCM with template	
Hole $e > 0.16\,\text{mm}$	27	23	85 %	17	63 %	24	89 %
Hole $e < 0.16\,\text{mm}$	8	2	25 %	2	25 %	6	75 %
Elevation $e > 0.16\,\text{mm}$	20	5	25 %	9	45 %	6	30 %
Elevation $e < 0.16\,\text{mm}$	14	0	0 %	0	0 %	0	0 %
Slit[8] $e > 0.16\,\text{mm}$	9	4	44 %	6	67 %	8	89 %
Slit[9] $e < 0.16\,\text{mm}$	22	0	0 %	1	5 %	1	5 %
Total	100	34		35		45	
False defect	64 pictures	8	12 %	44	69 %	2	3 %[10]

Table 5.11: Results for the second test with more images.

Taking into consideration the minimum detectable size (1.6 mm height difference or length larger than 0.34 mm and height difference of 0.16 mm) from section 5.3, the number of these defects does not change but only the total number of defects taken into account which is no longer 100 but 56. So the final detection and the false detection rate are increased:

	Detection rate	False detection rate
GLCM	57 %	12 %
FFT	57 %	69 %
GLCM with template	68 %	3 %

Table 5.12: Results considering only the theoretically detectable defects.

However, there are still defects that are not recognized. Especially the elevations often fail to be inspected. When ignoring all bimples among the defects, the detection rates increase to 75 % for the GLCM method, 64 % for the FFT method and 89 % for the GLCM with template method. Overall the smallest deepening to be always detected is of diameter 0.6 mm and depth 0.2 mm. And the smallest elevation being always detected if of height 2 mm.

5.6 Processing time optimization

The processing time will be measured on the execution of the Python program and it will be assumed that the data of the different cameras is treated simultaneously. Thus, the inspection time of all cameras is considered equal to the inspection time of one camera. The processing is split in:

[8] With a length $L > 10\,\text{mm}$and a width $l < 0.3\,\text{mm}$.

[9] With a length $L > 10\,\text{mm}$ and a width $l < 0.3\,\text{mm}$.

[10] The border of the disc is considered a legitimate defect for this algorithm, but as it is not part of the region of interest, it is not count.

- the correction time (HDR, lens and perspective correction and mean threshold),
- the detection time (eather GLCM, FFT or GLCM with template) and
- the displaying time.

These values were measured via the python code and the average computed for different pictures.

	Correction	GLCM	FFT	GLCM w/ template	Displaying
Mean (s)	0.391	11.269	24.270	5.148	0.041
Relative STD (%)	0.171	0.476	1.013	1.141	19.240

Table 5.13: Processing time.

So the processing time is 11.701 s for the GLCM, 24.702 s for the FFT and 5.580 s for the GLCM with template method. To get the entire inspection time, the acquisition time of $\frac{1}{1000} + \frac{1}{4000} + \frac{1}{10000} \simeq 1.35 \cdot 10^{-3}$ s must be added which is negligible in this case. From table 5.13 can be seen that the most time consuming part is the detection time which is in fact the longest for the FFT and the fastest for the GLCM with template method.

These duration could be reduced by using less features and less block-sizes or setting less overlap. For example, if no overlap is used instead of the half of a block-size, the detection times would be divided by two[11] . If only half of the features are used to define the defects, the time is again divided by two. In this way the inspection time could be reduced considerably (approximately divided by four in the example). However these changes could negatively affect the accuracy of the inspection. To test this, the last experiment is repeated using only one feature and block-size and without overlap. For both GLCM methods the kept feature is the cluster tendency with block-size 16 pixels and for the FFT algorithm the retained filter is the "boxcar" filter with a block-size of 32 pixels[12].

Table 5.14 resumes the final inspection times and the corresponding detection rates (counting only the theoretically detectable defects) depending on used features, block-size and overlap. The definitions for "test1", "test2" and "test3" can be found in table 5.15.

[11] Except for the GLCM with template method as it allready did not use overlap.

[12] This modes have been chosen because they showed the lowest relative standard deviation while determining the thresholds.

	GLCM			FFT			GLCM with template		
	Time (s)	Detection rate (%)	False defect (%)	Time (s)	Detection rate (%)	False defect (%)	Time (s)	Detection rate (%)	False defect (%)
test1	11.70	57	12	24.70	57	69	5.58	68	3
test2	1.76	58	14	1.37	69	70	2.02	71	9
test3	0.742	59	52						

Table 5.14: Influence of the algorithm complexity on the results.

		test1	test2	test3
GLCM	Features	cluster tendency, maximum probability	cluster tendency	cluster tendency
	Thresholds[13]	[0.2, 0.148], [0.47, 0.18]	[0.47]	[0.2]
	Block size	(16 x 16), (32 x 32)	(16 x 16)	(32 x 32)
	Overlap	2	none	none
GLCM with template	Features	cluster tendency, entropy, maximum probability, uniformity of energy	cluster tendency	
	Thresholds	[0.52, 0.8, 0.43, 0.49]	[0.52]	
	Block size	(16 x 16)	(16 x 16)	
	Overlap	2	none	none
FFT	Features	boxcar, triang, hamming, bartlett	boxcar	
	Thresholds	[10.5, 4.6, 4.8, 4.6], [18.2, 7.7, 8.2, 7.5], [32, 12.9, 13.9, 13]	[10.5]	
	Block size	(32 x 32), (64 x 64), (128 x 128)	(16 x 16)	
	Overlap	2	none	none

Table 5.15: Legend for test1, test2 and test3.

There is no significant change between the rates from test1 and test2. They even seem to be better. The FFT method also seems to have improved. But its false defect detection rate is not acceptable, too many areas are reported as defective when they are not and the areas are not always located at the disc's borders. This algorithm can't be considered as functional. Because of this, test3 was only applied on the GLCM algorithm. Its inspection time was under one second but the false detection rate has has almost quadrupled.

5.7 Evaluation of results

For each requirement made in section 3.1, the ideal result and the practical results are elaborated then compared with the desired result in order to validate (or not) the program.

[13] One array gives the thresholds for one block-size. Within an array the thresholds are listed correspondantly to the features listing on the line above.

Nr.	Description	Imp[14]	Characteristics	Results
1.	Max testing area	3	Min 373 cm^2	- ideal case: 38.6 cm^2 with four cameras, - real case: 15.1 cm^2 with four cameras;
2.	Inspection rate	2	Min 1 Hz	0.76 Hz (but it can be improved with better processors)
3.	Acquisition time	3	Max 26 µs	1.35 ms
4.	Accuracy	1	Min 5 µm	- smallest depth 0.2 mm, - smallest diameter 0.6 mm.
5.	Detection rate	1	90 %	- GLCM: 57 % - FFT: 57 % - GLCM w/ template: 68 %
6.	False defect rate	1	Max 10 %	- GLCM: 12 % - FFT: 69 % - GLCM w/ template: 3 %
7.	Risk group according to CEI EN 62471:2010	1	exempt	Datasheet of light source
8.	Invariance to the part position	2	rotation translation	only fulfilled by the GLCM with template algorithm

Table 5.16: Fulfillment of the requirements table 3.1.

1. Maximum testing area. In the ideal scenario, the testing area of one camera is $71.7 \cdot 53.8 \simeq 3.86 \cdot 10^3$ mm^2. It was determined that four cameras are necessary to be able to really detect every error. By assuming that the plate is a square (worst case scenario), the side would be around 193 mm and three sets of four cameras would be enough to inspect the widht of the plate. The plate can then be moved forward in order to inspect the entire surface.

In practice, the recorded area is of $44.8 \cdot 33.6 \simeq 1.51 \cdot 10^3$ mm^2. So 5 sets of four cameras are needed (20 cameras) which is still a viable solution. The requirement is fulfilled.

[14] Importance factor: 1 most important - 5 least important.

2. Inspection rate. The required inspection time is one second. This duration is exceeded for every test from table 5.14 except for the test3, but the false defect rate is much higher than in the other two tests. To fulfill the requirements a compromise must be found. In the requirements table 3.1 , the inspection rate is rated less important than the requirement of the false defect detection rate. Thus, the most adapted solution would be to use the GLCM algorithm under the conditions of test2. It is only 76 % over the requirement and it can be improved by using a computer with a faster processor.

3. Acquisition time. In the ideal case, the camera exposure can be set to 24 µs which fulfills the requirement number 3. However, in practice the images thus obtained are too dark to recognize the grid correctly. An image of the table was taken with an exposure of 1/41667 seconds. The highest grayscale of this picture is 71 (of 255), but its mean grayscale value is 2.79 and its median is 2 which means that over the half of all pixels have the value 0 or 1. That is not enough to threshold the image correctly. The image is black, the grid is not distinguishable.

For the experiments, three pictures of different exposures have been combined. The exposure times were arbitrarily chosen so that the limits of the grid could be observed in the darkest and most exposed areas. So the real acquisition time is of $\frac{1}{1000} + \frac{1}{4000} + \frac{1}{10000} \simeq 1.35 \cdot 10^{-3}$ s, over 52 times longer than the whished 26 µs. However, this goal value was computed from the smallest defect to detect (assuming it will be at least 5 µm large). This value has been underestimated. According to section 5.3, the smallest defect to be detected is around 0.6 mm, so 120 times larger as expected.

But what actually needs to be considered is that the part moves less than the length of the captured area of a pixel during the acquisition time. The part is moving at 19 cm s^{-1} and in this case one pixel is 33 µm wide which gives an acquisition time of 0.174 ms. In the ideal case the calculated observed pixel width was larger (56 µm). That would give an acceptable acquisition time of 0.295 ms. So even by considering the ideal observed pixel area, the actual acquisition time of the system is too long to be used on a constant continuous production line. For it to be suitable, the chain would have to be slowed down at times or stopped completely for 1.35 ms.

4. Accuracy. The theoretical accuracy is determined in section 5.3. In practice, the smallest defect that has been detected is of depth 0.2 mm and diameter 0.6 mm. The required accuracy of 5 µm is not achieved.

5. Detection rate. The detection rates are elaborated in section 5.5 with 64 different images of discs with a total number of 59 theoretically detectable defects. Only the GLCM with template algorithm has better results than the two other algorithms, it is however only usable for the inspection of plane objects. When elevations are not considered, its detection rate even gets to 89 %.

6. False defect rate. The detection rates are elaborated in section 5.5 with 64 different images of discs with a total number of 59 theoretically detectable defects. The requirement of maximum 10 % false defect detection is only obtained for the GLCM algorithm with template and nearly obtained with the simple GLCM algorithm.

8. Invariance to the part position. Only the GLCM with template algorithm is completely independant to the part's orientation or positioning. The other two methods need the inspected part to be precisely positioned because the image to inspect is compared to a reference image taken in advance. When these pictures do not coincide, all geometrical feature of the part can be considered a defect.

Overall, the algorithm that meets the requirements the best is the GLCM with template algorithm. It has the shortest processing time, the best detection rate and lowest false detection rate. The only issue is that it is only suitable for the inspection of flat surfaces. For the defect detection of metallic bipolar plates it is not adapted. For this, the best solution is the GLCM comparison method which has a processing time of less than 2 seconds, a detection rate of over 50 %, a false detection rate of 14 % and showed acceptable results with the 3D printed BPP. Unfortunately, even this method does not enable to find the defects on the metallic BPP samples because of their random curvature, caused by the thin sheet metal, and their too small defects for this system.

Chapter 6

Discussion and Conclusion

6.1 Discussion

Programming in Xeidana. For now, the system has been tested with one camera on a static environment coded in Python. The next step would be to test it on a test bench accomplishing the inspection surface requirement (use of several cameras at a time) coded in Xeidana and using a moving part. Xeidana is an extensible environment for industrial data analysis developed by Fraunhofer IWU. It is a graphical programming language where the different image processing functions are integrated in modules using parallelization[1] [xei].

Defect classification. The simplest way to implement a defect classification in the existing program would be to look for similarities in the thresholds according to the observed defect. For example, as mentioned in section 5.2, only the holes were distinguishable on the maximum probability and uniformity of energy matrix feature matrices. This could also be the reason why most bumps or bimples were not recognized by the system. Maybe their threshold is slightly different from the final chose thresholds an this led to their non-recognition. To verify this, the threshold evaluation must be done on a larger amount of samples with well-defined defects.

Another way of classifying the defects would be to look at every angle of the Gray-Level Co-occurence Matrix separately and thus obtain more feature

[1] each processing module works prarallel to the others; incoming data is processed and sent to the subsequent module

matrices. Each new feature matrix can present a difference depending on which type of defect is analyzed. Increasing the amount of angles for the GLCM computation could also increase the chance of classifying the defects. If there is the possibility to produce more samples, the defect classification could also be done by a trained artificial intelligence. Data augmentation might as well be used to increase the amount of available input data [SK19].

Accuracy optimization. One way to optimize accuracy without changing equipment would be to analyze only the top of the image. As seen in section 5.3, the theoretical minimum detectable height difference is at the opposite side of the camera inclination. When the system is used as inline inspection system, the parts will move under the camera and the entire surface will be scanned without adding new cameras. But this requires the acquisition of a larger number of images. Another way to reduce the minimum size of detectable defects is to take several pictures of the part on fixed predefined positions, so even the smallest defects do not hide in a white square of the grid.

Otherwise, the accuracy can be improved by using a camera with higher resolution. It would be possible to obtain an accuracy more than ten times better with a 151MP camera then with a 1 MP camera. However, a higher resolution requires a larger number of different observed positions (more pictures) in order to avoid the defect to be hidden in the grid projection. This increases the inspection time.

Application on BPPs. The program is not applicable for the inspection of metallic BPPs because they present too much curvature due to the very thin material. One solution that has not been tested yet is to use a transparent plate (PVC or glass) to flatten the plate. But, as mentioned in section 5.4, this would bring complications because of the different refraction indice of the transparent plate which will deflect the light rays. However, it could be possible to test graphite BPPs. As these plates are more rigid, they do not have random curvatures. The experimental conditions would be close to the tests made with the 3D printed parts in section 5.4.

Comparison with no grid projection. The purpose of this work was to create a new type of inspection system using a projected grid. For the developed system to be of interest, it must have an advantage over a system without grid projection. To verify this, the GLCM comparison method is tested on 64 disc pictures without grid projection, only using ambiant light for lighting with an exposure of 23 ms. The disc defects and positions are exactly the same as for the previous 64 disc pictures with grid. The results are described in table 6.1.

Types of defects	Number of defects	GLCM with grid		GLCM without grid	
Hole $e > 0.16\,\mathrm{mm}$	27	23	85 %	27	100 %
Hole $e < 0.16\,\mathrm{mm}$	8	2	25 %	8	100 %
Elevation $e > 0.16\,\mathrm{mm}$	20	5	25 %	19	95 %
Elevation $e < 0.16\,\mathrm{mm}$	14	0	0 %	14	100 %
Slit[2] $e > 0.16\,\mathrm{mm}$	9	4	44 %	9	100 %
Slit[3] $e < 0.16\,\mathrm{mm}$	22	0	0 %	22	100 %
Total	100	34		99	
False defect	64 pictures	8	12 %	28	44 %

Table 6.1: Comparison of the GLCM algorithm used on pictures with and without the grid projection.

Applying the exact same algorithm on pictures without the grid projection leads to a detection rate of 99 %, detecting even the smallest defects, but it also detects non-topographical defects like color variations and thus have a higher false detection rate. For better visualization, the resulting images from sample number 3 and 25 are shown in appendix I.6. The higher false detection rate is also caused by the lighting variation of the daylight. Better results are expected when using steady artificial light sources.

In the particular case of inspecting die cut metallic discs, the usage of a grid projection does not add value to the inspection system. However, without pattern projection, the system is sensible to two-dimensional defects, so the grid projection might be more suitable in cases where the surface of the parts has color variations that are not considered defects. For instance, if something was printed on them for aesthetic or informative reasons. It could also be more useful for defects that do not scatter light in all directions and therefore have little chance of being detected by systems with conventional light sources. Finally, the grid projection can be used to create a 3D reconstruction of the analyzed object using triangulation [Gen11].

[2] With a length $L > 10\,\mathrm{mm}$ and a width $l < 0.3\,\mathrm{mm}$.

[3] With a length $L > 10\,\mathrm{mm}$ and a width $l < 0.3\,\mathrm{mm}$.

6.2 Conclusion

To summarize, research was conducted to determine the utility of a metallic bipolar plate, how it is manufactured, and what inspection systems exist that could be applied to it. A list of requirements has been established for the system to meet the industrial demands and boundaries. From this, a set-up was developed with the available equipment.

A theoretical model was implemented to evaluate the minimum size of the theoretically detectable defect depending on the camera inclination to the part, the camera-part distance and the picture resolution. It showed that, the stronger the camera inclination and the higher the picture resolution, the smaller the detectable defects; the larger the camera-part distance, the more even the size of the minimum detectable defect on the picture. By using the 1MP camera at 24° and the grid projector (PTGR020035P) at their minimum working distances, the defects must be larger than $0.285\,\mathrm{mm}^2$ and present a height difference of 0.4 mm to be detectable. If their surface is smaller, their height difference with the original position must be 2 mm.

Four algorithms were programmed and tested, first on metallic BPP samples. The results were not evaluable because of the very thin material that caused random curbations. These deformations distorted the grid which led to the whole part being recognized as a defect. Second, the program was tested on 3D-printed BPP samples (scale 1:2 to the metallic samples) and on metallic discs. The statistical comparison was rejected after the first attempt because the defect detection was not based on the deformation of the grid. The GLCM comparison was the only algorithm that showed readable results with the 3D-printed BPPs. It had a 53 % detection rate and 9 % false detection. With the metallic discs the detection rate was of 57 % and the false detection of 12 %. A detection frequency of around 0.76 plates/s was measured. The FFT comparison showed no readable results with the 3D-printed samples and unsatisfactory results with the metalic discs. With a 57 % detection rate and 69 % false detection it was stated as not reliable enough. Finally, the GLCM comparison with template presented the best results with 68 % detection rate and 3 % false detection, but could only be applied on flat surfaces.

It was also discussed that the GLCM algorithm gave better results on metal discs without using the grid projection, illuminated only by daylight. Future research should focus on testing the algorithms on applications where the grid provides an added-value: parts with an unimportant or wanted discoloration or with larger defects which don't scatter the light; and extending the GLCM comparison with template to more complicated surfaces.

Appendix A

Bibliography

[ace] *Fuel types of new cars: battery electric 9.1%, hybrid 19.6% and petrol 40.0% market share full-year 2021*, https://www.acea.auto/fuel-pc/fuel-types-of-new-cars-battery-electric-9-1-hybrid-19-6-and-petrol-40-0-market-share-full-year-2021/, Accessed: 01.08.2022.

[AOP19] Oluwaseun Ayotunde Alo, Iyiola Olatunji Otunniyi, and HCvZ Pienaar, *Manufacturing methods for metallic bipolar plates for polymer electrolyte membrane fuel cell*, Materials and Manufacturing Processes **34** (2019), no. 8, 927–955.

[AS09] José Manuel Andújar and Francisca Segura, *Fuel cells: History and updating. a walk along two centuries*, Renewable and sustainable energy reviews **13** (2009), no. 9, 2309–2322.

[azo] *Carbon - graphite materials*, https://www.azom.com/properties.aspx?ArticleID=516, Accessed: 06.08.2022.

[AZP$^+$19] Buddhika Abeyrathna, Peng Zhang, Michael P Pereira, Daniel Wilkosz, and Matthias Weiss, *Micro-roll forming of stainless steel bipolar plates for fuel cells*, International Journal of Hydrogen Energy **44** (2019), no. 7, 3861–3875.

[BB02] John R Bowler and Nicola Bowler, *Evaluation of the magnetic field near a crack with application to magnetic particle inspection*, Journal of physics D: applied physics **35** (2002), no. 18, 2237.

[BCZ20] Colin Bell, Jonathan Corney, and Nicola Zuelli, *A state of the art review of hydroforming technology*, International Journal of Material Forming **13** (2020), no. 5, 789–828.

[BHA19] Alexander Bauer, Sebastian Härtel, and Birgit Awiszus, *Manufacturing of metallic bipolar plate channels by rolling*, Journal of Manufacturing and Materials Processing **3** (2019), no. 2, 48.

[BQO+10] Yukihiro Bandoh, Guoping Qiu, Masahiro Okuda, Scott Daly, Til Aach, and Oscar C Au, *Recent advances in high dynamic range imaging technology*, 2010 IEEE International Conference on Image Processing, IEEE, 2010, pp. 3125–3128.

[cal] *Camera calibration*, `https://opencv24-python-tutorials.readthedocs.io/en/stable/py_tutorials/py_calib3d/py_calibration/py_calibration.html`, Accessed: 29.07.2022.

[Cam13] Flake C Campbell, *Inspection of metals: understanding the basics*, ASM International, 2013.

[can] *Canny edge detection*, `https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html`, Accessed: 26.07.2022.

[CCM+20] Tamás Czimmermann, Gastone Ciuti, Mario Milazzo, Marcello Chiurazzi, Stefano Roccella, Calogero Maria Oddo, and Paolo Dario, *Visual-based defect detection and classification approaches for industrial applications—a survey*, Sensors **20** (2020), no. 5, 1459.

[DM08] Paul E Debevec and Jitendra Malik, *Recovering high dynamic range radiance maps from photographs*, ACM SIGGRAPH 2008 classes, 2008, pp. 1–10.

[Ede15] Ottmar Edenhofer, *Climate change 2014: mitigation of climate change*, vol. 3, Cambridge University Press, 2015.

[FFF+17] Paulo MF Forte, Paulo Eduardo Reis Felgueiras, Flavio P Ferreira, MA Sousa, Eduardo J Nunes-Pereira, Boris PJ Bret, and Michael S Belsley, *Exploring combined dark and bright field illumination to improve the detection of defects on specular surfaces*, Optics and Lasers in Engineering **88** (2017), 120–128.

[Gen11] Jason Geng, *Structured-light 3d surface imaging: a tutorial*, Advances in Optics and Photonics **3** (2011), no. 2, 128–160.

[GMR04] A Gachagan, A McNab, and P Reynolds, *Analysis of ultrasonic wave propagation in metallic pipe structures using finite element modelling techniques*, IEEE Ultrasonics Symposium, 2004, vol. 2, IEEE, 2004, pp. 938–941.

[Har78a] Fredric J Harris, *On the use of windows for harmonic analysis with the discrete fourier transform*, Proceedings of the IEEE **66** (1978), no. 1, 51–83.

[Har78b] ______, *On the use of windows for harmonic analysis with the discrete fourier transform*, Proceedings of the IEEE **66** (1978), no. 1, 51–83.

[hdr] *Exposure fusion using opencv (c++/python)*, `https://learnopencv.com/exposure-fusion-using-opencv-cpp-python/`, Accessed: 29.07.2022.

[HFY20] Yanqing Han, Junfeng Fan, and Xiaozhong Yang, *A structured light vision sensor for on-line weld bead measurement and weld quality inspection*, The International Journal of Advanced Manufacturing Technology **106** (2020), no. 5, 2065–2078.

[HGL+21] Yanchen He, Jinxi Gong, Huixia Liu, Youjuan Ma, Jiaxin Lu, and Xiao Wang, *Investigation on formability improvement in laser shock hydroforming*, International Journal of Material Forming **14** (2021), no. 5, 855–869.

[HIZA18] Lei Huang, Mourad Idir, Chao Zuo, and Anand Asundi, *Review of phase measuring deflectometry*, Optics and Lasers in Engineering **107** (2018), 247–257.

[HS95] Richard I Hartley and Peter Sturm, *Triangulation*, International Conference on Computer Analysis of Images and Patterns, Springer, 1995, pp. 190–197.

[HSL19] Chih-Cheng Hung, Enmin Song, and Yihua Lan, *Image texture analysis*, Springer, 2019.

[Hyk14] Jon Hykawy, *A stress test on future graphite pricing*, Industry report, STORMCROW (2014).

[IKL+11] Hun Suk Im, Sang-Kyung Kim, Seongyop Lim, Dong-Hyun Peck, Doohwan Jung, and Won Hi Hong, *Study on the water flooding in the cathode of direct methanol fuel cells*, Journal of Nanoscience and Nanotechnology **11** (2011), no. 7, 5788–5794.

[IMA] Stemmer IMAGING, *Ccd oder cmos: Können cmos-sensoren in allen fällen ccds ersetzen?*, Allied Vision, 1–8.

[JJ05] L Jagannathan and CV Jawahar, *Perspective correction methods for camera based document analysis*, Proc. First Int. Workshop on Camera-based Document Analysis and Recognition, 2005, pp. 148–154.

[JJHK14] Min-Geun Jeong, Chul-Kyu Jin, Gyu-Wan Hwang, and Chung-Gil Kang, *Formability evaluation of stainless steel bipolar plate considering draft angle of die and process parameters by rubber forming*, International journal of precision engineering and manufacturing **15** (2014), no. 5, 913–919.

[JMK+17] Gyeongju Je, Dror Malka, Hyesu Kim, Sungmoo Hong, and Bosung Shin, *A study on micro hydroforming using shock wave of 355 nm uv-pulsed laser*, Applied Surface Science **417** (2017), 244–249.

[KFRF12] Shahram Karimi, Norman Fraser, Bronwyn Roberts, and Frank R Foulkes, *A review of metallic bipolar plates for proton exchange membrane fuel cells: materials and fabrication methods*, Advances in Materials Science and Engineering **2012** (2012).

[led] *Led inspection lighting*, `https://www.metaphase-tech.com/applications/led-inspection-lighting/`, Accessed: 22.03.2022.

[LH10] Yanxiong Liu and Lin Hua, *Fabrication of metallic bipolar plate for proton exchange membrane fuel cells by rubber pad forming*, Journal of Power Sources **195** (2010), no. 11, 3529–3535.

[MBJHG16] N Mohammadtabar, M Bakhshi-Jooybari, SJ Hosseinipour, and AH Gorji, *Feasibility study of a double-step hydroforming process for fabrication of fuel cell bipolar plates with slotted interdigitated serpentine flow field*, The International Journal of Advanced Manufacturing Technology **85** (2016), no. 1, 765–777.

[mea] *Miscellaneous image transformations, adaptivethresholdtypes*, `https://docs.opencv.org/4.x/d7/d1b/group__imgproc__misc.html#ggaa42a3e6ef26247da787bf34030ed772cad0c5199ae8637a6b195062fea4789fa9`, Accessed: 29.07.2022.

[MKVR07] Tom Mertens, Jan Kautz, and Frank Van Reeth, *Exposure fusion*, 15th Pacific Conference on Computer Graphics and Applications (PG'07), IEEE, 2007, pp. 382–390.

[MKVR09] ———, *Exposure fusion: A simple and practical alternative to high dynamic range photography*, Computer graphics forum, vol. 28, Wiley Online Library, 2009, pp. 161–171.

[NMD14] Nirbhar Neogi, Dusmanta K Mohanta, and Pranab K Dutta, *Review of vision-based steel surface inspection systems*, EURASIP Journal on Image and Video Processing **2014** (2014), no. 1, 1–19.

[OBM15] Romana Oancea, Vlad Bârsan, and Cristian Molder, *Perspective correction for image processing*, International conference KNOWLEDGE-BASED ORGANIZATION, vol. 21, 2015, pp. 712–721.

[OBS01] Alan V Oppenheim, John R Buck, and Ronald W Schafer, *Discrete-time signal processing. vol. 2*, Upper Saddle River, NJ: Prentice Hall, 2001.

[ope] *Morphological transformations*, `https://docs.opencv.org/4.x/d9/d61/tutorial_py_morphological_ops.html`, Accessed: 29.07.2022.

[opta] *f/# and depth of field*, `https://www.opto-e.com/basics/f-and-depth-of-field`, Accessed: 29.07.2022.

[optb] *Structured illumination - opto engineering*, `https://www.opto-e.com/basics/structured-illumination`, Accessed: 30.03.2022.

[OPT22] OPTO Engineering, *Strobe controllers, 1 channel, instructions manual*, 07 2022, v1.0.

[pat] *Offenlegungsschrift: Bezeichnung: Verfahren zur detektion von topografischen fehlern auf oberflächen eines bauteils.*

[PC18] Nicholas Paul and ChanJin Chung, *Application of hdr algorithms to solve direct sunlight problems when autonomous vehicles using machine vision systems are driving into sun*, Computers in Industry **98** (2018), 192–196.

[pCOG05] pco. Computer Optics GmbH, *Smear in ccd-sensors*, 1–9.

[pi05] pco. imaging, *blooming - anti-blooming*, the cooKe corporation (2005), 1–2.

[Pie20] Pierer, *Sensordatenfusion zur prozessintegrierten qualitätsüberwachung für fehlerfreie produkte*, Frauenhofer IWU, Optische Inline-Inspektion (2020), 1–10.

[PL^{+}09] Linfa Peng, Xinmin Lai, et al., *Effect of dimensional error of metallic bipolar plate on the gdl pressure distribution in the pem fuel cell*, International journal of hydrogen energy **34** (2009), no. 2, 990–997.

[Pop14] Ovidiu Popovici, *The study of the efficiency of a direct methanol fuel cell*, Journal of Electrical and Electronics Engineering **7** (2014), no. 2, 31.

[PPW19] Sebastian Porstmann, Allan Christian Petersen, and Thomas Wannemacher, *Analysis of manufacturing processes for metallic and composite bipolar plates*, Konferenzband der ersten FC^3 Fuel Cell Conference Chemnitz 26. und 27. November 2019, 2019, p. 25.

[PWD20] Sebastian Porstmann, Thomas Wannemacher, and W-G Drossel, *A comprehensive comparison of state-of-the-art manufacturing methods for fuel cell bipolar plates including anticipated future industry trends*, Journal of Manufacturing Processes **60** (2020), 366–383.

[RBS99] Mark A Robertson, Sean Borman, and Robert L Stevenson, *Dynamic range improvement through multiple exposures*, Proceedings 1999 International Conference on Image Processing (Cat. 99CH36348), vol. 3, IEEE, 1999, pp. 159–163.

[RH07] Sergio Rapuano and Fred J Harris, *An introduction to fft and time domain windows*, IEEE instrumentation & measurement magazine **10** (2007), no. 6, 32–44.

[RNJ^{+}21] Uwe Reimer, Ekaterina Nikitsina, Holger Janßen, Martin Müller, Dieter Froning, Steven B Beale, and Werner Lehnert, *Design and modeling of metallic bipolar plates for a fuel cell range extender*, Energies **14** (2021), no. 17, 5484.

[SGTL18] Xiaohong Sun, Jinan Gu, Shixi Tang, and Jing Li, *Research progress of visual inspection technology of steel products—a review*, Applied Sciences **8** (2018), no. 11, 2195.

[Sid20] Marco Siddi, *The european green deal: Asseasing its current state and future implementation.*

[SJDH11] PV Suresh, S Jayanti, AP Deshpande, and P Haridoss, *An improved serpentine flow field with enhanced cross-flow for fuel cell applications*, International journal of hydrogen energy **36** (2011), no. 10, 6067–6072.

[SK19] Connor Shorten and Taghi M Khoshgoftaar, *A survey on image data augmentation for deep learning*, Journal of big data **6** (2019), no. 1, 1–48.

[SKPK20] Marco Sauermoser, Natalya Kizilova, Bruno G Pollet, and Signe Kjelstrup, *Flow field patterns for proton exchange membrane fuel cells*, Frontiers in Energy Research **8** (2020), 13.

[stea] *Stainless steel metalminer prices*, https://agmetalminer.com/mmwp/metal-prices/stainless-steel/, Accessed: 06.04.2022.

[steb] *Stainless steel: Tables of technical properties*, https://www.worldstainless.org/Files/issf/non-image-files/PDF/Euro_Inox/tables_TechnicalProperties_EN.pdf, Accessed: 06.04.2022.

[SVUB12] Bino Sebastian V, A Unnikrishnan, and Kannan Balakrishnan, *Gray level co-occurrence matrices: generalisation and some new features*, arXiv preprint arXiv:1205.4831 (2012).

[SWHD10] J Shang, L Wilkerson, S Hatkevich, and GS Daehn, *Commercialization of fuel cell bipolar plate manufacturing by electromagnetic forming*.

[SYG$^+$21] Nima Shaigan, Xiao-Zi Yuan, François Girard, Khalid Fatih, and Mark Robertson, *Standardized testing framework for quality control of fuel cell bipolar plates*, Journal of Power Sources **482** (2021), 228972.

[THM07] H Tawfik, Y Hung, and D Mahajan, *Metal bipolar plates for pem fuel cell—a review*, Journal of power sources **163** (2007), no. 2, 755–767.

[tita] *Titanium alloys - ti6al4v grade 5*, https://www.azom.com/properties.aspx?ArticleID=1547, Accessed: 06.04.2022.

[titb] *Titanium price*, https://www.metalary.com/titanium-price/, Accessed: 06.04.2022.

[TSTR05] Gui Yun Tian, Ali Sophian, David Taylor, and John Rudlin, *Multiple sensors on pulsed eddy-current detection for 3-d subsurface crack assessment*, IEEE Sensors Journal **5** (2005), no. 1, 90–96.

[TVGMM17] Zhongwei Tang, Rafael Grompone Von Gioi, Pascal Monasse, and Jean-Michel Morel, *A precision analysis of camera distortion models*, IEEE Transactions on Image Processing **26** (2017), no. 6, 2694–2704.

[VD19] Nikola Vukašinović and Jože Duhovnik, *Optical 3d geometry measurments based on laser triangulation*, Advanced CAD Modeling, Springer, 2019, pp. 191–216.

[Vis18] Arnt Inge Vistnes, *Interference—diffraction*, Physics of Oscillations and Waves, Springer, 2018, pp. 419–473.

[WCH$^+$92] Juyang Weng, Paul Cohen, Marc Herniou, et al., *Camera calibration with distortion models and accuracy evaluation*, IEEE Transactions on pattern analysis and machine intelligence **14** (1992), no. 10, 965–980.

[WPG$^+$19] Thomas Wiener, Alexander Pierer, Leutrim Gjakova, Michael Hoffmann, and Sven Barthel, *Inline-qualitätsüberwachung beschichteter, metallischer bipolarplatten nach dem umformprozess*, Konferenzband der ersten FC^3 Fuel Cell Conference Chemnitz 26. und 27. November 2019, 2019, p. 157.

[xei] *Xeidana®neue software für die automatisierte qualitätssicherung*, https://www.xeidana.de/, Accessed: 06.08.2022.

[XXZ$^+$11] Jing Xu, Ning Xi, Chi Zhang, Quan Shi, and John Gregory, *Real-time 3d shape inspection system of automotive parts based on structured light pattern*, Optics & Laser Technology **43** (2011), no. 1, 1–8.

[YP99] Orly Yadid-Pecht, *Wide-dynamic-range sensors*, Optical Engineering **38** (1999), no. 10, 1650–1660.

[YYZ$^+$17] Temesgen A Yemata, Qun Ye, Hui Zhou, Aung KK Kyaw, Wee S Chin, and Jianwei Xu, *Conducting polymer-based thermoelectric composites: Principles, processing, and applications*, Hybrid Polymer Composite Materials, Elsevier, 2017, pp. 169–195.

[ZDC$^+$15] Yirong Zou, Dong Du, Baohua Chang, Linhong Ji, and Jiluan Pan, *Automatic weld defect detection method based on kalman filtering for real-time radiographic inspection of spiral pipe*, Ndt & E International **72** (2015), 1–9.

[ZWZ22] Jie Zhang, Ruichuan Wang, and Yun Zeng, *Hydroforming rules and quality control parameters analysis for metal bipolar plate*, Engineering Failure Analysis **132** (2022), 105919.

Appendix B

Index

Appendix C

Comparison of different defect detection methods [SGTL18]

Method	Advantages	Disadvantages
Mathematical morphology	Computational simplicity, geometric representation of texture image, highly suitable for random or natural textures	Only implemented on non-periodic steel defects
Co-occurence matrix	Extracting spatial relationship of pixels with different statistical computations, high accuracy rate	Difficult to judge the optimal displacement vector
Histogram properties	Translation and rotation invariance, simple calculation	Low detection rate (50 - 70 %) for irregular textures
Local binary pattern	Strong recognition ability, simple calculation, rotation and gray invariance	Too dependent on the gray value of the center point pixel
Spatial filtering	A more centralized text-based approach	Difficult to determine the optimal filter parameters, high computation cost
Frequency filtering	Invariant to shift, rotation and scaling,suitable for the detection of global and local defects, short calculation time	No spatial orientation, not suitable for random texture detection
Gabor transform	Suitable for high dimensional feature space, reduced computational complexity,suitable for defect detection in airspace and frequency domain	Determination of optimal filter parameters, no rotation invariance
Wavelet transform	Suitable for multi-scale image analysis, high detection rate (83 - 97 %), efficient image compression with less information loss	Easy affection by feature correlations between the scales
Multiscale geometric analysis	Good at image processing of strong noise background, compression with less information	Redundancy problem
Fractal model	Remain invariant to large geometric transformations and lighting variations	Low characteristics dimensions lead to weak judgment
Markov random field model	Can be used with statistical and spectral methods for segmentation applications, captures the local texture orientation information	Not invariant to rotation and scaling, can not detect small defects, not suitable for global texture analysis, strong spatial constraint
Artificial neural networks	Real-time performance, can learn complex nonlinear input-output relationships	High calculation cost, needs 100 - 4000 images
Convolutional neural network	Does everything from image input to results output, high detection rate (95 - 100 %)	Needs 4000 - 10 000 images
Moving center hypersphere	Not sensitive to noise, high accuracy (93 - 96 %)	Difficult optimal choice of parameters
Sparse coding	Can be used in input phase and output phase	Long calculation time (>45 s)

Table C.1: Comparison of the different defect detection methods.

To summarize:

- the statistical methods are, the simplest to implement and are often invariant to rotation and translations, but it sometimes affect the detection rate in a wrong way;
- the filtering methods; some are invariant to shift and rotation and good at image processing of strong noises but often difficult to determine the optimal parameters;
- the model methods is not used a lot and it is difficult to find scientific papers using it;
- the machine learning methods have the best accuracy in defect detection, but they need data bases of images that are between 100 and 10 000 images to get this precision and reliability.

Appendix D

Optics

D.1 Diffraction

To verify if the pattern will not cause a diffraction the following equation is used [Vis18]:

$$\theta = \frac{\lambda}{t} \tag{D.1}$$

Where θ is the angle of the flare, λ is the wavelength of the light and t is the line thickness of the pattern, as sketch in figure D.1.

In this case $t = 0.035\,\mathrm{mm}$ and $\lambda = 618\,\mathrm{nm}$, so $\theta = 0.0177\ ^\circ$.

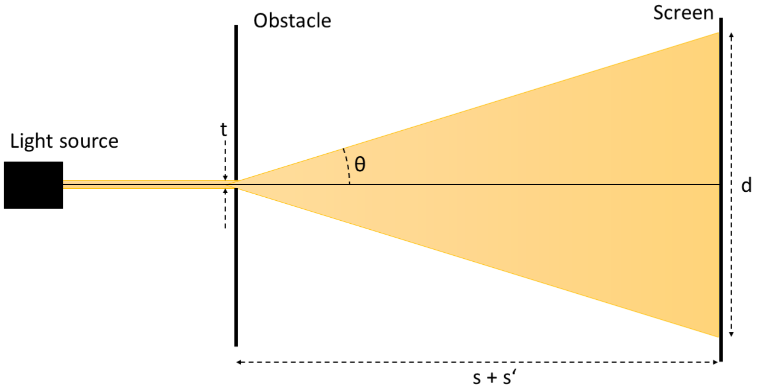


Figure D.1: Schematic of the diffraction phenomenon.

From figure D.1 the following equation can be deduced:

$$s + s' = \frac{d}{2 \cdot tan(\theta)} \tag{D.2}$$

In order to have the best precision the distance d must be smaller than the thickness of the projected line: $d \leq \frac{t_p}{M}$ is needed, so the maximum working distance is $(s + s')_{max} = \frac{t_p}{2 \cdot tan(\theta)} = \frac{0.163}{2 \cdot tan(0.0177)} = 264\,\text{mm}$.

D.2 Focal length equation

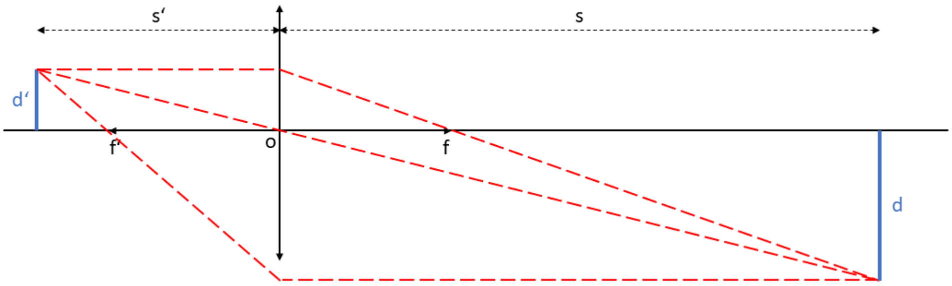


Figure D.2: Optical system with a convergent lens.

Using twice Thales's theorem in figure D.2, the following equalities are obtained:

$$\frac{f}{s} = \frac{d'}{d+d'} \tag{D.3}$$

$$\frac{f'}{s'} = \frac{d}{d+d'} \tag{D.4}$$

$$D.3 + D.4 \rightarrow \frac{f}{s} + \frac{f'}{s'} = \frac{d+d'}{d+d'} = 1 \tag{D.5}$$

As $f' = f$, the final equation is obtained.

$$\frac{1}{s} + \frac{1}{s'} = \frac{1}{f} \tag{D.6}$$

D.3 Triangulation

When a point P is visible by two cameras, its position can be described by two different vectors (x_1, y_1, z_1) and (x_2, y_2, z_2) ; one in the first camera coordinate system (X_1, Y_1, Z_1) and the other and the second camera coordinate system (X_2, Y_2, Z_2). For both vectors, only the value on the depth axis Z is unknown. Knowing the focal lengths of the cameras and their orientation, the observed point is located on a known line in space. To find out the depth, the intersection between the two lines must be computed [HS95]. It is the same problem when using a laser/projector instead of the second camera [VD19]. Here the laser must project light on the observed point instead of receiving it creating the line in spacing.

D.4 Demonstration of section 3.4

When x is the orthogonal distance between the closest and the farthest point the camera captures.

$$x = W \cdot sin(\alpha) \tag{D.7}$$

This distance must be smaller than the DoF in order to have a sharp picture, so:

$$0 < \alpha \leq \sin^{-1}\left(\frac{DoF}{W}\right) \tag{D.8}$$

Smallest DoF is 27.6 mm and W is 53.9 mm, so $\alpha \leq 30.8°$.

D.5 Working principle of Mertens fusion

It is explained in [MKVR09]. To sum up, every pixel $(x, y)_k$ of every picture k of a set of n pictures is assigned a weight $W_{x,y,k}$. This weight is the normalized multiple of the following three features [hdr]:

- $C_{x,y,q}$, the contrast: absolute value of the Laplacien filter response;
- $S_{x,y,q}$, the saturation: standard deviation;
- $E_{x,y,q}$, the well-exposedness: Gauss curve with $\sigma = 0.2$,
$$E_{x,y,q} = \exp\left(-\frac{(g((x,y))_q - 0.5)^2}{2 \cdot 0.2^2}\right)$$
.

$$W_{x,y,q} = \frac{C_{x,y,q} \cdot S_{x,y,q} \cdot E_{x,y,q}}{\sum_{p=1}^{n} C_{x,y,p} \cdot S_{x,y,p} \cdot E_{x,y,p}} \tag{D.9}$$

Thus, n weight maps are obtained. The new grayscale value function g_{HDR} is then defined by equation D.10.

$$g_{HDR}((x,y)) = \sum_{p=1}^{n} W_{x,y,p} \cdot g((x,y))_p \tag{D.10}$$

D.6 Working principle of the distortion correction [cal]

Every pixel of an image is located by its horizontal and vertical position (x, y). Because of the distortion, the actual location $(x_{correct}, y_{correct}$ is not the same as the one on the image. To correct the radial distortion, it is assumed that the actual location can be approximated by a polynomial of the radial distance to the image center r like in equations D.11 with k_1, k_2 and k_3 being real constants.

$$\begin{aligned} x_{correct} &= x \cdot (1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6) \\ y_{correct} &= ytimes \cdot (1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6) \end{aligned} \tag{D.11}$$

To correct the tangential distortion, it is assumed that the actual location can be approximated by the system in equation D.12 with p_1 and p_2 being real constants.

$$\begin{aligned} x_{correct} &= x + \left(2p_1 xy + p_2\left(r^2 + 2x^2\right)\right) \\ y_{correct} &= y + \left(2p_2 xy + p_1\left(r^2 + 2y^2\right)\right) \end{aligned} \tag{D.12}$$

To undistort pictures from a camera for every position there is also the need for a camera matrix Mtx as in equation D.13 with (f_x, f_y) being the focal length of the camera in respectively x and y direction and (c_x, c_y) the location of the optical center.

$$Mtx = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{D.13}$$

To calibrate and find the correct values for the constants k_1, k_2, k_3, p_1, p_2 and Mtx, pictures of a chessboard are used. The corners of a chessboard are easy to recognize because of its clear black to white transition and they are all at equal distance from each other which makes it simple to find their actual position. Using this information and the size of the chessboard, the constants' values can be found.A seven times seven chessboard has been printed and 20 images have been taken with the 1MP camera from different perspectives. The code and representative pictures of the chessboard corners can be found in appendix H. The resulting constants are:

$$\begin{aligned} (k_1, k_2, k_3, p_1, p_2) &= (-0.2175, -10.985, 0.0567, 0.0020, 112) \\ Mtx &= \begin{bmatrix} 5211 & 0 & 699 \\ 0 & 5049 & -34. \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{D.14}$$

D.7 equations of the statistical features

Mean:

$$\text{mean}(B) = \frac{\sum_{(x,y)\in B} f((x,y))}{|B|} \tag{D.15}$$

Variance:

$$\text{var}(B) = \frac{\sum_{(x,y)\in B}(f((x,y)) - \text{mean}(B))^2}{|B|} \tag{D.16}$$

Median: If Bs is the set of B pixels sorted by their f value and Bs_m is the m-est pixel of this set

$$\text{median}(B) = \begin{cases} f(Bs_{\frac{|B|+1}{2}}) & \text{if} \quad |B| \equiv 1 \pmod 2 \\ \frac{f(Bs_{\frac{|B|}{2}})+f(Bs_{\frac{|B|}{2}+1})}{2} & \text{if} \quad |B| \equiv 0 \pmod 2 \end{cases} \tag{D.17}$$

Standard deviation:

$$\text{std}(B) = \sqrt{\text{var}(B)} \tag{D.18}$$

Skewness:

$$\text{skew}(B) = \frac{\sum_{(x,y)\in B}(f((x,y)) - \text{mean}(B))^3}{\text{std}(B)^3} \tag{D.19}$$

Kurtosis:

$$\text{kurto}(B) = \frac{\sum_{(x,y)\in B}(f((x,y)) - \text{mean}(B))^4}{\text{std}(B)^4} \tag{D.20}$$

D.8 equations of the GLCM features

By noting N the number of rows/columns of the GLCM and $\forall i, j \in [\![0..N-1]\!]^2, G(i,j)$ the value of the GLCM on row i an column j, the formulas for the features can be written as follows.

Cluster tendency:

$$\text{cluster} = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}(i+j-2\mu)^2 \cdot G(i,j) \tag{D.21}$$

$$\mu = \frac{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} G(i,j)}{N}$$

Contrast:

$$\text{contrast} = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}|i-j|^2 \cdot G(i,j) \tag{D.22}$$

Correlation:

$$\text{correlation} = \frac{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} ijG(i,j) - \mu_1\mu_2}{\sigma_1^2\sigma_2^2} \tag{D.23}$$

$$\mu_1 = \sum_{i=0}^{N-1} i \sum_{j=0}^{N-1} G(i,j)$$

$$\mu_2 = \sum_{j=0}^{N-1} j \sum_{i=0}^{N-1} G(i,j)$$

$$\sigma_1^2 = \sum_{i=0}^{N-1} (i-\mu_1)^2 \sum_{j=0}^{N-1} G(i,j)$$

$$\sigma_2^2 = \sum_{j=0}^{N-1} (j-\mu_2)^2 \sum_{i=0}^{N-1} G(i,j)$$

Dissimilarity:

$$\text{dissimilarity} = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \frac{G(i,j)}{1+|i-j|} \tag{D.24}$$

Entropy:

$$\text{entropy} = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} G(i,j)\cdot\log(G(i,j)) \tag{D.25}$$

Homogeneity:

$$\text{homogeneity} = \text{dissimilarity} \tag{D.26}$$

Inverse difference moment:

$$\text{invDiffMoment} = \sum_{i=0}^{N-1}\sum_{j\neq i} \frac{G(i,j)}{|i-j|^3} \tag{D.27}$$

Maximum probability:

$$\text{maxProba} = \max_{i,j} G(i,j) \tag{D.28}$$

Uniformity of energy:

$$\text{uniformity} = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} (G(i,j))^2 \tag{D.29}$$

D.9 equation of the passing windows

Noting win the window functions, MxM the size of the block in pixels^2 and (m,n) the location of a pixel in the block with $(m,n) \in [\![-\frac{M}{2} .. \frac{M}{2}]\!]^2$. Defining $N = \frac{M}{2}$.

[OBS01] "boxcar" also known as rectangular window is defined by:

$$win(m,n) = \begin{cases} 1 & \text{if } \sqrt{m^2+n^2} \leq N, \\ 0 & \text{else.} \end{cases} \tag{D.30}$$

"triang" also known as Bartlett (triangular) window is defined by [OBS01]:

$$win(m,n) = \begin{cases} \frac{\sqrt{m^2+n^2}}{N} & \text{if } \sqrt{m^2+n^2} \leq N, \\ 0 & \text{else.} \end{cases} \tag{D.31}$$

"hann" also known as Hanning window is defined by [OBS01]:

$$win(m,n) = \begin{cases} 0.5 - 0.5\cos\left(\frac{\pi\sqrt{m^2+n^2}}{N}\right) & \text{if } \sqrt{m^2+n^2} \leq N, \\ 0 & \text{else.} \end{cases} \tag{D.32}$$

"hamming" called Hamming window is defined by [OBS01]:

$$win(m,n) = \begin{cases} 0.54 - 0.46\cos\left(\frac{\pi\sqrt{m^2+n^2}}{N}\right) & \text{if } \sqrt{m^2+n^2} \leq N, \\ 0 & \text{else.} \end{cases} \tag{D.33}$$

"blackman" called Blackman window is defined by [OBS01]:

$$win(m,n) = \begin{cases} 0.42 - 0.5\cos\left(\frac{\pi\sqrt{m^2+n^2}}{N}\right) + 0.08\cos\left(\frac{2\pi\sqrt{m^2+n^2}}{N}\right) & \text{if } \sqrt{m^2+n^2} \leq N, \\ 0 & \text{else.} \end{cases} \tag{D.34}$$

"parzen" also called de la Vallé-Poussin window is defined by [Har78b]:

$$win(m,n) = \begin{cases} 1.0 - 6[\frac{\sqrt{m^2+n^2}}{N}]^2[1.0 - \frac{\sqrt{m^2+n^2}}{N}] & \text{if } \sqrt{m^2+n^2} \leq \frac{N}{2}, \\ 2[1.0 - \frac{\sqrt{m^2+n^2}}{N}]^3 & \text{if } \frac{N}{2} \leq \sqrt{m^2+n^2} \leq N, \\ 0 & \text{else.} \end{cases} \tag{D.35}$$

"blackmanharris" called Blackman-Harris window is defined by [Har78b]:

$$win(m,n) = \begin{cases} 0.35875 - 0.48829\cos\left(\frac{\pi\sqrt{m^2+n^2}}{N}\right) & \\ +0.14128\cos\left(\frac{2\pi\sqrt{m^2+n^2}}{N}\right) - 0.01168\cos\left(\frac{3\pi\sqrt{m^2+n^2}}{N}\right) & \text{if } \sqrt{m^2+n^2} \leq N, \\ 0 & \text{else.} \end{cases} \tag{D.36}$$

D.10 Explanation of equation 5.1

From equation 3.3,

$$d = \frac{d'}{f} \cdot (u - BFL)$$

Keeping d' and f constant, and to different values for d (d_1 and d_2) and u (u_1 and u_2), the following two equations are obtained:

$$d_1 = \frac{d'}{f} \cdot (u_1 - BFL) \tag{D.37a}$$

$$d_2 = \frac{d'}{f} \cdot (u_2 - BFL) \tag{D.37b}$$

$$D.37b - D.37a \rightarrow d_2 - d_1 = \frac{d'}{f} \cdot (u_2 - u_1) \tag{D.38}$$

Noting $d_2 - d_1 = \Delta d$ and $u_2 - u_1 = \Delta u$, the equation 5.1 is obtained.

D.11 Explanation of equation 5.4

First, β_x and β_y. In figure 5.11, the red letters are for better orientation. The triangles ABC and ABD are both right at the angle B and the points B, C and D are aligned. So, $x.p$ = CD = BD - BC, BC = AB tan(α) and BD = AB tan($\alpha + \beta$). This leads to:

$$x.p = AB \cdot (\tan(\alpha + \beta) - \tan(\alpha))$$

In addition, AB = $D\cos(\alpha)$, so the following is obtained:

$$x.p = D\cos(\alpha) \cdot (\tan(\alpha + \beta) - \tan(\alpha))$$

Now β needs to be isolated and equation D.41 is obtained.

$$\begin{aligned} x.p &= D\cos(\alpha) \cdot (\tan(\alpha + \beta) - \tan(\alpha)) \\ \Leftrightarrow \tan(\alpha + \beta) &= \frac{x.p}{D\cos(\alpha)} + \tan(\alpha) \\ \Rightarrow \beta &= \arctan\left(\frac{x.p}{D\cos(\alpha)} + \tan(\alpha)\right) - \alpha \end{aligned} \tag{D.39}$$

For β_y it is the same but α equals 0 and D is replaced by the distance AD. Now AD can be written as $\frac{\text{AB}}{\cos(\alpha+\beta)} = \frac{D\cos(\alpha)}{\cos(\alpha+\beta)}$. So the second equation of D.41 is obtained.

$$\begin{aligned} \beta_y &= \arctan\left(\frac{x.p}{\text{AB}\cos(0)} + \tan(0)\right) - 0 \\ \Leftrightarrow \beta_y &= \arctan\left(\frac{x.p \cdot D\cos(\alpha)}{\cos(\alpha + \beta)}\right) \end{aligned} \tag{D.40}$$

So,

$$\begin{aligned} \beta_x &= \arctan\left(\frac{x \cdot p}{D\cos(\alpha)} + \tan(\alpha)\right) - \alpha \\ \beta_y &= \arctan\left(\frac{y \cdot p\cos(\alpha + \beta_x)}{D\cos(\alpha)}\right) \end{aligned} \tag{D.41}$$

With p the size of a projected pixel in mm and D the distance shown in figure 5.11 in mm. Now the angle difference depending on Δu, that will be called γ, needs to be found. For this, the problem is once again split in two axis.

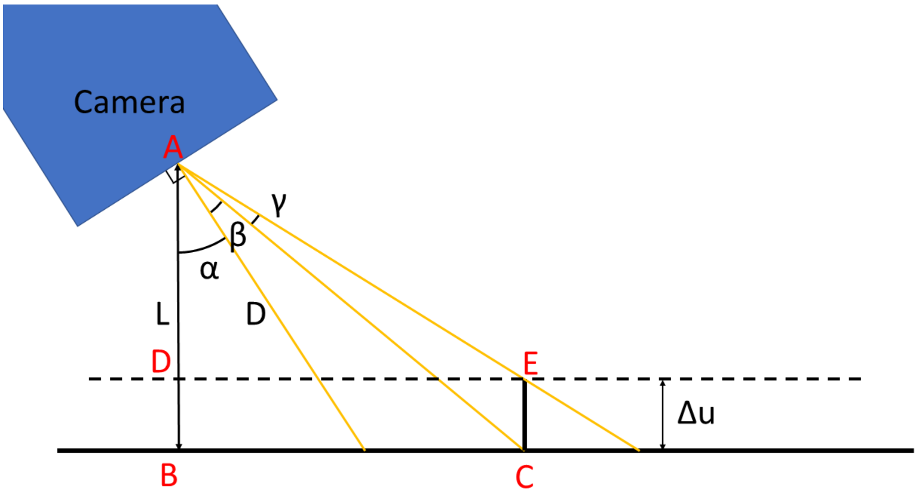


Figure D.3: More accurate scheme of the shifting problem.

From figure D.3 can be seen that DE = BC. Also, DE $= (L - \Delta u)\tan(\alpha + \beta + \gamma)$ and BC $= L\tan(\alpha + \beta)$, so:

$$\begin{aligned}(L - \Delta u)\tan(\alpha + \beta + \gamma) &= L\tan(\alpha + \beta)\\ \Leftrightarrow \tan(\alpha + \beta + \gamma) &= \frac{L}{L + \Delta u}\tan(\alpha)\\ \Rightarrow \gamma &= \arctan\left(\frac{L}{L + \Delta u}\tan(\alpha)\right) - (\alpha + \beta)\end{aligned} \tag{D.42}$$

As $L = D\cos(\alpha)$, the equation for γ_x is obtained.

For γ_y it is the same principle as for finding β_y. $\alpha = 0$ and D is replaced by $D \rightarrow \frac{D\cos(\alpha)}{\cos(\alpha+\beta_x+\gamma_x)}$. Thus,

$$\gamma_y = \arctan\left(\frac{\frac{D\cos(\alpha)}{\cos(\alpha+\beta_x+\gamma_x)}\cos(\beta_y)}{\frac{D\cos(\alpha)}{\cos(\alpha+\beta_x+\gamma_x)}\cos(\beta_y) + \Delta u}\tan(\beta_y)\right) - \beta_y \tag{D.43}$$

The results are shown in equation D.44.

$$\begin{aligned}\gamma_x &= \arctan\left(\frac{D\cos(\alpha + \beta_x)}{D\cos(\alpha + \beta_x) + \Delta u}\tan(\alpha + \beta_x)\right) - (\alpha + \beta_x)\\ \gamma_y &= \arctan\left(\frac{\frac{D\cos(\alpha)}{\cos(\alpha+\beta_x+\gamma_x)}\cos(\beta_y)}{\frac{D\cos(\alpha)}{\cos(\alpha+\beta_x+\gamma_x)}\cos(\beta_y) + \Delta u}\tan(\beta_y)\right) - \beta_y\end{aligned} \tag{D.44}$$

Finally, the shift on both axis is found using equation 5.3 with $\alpha + \beta_x + \gamma_x$

and $\beta_y + \gamma_y$ instead of α to get respectively Δx and Δy. After simplifying:

$$\begin{aligned}\Delta x &= \Delta u \cdot \frac{D\cos(\alpha+\beta_x)}{D\cos(\alpha+\beta_x)+\Delta u}\left(\frac{x\cdot p}{D\cos(\alpha)}+\tan(\alpha)\right)\\ \Delta y &= \Delta u \cdot \frac{\frac{y\cdot p\cdot\cos(\beta_y)\cdot\cos(\alpha+\beta_x)}{\cos(\alpha+\beta_x+\gamma_x)}}{\frac{D\cos(\alpha)}{\cos(\alpha+\beta_x+\gamma_x)}\cos(\beta_y)+\Delta u}\end{aligned} \tag{D.45}$$

Appendix E

Sensitivities

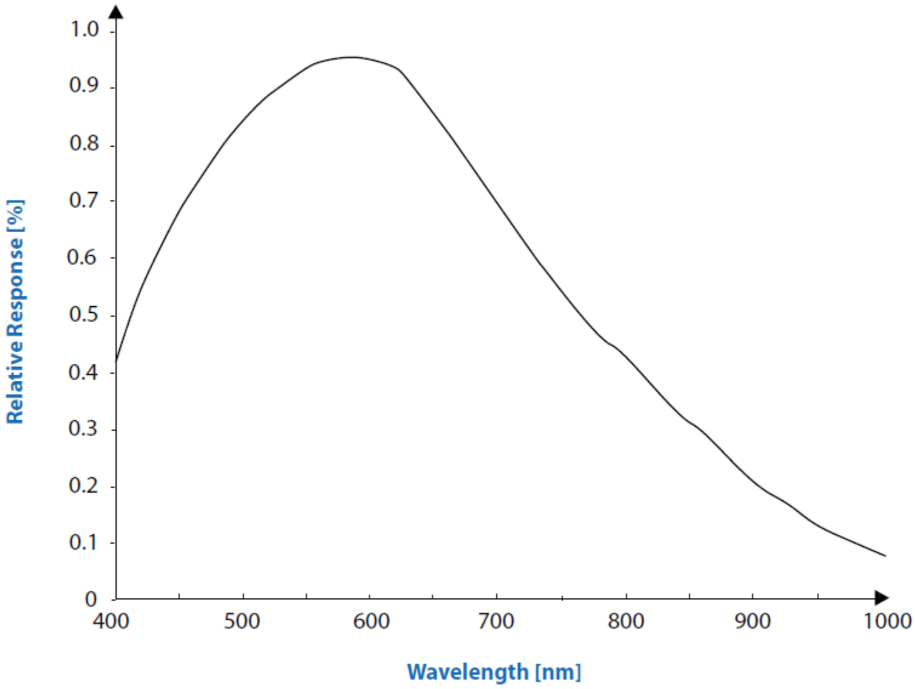


Figure E.1: Spectral sensitivity diagram of the ICX445ALA monochrome sensor in the 1MP monochrom camera - courtesy of Sony Deutschland GmbH.

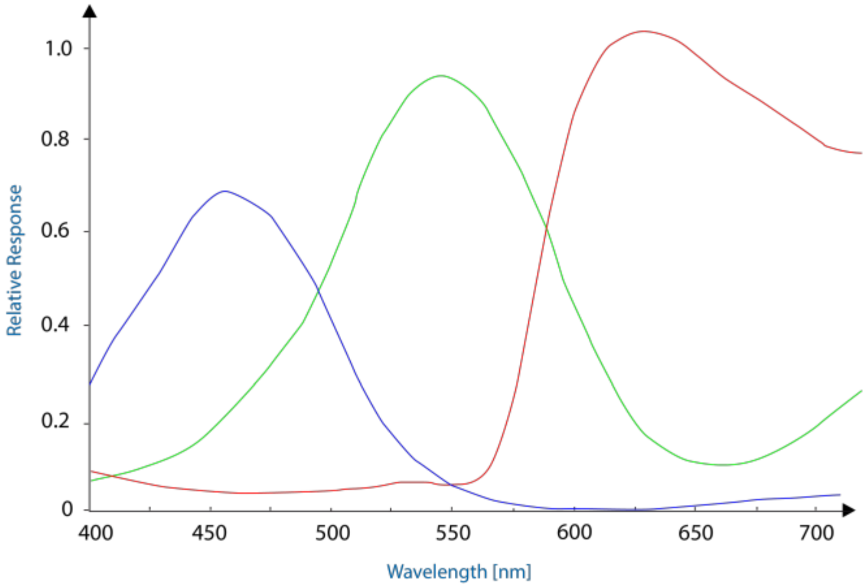


Figure E.2: Spectral sensitivity diagram of the ICX445AQA color sensor in the 1MP color camera - courtesy of Sony Deutschland GmbH.

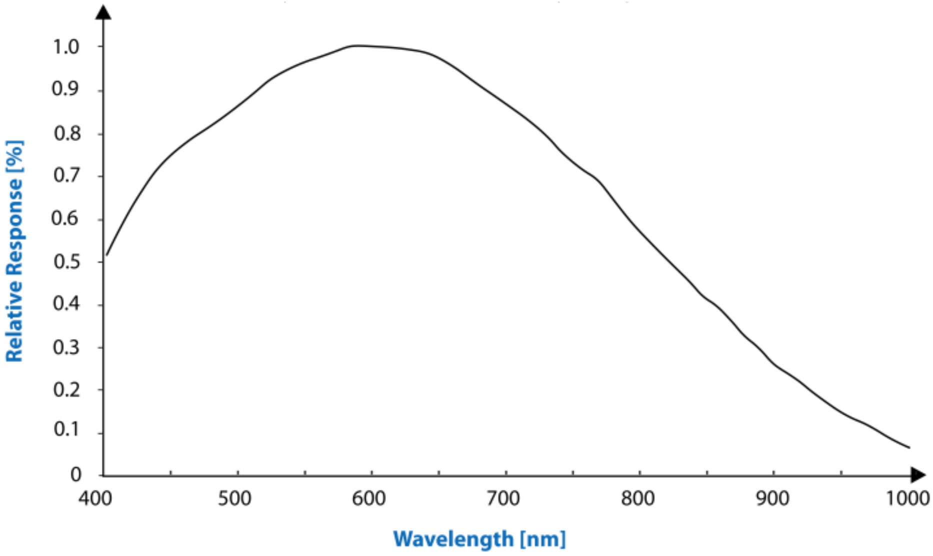


Figure E.3: Spectral sensitivity diagram of the IMX264LLR-C monochrome sensor in the 5MP monochrome camera - courtesy of Sony Deutschland GmbH.

Appendix F

Thresholds

F.1 Threshold 1

	disc1	disc2	disc3	disc4	disc5	Average
GLCM						
[128. 128]						
cluster	0.09	0.1	0.1	0.1	0.1	**0.098**
correlation	bad	bad	0.08	0.08	0.08	**0.08**
entropy	bad	bad	0.02	0.02	0.018	**0.019333333**
max proba	0.04	0.05	0.04	0.04	0.05	**0.044**
unif nrj	bad	bad	0.01	0.01	bad	**0.01**
[64. 64]						
cluster	0.3	0.17	0.13	0.16	0.2	**0.192**
correlation	bad	bad	bad	bad	bad	**bad**
entropy	bad	bad	0.03	0.06	0.03	**0.04**
max proba	0.15	0.1	0.07	0.07	0.1	**0.098**
unif nrj	bad	bad	0.027	0.03	0.03	**0.029**
[32. 32]						
cluster	0.54	0.3	0.25	0.3	0.3	**0.338**
correlation	bad	bad	bad	bad	bad	**bad**
entropy	0.5	0.1	0.2	0.28	0.28	**0.272**
max proba	0.3	0.1	0.2	0.2	0.2	**0.2**
unif nrj	0.3	0.1	0.13	0.11	0.15	**0.158**
[16. 16]						
cluster	0.8	0.4	0.5	0.4	0.6	**0.54**
correlation	bad	bad	bad	bad	bad	**bad**
entropy	0.9	0.4	0.4	0.4	0.4	**0.5**
max proba	0.5	0.3	0.31	0.33	0.4	**0.368**
unif nrj	0.5	0.3	0.4	0.35	0.3	**0.37**

Table F.1: First threshold values using the GLCM algorithm on fives discs.

	disc1	disc2	disc3	disc4	disc5	Average
STATISTICAL						
[16. 16]						
mean	93	bad	22	30	40	**46.25**
var	3000	bad	bad	3000	bad	**bad**
median	110	bad	bad	30	50	**63.33333333**
std	bad	bad	bad	30	bad	**bad**
[32. 32]						
mean	60	bad	bad	30	40	**43.33333333**
var	4000	bad	bad	2000	bad	**bad**
median	100	bad	bad	30	50	**60**
std	bad	bad	bad	22	bad	**bad**
[64. 64]						
mean	44	bad	bad	22	50	**38.66666667**
var	bad	bad	bad	bad	3000	**3000**
median	65	bad	bad	23	50	**46**
std	bad	bad	bad	bad	30	**30**
[128. 128]						
mean	bad	bad	bad	bad	40	**bad**
var	bad	bad	bad	bad	3000	**bad**
median	bad	bad	bad	20	40	**30**
std	bad	bad	bad	bad	30	**bad**

Table F.2: First threshold values using the statistical algorithm on fives discs.

	disc1	disc2	disc3	disc4	disc5	Average
FFT						
[32. 32]						
boxcar	5	11	11	9	11	**9.4**
triang	2	4.2	5	4.2	4.2	**3.92**
hamming	2	4.5	5	4.5	4.6	**4.12**
bartlett	2	4.2	4.6	4.2	4.5	**3.9**
[64. 64]						
boxcar	18	17	18	17	18	**17.6**
triang	7	7	7	7	8	**7.2**
hamming	8	8	8	8	8	**8**
bartlett	7	7	7	7	7	**7**
[128. 128]						
boxcar	31	31	32	28	32	**30.8**
triang	13	13	13	12	13	**12.8**
hamming	14	14	14	12	14	**13.6**
bartlett	13	13	13	12	13	**12.8**
[16. 16]						
boxcar	6	6	6	6	6.3	**6.06**
triang	2.5	2.5	2.5	2.5	2.6	**2.52**
hamming	2.8	2.8	2.8	2.8	2.8	**2.8**
bartlett	2.5	2.5	2.5	2.5	2.45	**2.49**

Table F.3: First threshold values using the FFT algorithm on fives discs.

	disc1	disc2	disc3	disc4	disc5	Average
GLCM template						
cluster	0.4	0.4	0.4	0.4	0.4	**0.4**
correlation	bad	bad	bad	2	2.5	**2.25**
entropy	0.8	0.8	0.8	0.8	0.8	**0.8**
max proba	0.4	0.4	0.4	0.4	0.4	**0.4**
unif nrj	0.4	0.4	0.4	0.4	0.4	**0.4**

Table F.4: First threshold values using the GLCM with template algorithm on fives discs.

F.2 Threshold 2

	disc1	disc2	disc3	disc4	disc5	disc6	Average
GLCM							
[64. 64]							
cluster	0.2	0.2	bad	0.1	0.2	0.2	**0.16**
correlation	bad	bad	bad	bad	bad	bad	**bad**
entropy	bad	bad	bad	bad	bad	0.1	**0.1**
max proba	0.1	0.07	bad	0.05	0.1	0.1	**0.084**
unif nrj	0.1	bad	bad	bad	bad	0.3	**0.2**
[32. 32]							
cluster	0.2	0.2	0.2	0.2	0.2	0.2	**0.2**
correlation	bad	bad	bad	bad	bad	**bad**	
entropy	0.2	bad	bad	bad	0.2	0.2	**0.2**
max proba	0.2	0.1	0.08	0.1	0.2	0.2	**0.147**
unif nrj	0.2	bad	bad	bad	0.15	0.1	**0.15**
[16. 16]							
cluster	0.8	0.3	0.2	0.4	0.8	0.3	**0.47**
correlation	bad	bad	bad	bad	bad	**bad**	
entropy	0.5	bad	0.5	bad	0.4	0.3	**0.425**
max proba	0.2	0.1	0.2	0.2	0.2	0.2	**0.18**
unif nrj	0.3	bad	0.2	0.1	0.2	0.2	**0.2**

Table F.5: Redefined threshold values using the GLCM algorithm on the six discs presented in section 5.2.

	disc1	disc2	disc3	disc4	disc5	disc6	Average
FFT							
[32. 32]							
boxcar	11	11	11	10	10	10	**10.5**
triang	5	4.4	bad	4.5	4.5	4.5	**4.6**
hamming	5	5	bad	4.2	5	5	**4.8**
bartlett	5	4.6	bad	4.5	4.5	4.5	**4.6**
[64. 64]							
boxcar	18	19	18	18	18	18	**18.2**
triang	8	7.5	7.8	7.5	7.5	8	**7.7**
hamming	8.6	8	8.4	8	8	8	**8.2**
bartlett	8	7.5	7.7	7	7.5	7	**7.5**
[128. 128]							
boxcar	32	32	33	32	31	32	**32**
triang	13	12.8	12.9	12.5	13	13	**12.9**
hamming	14	13.9	14	13.5	14	14	**13.9**
bartlett	13	12.8	12.9	13	13	13	**13**
[16. 16]							
boxcar	7	6.4	bad	bad	6	6	**6.3**
triang	2.5	2.8	bad	bad	2.5	2.5	**2.58**
hamming	2.8	2.9	bad	bad	2.8	2.8	**2.83**
bartlett	2.5	2.6	bad	bad	2.5	2.5	**2.53**

Table F.6: Redefined threshold values using the FFT algorithm on the six discs presented in section 5.2.

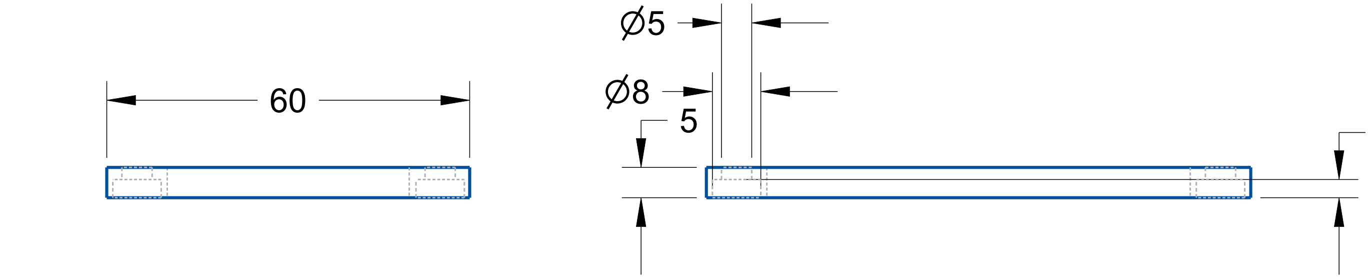
	disc1	disc2	disc3	disc4	disc5	disc6	Average
GLCM template							
cluster	0.5	0.6	0.6	0.5	0.5	0.4	**0.52**
correlation	bad	bad	bad	bad	bad	bad	**bad**
entropy	0.8	0.8	0.8	0.8	0.8	0.8	**0.8**
max proba	0.4	0.5	0.5	0.4	0.4	0.4	**0.43**
unif nrj	0.5	0.55	0.5	0.5	0.5	0.4	**0.49**

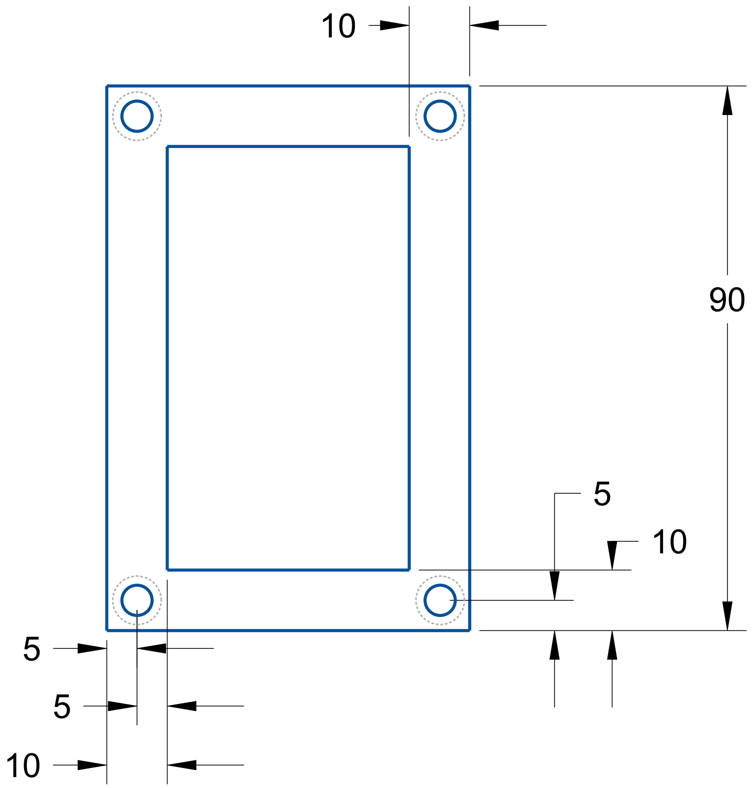
Table F.7: Redefined threshold values using the GLCM with template algorithm on the six discs presented in section 5.2.

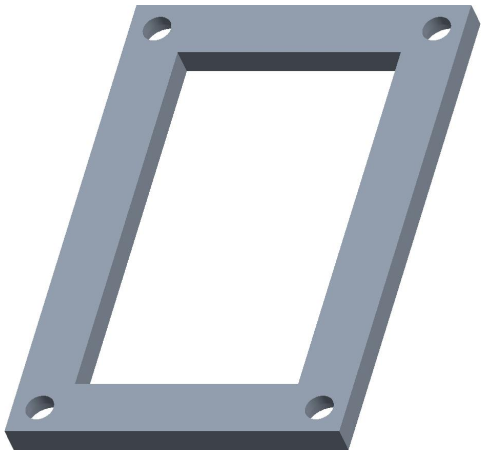
Appendix G

Drawings

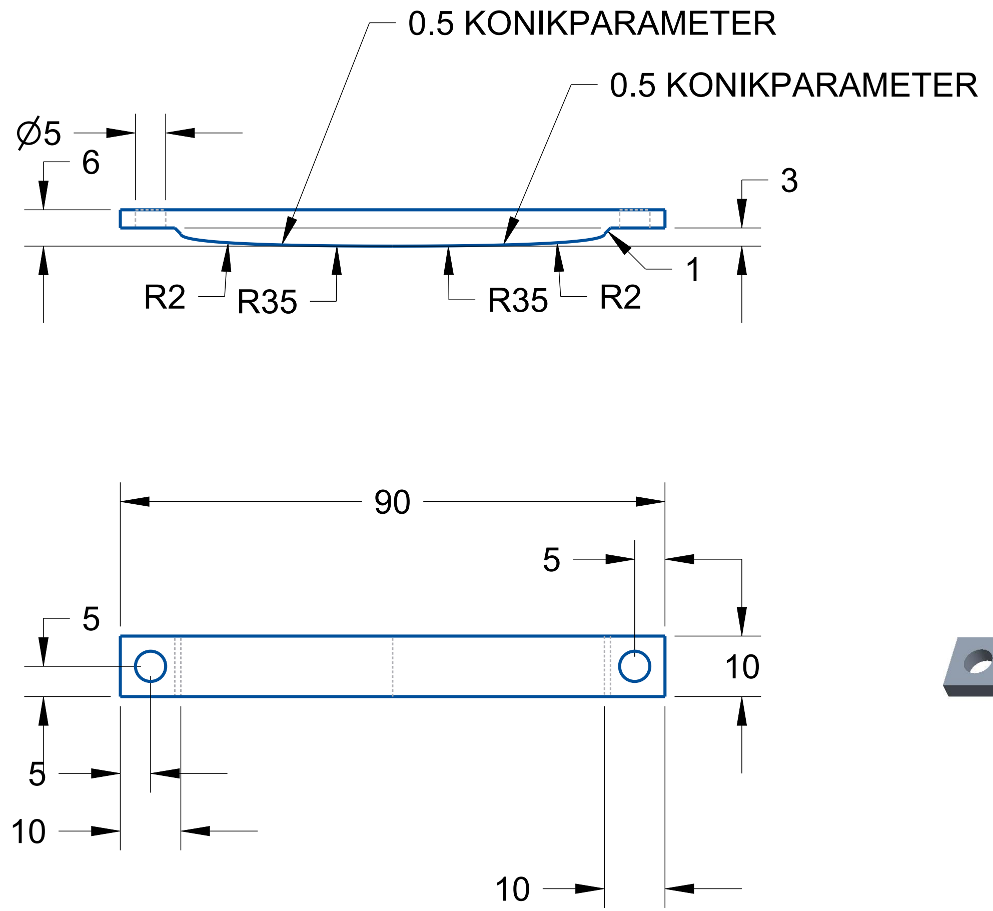
The CAD files can be found in the file sent along with the report under the respective names "bpp__bracket.prt", "bpp__bracket2.prt" and "bpp.prt".



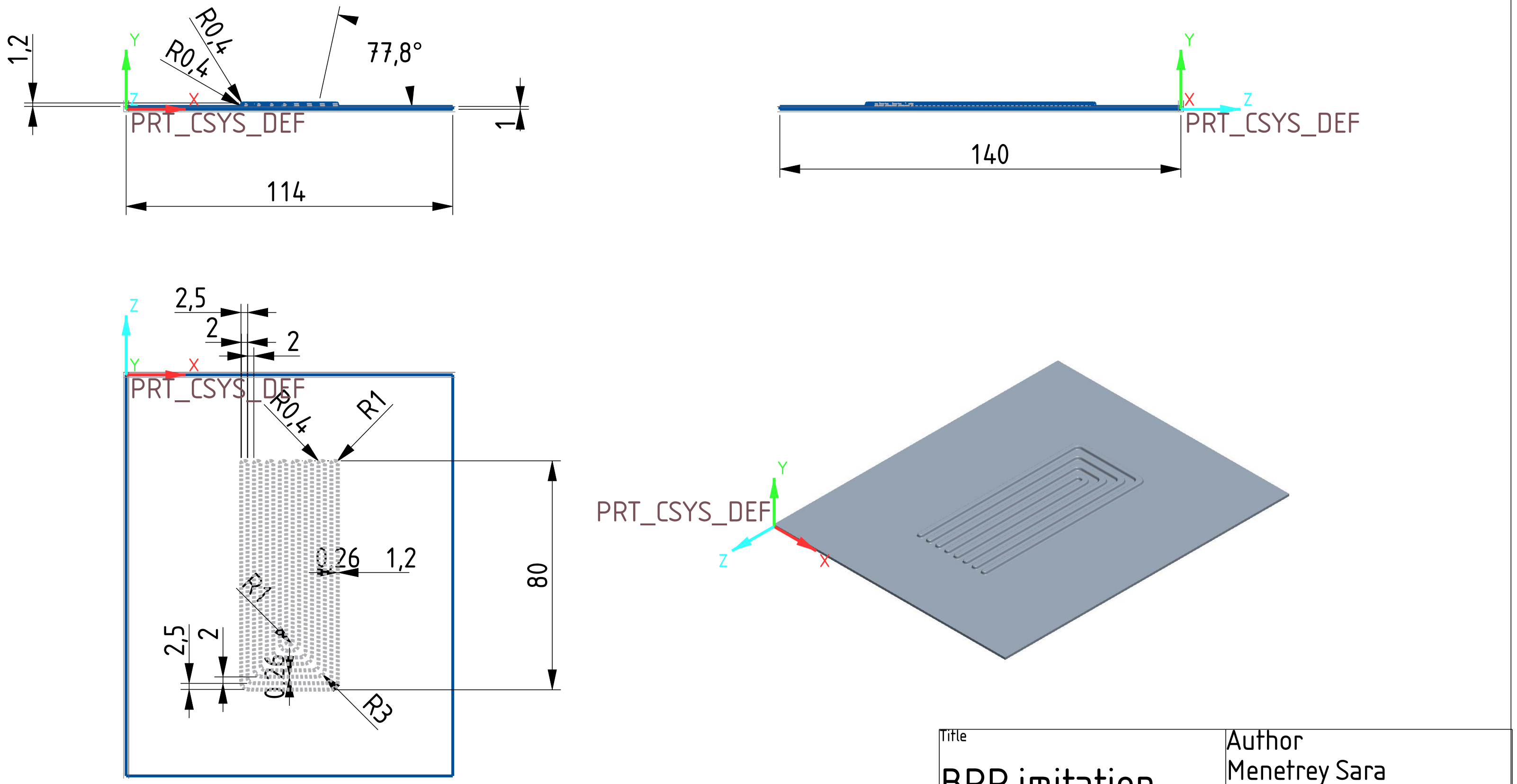




Title Bracket 1		Author Menetrey Sara 3D printed	black tough PLA
Scale 1:1	Size	Simplified drawing (no tolerances)	Sheet 1/2



Title		Author Menetrey Sara	
Bracket 2		3D printed	Black Tough PLA
Scale	Size		Sheet
1:1		Simplified drawing (no tolerances)	2/2



Title		Author	
BPP imitation		Menetrey Sara	
		3D printed	White PLA
Scale	Size		Sheet
1:2		Simplified drawing (no tolerances)	1/1

Appendix H

Code

Algorithm	Library	Functions
HDR	OpenCV cv2	- createMergeMertens(), - process()
Camera lens bias	OpenCV cv2	- findChessboardCorners(), - calibrateCamera(), - getOptimalNewCameraMatrix(), - undistort()
Perspective	OpenCV cv2	- getPerspectiveTransform(), - warpPerspective()
Statistical	Numpy	- mean, - var, - median, - std, - skew and - kurto
Opening	OpenCV cv2	- getStructuringEelement(cv2.MORPH_RECT, k_{size}), - morphologyEx(img, cv2.MORPH_OPEN,kernel)
Threshold	OpenCV cv2	- adaptiveThreshold(cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY_INV)
GLCM	skimage.feature	- graycomatrix()
FFT	skimage.filters numpy.fft	- window() - fft2(), - fftshift()
GLCM template	numpy.fft	- fft2(), - fftshift()

Table H.1: Summary of the prefabricated functions used in the different algorithms

Camera to projector position.

```
# Name : reflection.py
# Author : Sara Menetrey
# Date : 10.08.2022
#comparison of the gray level distribution of pictures

import matplotlib as mpl
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import math
import glob

images =
    glob.glob(r"C:\Users\Sara\Documents\Fraunhofer\pictures\angle\reflection\1\*.bmp")
    #Download all file names

# Threshold the image "img" at the limit value "lim"
def threshold(img,lim):
    imgc = np.copy(img)
    imgc[imgc > lim] = 255
    imgc[imgc <= lim] = 0
    return imgc

# Plot the original images
titles = [’11 deg’,’9 deg’,’6 deg’,
          ’3 deg’,’0 deg’,"-3 deg",
          ’-5 deg’,’-11 deg’,"-12 deg"]
fig = plt.figure(figsize=(11,9))
for i in range(0, len(images)):
    img = mpimg.imread(images[i])
    plt.subplot(3,3,i+1),plt.imshow(img, cmap=mpl.cm.gray)
    plt.title(titles[i])
plt.savefig(r"C:\Users\Sara\Documents\Fraunhofer\pictures\angle\reflection\1\init.PNG")
    # save figure

# Plot the images thresholded at 254 (from 255)
# only the point with a grayscale of 255 stay at 255, the other
    points are set to 0
fig = plt.figure(figsize=(11,9))
for i in range(0, len(images)):
    img = mpimg.imread(images[i])
    thres = threshold(img, 254)
    plt.subplot(3,3,i+1),plt.imshow(thres, cmap=mpl.cm.gray)
    plt.title(titles[i])
plt.savefig(r"C:\Users\Sara\Documents\Fraunhofer\pictures\angle\reflection\1\res.PNG")
    #save figure
```

HDR. Source: https://docs.opencv.org/4.x/d2/df0/tutorial_py_hdr.html

```
# Name : HDR.py
# Source: https://docs.opencv.org/4.x/d2/df0/tutorial_py_hdr.html
```

```
import cv2 as cv
import numpy as np

# Loading exposure images into a list
img_fn =
    [r"C:\Users\Sara\Documents\Fraunhofer\pictures\hdr\3_4_10.bmp".r"C:\Users\Sara\Documents\Fra
    r"C:\Users\Sara\Documents\Fraunhofer\pictures\hdr\3_4_1000.bmp"]
img_list = [cv.imread(fn) for fn in img_fn]
exposure_times = np.array([1/10, 1/120, 1/1000], dtype=np.float32)

# Merge exposures to HDR image
merge_debevec = cv.createMergeDebevec()
hdr_debevec = merge_debevec.process(img_list,
    times=exposure_times.copy())
merge_robertson = cv.createMergeRobertson()
hdr_robertson = merge_robertson.process(img_list,
    times=exposure_times.copy())

# Tonemap HDR image
tonemap1 = cv.createTonemap(gamma=2.2)
res_debevec = tonemap1.process(hdr_debevec.copy())
res_robertson = tonemap1.process(hdr_robertson.copy())

# Exposure fusion using Mertens
merge_mertens = cv.createMergeMertens()
res_mertens = merge_mertens.process(img_list)

# Convert datatype to 8-bit and save
res_debevec_8bit = np.clip(res_debevec*255, 0, 255).astype('uint8')
res_robertson_8bit = np.clip(res_robertson*255, 0,
    255).astype('uint8')
res_mertens_8bit = np.clip(res_mertens*255, 0, 255).astype('uint8')
cv.imwrite(r"C:\Users\Sara\Documents\Fraunhofer\pictures\hdr\3_4ldr_debevec2.bmp",
    res_debevec_8bit)
cv.imwrite(r"C:\Users\Sara\Documents\Fraunhofer\pictures\hdr\3_4ldr_robertson2.bmp",
    res_robertson_8bit)
cv.imwrite(r"C:\Users\Sara\Documents\Fraunhofer\pictures\hdr\3_4fusion_mertens2.bmp",
    res_mertens_8bit)
```

Distortion correction. Source: https://opencv24-python-tutorials.readthedocs.io/en/stable/py_tutorials/py_calib3d/py_calibration/py_calibration.html

```
# Name : calibration.py
# Source:
    https://opencv24-python-tutorials.readthedocs.io/en/stable/py_tutorials/py_calib3d/py_calibr

import numpy as np
import cv2
import glob
import matplotlib as mpl
import matplotlib.pyplot as plt

# termination criteria
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30,
    0.001)

# prepare object points, like (0,0,0), (1,0,0), (2,0,0) ....,(6,5,0)
objp = np.zeros((6*6,3), np.float32)
objp[:,:2] = np.mgrid[0:6,0:6].T.reshape(-1,2)

# Arrays to store object points and image points from all the images.
objpoints = [] # 3d point in real world space
imgpoints = [] # 2d points in image plane.

images =
    glob.glob(r"C:\Users\Sara\Documents\Fraunhofer\pictures\chessboard\t6x6\*.bmp")

for fname in images:
    img = cv2.imread(fname)

    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    print("done")
    # Find the chess board corners
    ret, corners = cv2.findChessboardCorners(gray, (6,6),None)

    # If found, add object points, image points (after refining them)
    if ret == True:
        print("chessboard found")
        objpoints.append(objp)

        corners2 =
            cv2.cornerSubPix(gray,corners,(11,11),(-1,-1),criteria)
        imgpoints.append(corners2)

        # Draw and display the corners
        img = cv2.drawChessboardCorners(img, (6,6), corners2,ret)
        cv2.imshow('img',img)
        cv2.waitKey(500)

cv2.destroyAllWindows()

ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints,
    imgpoints, gray.shape[::-1],None,None)
print(mtx)
print(dist)
img =
    cv2.imread(r"C:\Users\Sara\Documents\Fraunhofer\pictures\chessboard\t6x6\t.bmp")
h,  w = img.shape[:2]
newcameramtx,
    roi=cv2.getOptimalNewCameraMatrix(mtx,dist,(w,h),1,(w,h))

# undistort
```

```
dst = cv2.undistort(img, mtx, dist, None, newcameramtx)

# crop the image
x,y,w,h = roi
dst = dst[y:y+h, x:x+w]
plt.imshow(dst,cmap = mpl.cm.gray)
plt.show()
```

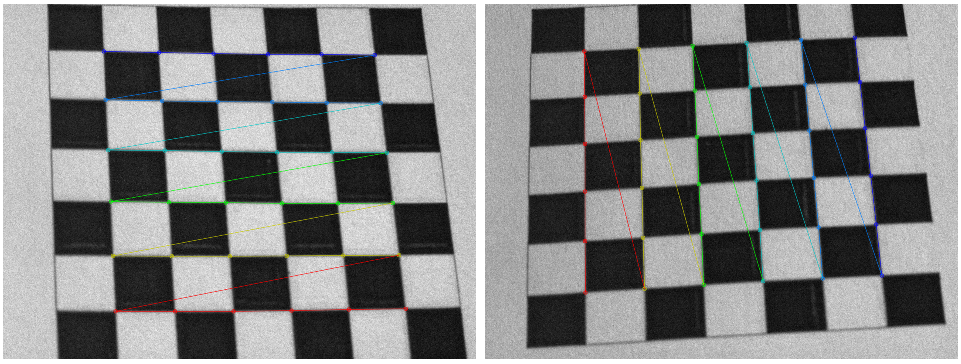


Figure H.1: Examples of the used chessboard pictures and the detected corners.

Perspective correction.

```
# img : image to be corrected
# start : array of four coordinates of points in img
# stop : array of four coordinates of the same points in the
    corrected picture

def perspective(img, start, stop, inv = False):
    (h, w) = img.shape[:2]
    center = (w / 2, h / 2)
    M = cv.getPerspectiveTransform(start, stop)
    if inv == True:
        M = np.linalg.inv(M)
    perspectived = cv.warpPerspective(img, M, (w,h))
    return perspectived
```

Final program.

```
# Name : Solution.py
# Author : Sara Menetrey
# Date : 10.08.2022

import cv2 as cv
import matplotlib as mpl
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import math
from skimage.feature import graycomatrix, greycoprops
from skimage import io, color, img_as_ubyte
from scipy import signal
from skimage.filters import window
```

```
# Perspective correction
# img : image to be corrected
# start : array of four coordinates of points in img
# stop : array of four coordinates of the same points in the
    corrected picture
# inv : boolean, if true, start and stop are inverted
def perspective(img, start, stop, inv = False):
    (h, w) = img.shape[:2]
    center = (w / 2, h / 2)
    M = cv.getPerspectiveTransform(start, stop)
    if inv == True:
        M = np.linalg.inv(M)
    perspectived = cv.warpPerspective(img, M, (w,h))
    return perspectived

def correctPerspective(img, inv=False):
     # perspective : begin values are measured on the undistorted
         image so that it follows the projected pattern and end is
         chosen in order to form a rectangle
    # 1 MP camera
    begin =
        np.array([[108,28],[25,711],[1226,788],[1232,105]],np.float32)
        # coordination read on a picture taken without part
    end =
        np.array([[25,28],[25,788],[1232,788],[1232,28]],np.float32)
         # coordination determined for the points above to form a
        rectangle
    # 5 MP camera
    #begin =
        np.array([[281,1850],[2276,1248],[1839,83],[141,609]],np.float32)
    #end =
        np.array([[141,1850],[2276,1850],[2276,83],[141,83]],np.float32)
    if inv :
        perspect = perspective(img,end, begin)
    else:
        perspect = perspective(img,begin, end)
    return perspect

# Morphological opening
# img : image to be opened
# kSize : (int,int), size of the kernel
def opening(img, kSize):
    kernel = cv.getStructuringElement(cv.MORPH_RECT, kSize)
    opening = cv.morphologyEx(img, cv.MORPH_OPEN, kernel)
    return opening

# Distortion correction
# img : image to undistort
#using the previously generated "mtx" and "dist" with the camera
    calibration code
```

```
def undistort(img):
    # Undistort : mtx and dist are given by calibration.py
    h,  w = img.shape[:2]
    # 1 MP camera
    mtx = np.array([[ 5.21118799e+03, 0.00000000e+00 ,
        6.99298702e+02],
                  [ 0.00000000e+00, 5.04923313e+03, -3.46575305e+01],
                  [ 0.00000000e+00, 0.00000000e+00, 1.00000000e+00]])
    dist = np. array([[-2.17548569e-01, -1.09851687e+01,
        5.67463939e-02, 2.02377062e-03,
                  1.12005259e+02]])
    # 5 MP camera
    #mtx = np.array([[4.76098633e+03, 0.00000000e+00,
        1.11830765e+03],
    #               [0.00000000e+00, 5.18668524e+03, 9.42253265e+02],
    #               [0.00000000e+00, 0.00000000e+00, 1.00000000e+00]])
    #dist = np. array([[ 9.08388442e-01, -2.66155902e+01,
        -1.10955365e-02, 4.90286863e-02, 2.07598091e+02]])
    newcameramtx,
        roi=cv.getOptimalNewCameraMatrix(mtx,dist,(w,h),1,(w,h))
    dst = cv.undistort(img, mtx, dist, None, newcameramtx)
    return dst

# Opening and Adaptive threshold
def threshold(img):
    # opening : kernel size chosen arbitrarily for the pattern to be
        a little more defined
    opened = opening(img, (10,10))
    # Mean threshold inverted
    thres =
        cv.adaptiveThreshold(opened,255,cv.ADAPTIVE_THRESH_MEAN_C,cv.THRESH_BINARY_INV,11,2)
    return thres


####
################################### GLCM
    ############################################
####

# img : block of the image
# mode : string, name of the used feature
# d : int, shift used in the GLCM computation
def GLCM(img, mode,d=1):
    (h, w) = img.shape[:2]
    angles = [0, np.pi/4, np.pi/2, 3*np.pi/4]                # all
        angles taken into account for the GLCM
    glcm = graycomatrix((img/255).astype(int),[d],angles,2,
        symmetric=False)  # Computes the GLCM
    glcmfeat = []                               # list of the
        feature values for every angle
    for i in range(len(angles)):
        glcmAngle = glcm[:,:,0,i]                         # GLCM for
            angle number i
        glcmAnorm = glcmAngle/np.sum(glcmAngle)              #
            normalized GLCM for angle number i
        if mode == "Cluster":
            glcmfeat.append(cluster(glcmAnorm))
        if mode == "Correlation":
            glcmfeat.append(correlation(glcmAnorm))
        if mode == "Entropy":
            glcmfeat.append(entropy(glcmAnorm))
        if mode == "Maximum probability":
            glcmfeat.append(maxProba(glcmAnorm))
        if mode == "Uniformity of Energy":
            glcmfeat.append(unifNRJ(glcmAnorm))
        if mode == "Contrast":
            glcmfeat.append(contrast(glcmAnorm))
        if mode == "Dissimilarity":
            glcmfeat.append(dissimilarity(glcmAnorm))
        if mode == "invDiffMoment":
            glcmfeat.append(invDiffMoment(glcmAnorm))
    return np.mean(glcmfeat)                           # returns the
        mean value of the angles feature values

## Features

# Mean
def mu(img):
    return np.mean(img)

# Cluster tendency
def cluster(glcm):
    Clu = 0
    m = mu(glcm)
    for i in range(len(glcm)):
        for j in range(len(glcm[0])):
            Clu += (i+j-2*m)**2*glcm[i,j]
    return Clu

# Contrast
def contrast(glcm):
    Con = 0
    for i in range(len(glcm)):
        for j in range(len(glcm[0])):
            Con += np.abs(i-j)**2*glcm[i,j]
    return Con

def mu1(glcm):
    m1 = 0
    for i in range(len(glcm)):
        P = 0
        for j in range(len(glcm[0])):
            P += glcm[i,j]
        m1 += P*i
    return m1

def mu2(glcm):
    m2 = 0
    for j in range(len(glcm[0])):
        P = 0
        for i in range(len(glcm)):
            P += glcm[i,j]
        m2 += P*j
    return m2

def sigma1(glcm):
    s1 = 0
    for i in range(len(glcm)):
        P = 0
        for j in range(len(glcm[0])):
            P += glcm[i,j]
        s1 += P*(i-mu1(glcm))**2
    return s1

def sigma2(glcm):
    s2 = 0
    for j in range(len(glcm[0])):
        P = 0
        for i in range(len(glcm)):
            P += glcm[i,j]
        s2 += P*(j-mu2(glcm))**2
    return s2

# Correlation
def correlation(glcm):
    P = 0
    for i in range(len(glcm)):
        for j in range(len(glcm[0])):
            P += i*j*glcm[i,j]
    if sigma1(glcm) == 0 or sigma2(glcm)==0:
        Cor = 0
    else:
        Cor = (P -mu1(glcm)*mu2(glcm))/(sigma1(glcm)*sigma2(glcm))
    return Cor

# Dissimilarity
def dissimilarity(glcm):
    D = 0
    for i in range(len(glcm)):
        for j in range(len(glcm[0])):
            D += glcm[i,j]/(1+np.abs(i-j))
    return D

# Entropy
def entropy(glcm):
    E = 0
    for i in range(len(glcm)):
        for j in range(len(glcm[0])):
            if glcm[i,j] != 0:
                E += glcm[i,j] * np.log(glcm[i,j])
    return -E

# Inverse difference moment
def invDiffMoment(glcm,k=3):
    I = 0
    for i in range(len(glcm)):
        for j in range(len(glcm[0])):
            if i != j:
                I += glcm[i,j]/(np.abs(i-j)**k)
    return I

# Maximum probability
def maxProba(glcm):
    return np.max(glcm)

# Uniformity of energy
def unifNRJ(glcm):
    U = 0
    for i in range(len(glcm)):
        for j in range(len(glcm[0])):
            U += glcm[i,j]**2
    return U

# Comparison of the GLCM feature values of img1 and img2 using the
    mode
# img1 : reference block of image
# img2 : block of image to inspect
# mode: string, name of the feature
def compare(img1, img2, mode):
    feat1 = GLCM(img1, mode)
    feat2 = GLCM(img2, mode)
    return np.abs(feat1-feat2)      # comparison done by computing
        the absolute value of the difference


#Blockwise comparison of img1 with img2 of the mode "mode" with
    blocksizes of "blocksize" starting from the start coordinates
    and with an overlap of "overlap"
# img1 : reference image
# img2 : image to inspect
# blocksize : [int, int], size of the blocks to be compared
# mode : string, name of the used feature
# start : [int, int], coordinates of the starting point
# overlap: int, block overlap as explained in the algorithm section
def compare2img(img1, img2, blocksize,mode, start =[0,0], overlap =
    1):
    xstart, ystart = start[0], start[1]
    H, W = img1.shape                  # dimensions of the images
    h, w = blocksize                 # dimensions of the block
    resu = np.zeros((H,W))             # resulting feature matrix
    for i in np.arange(xstart, H-h+1, int(h/overlap)):
        for j in np.arange(ystart,W-w+1,int(w/overlap)):
            D = compare((img1[i:i+h, j:j+w]).astype(int),
                (img2[i:i+h, j:j+w]).astype(int), mode)  # block
                comparison
            resu[i:i+int(h/overlap), j:j+int(w/overlap)] = D *
                np.ones((int(h/overlap),int(w/overlap))) # writing
                the result of the block comparison on the
                corresponding coordinates
    return resu

# Final function combining the feature matrix for the different
    modes using the predefined thresholds
def GLCMfinal(img1, img2):
    Blocksizes = np.array([[16,16]]) #,[16,16]])      # used
        blocksizes
    Features = np.array(["Cluster"])#,"Maximum probability"])  #
        used modes
    Thresholds = np.array([[0.47]])              # corresponding
        thresholds
 #   Thresholds = np.array([[0.2,0.148],[0.47,0.18]])
#    Thresholds =
    np.array([[0.098,0.0193,0.044,0.01],[0.192,0.04,0.098,0.029],[0.338,0.272,0.2,0.158],[0.54,0
 #   Thresholds =
     np.array([[0.09,0.15,0.04,0.046,0.03],[0.2,0.5,0.12,0.1,0.09],[0.38,0.6,0.36,0.3,0.27]])

    resu = np.zeros_like(img1).astype(np.float64)
    for n in range(len(Blocksizes)):
 #       fig = plt.figure(figsize=(25,18))                    ## to
     comment
 #       plt.title("Comparisons with blocksize "+str(Blocksizes[n]))
     ## to comment
        for k in range(len(Features)):
            res = compare2img(img1,img2, Blocksizes[n], Features[k])
 #           ax  = fig.add_subplot(2,3,k+1)                   ## to
     comment
 #           ax.imshow(res, cmap = mpl.cm.gray)               ## to
     comment
 #           ax.set_title("Feature: "+Features[k])            ## to
     comment
            lim = Thresholds[n,k]
            imgc = res
            imgc[imgc > lim] = 255
            imgc[imgc <= lim] = 0
            resu += imgc
    resu = resu/(len(Blocksizes)*len(Features))   # rearanging the
        resulting matrix on the scale of 0 to 255
    return resu
```

```
####
#################################### STATS
    #############################################
####

# img1 : reference image
# img2 : image to inspect
# blocks : [int,int], dimensions of the block for the blockwise
    comparison
# mode : string, used mode
# overlap : int, block overlap as explained in the algorithm section
def stats_compare(img1, img2, blocks, mode, overlap = 1):
    h, w = blocks                # block dimensions
    H, W = img1.shape               # image dimensions
    resu = np.zeros((H,W))          # resulting feature matrix
    for i in range(0, H-h,int(h/overlap)):
        for j in range(0,W-w,int(w/overlap)):
            block1 = img1[i:i+h,j:j+w]      # block from the
                reference image
            block2 = img2[i:i+h,j:j+w]      # block from the image
                to inspect
        # mean
            if mode == "mean":
                mean1 = np.mean(block1)
                mean2 = np.mean(block2)
                D = np.abs(mean1 - mean2)
        # variance
            if mode == "var":
                var1 = block1.var()
                var2 = block2.var()
                D = np.abs(var1 - var2)
        # median
            if mode == "median":
                median1 = np.median(block1)
                median2 = np.median(block2)
                D = np.abs(median1 - median2)
        # standard deviation
            if mode == "std":
                std1 = np.std(block1)
                std2 = np.std(block2)
                D = np.abs(std1 - std2)
        # skewness
            if mode == "skew":
                skew1 = stat.skew(block1)[0]
                skew2 = stat.skew(block2)[0]
                D = np.abs(skew1 - skew2)
        # kurtosis
            if mode == "kurto":
                kurto1 = stat.kurtosis(block1)[0]
                kurto2 = stat.kurtosis(block2)[0]
                D = np.abs(kurto1 - kurto2)
            resu[i:i+int(h/overlap),j:j+int(w/overlap)] = D +
                np.zeros((int(h/overlap),int(w/overlap))) # writing
                the result of the block comparison on the
                corresponding coordinates
    return resu


# Final function combining the feature matrix for the different
    modes using the predefined thresholds
def Statsfinal(img1, img2):
    blocksizes = np.array([(16,16),(32,32),(64,64)])     # used
        blocksizes
    modes = np.array(["mean", "median"])         # used modes
    Thresholds = np.array([[46.25,63.33],[43.33,60],[38.66,46]]) #
        corresponding thresholds
#    Thresholds =
    np.array([[12,100,12,1000],[9,1000,9,1000],[4,1000,4,1000],[100,1000,1000,1000]])
    resu = np.zeros_like(img1).astype(np.float64)
    for n in range(len(blocksizes)):
        fig = plt.figure(figsize=(25,18))                     ## to
            comment
        plt.title("Comparisons with blocksize "+str(blocksizes[n]))
            ## to comment
        for k in range(len(modes)):
            stats = stats_compare(img1, img2, blocksizes[n], modes[k])
            ax  = fig.add_subplot(2,2,k+1)                   ## to
                comment
            ax.imshow(stats, cmap = mpl.cm.gray)             ## to
                comment
            ax.set_title("Mode: "+modes[k])                  ## to
                comment
            lim = Thresholds[n,k]
            imgc = stats
            imgc[imgc > lim] = 255
            imgc[imgc <= lim] = 0
            resu += imgc
    resu = resu/(len(blocksizes)*len(modes))  # rearanging the
        resulting matrix on the scale of 0 to 255
    return resu


####
################################### GLCM TEMP
    ############################################
####

# Comparison of the image to inspect with the created template using
    the mode "mode" from the starting point "start"
# img : image to inspect
# mode : string, name of the used feature
# start : [int,int], coordinates of the starting point
```

```
def compareAll(img, mode, start = [0,0]):
    temp = createTemp(img)              # template
    xstart, ystart = start[0], start[1]           # starting point
    h, w = temp.shape                   # block dimensions
    resu = np.zeros((len(img),len(img[0])))      # resulting feature
        matrix
    for i in np.arange(xstart, len(img)-h+1, h):
        for j in np.arange(ystart,len(img[0])-w+1,w):
            D = compare((img[i:i+h, j:j+w]).astype(int), temp, mode)
            for n in range(5):
                resu[i:i+h, j:j+w] = D * np.ones((h,w))    # writing
                    the result of the block comparison on the
                    corresponding coordinates
    return resu


# Creation of the template from the image to inspect
# img : image to inspect
def createTemp(img):
    patternW, patternH, lineW, lineH = FFT(img)     # dimensions of
        the grid pattern
    patternW, patternH, lineW, lineH = round(patternW),
        round(patternH), round(lineW)-1, round(lineH)-1 #
        transformation of the dimensions into integers
  #  print(patternW, patternH, lineW, lineH)
    # cell pattern : dimensions given by a fourier analysis of the
        image "thres" in fft.py
   # creation of the template matrix with this dimensions
    line = np.concatenate((np.zeros(patternW-lineW),np.ones(lineW)))
    temp1 = np.array([line for i in range(patternH-lineH)])
    temp2 = np.array([np.ones(patternW) for i in range(lineH)])
    temp = np.concatenate((temp1,temp2)).astype(int)
   # temp = np.concatenate((temp,temp)).astype(int)
    #temp = np.concatenate((temp,temp), axis=1)
  #  print(temp)
    return temp

# Fourier analyzes to determine the dimensions of the pattern
#img : image to inspect
def FFT(img):
    shape = img.shape[:2]

    fft_img = np.zeros_like(img,dtype=complex)
    fft_img = np.fft.fftshift(np.fft.fft2(img))  # shifted fast
        fourier transform of the image
    abs_fft_img = np.abs(fft_img)        # absolute of the transform

# Analysis on the image length
    f_s = len(abs_fft_img)          # sample rate (here the length of
        the image)
    indices = (-abs_fft_img[int(f_s/2),:]).argsort()  # returns the
        indices that would sort an array
    i = 1
    while np.abs(indices[0]-indices[i])<10:  # assures that the pic
        is not part of the mean pic anymore
        i +=1
    patternW =
        np.round(len(abs_fft_img[0])/np.abs(indices[i]-indices[0]))
         # grid pattern length

# Analysis on the image width
    f_s = len(abs_fft_img[0])         # sample rate (here the width
        of the image)
    indices = (-abs_fft_img[:,int(f_s/2)]).argsort()  # returns the
        indices that would sort an array
    i = 1
    while np.abs(indices[0]-indices[i])<10:  # assures that the pic
        is not part of the mean pic anymore
        i +=1
    patternH =
        np.round(len(abs_fft_img)/np.abs(indices[i]-indices[0]))   #
        grid pattern width

   # print("The pattern's dimensions W x H are = "+str(patternW)+" x
       "+str(patternH)+" in pixels")
   # print("Vertical stripes should be of thickness =
       "+str(20/55*patternW)+" pixels")
   # print("Horizontal stripes should be of thickness =
       "+str(20/55*patternH)+" pixels")
    return (patternW, patternH,20/55*patternW, 20/55*patternH)


# Final function combining the feature matrix for the different
    modes using the predefined thresholds
def GLCMtempfinal(img):
    start = [0,10]          # starting point
    features = np.array(["Cluster"])#,"Entropy","Maximum
        probability","Uniformity of Energy"])  # used features
    fig = plt.figure(figsize=(25,18))
    Thresholds = np.array([0.52])                             #
        corresponding thresholds
 #   Thresholds = np.array([0.52,0.8,0.43,0.49])
  #  Thresholds = np.array([0.6,-5,0.9,0.47,0.57])
    result = np.zeros_like(img).astype(np.float64)
    for k in range(len(features)):
        res = compareAll(img, features[k], start)
    #    ax  = fig.add_subplot(2,3,k+1)                     ## to
        comment
    #    ax.imshow(res, cmap = mpl.cm.gray)               ## to
        comment
    #    ax.set_title("Feature: "+features[k])                   ##
        to comment
        lim = Thresholds[k]
        imgc = res
        if lim <0:
            imgc[imgc > -lim] = 255
            imgc[imgc <= -lim] = 0
        else:
            imgc[imgc >= lim] = 255
            imgc[imgc < lim] = 0
            imgc = 255-imgc
        result += imgc
    result = result/(len(features))   # rearanging the resulting
        matrix on the scale of 0 to 255
    return result




####
################################## FFT
    ##########################################
####

#img1 : block of reference image
#img2 : block of image to inspect
#blocksize : [int,int], block dimensions
#filtertype : string, name of the used passing window
#overlap : int, block overlap as explained in the algorithm section
def fft_compare(img1, img2, blocksize, filtertype, overlap = 1):
    if img1.shape != img2.shape:
        return "error: the images are not the same shape"
    else:
        h, w = blocksize                       # block dimensions
        H, W = img1.shape                   # image dimensions
        resu = np.zeros((H,W))                 # resulting feature
            matrix
        img1, img2 = img1/float(2**8), img2/float(2**8)
        for i in range(0,H-h+1,int(h/overlap)):
            for j in range(0, W-w+1, int(w/overlap)):
                # defining the blocks and passing window
                wimg1 = img1[i:i+h, j:j+w] * window(filtertype, (h,
                    w))
                wimg2 = img2[i:i+h, j:j+w] * window(filtertype, (h,
                    w))
                # producing the fft of the blocks
                fft_img1 = np.zeros_like(wimg1,dtype=complex)
                fft_img1 = np.fft.fftshift(np.fft.fft2(wimg1))
                fft_img2 = np.zeros_like(wimg2,dtype=complex)
                fft_img2 = np.fft.fftshift(np.fft.fft2(wimg2))
                # comparing both blocks
                D = np.mean(abs(fft_img1-fft_img2))
                # insert the result of the comparison in the result
                    table
                resu[i:i+int(h/overlap), j:j+int(w/overlap)] = D +
                    np.zeros((int(h/overlap),int(w/overlap)))
```

```
       return resu

# Final function combining the feature matrix for the different
    modes using the predefined thresholds
#img1 : reference image
#img2 : image to inspect
def FFTfinal(img1,img2):
    blocksizes = np.array([(32,32)])#,(64,64),(128,128)])
          # used blocksizes
    filters = np.array([’boxcar’])#,’triang’,’hamming’,’bartlett’])
              # used passing windows
    Thresholds =
       np.array([[10.5]])#,4.6,4.8,4.6],[18.2,7.7,8.2,7.5],[32,12.9,13.9,13]])
       # corresponding thresholds
#   Thresholds =
    np.array([[10.5,4.7,5,4.6],[18.5,9,9.5,9],[36,15,16,15],[9.7,4.5,2.9,2.8]])
    resu = np.zeros_like(img1).astype(np.float64)
    for n in range(len(blocksizes)):
   #    fig = plt.figure(figsize=(25,18))                    ## to
      comment
   #    plt.title("Comparisons with blocksize "+str(blocksizes[n]))
      ## to comment
       for k in range(len(filters)):
           fft1 = fft_compare(img1, img2, blocksizes[n], filters[k])
   #        ax  = fig.add_subplot(2,2,k+1)                   ## to
      comment
   #        ax.imshow(fft1, cmap = mpl.cm.gray)              ## to
      comment
   #        ax.set_title("Filter: "+filters[k])              ## to
      comment
           lim = Thresholds[n,k]
           imgc = fft1
           imgc[imgc > lim] = 255
           imgc[imgc <= lim] = 0
           resu += imgc
    resu = resu/(len(blocksizes)*len(filters))   # rearanging the
       resulting matrix on the scale of 0 to 255
    return resu




def limit(img):
    points = []
    lim = 52
    h, w = img.shape
    for i in range(1,h-1):
       for j in range(1,w-1):
           if img[i,j]>lim and (img[i+1,j]<=lim or img[i+1,j+1]<=lim
               or img[i+1,j-1]<=lim or img[i,j+1]<=lim or
               img[i,j-1]<=lim or img[i-1,j+1]<=lim or
               img[i-1,j]<=lim or img[i-1,j-1]<=lim):
              points.append((j,i))
    return points


def thresh(imgc, lim):
    imgc[imgc > lim] = 255
    imgc[imgc <= lim] = 0









    #disc3 : [24,9,3]
    #disc2 : [22,12,5,1]
for i in [x for x in range(1,14) if (x != 4 and x!=5)]:
    for j in range(1,5):

# read reference image
       ref =
           mpimg.imread("C:\\Users\\Sara\\Documents\\Fraunhofer\\pictures\\strobe\\1MP\

       if len(ref.shape)>2:
           ref = ref[:,:,0]
# read image to inspect
       img2 =
           mpimg.imread("C:\\Users\\Sara\\Documents\\Fraunhofer\\pictures\\strobe\\1MP\
       if len(img2.shape)>2:
           img2= img2[:,:,0]
       print("done")

# Distortion correction
       dst1 = undistort(ref)
   dst2 = undistort(img2)
# Perspective correction
       perspect1 = correctPerspective(dst1)
   perspect2 = correctPerspective(dst2)
# Opening and adaptive threshold
       thres1 = threshold(perspect1)
   thres2 = threshold(perspect2)
       #plt.figure()
       #plt.imshow(thres1, cmap = mpl.cm.gray)



       #plt.figure()
       #plt.imshow(thres2, cmap =mpl.cm.gray)
       #print("done")


       starttime = time.time()
           #####################################################################################

       resu = GLCMfinal(thres1,thres2)
       print("glcm time = "+str((time.time()-starttime))+" s")
           #####################################################################################

     #  plt.figure()
     #  plt.imshow(resu, cmap=mpl.cm.gray)
     #  plt.figure()

# Undo perspective correction
       resu = correctPerspective(resu, True)
# Normal threshold at the middle value, so the defect was recognized
    on more than half of the feature matrices
       thresh(resu, 128)

# Defect displaying
# Source :
    https://answers.opencv.org/question/140096/change-colour-of-canny-output/
     # step 1: make a3 channel, bgr or rgb image from the canny
         output:
       edges = cv.Canny(resu.astype(np.uint8),100,150)
       rgb = cv.cvtColor(edges, cv.COLOR_GRAY2RGB) # RGB for
           matplotlib, BGR for imshow() !
       rgbimg = cv.cvtColor(img2, cv.COLOR_GRAY2RGB)
     # step 2: now all edges are white (255,255,255). To make it
         red, multiply with another array:
       rgb *= np.array((1,0,0),np.uint8) # set g and b to 0, leaves
           red
       rgb = cv.dilate(rgb,(3,3),iterations = 3)
    # step 3: compose:
       out = np.bitwise_or(rgbimg, rgb)
       plt.imshow(out)
       #for point in limit(resu):
        #plt.scatter(point[0],point[1],s = 1,color='r')
       plt.title("GLCM comparison after thresholding and combining")
       file =
           "C:\\Users\\Sara\\Documents\\Fraunhofer\\pictures\\strobe\\1MP\\test2s3\\glcmMin"+str
       plt.savefig(file)

       print("done")

    #resu = Statsfinal(perspect1,perspect2)
    ##plt.figure()
    ##plt.imshow(resu, cmap=mpl.cm.gray)
    #plt.figure()
```

```
#    starttime = time.time()
    #################################################################################

#    resu = GLCMtempfinal(thres2)
#    print("glcmtemp time = "+str((time.time()-starttime))+" s")
    #################################################################################

##   plt.figure()
##   plt.imshow(resu, cmap=mpl.cm.gray)
#    plt.figure()

#    resu = correctPerspective(resu, True)
#    thresh(resu, 128)


#    # step 1: make a3 channel, bgr or rgb image from the canny
    output:
#    edges = cv.Canny(resu.astype(np.uint8),100,150)
#    rgb = cv.cvtColor(edges, cv.COLOR_GRAY2RGB) # RGB for
    matplotlib, BGR for imshow() !
#    rgb = cv.dilate(rgb,(3,3),iterations = 3)
#    rgbimg = cv.cvtColor(img2, cv.COLOR_GRAY2RGB)
#    # step 2: now all edges are white (255,255,255). To make it
    red, multiply with another array:
#    rgb *= np.array((1,0,0),np.uint8) # set g and b to 0, leaves
    red :)
#    # step 3: compose:
#    out = np.bitwise_or(rgbimg, rgb)
#    plt.imshow(out)
#    #plt.title("GLCM comparison with temp after thresholding and
    combining")
#    #file =
    "C:\\Users\\Sara\\Documents\\Fraunhofer\\pictures\\strobe\\1MP\\disc2noOpen\\glcm
#    #plt.savefig(file)
#    plt.title("GLCM comparison after thresholding and combining")
#    file =
    "C:\\Users\\Sara\\Documents\\Fraunhofer\\pictures\\strobe\\1MP\\test2s1\\glcmTemp
#    plt.savefig(file)

#    print("done")
# resu = correctPerspective(resu, True)
# thresh(resu, 128)
# # step 1: make a3 channel, bgr or rgb image from the canny
    output:
# edges = cv.Canny(resu.astype(np.uint8),100,150)
# rgb = cv.cvtColor(edges, cv.COLOR_GRAY2RGB) # RGB for
    matplotlib, BGR for imshow() !
# rgb = cv.dilate(rgb,(3,3),iterations = 3)
# rgbimg = cv.cvtColor(img2, cv.COLOR_GRAY2RGB)
#  # step 2: now all edges are white (255,255,255). To make it
    red, multiply with another array:
```

```
# rgb *= np.array((1,0,0),np.uint8) # set g and b to 0, leaves
    red :)
# # step 3: compose:
# print(rgb.shape)
# print(rgbimg.shape)
# out = np.bitwise_or(rgbimg, rgb)
# plt.imshow(out)
# plt.title("Statistical comparison after thresholding and
    combining")
# file =
    "C:\\Users\\Sara\\Documents\\Fraunhofer\\pictures\\strobe\\1MP\\disc2noOpen\\mean"+str(i)+
# plt.savefig(file)

# print("done")



#    starttime = time.time()
    ##########################################################################################

#    resu = FFTfinal(thres1,thres2)
#    print("fft time = "+str((time.time()-starttime))+" s")
    ##########################################################################################

#   # plt.figure()
#   # plt.imshow(resu, cmap=mpl.cm.gray)
#    plt.figure()

#    resu = correctPerspective(resu, True)
#    thresh(resu, 128)


# #  step 1: make a3 channel, bgr or rgb image from the canny
    output:
#    edges = cv.Canny(resu.astype(np.uint8),100,150)
#    rgb = cv.cvtColor(edges, cv.COLOR_GRAY2RGB) # RGB for
    matplotlib, BGR for imshow() !
#    rgb = cv.dilate(rgb,(3,3),iterations = 3)
#    rgbimg = cv.cvtColor(img2, cv.COLOR_GRAY2RGB)
#  #  step 2: now all edges are white (255,255,255). To make it
    red, multiply with another array:
#    rgb *= np.array((1,0,0),np.uint8) # set g and b to 0, leaves
    red :)
#    # step 3: compose:
#  #   print(rgb.shape)
#  #   print(rgbimg.shape)
#    out = np.bitwise_or(rgbimg, rgb)
#    plt.imshow(out)
#    plt.title("Frequencial comparison after thresholding and
    combining")
#    file =
    "C:\\Users\\Sara\\Documents\\Fraunhofer\\pictures\\strobe\\1MP\\disc3min\\fft"+str(i)+".pn
    #    plt.savefig(file)

    # print("done "+str(i))

plt.show()
```

Shift depending on Δu.

```
# Name : shift.py
# Author : Sara Menetrey
# Date : 10.08.2022

# This programm aims at computing the shifts dx and dy of a given
    point of coordinates x,y in shape(img)
# when the the parts point is at a height difference of du from its
    ideal location and the camera angle to the horizontal is alpha.

#Important:
# In this case, x and y are pixel coordinates so the width recording
    by a pixel "p" (in mm) needs to be multiplied in order to have
    the coordinates.
# Here, "p" is considered constant for the entire image even though
    it is not true because it is a function of the pixel coordinates
    p --> p(x,y).
# For more precise results, replace (x*p,y*p) with (X,Y) the point
    coordinates in mm.

# du in mm
# alpha in \textdegree
# x integer
# y integer

import math as ma
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D

d = 100           # mm, distance from image middle point to crossing
    point of all rays
H,W = 1280, 960   # pixels, camera resolution
p = 0.056         # mm, pixel side

def shift(du,alpha,x,y):
    x = x-W/2
    y = y-H/2
    alpha = np.pi/180*alpha
    betax = np.arctan(x*p/(d*np.cos(alpha))+np.tan(alpha))-alpha
    betay = np.arctan(y*p*np.cos(alpha + betax)/(d*np.cos(alpha)))
    Betax = betax + alpha
    Gammax =
        np.arctan(d*np.cos(Betax)*np.tan(Betax)/(d*np.cos(Betax)+du))

    dx = du * d* np.cos(Betax)*np.tan(Betax)/(d*np.cos(Betax) + du)
    dy = du * (np.cos(betay)*y*p*np.cos(Betax))
```

```
/np.cos(Gammax)/(d*np.cos(alpha)*np.cos(betay)/np.cos(Gammax)+du)

    return dx, dy

H,W = 1280, 960
Pts = []
for i in range(0,W,50):
    for j in range(0,H,50):
        Pts.append([i,j])
X = np.zeros(len(Pts))
Y = np.zeros(len(Pts))
for n in range(len(Pts)):
    X[n] = Pts[n][0]
    Y[n] = Pts[n][1]
plt.scatter(X*p,Y*p, marker = 's', label="ideal position")
alpha = 27
du = 5
for du in range(1,6,2):
    shiftedPts = []
    for ele in Pts:
        x, y = ele[0], ele[1]
        dx, dy = shift(du, alpha, x, y)
        shiftedPts.append([dx+x*p, dy+y*p])
    sX = np.zeros(len(Pts))
    sY = np.zeros(len(Pts))
    for n in range(len(Pts)):
        sX[n] = shiftedPts[n][0]
        sY[n] = shiftedPts[n][1]
    plt.scatter(sX,sY, marker = 's',label="shifted, du = "+str(du))
plt.legend(loc='lower right')
plt.title("Ideal and shifted positions of points \n with p =
    "+str(p)+" mm, d = "+str(d)+" mm and alpha =
    "+str(alpha)+"\textdegree.")
plt.show()
```

Minimum Δu.

```
DU = np.zeros(len(X))
for n in range(len(X)):
    du = 0
    x,y = X[n], Y[n]
    dx, dy = shift(du,alpha,x,y)
    while abs(dx)<p and abs(dy)<p:
        du += 0.01
        dx, dy = shift(du,alpha,x,y)
    DU[n] = du
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(X*p,Y*p,DU)
ax.set_ylabel('Position in image width (in mm)')
ax.set_xlabel('Position in image length (in mm)')
ax.set_zlabel('Height (in mm)')
plt.title("Minimum detectable height difference for every position
    \n with p = "+str(p)+" mm, d = "+str(d)+" mm and alpha =
    "+str(alpha)+"\textdegree.")
plt.show()
```

First attempt to predict the 2D image from a 3D object. The goal of this program is to give the 2D coordinates of a 3D point knowing the location of the camera in relation to the part. For this, the equations 5.4 from section 5.3 can be reused. Assuming all the 3D coordinates of a part are known and saved as (x_1, y_1, z_1) and the final 2D cooridnates are called (x_2, y_2), then $z_1 = \Delta u$, $x_1 = x \cdot p$ and $y_1 = y \cdot p$:

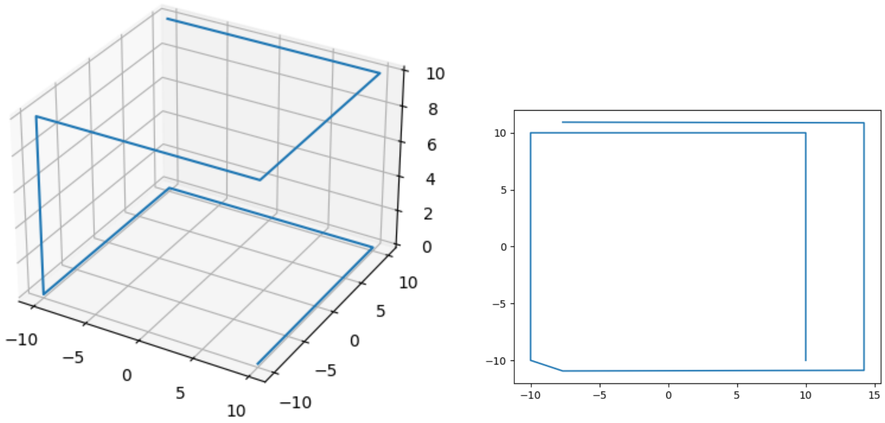
$$\begin{aligned} x_2 &= x_1 + \Delta x \\ y_2 &= y_1 + \Delta y \end{aligned} \tag{H.1}$$

$$\begin{aligned} x_2 &= x_1 + z_1 \cdot \frac{D\cos(\alpha+\beta_x)}{D\cos(\alpha+\beta_x)+z_1}\left(\frac{x_1}{D\cos(\alpha)} + \tan(\alpha)\right) \\ y_2 &= y_1 + z_1 \cdot y_1 \cdot \frac{\cos(\beta_y)}{\frac{D\cos(\alpha)}{\cos(\alpha+\beta_x+\gamma_x)}\cos(\beta_y)+z_1} \end{aligned} \tag{H.2}$$

Translating this formula into Python code, the following program and results are obtained. For the computed to be compared with the real image, the real image's distortion must be corrected.

```
# Name : 3Dto2D.py
# Author : Sara Menetrey
# Date : 10.08.2022
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

cube =
    np.array([[-10,10,10],[10,10,10],[10,-10,10],[-10,-10,10],[-10,-10,0],[-10,10,0],[10
    # coordinates in mm

x = cube[:,0]
y = cube[:,1]
z = cube[:,2]

plt.figure()
ax = plt.axes(projection='3d')
ax.plot3D(cube[:,0], cube[:,1], cube[:,2])

alpha = 20 # angle in degrees
alpha = alpha*np.pi/180 # angle in rad
D = 100 # object to lens distance in mm

betax = np.arctan(x/(D*np.cos(alpha))+np.tan(alpha))-alpha
betay = np.arctan(y*np.cos(alpha + betax)/(D*np.cos(alpha)))
Betax = betax + alpha
Gammax = np.arctan(D*np.cos(Betax)*np.tan(Betax)/(D*np.cos(Betax)+z))

L = D* np.cos(alpha)/np.cos(Gammax)
```

```
xp2 = x + z * D* np.cos(Betax)*np.tan(Betax)/(D*np.cos(Betax) + z)
yp2 = y + z * np.cos(betay)*y/(L*np.cos(betay) + z)

plt.figure()
plt.plot(xp2,yp2)
plt.show()
```



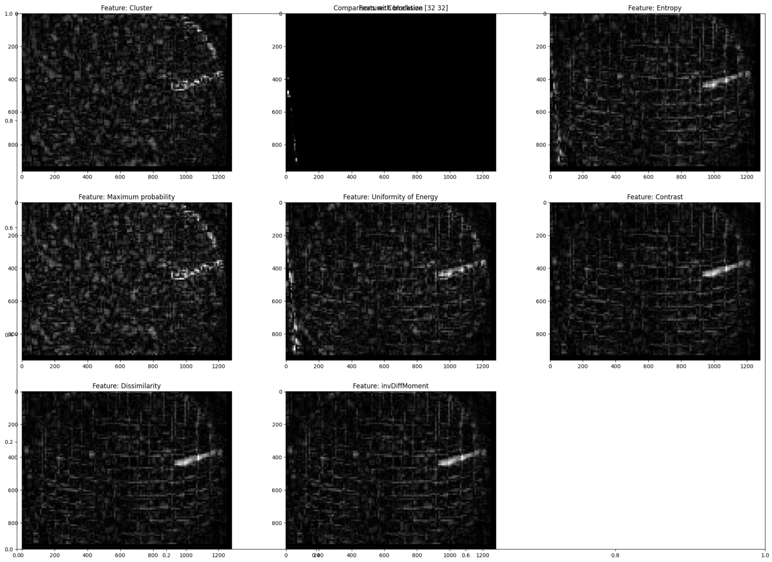
(a) : 3D representation of a rectangular cuboid.

(b) : 2D representation of the camera vision of the same cuboid.

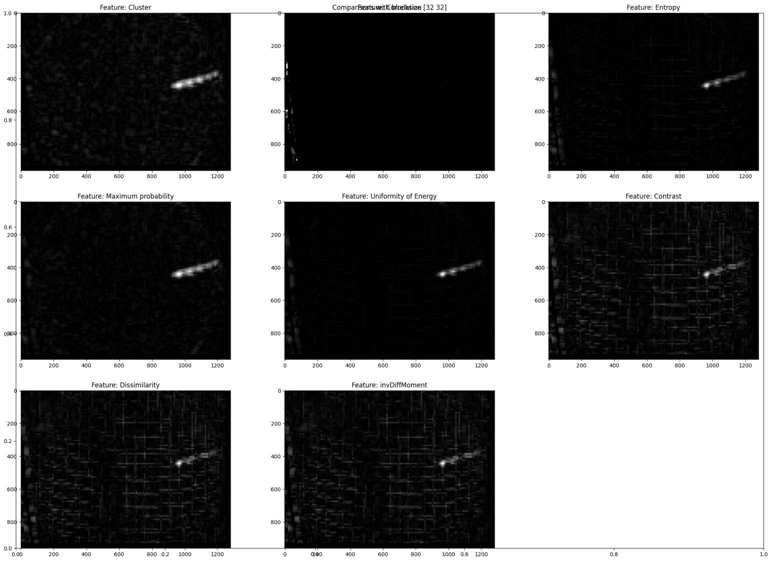
Figure H.2: Results from the Python program above.

Appendix I

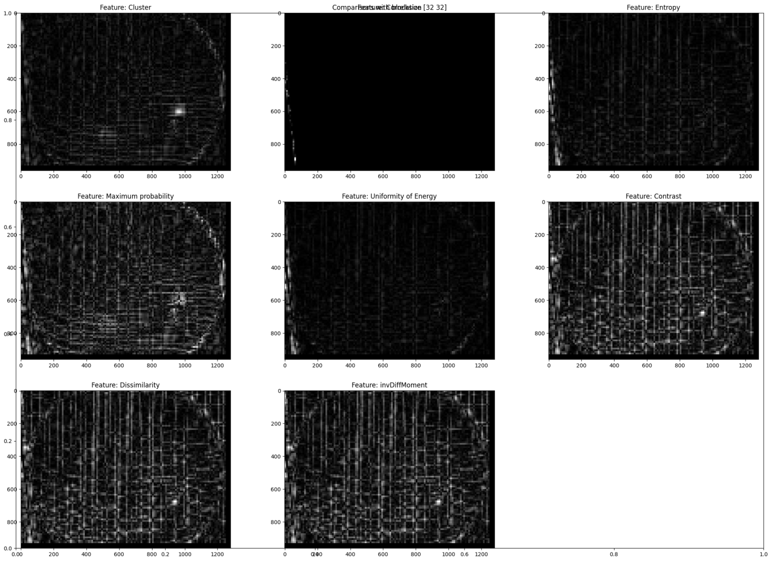
Images



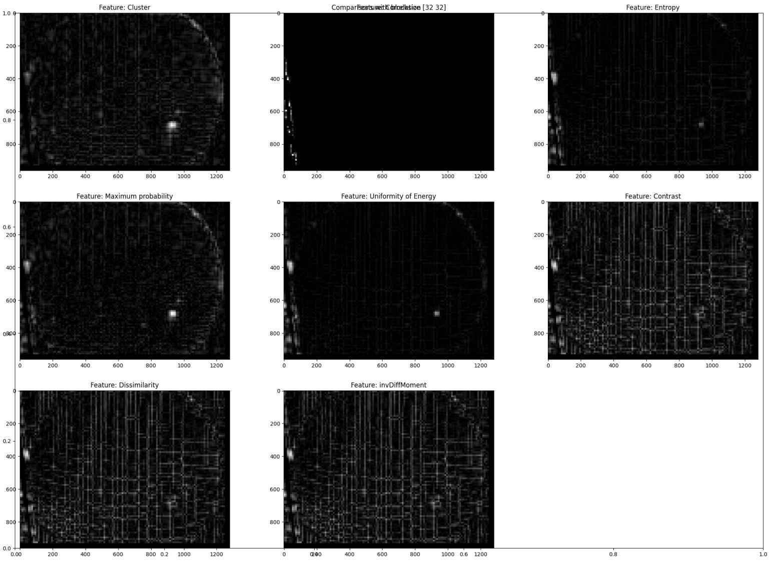
(a) : without opening



(b) : with opening

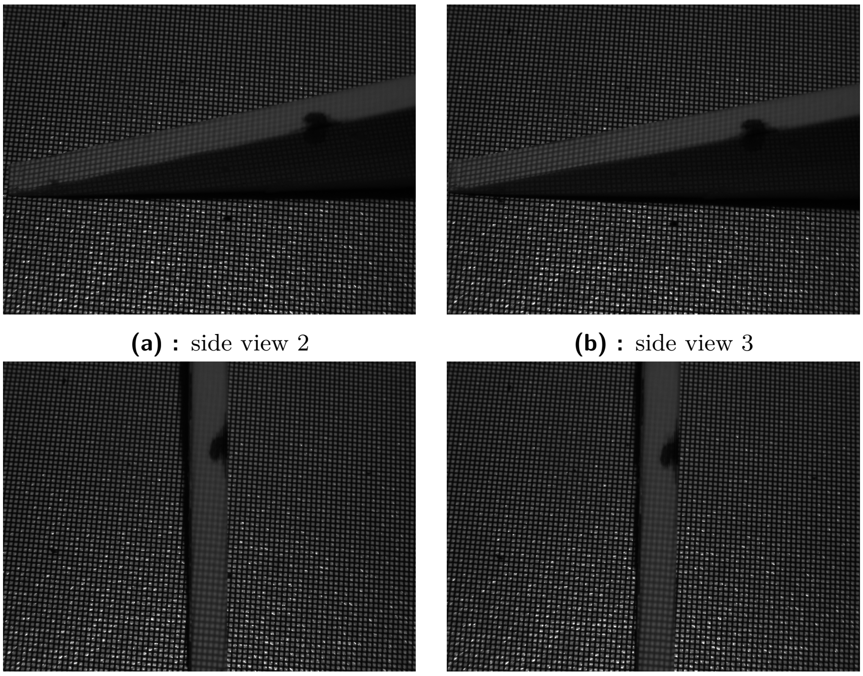


(c) : without opening



(d) : with opening

Figure I.1: Obtained matrices after applying the GLCM algorithm with and without opening.



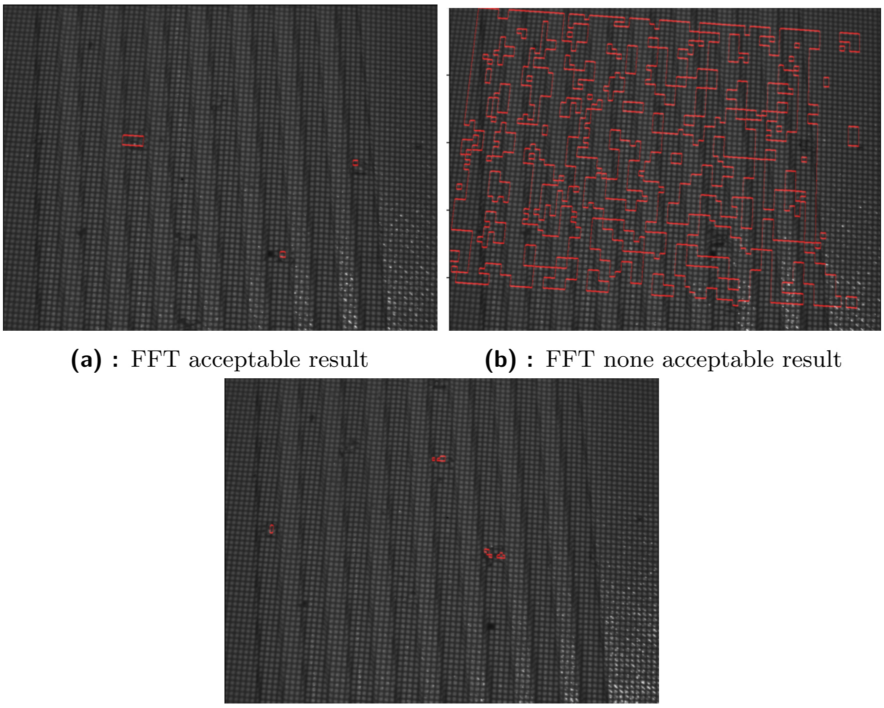
(a) : side view 2

(b) : side view 3

(c) : front view 2

(d) : front view 3

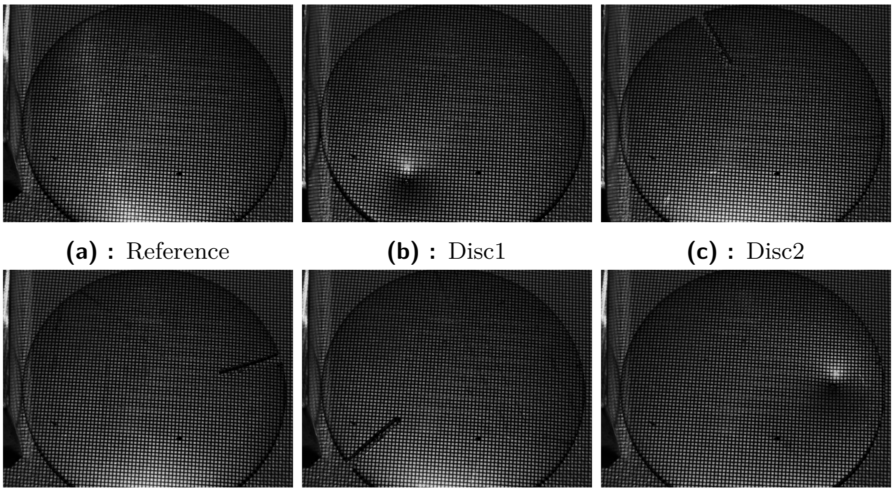
Figure I.2: Pictures to determine the DoF of the projector.



(a) : FFT acceptable result **(b) :** FFT none acceptable result

(c) : GLCM acceptable result

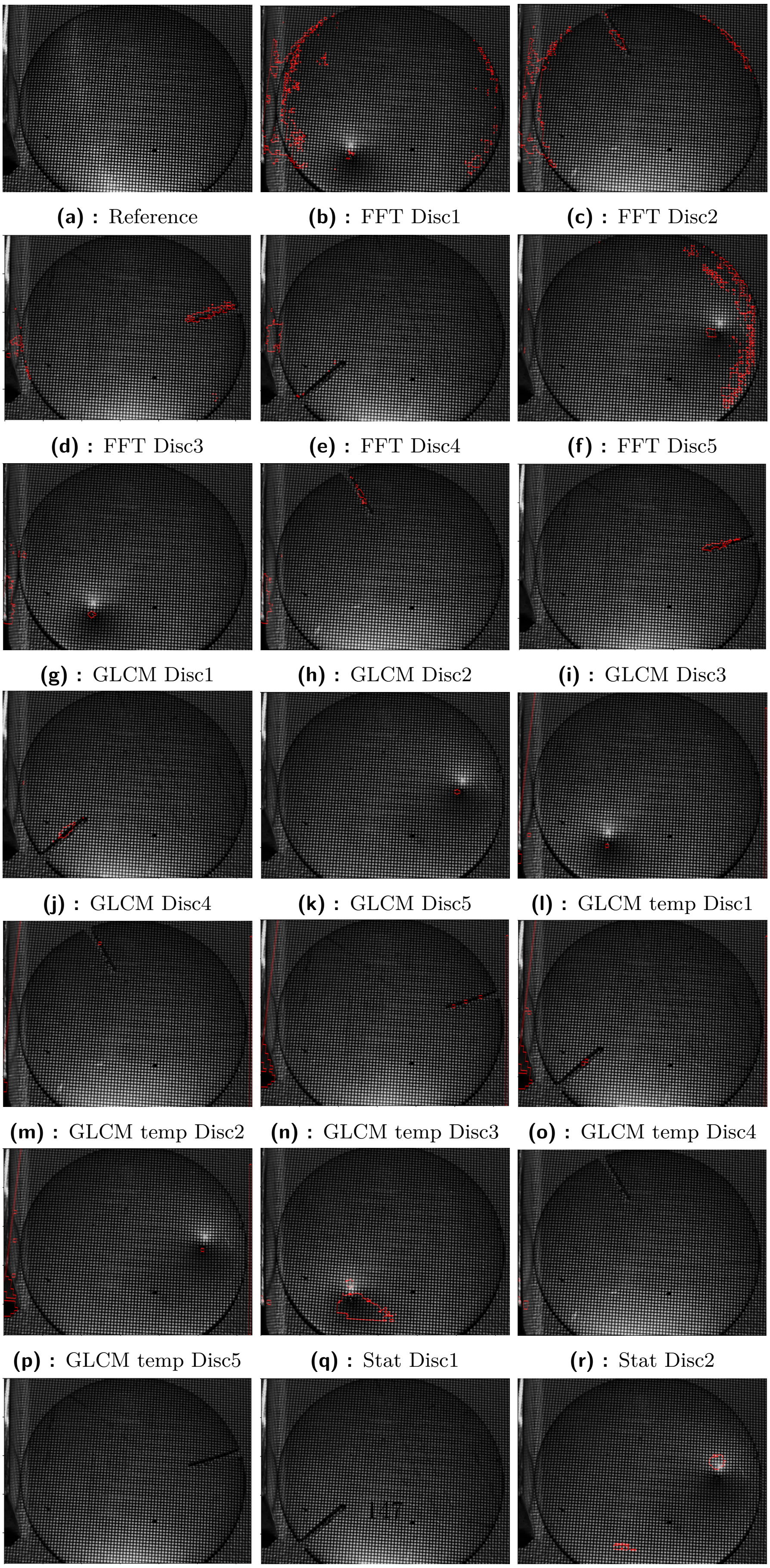
Figure I.3: Example of results with the 3D printed plates.



(a) : Reference **(b) :** Disc1 **(c) :** Disc2

(d) : Disc3 **(e) :** Disc4 **(f) :** Disc5

Figure I.4: Five disc pictures used to determine the first thresholds in test one 5.1.1.



(a) : Reference

(b) : FFT Disc1

(c) : FFT Disc2

(d) : FFT Disc3

(e) : FFT Disc4

(f) : FFT Disc5

(g) : GLCM Disc1

(h) : GLCM Disc2

(i) : GLCM Disc3

(j) : GLCM Disc4

(k) : GLCM Disc5

(l) : GLCM temp Disc1

(m) : GLCM temp Disc2

(n) : GLCM temp Disc3

(o) : GLCM temp Disc4

(p) : GLCM temp Disc5

(q) : Stat Disc1

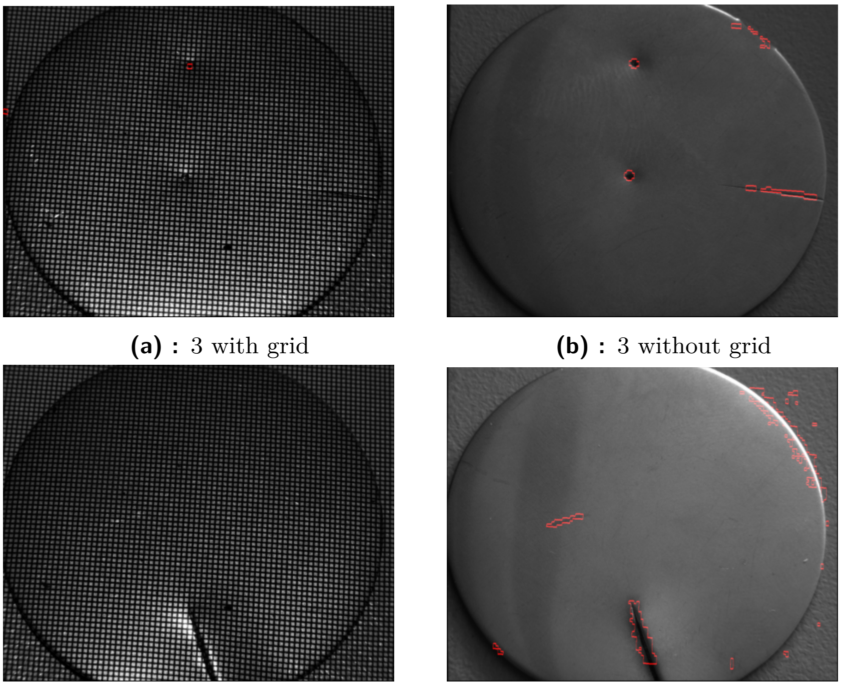
(r) : Stat Disc2

(s) : Stat Disc3

(t) : Stat Disc4

(u) : Stat Disc5

Figure I.5: Results for the five disc pictures of the first test 5.1.2.



(a) : 3 with grid

(b) : 3 without grid

(c) : 25 with grid

(d) : 25 without grid

Figure I.6: GLCM comparison results of sample number 3 and 25. Test conditions: "test1"

I.1 Value tables

Correction	GLCM	FFT	GLCM with template	Displaying
0.39	11	24.0878	5.158186	0.039
0.391	11.223	24.345	5.3148	0.067
0.39145	11.283	24.16	5.12688	0.041
0.39146	11.38676	24.2319	5.0944	0.037991
0.39146	11.3459	24.9711	5.15533	0.039
0.39145	11.2266	24.25324	5.083	0.039
0.39145	11.2711	24.0511	5.1137	0.04
0.39145	11.227	24.3692	5.137	0.039
0.39	11.234	24.2943	5.13126	0.04
0.39129	11.2564	24.2369	5.16	0.03898
0.39245	11.272	24.209	5.17	0.04
0.39145	11.206	24.02914	5.13176	0.039

Table I.1: Recording of the execution time of the different parts of the program for different images.

GLCM	FFT	GLCM with template
1.344	1.918	1.614
1.327	0.932	1.591
1.329	0.921	1.59
1.332	0.914	1.599
1.33	0.924	1.598
1.32	0.923	1.58
1.326	0.916	1.592
1.334	0.932	1.582
1.354	1.022	1.583

Table I.2: Recording of the execution time of the different parts of the program for different images with the optimized program.