CTU

CZECH TECHNICAL UNIVERSITY IN PRAGUE

F3

Faculty of Electrical Engineering
Department of Measurement

Master's thesis

Temperature compensation of microbolometric image sensor and implementation of autofocus function

Miroslav Tržil

May, 2022

Supervisor: Ing. Radek Sedláček, Ph.D

Field of study: Cybernetics and Robotics

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Tržil** Jméno: **Miroslav** Osobní číslo: **466110**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra měření**

Studijní program: **Kybernetika a robotika**

Studijní obor: **Kybernetika a robotika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Teplotní kompenzace mikrobolometrického obrazového snímače a implementace funkce automatického ostření

Název diplomové práce anglicky:

Temperature compensation of microbolometric image sensor and implementation of autofocus function

Pokyny pro vypracování:

1. Seznamte se s principem mikrobolometrického snímače firmy ULIS 640px.
2. Změřte teplotní závislost tohoto snímače na okolní teplotě s ohledem na vhodné nastavení dynamického rozsahu -15°C až 150 °C.
3. Na základě rešerše odborné literatury navrhněte vhodnou metodu pro teplotní kompenzaci snímače pomocí řízení napětí detektoru GSK a GFID pro správně zvolenou délku integrace a implementujte příslušný modul pro FPGA obvod v jazyce pro popis hardware (VHDL/Verilog).
4. Správnou funkci použité metody pro teplotní kompenzace snímače ověřte měřením v klimatické komoře pro rozsahu teplot 0 až 60 °C.
5. V jazyce pro popis hardware (VHDL/Verilog) navrhněte modul pro FPGA obvod pro vyhodnocení ostrosti obrazové informace ze snímače. Výstup tohoto modulu použijte pro zpětnovazební řízení motoru ostření (vytvoření funkce automatického ostření).

Seznam doporučené literatury:

[1] Torres, S. N.; Hayat, M. M.: Compensation for Nonuniformity and Drift in Infrared Focal-Plane Arrays: A Kalman-Filtering Approach. 2001. http://www.eece.unm.edu/faculty/hayat/Kalman.pdf
[2] Kang, C. G.: Computer Simulation of Compensation Method for Infrared Focal Plane Array. Journal Beijing Institute of Technology, Vol.9, No.3, P.330--334, Beijing 2000.
[3] Venkateswarlu, R., Er, M. H., Gan, Y. H., and Fong, Y. C., “Nonuniformity compensation for IR focal plane array sensors”, in Infrared Technology and Applications XXIII, 1997, vol. 3061, pp. 915–926. doi:10.1117/12.280310.
[4] W. Minkina, S. Dudzik, Infrared Thermography: Errors and Uncertainties, Wiley Online Library, 2009.
[5] Lane, B., Whitenton, E. P. : Calibration and Measurement Procedures for a High Magnification Thermal Camera, NIST, 2015, http://dx.doi.org/10.6028/NIST.IR.8098.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Radek Sedláček, Ph.D. katedra měření FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: _____ Termín odevzdání diplomové práce: **04.01.2022**

Platnost zadání diplomové práce:
do konce letního semestru 2022/2023

Ing. Radek Sedláček, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

Declaration

I declare that the presented work was developed independently and that I have listed all sources of the information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, May 20, 2022

. . . . . .
Miroslav Tržil

Acknowledgement

I would like to express my appreciation to my supervisor Ing. Radek Sedláček, Ph.D for his guidance and valuable feedback. I could not have undertaken this journey without my colleagues at Workswell s.r.o, who contributed with their expertise and provided equipment used in this thesis. I am also grateful to all academic workers from the Czech Technical University in Prague, who have a significant impact on my education. Last but not least I would like to thank my family, especially my parents and partner, for their support throughout my studies.

Abstrakt

Diplomová práce se zabývá nastavením mikrobolometrického obrazového snímače použitého v nechlazené, dlouhovlnné infračervené kameře a následným použitím snímaného obrazu pro ostření kamery. Protože výstupní signál z použitého mikrobolometrického senzoru je možné ovlivnit nastavením pěti vstupních parametrů (velikostí třech řídících napětí V_A, V_B a V_{off}, dobou integrace a velikostí integračního kondenzátoru), provedl jsem se v první části experimentální měření a zdokumentování vlivu všech těchto parametrů na výstupní signál snímače. Vliv všech těchto parametrů jsem současně měřil v závislosti na externím vlivu okolní teploty. Na základě těchto měření jsem poté navrhl dvě metody stabilizace teplotní závislosti mikrobolometrického senzoru. První metoda je vhodná pro aplikace, které pouze zobrazují dopadající infračervené záření (např. puškohled). Protože tato metoda nemá vliv na nehomogenitu zesílení jednotlivých pixelů obrazového snímače, je obraz získaný z kamery subjektivně kvalitnější. Druhá metoda kompenzace vnější teploty je určena pro přesný výpočet a zjištění teploty předmětu snímaného kamerou (radiometrie). Tato metoda kompenzace nastavuje parametry senzoru tak, aby tělesa, která vyzařují stejná množství energie, byla zobrazována vždy stejnou výstupní hodnotou bez ohledu na teplotu okolí. V druhé části jsem se zabýval využitím získaného obrazu ze snímače pro ostření kamery. Zde jsou popsány a porovnány metody výpočtu ostrosti obrazu a popsány algoritmy, které lze na základě ostrosti obrazu použít pro hledání nejlepšího zaostření kamery.

Klíčová slova: nechlazená infračervená kamera, microbolometer, automatické ostření, měření ostrosti obrazu.

Abstract

Thermal cameras, non-contact devices that convert infrared energy into a visual image, have a broad sphere of application ranging from epidemiology to building and construction. The objectives of this master thesis are to first develop a process of setting up a microbolometer infrared detector used in uncooled, long-wave infrared cameras and second to use the resultant image for camera focusing purposes. As the output signal from the used microbolometric detector can be influenced by adjusting five input parameters (integration time, integration capacitor size and three controlled voltages G_A, G_B and V_{off}), an experimental measurement assessing influence each of these parameters was performed. Simultaneously, these measurements were assessed for their dependence on ambient temperature, where a strong dependence was found. Based on these results, two compensation methods of ambient temperature dependence were designed. The first method is suitable for an application that only displays incident infrared radiation, an example of which may be a digital riflescope. Because this method does not amplify the influence of individual pixels of the image sensor on nonuniformity, the image obtained from the camera is subjectively better. The second method is intended for accurate calculation and determination of the temperature of the captured object (radiometry) for industry and research purposes. This compensation method adjusts the sensor parameters in the way that subjects emitting the same amount of energy are always displayed with the same output value, regardless of ambient temperature. Further, methods to assess sharpness of an image were researched and based on experimental evaluation the Prewit derivation evaluated with the threshold sum was found to bring the best results. Lastly, four algorithms allowing to find the maximum sharpness were evaluated, and based on secondary research, the hill search algorithm was selected as the best option.

Keywords: Uncooled infrared camera, Microbolometer, Uncooled infrared detector, Autofocus, Sharpness-maximization autofocus.

Contents

List of Figures

List of Tables

Abbreviations

ADC	Analog-to-digital converter
BB	Black body
CCD	Charge-coupled devices
c	Speed of light ($\approx 3.00 \cdot 10^{8}\ m\,s^{-1}$)
EPROM	Erasable programmable read-only memory
FIFO	First in first out
FPA	focal-plane array
FPGA	Field programmable gate array
FRAM	Ferroelectric Random Access Memory
Full HD	Full High Definition
h	Planck constants ($\approx 6.63 \cdot 10^{-34}\ m^2\,kg\,s^{-1}$)
I2C	Inter-Integrated Circuit
IR	Infrared
IR LED	infrared light-emitting diode
k	Boltzmann constant ($\approx 1.38 \cdot 10^{-23}\ m^2\,kg\,s^{-2}\,K^{-1}$)
LED	Light emitting diode
LSB	Least significant byte
LWIR	Long Wave Infrared
MWIR	Mid-Wave Infrared
NETD	Noise Equivalent Temperature Difference
NUC	Non-uniformity correction
NIR	Near Infrared
PCB	Printed circuit board
PWM	Pulse Width Modulation
SWIR	Short Wave Infrared
VLWIR	Very Long Wave Infrared

Chapter 1
Introduction

FLIR, one of the largest producers of thermal cameras, defines thermal camera as „non-contact device that detects infrared energy (heat) and converts it into a visual image“ [1]

Even though the human eyes have brilliant performance, they can capture and interpret only a tiny fraction of the electromagnetic spectrum. This spectrum is called the visible spectrum and has a wavelength of 0.4 – 0.7 µm [2]. The human eye does not have flat sensitivity, but respects the photopic curve with maximum sensitivity at 0.555 µm. In dark conditions, the photopic curve is replaced with the scotopic curve with a new peak at 0.510 µm. Although visible light is essential for humans, there is a lot of information hidden out of the light that humans cannot see. This is the area in which the infrared cameras operate.

The objectives of this master thesis are to first develop a process of setting up an uncooled, long-wave infrared camera and to use the resultant image for automatic camera focusing purposes. The camera setting is very important for the image quality. This thesis aims to regulate the cameras' parameters in such a way that the camera will display the object independently of the ambient temperature.

Infrared cameras play an important role in various fields of application. Firstly, in building management and construction field, infrared cameras can provide time effective and non-invasive analysis of thermal isolation of the building. The thermal cameras highlight areas with above-average temperatures and hence allow to detect heat-loss points. This information will allow not only to improve the building design while under construction, but also will remain essential for those managing the building or occupying the house, as it can be used to optimize air conditioning and heating. This optimization will lead to both cost savings and lowering the environmental impact. Secondly, thermal cameras can be used for early fault detection. For example, a hot water leakage will be detectable in very early stages and can be resolved prior further damage is caused. The advantage of infrared cameras in these scenarios over other, more traditional methods, is their non-invasive nature.

Continuing on the topic of fault detection, the increase in temperature in temperature is a symptom of many failures in both electronics and mechanics. In developing electric circuits, temperature gradients can bring to light short circuits and overcurrent. Modern electronics use many small footprint integrated circuits, which makes it challenging to analyze such a printed circuit board (PCB) using conventional methods. The infrared camera can take one picture of the entire PCB, and the fault will literally lights up.

Thermography is also well suited for regular inspections of electrical distribution systems. The rise in temperature may be caused by loss of connection, insulation problems, and others, and may lead to unplanned outages. The main benefit of the thermal camera is that during an inspection, the systems can operate normally and there is no need to interrupt the power delivery.

Thermography also proved to be an asset in medicine and epidemiology, as raised body temperature is a common symptom to many infectious diseases. Preventing the spread of diseases (especially SARS-CoV-2) has become essential in the past two years. Infrared camera systems play an important role in solving this critical problem. While detecting potentially ill humans, the measurement can be proceeded contactless and within a significant distance. This makes infrared cameras a great solution for high volume traffic points as well as the overall increase in protection

for the operating staff due to their separation from the potentially contentious population. This and the processing speed of the measurements are the main advantages of this solution over more conventional methods. Moreover, the whole process can be automatized by finding human faces using a well-known algorithm on a visible camera as described in [3] and [4]. Thermal cameras can be used in medicine not only as a noncontact thermometer but also as a diagnostic tool. Infrared imaging was found to be nearly as good as mammography in detecting breath cancer. Sensitivity, specificity, and false positive rate have similar results to mammography, unlike the false positive rate, which is approximately two times higher [5]. Thermal imaging is a non-invasive technique that does not use X-rays, making it less harmful. Furthermore, in recent years, deep neural networks have allowed us to predict breast cancer as presented in [6].

Infrared technology has found its place in chemistry as well. Infrared spectroscopy is one of the widely used spectroscopic methods due to its sensitivity and simplicity for users. Furthermore, the results are quickly obtained and contain numerous information on each bond vibration in the molecule connected with infrared light absorption and change of the dipole moment. More information, including sensors used, can be found in [7].

As the field of application is large, there are many IR cameras already available on the market today. The cheap one starts as low as 5000 CZK (VOLTCRAFT WB-80 [8]), which is affordable for a wide range of potential customers. The price of an expensive industrial IR camera system can be higher than a new car. The price of the camera depends on its parameters described in the following.

The most expensive cameras are those with cooled detectors. These detectors are cooled to temperatures lower than -120 °C. The temperature depends on the used technology InSb, MCT, X-Hot. Cameras equipped with cooled detectors have outstanding performance. They are not blinded by self-radiation, which is a consequence of higher operating temperature in not cooled cameras and devalues the obtained results. Due to this, the detectors have a lower noise level, and therefore, the sensitivity of the detectors increases. This makes those cameras perfect for long-range lenses. Cooled thermal cameras are usually intended as MWIR or SWIR.

On the other hand, cooled sensors are heavier, bigger and consume more energy compared to uncooled ones. Before the camera can be used, the sensor has to be cooled down to the operating temperature. The cooling system contains moving parts, which can produce unwanted noise and can wear out. The cooling system uses helium gas, which slowly passes out because of the small volume of helium atoms or dimer molecules. The cooled detectors have a limited mean time to failure from 8000 to 30000 hours, after which the camera needs to be serviced.

The second possible type of sensor is an uncooled one. As the term indicates, these sensors operate at ambient temperature. These sensors are compact and look similar to standard charge-coupled devices (CCDs) used in visible-light cameras. Uncooled bolometer cameras are perfectly suited for mobile applications because of small dimensions, low power consumption, and lightweight (compared to cooled ones). Because they do not contain moving parts or small molecular gases (such as helium), they require much less maintenance than cooled sensors, when operating in the same environment.

For IR cameras, the trend is similar to that in other electronics: The necessity to have higher resolution for a lower price and smaller physical dimensions as time goes on. The higher resolution plays an important role in radiometry. Similarly to visible-light photography, higher resolution means higher image quality. Moreover, in IR imaging, higher resolution means more accurate thermal measurement. When the camera has low resolution, the measured subject is captured by fewer pixels. This may result in inaccuracy in temperature determination, which in turn can lead to incorrect conclusions. As an example, if one has two cameras with the same optical systems, one with a resolution of 160 x 120 pixels and the second 320 x 240 pixels. The value of one pixel

of the first camera equals on an average of 4 pixels of the second camera. If the object is one pixel big, the lower-resolution camera shows an average of 3 background temperatures and one object temperature. This phenomenon results in lower temperature capture by the lower resolution camera. The higher resolution sensor makes camera measurements more accurate. There is a Full High Definition sensor (Full HD: 1920 x 1080 pixels) already available on the market.

The next important parameter in choosing the IR sensor is the pitch of the pixels. The pitch of the pixels is the distance from the center of a pixel to the center of the adjusted pixel. A lower pixel pitch means a smaller footprint of the sensor with the same resolution, which results in a smaller and lighter camera. On the other hand, smaller pixels can capture less energy, which can reflect in a higher noise of the picture.

Next, it is essential to classify the quality of the image. Infrared systems are very sensitive to any kind of noise. Noise can be divided into two groups. First, spatial noise can be seen when a homogeneous surface is captured. Instead of all pixels having the same value, the values vary with some time-invariable pattern. Because this pattern does not change between frames, it is sometimes referred to as **Fixed Pattern Noise**. This Fixes Pattern Noise can be canceled using non-uniformity correction, as described later.

The second noise category is temporal noise, which changes from one frame to another. The temporal noise that occurs in microbolometers is well described in [9]. One of the most common types of temporal noise is **Johnson noise** (sometimes called Nyquist noise). This noise is caused by the spontaneous motion of charge carriers in an electrical conductor. The effect of this noise is directly proportional to the temperature of the conductor and can be reduced by increasing the integration time. The Johnson noise is often described as a Gaussian white noise because of the nearby uniform distribution of power over the frequency spectrum (similarly to white light, which contains all the colors in the spectrum to an equal extent). Next, **Flicker noise** is another major noise, especially at low frequency. This noise is often referred to as 1/f noise or pink noise because of the spectral density of the 1 / f power. The flicker noise is associated with a DC current and can be found in all active electronic components, as well as some of the passive ones. Not only the electronic, but the temperature plays an important role in noise. **Temperature fluctuation noise** is caused by fluctuations in the temperature of the thermal detector. This fluctuation arises from the heat exchange (by conduction and radiation) with the sensor surroundings.

It turns out that it is useful to classify camera or sensor noise by a single number to enable comparison. The objective parameter is better than subjective parameters, such as the minimum resolvable temperature difference or the minimum detectable temperature difference because it has excellent repeatability and the value is the same, regardless of the person performing the measurement. Noise Equivalent Temperature Diffece (NETD) is used in infrared imaging. NETD is defined as an equivalent black-body target to the background temperature difference that produces a peak signal-to-rms noise ratio (SNR) of one at video output [10].

$$NETD = \frac{\frac{\Delta T}{\Delta S}}{N} \tag{1}$$

, where Δ T is the difference between the target and the background. Δ S is the response of the system signal voltage to the input Δ T and N is the root mean square deviation of RMS from the mean video output, which is commonly referred to as the standard deviation. The exact procedure is described in [10]. IR detector manufacturers present the Noise Equivalent Temperature Difference (NETD), which indicates the level of ability to distinguish among small differences in the image, as an indicator of sensor quality. The NETD is the equivalent temperature (in kelvins) of the noise in the whole sensor. Better sensors have lower NETD. 20 mK is considered a significant difference in the image quality obtained [11]. The NETD parameterize only the temporary noise.

The following two Sections 1.1 and 1.2 introduce the basic knowledge about infrared light and the infrared camera. The rest of the work has been divided into two parts. Section 2 describes the measure behavior and possible settings of the microbolometer array and ends with the ambient temperature compensation. The second part of the thesis covers the camera automatic focusing and can be found in Section 2.8. Firstly, the camera hardware related to focusing is described as well as the controlling firmware. The next Section 3.1 presents autofocus principles. From these principles the image sharpness-based autofocus principle was chosen, and Section 3.2 focused on it.

1.1 Infrared Light

Every physical object with a temperature higher than absolute zero (0 K = -273.15 °C) spontaneously emits electromagnetic radiation.

This radiation was described by Max Planck. [12] Planck's radiation law gives the relationship between spectral radiance I with wavelength λ and the absolute temperature of the body T as described in Equation2:

$$I = \frac{2hc^2}{\lambda^5} \frac{1}{e^{\frac{hc}{\lambda kT}} - 1} \tag{2}$$

,where h is the Planck constant, c is the speed of light, and k is the Boltzmann constant.

From Planck's law, one can determine the wavelength at which the spectral radiance has a maximum value depending on the temperature by deriving the Planck low (Equation 3), and finding at which lambdas the equation equals zero.

$$\frac{dI}{d\lambda} = \frac{2hc^2}{\lambda^6 \left(e^{\frac{hc}{\lambda kT}} - 1\right)^2} \left(5 - 5e^{\frac{hc}{\lambda kT}} + \frac{hc}{\lambda kT} e^{\frac{hc}{\lambda kT}}\right) \stackrel{!}{=} 0 \tag{3}$$

As c is the speed of light and h is the Boltzmann constant, the fractal before the bracket in Equation 3 cannot equal zero. Substitution $x = \frac{hc}{\lambda kT}$ was made on the rest of Equation 3 to obtain Equation 4:

$$(x - 5)e^x + 5 = 0 \tag{4}$$

The trivial solution for Equation 4 cannot be used because either speed of light or Boltzmann constant equals zero and neither wavelength nor Boltzmann constant nor temperature goes to infinity. The second solution was obtained from a numerical solver: $x \approx 4.965$. From the revert substitution, the Wien law was obtained:

$$\lambda = \frac{hc}{4.965kT} \tag{5}$$

If the Plank's law is integrated, the Stefan–Boltzmann law is obtained (Equation 6). The law describes how much energy P the object of surface area A radiated.

$$\frac{P}{A} = \frac{2\pi^5 k^4}{15h^3c^2} T^4 \tag{6}$$

In real life, the object radiates less power. The emissivity parameter (ϵ) was introduced to compensate for the difference between theoretical calculations and experimental values. The emissivity is defined as the ratio of the energy emitted by the measured material surface to the ideal emitter(black body). The emissivity is a number between 0 and 1, where 1 is the ideal emitter and 0 is the ideal reflector. The emissivity is highly dependent on the material's surface. Moreover, emissivity is not a constant, but is a function of many parameters (such as color, temperature,

wavelength, and surface structure). As an example, polished silver has very low emissivity (near 0). On the other hand, human skin has a very high emissivity (near 1). Objects with emissivity less than one are sometimes called gray bodies or imperfect black bodies. Equation 6 can be modified for an object with an emissivity (ϵ) for that gray body:

$$P = \frac{2\pi^5 k^4}{15h^3c^2} T^4 A\epsilon \tag{7}$$

Equation 7 indicates that the power radiated from an object is directly proportional to the fourth power of temperature, which indicates an extremely high sensitivity to the temperature difference. By contrast, the wavelength (with the maximal power) is indirectly dependent on temperature as described in Equation 5. Infrared light can be divided into smaller parts by its wavelengths. There are multiple wavelength classifications, and there are different boundaries between them. Infrared wavebands from Lynred [13] are presented in Table 1. The wavebands are ranges of wavelengths falling between limits, which are presented in the first column. The discontinuities between the Short, Mid, and Long Wave Infrared light are caused by high atmospheric absorption.

Wavelength	**Name**	**Temperature peaks**	**Applications**
0.8 μm ... 1 μm	Near Infrared (NIR)	2600°C ... 3000°C	Night vision (NIR LEDs required)
1 μm ... 2,7 μm	Short Wave Infrared (SWIR)	800°C ... 2600°C	Steel and glass Industry
3μm ... 5μm	Mid-Wave Infrared (MWIR)	300°C ... 700°C.	Fire detection
8μm ... 16μm	Long Wave Infrared (LWIR)	90°C ... -90°C	Security, surveillance, gas leak detections
16μm ... 50μm	Very Long Wave Infrared (VLWIR)	-90°C ... -200°C	Chemical analysis

Table 1: Infrared waveband.

The waveband names are listed in the second column. Temperatures that correspond to wavelength peaks were calculated using Equation 5 for each wavelength, which defines the wavebands. However, other wavelengths are also emitted by an object at a constant temperature. For that reason, the boundaries are not strict. Although the temperature of an object is not in the range of temperature peaks, the object can still emit in that waveband. These values were roundup and listed in the third column.

An example of a typical application is listed in the last column. The Near Infrared light is the closest to visible light and therefore, also some visible light cameras can see in this spectrum. If the manufacturer equips such a camera with an infrared light-emitting diode (IR LED), it could be used at night. In that case of use, the camera captures the reflected light from the surface of a reflecting object. Contrastly, in standard scenarios, the infrared camera captures the light emitted by the captured object itself.

1.2 Infrared camera

IR cameras consist of several parts. The first part of the camera, which is in contact with the infrared flux, is the IR lens. The same physical laws apply to the IR lenses as to visible-light lenses. The only difference is the material of the optics, because it requires a high transmission level in the infrared range. The glass, which has a brilliant transmission for visible light, reflects IR radiation. Thus, IR lenses are made of different materials, such as Germanium or Zinc Selenide. These materials work well for SWIR, MWIR, and LWIR, but they have a high reflection factor. For that reason, an antireflection coating is applied, which limits the lens to be used only in a small band of the IR spectrum. Another critical parameter is the working temperature, because the transmission of the used materials depends on its temperature. Infrared camera lenses have two fundamental parameters, the focal length and aperture, similar to a visible camera lens. The aperture is the optically open space where the flux can travel through the lens. The aperture mainly affects the amount of flux that hits the sensor. Usually, the len's aperture is specified as an f-number, the ratio of focal length to effective aperture diameter. Typically, the aperture is fixed for the IR camera lens. The focal length expresses how the optical system converges light. Negative focal length indicates that the system diverges light. The field of view (ω) can be computed on the basis of focal length (f):

$$\omega = 2atan\left(\frac{p \cdot N_{pix}}{2f}\right) \tag{8}$$

,where p is the pixel pitch and N_{pix} is a number of pixels. (The numerator is the size of the sensor)

The LWIR lens was used in this thesis. The lens has a focal length of 25.0 mm and the f-number is equal to f / 1.20. The field of view was calculated from equation 8 and equals 24.55 ° x 18,45 °. The Workswell Infrared Camera has the opportunity to change lenses to alter the field of view. Thus, in section 2.8 two more lenses from the same manufacturer were tested. The first lens used has a focal length of 14.2 mm and aperture-based f-number f/1.24 (file of view 41.93 ° x 32.06 °). The second lens used has a focal length of 7.48 mm and aperture-based f-number f/1.23. (file of view 72.05 ° x 57.22 °). All used lenses can operate with the 8 - 12 μm waveband light and have the same average transmission greater than 94% in this waveband.

The 'eye' of the camera is the Infrared sensor, sometimes called the focal-plane array (FPA). The FPA converts the infrared radiation to voltage.

The Long Wave infrared uncooled microbolometer array is used in this thesis. Because the manufacturer protects its know-how and does not wish to publish sensitive data, the principle of operation is described on a general sensor, for which the data are available from [15], [16], [17], [14], and [2]

The sensor is composed of pixels arranged in a regular two-dimensional grid (in this case 640 x 480) and a read-out integrated circuit (ROIC). The pixel is the smallest image sensor element in this context; however, the same term can be used in other contexts such as the smallest segment of digital photography. Each pixel is created with one small bolometer, that is why those IR sensors are referred to as microbolometer array. The example of such a pixel is displayed in Figure 1, where one can see the heat absorbing area (the square shape object in the center of the image), leg supports, and the substrate, from which the sensor is made.

For better imagination about working principle, the diagram in Figure 2 is presented. The diagram shows a vertical cut of a fictive bolometer, which simplified electrical schematic with the description can be found in section 1.2. Firstly, the IR flux hits the absorption layer. In this layer, the infrared light is converted into the heat. The IR flux should not be reflected, because reflection decreases the efficiency of light-heat conversion and may cause stray light [1]. Reflection may be

[1] Stray light is undesirable light in optical system. It may follow an unintended path or can be emitted by a source different than intended.

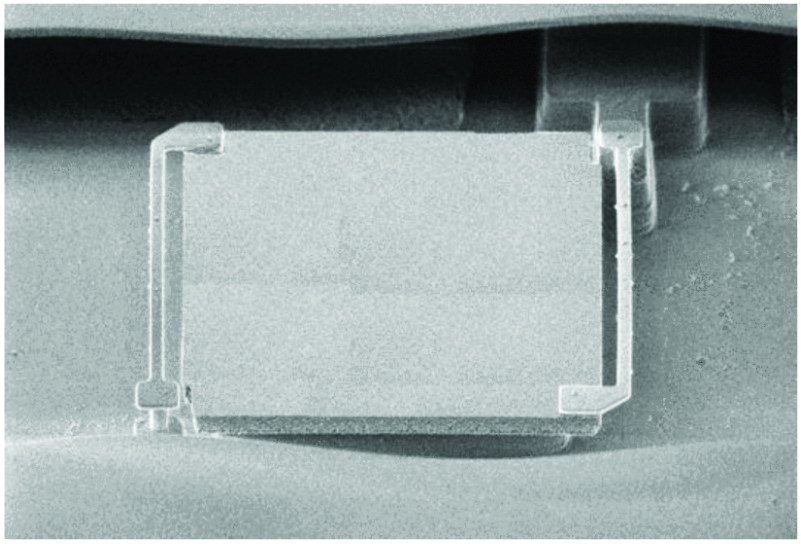


Figure 1: Scanning electron microscope photography of microbolometer pixel. The size of the pixel is 17 x 17 μm^2. The image is taken from [14].

limited with the antireflection coating or the material and thickness of the absorption layer. This layer can be made from Nb, NiCr, Ti, Zr as described in [18], or amorphous SiGe doped with hydrogenated boron [19] or TiN [20].

This absorption layer is thermally connected to the thermometer layer. If the absorption layer is made from electrically conductive material, these two layers must be electrically isolated (in Figure 2 marked as black lines). The temperature can be measured using a diode [21] or a thermistor. The sensor used in this thesis is thermistor-based. Thermistor is an electrical resistor sensitive to temperature. The thermistors are divided on the basis of their conduction model into the negative temperature coefficient and the positive temperature coefficient. Negative temperature coefficient thermistors are typically used as a thermometer in uncooled thermal camera sensors. The thermistors used in microbolometers can be made of amorphous silicon [20] or vanadium oxide VO_x [22].

The bolometer should be thermally isolated from the rest of the sensor. Mainly, this is reached by vacuum or air gap. Thus, the sensor is raised from the substrate, hanging on supporting legs. Typically the two legs are used. The legs have to be strong enough to carry the load of the bolometer. However, the wide legs result in temperature losses to the substrate, which may influence the sensor sensitivity. Long legs are used to increase thermal resistance, as can be seen in Figure 1. The thermal time constant must be compatible with the desired frame rate. The time constant is directly proportional to the heat capacity of the bolometer. The legs are used for electronic coupling of the bolometer and readout eletronics and may be metalized for higher conduction.

ROIC can be implemented in the substrate beneath the bolometer to save the chip footprint. A reflector made of metal (such as aluminum or gold) can be placed between the substrate and the bolometer to reflect the thermal flux going through the absorber. All pixels of the sensor must be thermally isolated from each other. In practice, this is done by physical separation. To increase the thermal insulation, some manufacturers place the microbolometer in vacuum.

If the FPA is the camera's 'eye', then the processing unit is the 'brain'. This unit has to read the sensor's analogue value and convert it into digital values. Digital values have to be further adjusted to obtain a nice picture. The first step is to replace dead pixels. During sensor manufacturing, some pixels are damaged and do not work. The value of those pixels is equal to saturation and remains

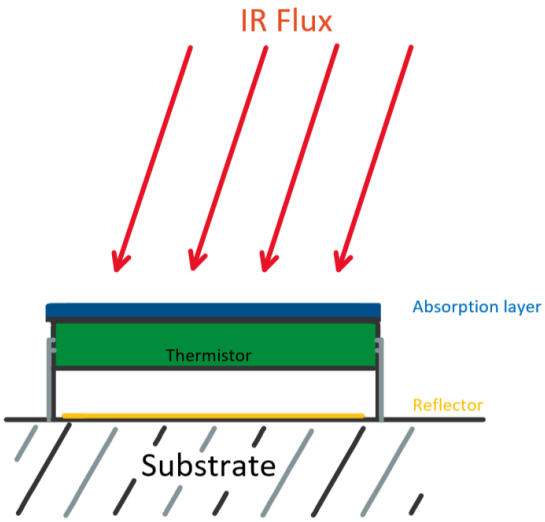


Figure 2: Working principle diagram of a bolometer.

the same, regardless of the source of IR radiation. Because such pixels would disturb the user, their value is replaced with an average value of the neighboring pixels. New dead pixels may arise during the life-cycle of the camera.

Even after the dead pixel is removed, the image is not usable due to variations in pixel-to-pixel sensitivity. Each pixel has a different gain and offset, making the image grainy. The image can be cleared by Non-Uniformity Correction (NUC), sometimes called flat-field correction. During this correction, the value of each pixel is corrected using a line approximation. The camera manufacturer has to compute the gain for each pixel. The offset of each pixel is usually recalibrated during regular camera operation because the offset of the pixels changes depending on ambient temperature. After the camera starts, the offset update is needed more frequently as the camera heats up. To obtain a new offset value for each pixel, the FPA is covered by an uniform surface. This surface can be the shutter or, if possible, an external object with a homogeneous surface can be used. If an external shutter is used, the NUC will also correct the lens imperfections, such as lens vignetting. As a result, each pixel working under the same conditions (mainly IR flux from the object) will have the same value after the NUC procedure. The offset update during the operation needs a shutter and when is performed some frames are skipped, because the shutter is closed and block the incoming radiation to capture only the background flux. This background flux is used to compute the offset of each pixel, which is saved in onboard memory. The price of the shutter and the frame skipping motivates some manufacturers not to perform the NUC offset update. Instead, they compute both gains and offset in advance. These cameras are called shutterless cameras. The principle of a shutterless camera is described in [23], where the temperature of the IR sensors is used to choose the offset correction from the memory. This means that the shutterless camera needs much more memory and cannot react to situations when the camera body is not uniformly heated.

In contrast to a visible camera, it is impossible to prohibit radiance from the sensor surroundings, because the camera body has a different temperature than absolute zero, and thus according to Planck law emits radiation. The processing unit produces heat while working, which complicates the situation, because the background radiation is not homogeneous. This phenomenon is partly solved by smart camera design and partly can be removed by the NUC. This makes it extremely important to correctly choose the size and position of the shutter.

In addition, the processing unit provides the representation of the data. The values for each

pixel are directly proportional to the capture of IR flux. Those values can be displayed, and this is enough for some applications (such as moisture detection or surveillance). For other applications, the exact temperature is essential. In these applications, the temperature could be computed from the flux and other parameters such as the emissivity of the object, the distance of an object, and the permeability of the environment. The camera that enables temperature reading is referred to as a radiometric camera.

The processing module can enable other features, such as digital zoom or increased contrast in the image. A frequently used feature is the coloration of the pixels according to the temperature or the capture flux. This feature is called temperature pallets. There exist many possible pallets for different use cases. For example, the *Rainbow* pallet uses all combination RGB colors (hot is white / red and coolest is blue) to increase the contrast of the image. The second example is *Isotherm*, which highlights a specific temperature range with a particular color (frequently red). The rest of the image reminds grayscale. The Isotherm pallet is perfect for detecting some error (for instance, overcurrent on the printed circuit board).

The Workswell Infrared Camera uses a field-programmable gate array (FPGA) as a processing unit due to the high complexity of this unit task. There is no processor or microcontroller in this camera, which means that everything has to be implemented in the FPGA. The great advantage of FPGA is that several processes can run simultaneously.

Chapter 2
Ambient temperature compensation of the infrared detector

The infrared flux from the captured object hits the infrared detector and causes uneven heating of individual pixels as described in Section 1.2. Each pixel consists of an infrared light absorber and a thermistor that measure the temperature of the absorber. The pixel temperature is read from all pixels and represents the image. The problem is that the absorber temperature is highly dependent on the ambient temperature. Moreover, sensor operations (such as reading the value of a certain pixel) cause sensor warming-up because current flows through the thermistor. Different settings of sensor parameters cause different sensor heating.

All voltages are presented as the digital level in the least significant bit (LSB). This section aims to compensate for the ambient-temperature effect on the infrared detector output.

2.1 State of the art

Only a few publications focus on the problem of ambient temperature compensation. A short review of them is listed below.

The [24] described the compensation in the ROIC based on the current mirror. This method enables the sensor to operate in a wide range of approximately 60 $°C$. This ROIC was tested with a 384 x 288 VO_x array at three environmental chamber temperatures: -10 $°C$, 30 $°C$ and 50 $°C$, showing a maximum difference of 0.545 V (out of 0.7 V - 4.3 V) for black body temperatures from -10 $°C$ to 110 $°C$.

The most related publication that was found is [25]. The aim of this article is not ambient temperature compensation, but ideal operation point selection. The sensor used in this article is similar to the one used in this thesis, but the frame rate used is twice higher in this thesis, which means that the integration time has to be twice shorter. The authors approximated the sensor output as $U = K_G * K_A + K_O$, where K_G denotes the gain coefficient, K_A is coefficient of active bolometer corresponding detected energy and the K_O is the offset coefficient. For each pixel, they measure the offset at the BB temperature of 10 $°C$ and compute the gain as the difference between the measured values at the 40 $°C$ and 10 $°C$ BB temperatures divided by the difference in the temperature of these BBs.

The next two methods provide compensation of the ambient temperature only by software processing. In [23] the authors propose to compute the corresponding voltage at ambient temperature T_0. To achieve this, they advise measuring the temperature of two black bodies at various ambient temperatures (with step of 5 K insight the working range). In the second step, the radiometry is computed using 2 K step for all BB ranges at ambient temperature T_0. Unfortunately, no results were published.

In [26] the same principle is used, but is more described, and a measurement is made. A photon 320 microbolometer camera located in an environmental chamber with a large-area black body is used for the measurement. The authors cycled the chamber temperature while changing the black body temperature. The temperature of the blackbody was changed from 10 $°C$ to 50 $°C$ with step of 10 $°C$. They present the relationship between the incoming flux and the output voltage of the IR sensor as linear. The correction for each pixel is computed by comparing the radiance at two

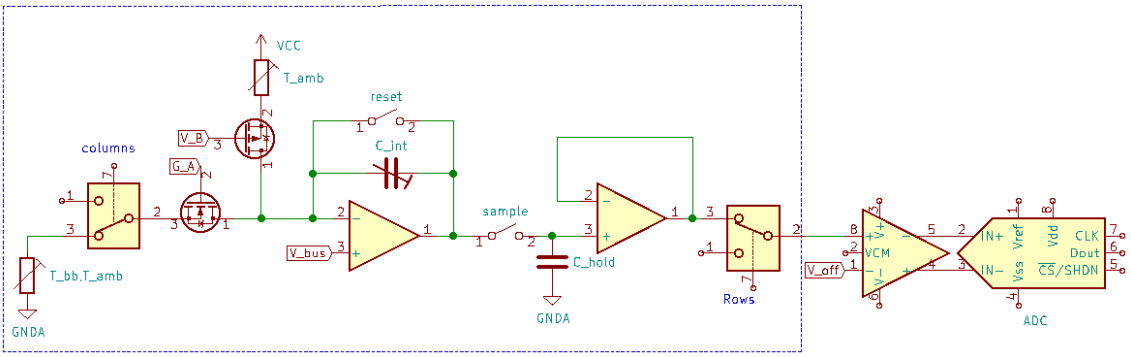


Figure 3: The simplified schema of microbolometer (in a blue dashed rectangle) and read-out circuit.

different chamber temperatures. For each pixel the correct value r_c is computed as

$$r_c = \frac{r + b\delta T}{1 - m\delta T} \tag{9}$$

, where *r* are the raw data from the sensor, *m* and *b* are the correction parameters. The results presented in the 24 hour experiment show good results for this method of uncertainties of +- 0.32 $°C$. The maximum error was close to 2 $°C$.

These two recently mentioned methods do not affect sensor behavior. Therefore, it can be used without knowing the sensor. However, the sensor value changes with the ambient temperature and, at some point, the value hits the ADC or sensor saturation. Accordingly, this method needs a value margin, which results in a lower dynamical range or lower sensitivity. This method performs ambient temperature drift compensation and a non-uniform correction at once.

2.2 Microbolometer properties

If the sensor output can be directly influenced, the whole sensor range can be used at each ambient temperature. This results in a higher sensitivity compared to the software output compensation presented in [26] and [23]. However, it is essential to understand its functionality to affect the sensor properly.

The simplified schema of the sensor (in a blue dashed rectangle, based on [27], [16], [28]) and the attached analog-to-digital converter (ADC) is shown in Figure 3. The pixel is represented as a resistor *R1* with variable resistance depending on the ambient temperature (T_{amb}) and the temperature of the captured object (T_{BB}). All pixels in the column share the rest of the electronics. ROIC automatically switches the pixel connected to the Q1 transistor according to the sensor clock. Because the ambient conditions have a more significant effect on the temperature of the pixels than infrared radiation, the ambient temperature is measured by the blind pixel (*R2*) and subtracted from the result. The blind pixel has the same construction as an active pixel, but it is isolated from incident IR radiation. Therefore, the thermisotr *R2* is sensitive only to ambient temperature. The *R2* thermistor is connected to the sensor power supply.

The microbolometer manufacturer provides the sensor with two metal oxide semiconductor field effect transistors (MOSFETs) to enable sensor tuning. The transistor *Q1* controls the current to the active bolometer *R1*. The transistor *Q1* is controlled by the V_A voltage. FPGA can set the voltage V_A by external 14-bits digital to analog converter (DAC). The blind thermistor compensation effect is weighted by the second *Q2* transistor, which is controlled by a V_B voltage. One blind microbolometer (*R2*) is used for one column of active pixels (out of 480).

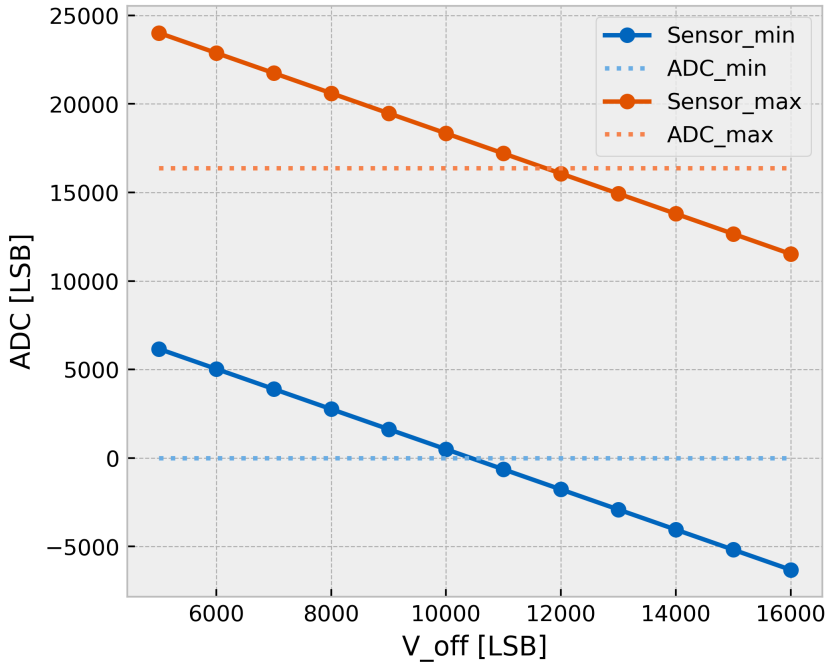


Figure 4: Affect of V_{off} on ADC value.

The resulting voltage is then integrated for t_{int} time. This time can be set through a communication bus. The integration capacitor C_{int} can be set to six discrete values on the same bus.

The one-pixel column was described above. The described circuit can be found 640 times in the entire sensor. All setting parameters are set for all columns, which means that there is only one set $\{V_A;V_B;t_{int};C_{int}\}$ for controlling the sensor.

The sensor works as follows. First, all pixels in the first column are integrated at the same time. Then the voltage is stored in the sample and hold circuit. The values are read pixel by pixel during next-line integration.

The sensor output is the voltage is compared to the V_{off} voltage and is converted by the differential ADC. The V_{off} is the last parameter that can influence the voltage level. FPGA can set V_{off} by DAC similarly to V_A and V_B. The resulting voltage is converted by a 14-bit ADC and processed in the FPGA. The effect of V_{off} on the ADC value can be seen in Figure 4. On the x-axis, the set value V_{off} is listed, and on the y-axis, the read ADC value is presented. The limitations of the ADC are highlighted by a horizontal level of light orange (ADC = 16383 LSB) and light blue (ADC = 0 LSB) horizontal level. The offset sensor minimum is marked with blue lines with dots, and the shifted sensor maximum is marked with an orange line with dots. The V_{off} only makes sense to set in values for which the ADC value of the shifted signal maximum does not exceed the maximum ADC value, and the ADC value of the offset signal minimum is greater than the minimum ADC value. This condition is valid only for V_{off} in the range from 10434 LSB to 11719 LSB.

The effects of other parameters were measured according to the procedure described in the next section.

2.3 Measuring Infrared detector thermal properties

All measurements were made with a particular camera and lens (focal length = 25.0 mm). The methodology described in the following lines was used for all measurements, unless otherwise noted.

The camera is located in the climatic chamber, which is set to a particular temperature (this temperature is called the ambient temperature or T_{amb}). First, the camera is turned on and placed in the environmental chamber. The measurement begins thirty minutes after the temperature of the climatic chamber has stabilized to allow the insight camera temperature to stabilize as well. The non-uniformity correction is not applied during the measurement while the dead pixels are removed. The camera is focused on the stabilized black body with fixed temperature. The distance between the sensor and the black body is one and a half meters, and there is no obstacle between the camera and the surface of the black body because the climatic chamber has a small hole created specially for this purpose. This small hole was tested to not affect the thermal stabilization of the climatic chamber or the sensor. The black body (BB) is the reference source of IR radiation with known high emissivity (near 1).

For measuring, only the average value of the middle 8 pixels (array 3 x 3 - 1) is captured and sent to the computer using the universal asynchronous receiver-transmitter (UART) communication protocol. The BBs have a limited size of the thermal controlled area, which is the reason why only the middle pixes are used. The surrounding pixels of the center are taken to ensure that the measured pixel is not faulty. The number eight was chosen because the integer divide by eight is equal to the bit shift by three to the right, and the shift operation is effective on FPGAs.

The central pixels value was measured 50 times to have the statistically correct value. The mean value of these 50 measurements is used in the following procedure.

For each measurement, the three temperature levels were also captured. These temperatures provide confirmation that the transient response was completed and may be used for ambient-temperature compensation. The first is the temperature of the bolometer. This temperature is read out form the bolometer. The second temperature is the shutter temperature, and the third temperature measures the FPGA temperature. The two last mentioned temperatures are measured by the camera's built-in temperature sensors.

2.4 Measured data evaluation procedure

For illustration, the data evaluation process is performed in the default sensor configuration, which the sensor manufacturer provides for each sensor.

Note that the integration time is two times higher than the developing camera can have, resulting in a fram erate of 30 frames per second (FPS) for the default sensor settings.

The demonstration data set was taken at ambient temperature $T_{amb} = 10\ °C$ for BB temperatures in the range of -15 °C to 650 °C. Data are shown in Figure 5. Every blue point corresponds to a mean value of 50 measurements of the average value of the eight central pixels (as described in the previous section). The mean value of the pixels can be found on the Y-axis. This is the raw ADC value, the minimum value is zero, and the maximum value is 16383 (marked by a red line), since the 14-bit ADS is used. On the X-axis the temperatures of the black bodies are listed.

The first thing to do is filter out the data that are not valid. If one of the three temperatures deviates more than 0.2 °C from an average value, the corresponding black body measurement is not used to prevent errors. This deviation can be caused by the change in the temperature of the climate chamber (opening the door, power supply outage, etc.) or when the measurement starts before the temperature is in steady state.

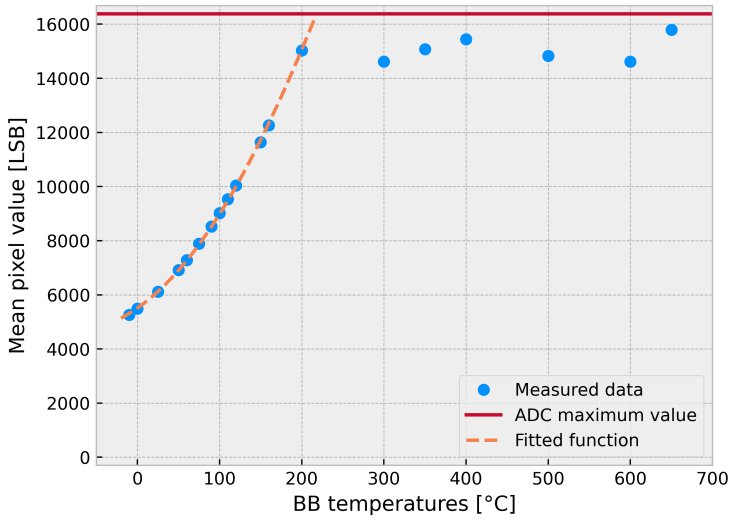


Figure 5: Default sensor configuration showing measured BB at blue points and the fitted RBO function by orange dashed line.

In the next step, the data points for which the ADC respective the sensor is saturated have to be filtered out. When the sensor is saturated, the value is not equated to the maximum possible value but rather to other values which, without knowing the context, may be considered as valid ones. The gradient was computed to detect those invalid data. Once the gradient is negative, all subsequent data (measured at higher BB temperature) were skipped. This data filtering turned out to be unreliable when two black body measurements were close to each other. The noise caused the condition to be triggered, mainly because of the lower BB temperatures. Then, experiments were carried out with other more complex filtrating conditions. The final and used condition predicts the position of the following measured points. If the measured point was lower than the predicted point by the threshold, then the point was considered invalid.

An important part of the post-processing of the data was determining the relationship between the measured data. The first attempt was to fit the measured data with a linear line similar to [25], in which the sensor was similar. Because the line is determined by two points, only two black bodies were measured. The result of the measurement and stabilization using the linear model does not meet the expectations, which results in a new measurement shown in Figure 5. Then several functions, including exponential and polynomials, were tested to fit the data. The function that best fits the results is shown in Equation 10 and is used to calculate the temperature of the measured ADC value [29]:

$$ADC = \frac{R}{e^{\frac{B}{T}} - 1} + O. \tag{10}$$

The Planck law (Equaion 2) can be obtained if R = $\frac{2hc^2}{\lambda^5}$, B = $\frac{hc}{\lambda k}$ and O = 0. This function is referred to as the RBO function in this thesis. The trust region reflective algorithm is used to fit the RBO function. The result can be seen in Figure 5 as a dashed orange line. Measurements reveal that parameter B can be considered as a constant. The set of measurements of all BB temperatures was described with two parameters, R and O.

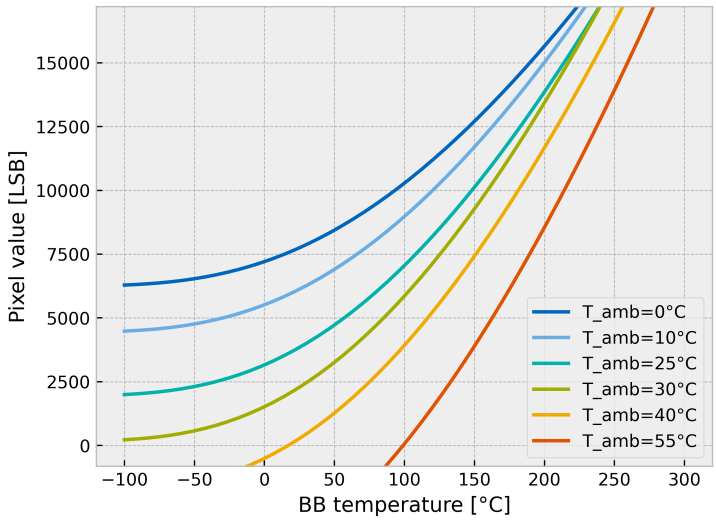
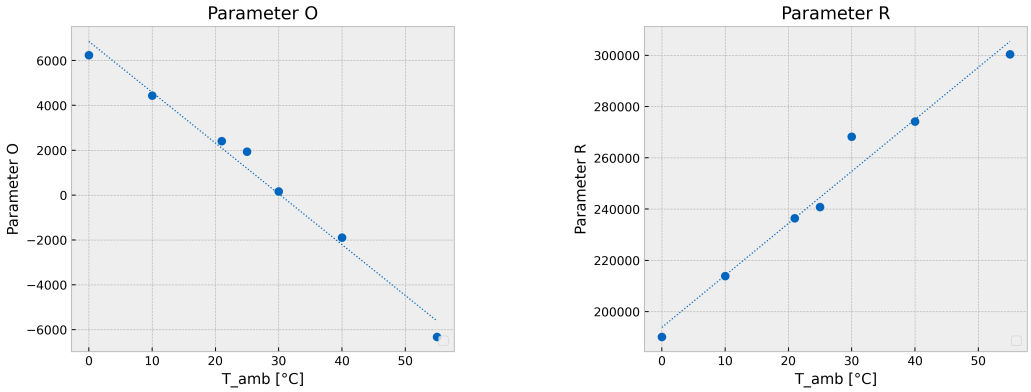


Figure 6: Default sensor configuration temperature showing the ambient temperature dependence. Each line represents the measurement on twenty BBs.



(a) Parameter O

(b) Parameter R

Figure 7: Influence of ambient temperature on RO parameters. Each point represents one RBO curve from Figure 5.

2.5 Measured infrared detector behavior

In this section, the measured characteristics of the infrared detector built into the Workswell Thermal Camera are presented.

2.5.1 Effect of ambient temperature on infrared sensor output

First, the influence of ambient temperature on sensor output was measured. Several ambient temperatures were set in the climate chamber and, for each ambient temperature, the measurements described in sections 2.3 and 2.4 were made. The measurement is shown in the graph in Figure 6. Each line corresponds to an ambient temperature constructed from nine black body temperatures. This measurement shows a substantial dependence on the output value on ambient temperature, which must be compensated for. Each of the functions shown in Figure 6 can be parameterised by two variables (R, O) according to Equation 10. The difference in ambient temperature causes a change in microbolometer behavior, which is described by changing the parameters R, O shown in the figure 7. As can be seen in the graph, the effect on both parameters looks linear, which makes it possible to approximate the data with a line using the least squares method (blue dotted line in the figure 7).

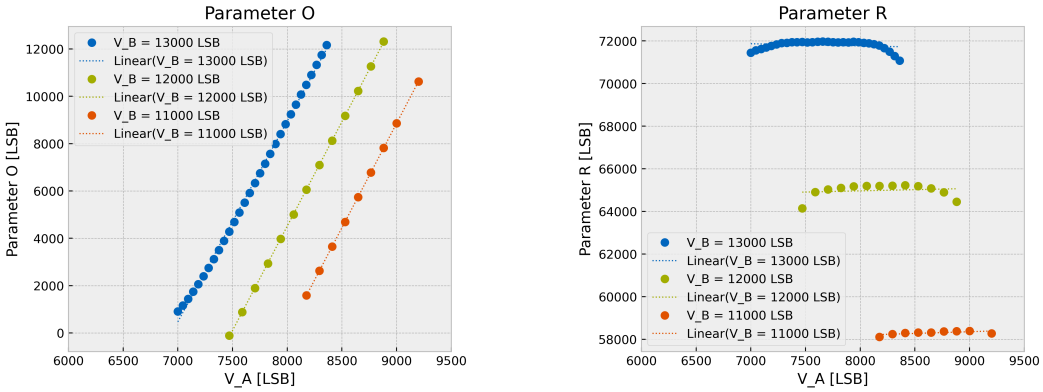
In the following subsections, all parameters that can be used for thermal compensation will be presented.

2.5.2 Effect of Voltage V_A on infrared sensor output

The influence of V_A on both parameters (R,O) was measured at an ambient temperature of $T_{amb} = 0° \ C$ for three different voltage values of V_B to see how they impact each other. The highest value of V_B was 13,000 LSB. This data set was measured with a small step of 50 LSB V_A to provide high precision. The other two set points (12000 LSB and 11000 LSB) were measured with a larger step of 120 LSB to confirm the conclusions of the first and more detailed set point. The measurement can be seen in Figure 8. A greater V_A causes a greater parameter O. On the other hand, the R parameter seems to be constant for the values in the middle part of the data. Near the end points of the line, the lines bend downward. This may be due to the fact that some of the black body measurements (blue points in Figure 5) cannot be used to fit the RBO functions because they are outside the range of the sensor. Although these points are taken into account, the effect of V_A on the R parameter seems to be negligible to the other measurements shown next.

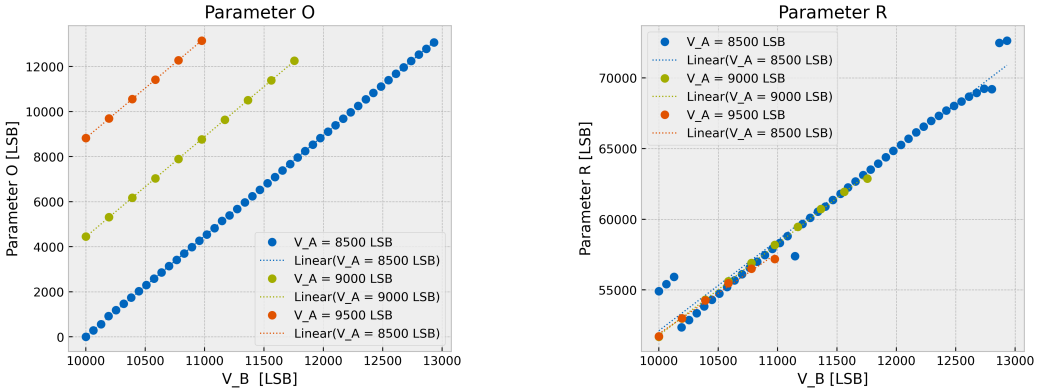
2.5.3 Effect of voltage V_B on infrared sensor output

Similarly to voltage V_A, voltage V_B was measured for three different V_A. All other parameters, including ambient temperature, are identical. Measurements are shown in Figure 9. The V_B has a linear response to the O parameter. Compared to V_A, the gain is approximately two times smaller. On the other hand, Figure 9b shows that V_B is proportional to a parameter R. Furthermore, this measurement confirms the measurement presented in Section 2.5.2, because the three different data sets V_A are on one line, which means that V_B does not affect the R parameter.



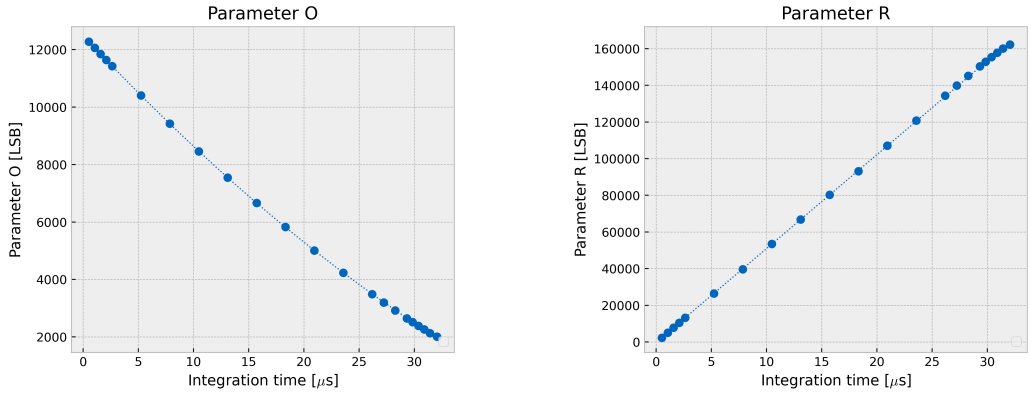
(a) Parameter O (b) Parameter R

Figure 8: Influence of V_A on RO parameters.



(a) Parameter O (b) Parameter R

Figure 9: Influence of V_B on RO parameters.



(a) Parameter O

(b) Parameter R

Figure 10: Integration time influence of RO parameters on the default configuration.

2.5.4 Effect of integration time on infrared sensor output

The Workswell Thermal Camera project restricts the maximum integration time due to the frame rate of 60 FPS. This maximal integration time can be computed as:

$$t_{int} = \frac{\frac{1}{Framerate}}{N_{rows}} = 34.7\ \mu s \tag{11}$$

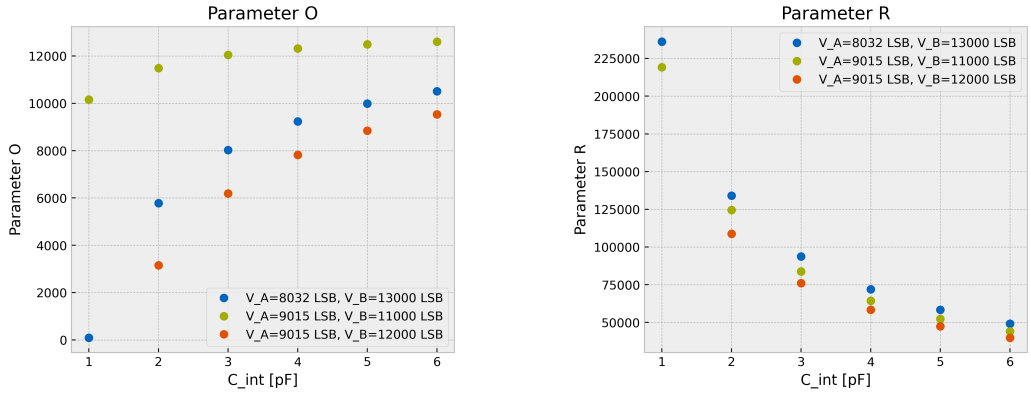
, where the frame rate is 60 frames per second (FPS) and N_{rows} is the total number of rows of pixels on the sensor (= 480).

Integration time can be changed in the range of 0.05 μs to 34.7 μs with a constant frame rate of 60 FPS. The effect of this parameter was measured for 22 different integration times and can be seen in Figure 10.

2.5.5 Effect of integration capacitor on infrared sensor output

The integration capacitor C_{int} can be set to six different discrete values via the serial interface I2C (Inter-Integrated Circuit) [2]. All possible values of C_{int} were measured, and the result is shown in Figure 12 for three different sensor settings. The green and orange points have the same V_B value. Unfortunately, C_{int} = 1 pF cannot be measured for the sensor configuration shown in orange color, because the measured BB levels were outside the range. However, a clear trend can be seen in the data. The dependence on the parameters R and O is not linear but logarithmic. A smaller integration capacitor means a higher R parameter and thus a higher range. On the basis of the other measurement, the effect on parameters such as V_B or V_A is not negligible.

[2] I2C is a synchronous serial communication bus.



(a) Parameter O

(b) Parameter R

Figure 11: Integration capacitor influence of RO parameters on the default configuration.

2.6 Ambient temperature drift compensation

The drift of the ADC value corresponding to one BB temperature described in the section 2.5.1 have to be compensate to enable the use of the camera in the entire range of operation of the camera. Two possible requirements can be made for the thermal drift compensation, dependant on camera usage. When temperature measurement is not needed (for example, for surveillance, hunting, etc.), the sensor output voltage only has to be in the ADC range for all ambient temperatures. The RBO function in this range is not important and can change its parameters based on the change in ambient temperature. Temperature ranges that do not enable temperature measurement are called non-radiometric. The nonradiometric range, which can capture the Black body temperature in range -15 $°C$ to 150 $°C$ is called High Gain in this thesis. The easy modification of the ambient-temperature drift compensation is essential for non-radiometric ranges. This range is described in Section 2.6.2

The radiometric range is used to compute the subject's temperature. To enable this, there should be a known dependence between the ambient temperature and the RBO parameters, because the RBO function is used to compute the temperature from the ADC value. This thesis aims to find the thermal compensation model in such a way that the RBO parameters are invariant to changes in ambient temperature. In other words,the ADC value of a BB-targeted pixel at constant temperature is constant for all ambient temperatures in the working range. This condition holds for all BB temperatures in range -15 $°C$ to 150 $°C$ and each of the BB temperatures must have a different ADC value. This range is denoted R1 in this thesis and is described in 2.6.3.

2.6.1 Thermal detector configuration finding

The main building block in ambient temperature drift compensation is to find the configuration in which the detector has specific RBO parameters. The target RBO parameters can be found using a Python[3] script with a graphical interface (see Appendix A.1). The script uses the numpy library [30] and matplotlyb [31] for visualization.

The user can set the RBO parameter and see the effect on the RBO curve and the BB temperature, which corresponds to 0, 10, 90 and 100 percent of ADC range. The BB temperature range that the camera should be able to capture is specified to be -15 $°C$ to 150 $°C$. Because each sensor can have different properties the margin 10 percent of ADC range is used. This means that the BB

[3]Python is high-level, interpreted programming language with garbage collector and dynamic data type.

of 150 $°C$ should be captured with the camera as a 0.9 ADC range and 0 $°C$ should correspond to the 0.1 ADC range. This margin should ensure that the sensor configuration can be used on different sensors.

The first step in the process of finding the detector configuration is to set the value C_{int}. This value is set based on the R parameter (which corresponds to the target Black body temperature range, in which the camera should see/measure). The C_{int} capacitor value has to be constant for all ambient temperature drift computation configurations, unless the nonuniformity correction gain matrix has to be updated. To compute the NUC gain matrix, two BB measurements are required, thus it is undesirable for the end user to perform it, and therefore it cannot be changed for ambient-temperature compensation.

This capacitor value was chosen as 1 pF because the BB range is relatively small and the maximum integration time is limited to 34.7 μs. If possible, the integration time is chosen as high as possible. The last parameter that can be used to adjust the R parameter is V_A. When it is impossible to reach the target value, the capacitor has to be changed or the time has to be reduced. In the last step, the parameter O is tuned using V_{off} and V_B. Because V_{off} has a limited effect on the ADC value, V_B is chosen. The V_{off} parameter can be used in the latter case for small ADC tuning.

The sensor configuration found based on the model created from the measurement was set on the camera and placed in the environmental chamber, where it was tested according to the procedure described in Sections 2.3 and 2.4. The values of V_B and V_A must be slightly adjusted because the measured RBO parameters do not correspond to the target ones.

Unfortunately, it turns out that V_B also has an effect on the R parameter, which is significant when precise values of the R and O parameters are required, and thus small parameter tuning is performed in the iterative process.

The first step is to determine the configurations that should be tested. The set of them is determined as all combinations of $\{V_{A-computed} - a; V_{A-computed}; V_{A-computed} + a\}$ and $\{V_{B-computed} - b; V_{B-computed}; V_{B-computed} + b\}$ where $V_{A-computed}, V_{B-computed}$ are obtained in the previous iteration step and *a*,*b* are constant, depending on the measured error. More constants a,b can be used in one iteration to save time.

In the second step, the set computed in the first step is measured according to the process described in Sections 2.3 and 2.4.

The last step is to compute the new values of $V_{A-computed}$ and $V_{B-computed}$ using a linear approximation (the presented measurement shows that V_A and V_B have a linear effect). If the newly computed values do not vary by less than 2 Least Significant byte (LSB), the value is considered as final, and the process is stopped.

2.6.2 High Gain sensor configuration

Because the High Gain sensor configuration has to be easy to find, only one parameter is controlled. According to Figure 6, the offset (parameter O) has the greatest impact on the result. The O parameter can be regulated using the V_A parameter.

Before starting the temperature compensation, the initial sensor configuration has to be found. For High Gain camera settings, it is favorable to choose the initial configuration in the highest ambient temperature from the operation range because when the ambient temperature decreases, the gain will also decrease, which results in range increasing. If the initial configuration is chosen at low ambient temperature, the range will be reduced, making it impossible to fulfill the BB temperature range. The down side of the broad range is the smaller sensitivity.

The initial sensor configuration was found using the methods described in Section 2.6.1. Then the climatic chamber was set to $0°C$. In the next step, the V_A parameter was found to obtain the

same O parameter. Because the parameter O is linearly dependent on V_A, only two measurements are needed to compute the new value V_A.

These two values were used for the first attempt of ambient temperature compensation. The value of V_A is regulated based on the temperature of the FPA according to the interpolation of the lines of the two measured V_A. This compensation mechanism was tested at ambient temperature $25°C$. The result is shown in Figure 12a as a yellow dotted curve. As can be seen, this regulation cannot be used due to the measurement of low BB temperatures. The offset shift is caused by the fact that the ambient temperature influences the impact of V_A on the sensor ADC output. The image shows the maximum ADC value by the black dotted line. Solid lines denote the measurement for which the V_A values were set to manually found values.

To compensate for this phenomenon, the correct value of V_A was found at $25°C$ using the same technique as for the ambient temperature of $0°C$. The formula to calculate the value V_A based on the temperature of the FPA can be extended by this measurement, resulting in a second-order polynomial:

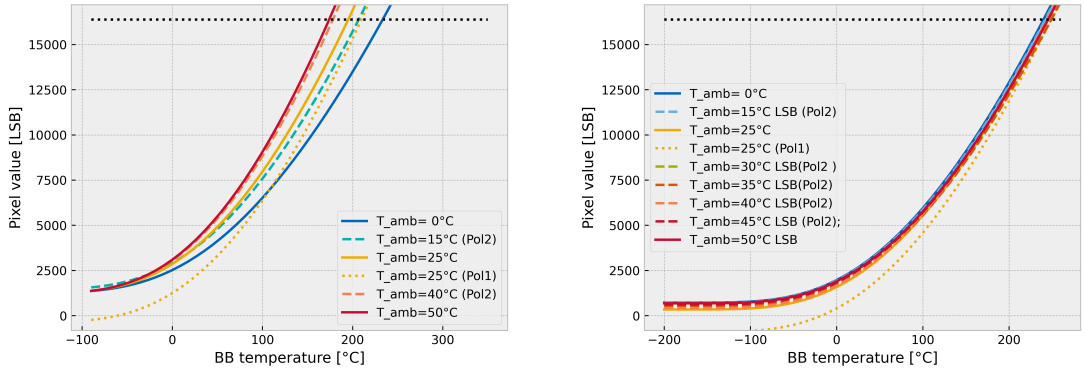
$$V_A = aT_{FPA}^2 + bT_{FPA} + c \tag{12}$$

Then the drift compensation of ambient temperature uses the second-order polynomial and was tested at ambient temperature 15 $°C$ and 40 $°C$. The result is shown in the same Figure 12a as the first-order polynomial method. The second-order polynomial method verification is visualized by the dashed curve. As can be seen in Figure 12a, the maximum BB temperature that the camera can capture depends on the ambient temperature and can vary from 173 $°C$ to 232 $°C$. This means that this compensation can be used for non-radiometric purposes only.

2.6.3 Ambient temperature drift compensation for radiometric purpose

The High Gain described above can display the desired BB range regardless of ambient temperature in the camera operating range. The issue with the High Gain range is that the parameter R changes, which decreases the sensitivity at low ambient temperatures and complicates the radiometric computations. This inspires the radiometric range R1. In contrast to High Gain, the initial configuration can be found at any ambient temperature. It is only necessary to take into account future regulation and choose the parameters in such a way that enables parameter tuning. In this work, the highest ambient temperature was chosen as the starting point. The configuration was found using the procedure described in Section 2.6.1. The integration time is set as high as possible to allow setting V_B to low values.

In the next step, the configuration is found at ambient temperature 0 $°C$, which is the lowest temperature at which the camera should operate. For this tuning, only V_A and V_B are used, and all other sensor parameters are kept constant. Similarly to High Gain, the first attempt of regulation is done using a first-order polynomial. In contrast to High Gain, R1 uses both V_A and V_B, which means that two equations are required. The result is shown in Figure 12b by the yellow dotted line. Similarly to High Gain, this regulation principle does not work well. The configuration was tuned at ambient temperature 25 $°C$ and both equations were extended to a second-order polynomial. The second-order compensation result was verified at 15 $°C$, 30 $°C$, 35 $°C$, 40 $°C$, 45 $°C$ and can be seen in Figure 12b displayed as dashed lines. As can be seen when comparing Figures 12a and 12b the compensation using both V_A and V_B achieves better results, the RBO curves are located close together. The ADC values corresponding to all BB temperatures in the range of -50 $°C$ to 150 $°C$ with a step of one were calculated. At each of these points, the standard deviation SD_i was calculated for all measured ambient temperatures. The maximum standard deviation is at the BB temperature 150 $°C$ $SD_{150} = 169\ LSB$, minimal at 11 $°C$ $SD_{11} = 128\ LSB$, and



(a) High Gain

(b) Radiometric

Figure 12: Ambient temperature drift compensation. The solid line represents measurement which uses sensors settings found to meet the specifications, the dotted line corresponds to measurement done with the sensor setting, which was computed using first order polynomial and the dashed lines was obtain, when the sensor configuration was calculated using second order polynomial.

average $SD_{AVG} = 136\ LSB$. Because the 14-bit ADS is used, the maximum ADC value is $2^{14} - 1 = 16383$. The maximal standard deviation corresponds to one percent of the ADC range.

2.7 Effect of both voltages V_A, V_B on infrared sensor output

The configuration findings shows dependency of parameter R to G_B, which was not found during the system identification. It was probably caused by the O parameter, which resulted in a rapid shift in the capture range, which means that the BBs were outside the sensor range, causing the inaccuracy. This dependence will be explored in this section.

To avoid the same issue, the measuring procedure was modified. The second objective of this measurement is to find an easier procedure for the configuration findings. The process described in Section 2.4 requires a much different BB temperature to interpolate the RBO function, which requires much time. In order to save time, a constant BB temperature is used in this section.

The cameras were placed in the Climate chamber, which was set at a temperature of 25 $°C$. The camera was focused on a black body with a temperature of 100 $°C$. The sensor settings were then changed. In Section 2.6.3 it is shown that V_A and V_B can be used to compensate the temperature of the infrared detector, therefore only those two were used. Then V_B was changed in the range of 10521 to 10980. This range was chosen based on the R1 configuration and divided into three subranges. The bigger step of 10 DAC bits was applied to ranges from 10521 LSB to 10721 LSB and 10780 LSB to 10980 LSB. The smaller step of 1 DAC bit was used in the central part 10702 LSB to 10779 LSB. In total, the set of possible voltages V_A had 100 values. Similarly, V_B was changed in three sub-ranges from 9151 LSB to from 9351 LSB and from 9410 LSB to 9610 LSB with a step of 10 DAC bits and from 9352 LSB to 9409 LSB with a step of one bit.

The response for each combination of sets V_A and V_B was measured and stored in 100 x 100 matrix $M_{measured}$. All values in one row were obtained with the same V_B and all values in the same column have the same V_A set in the infrared detector. The average temperature of all thermometers was also saved.

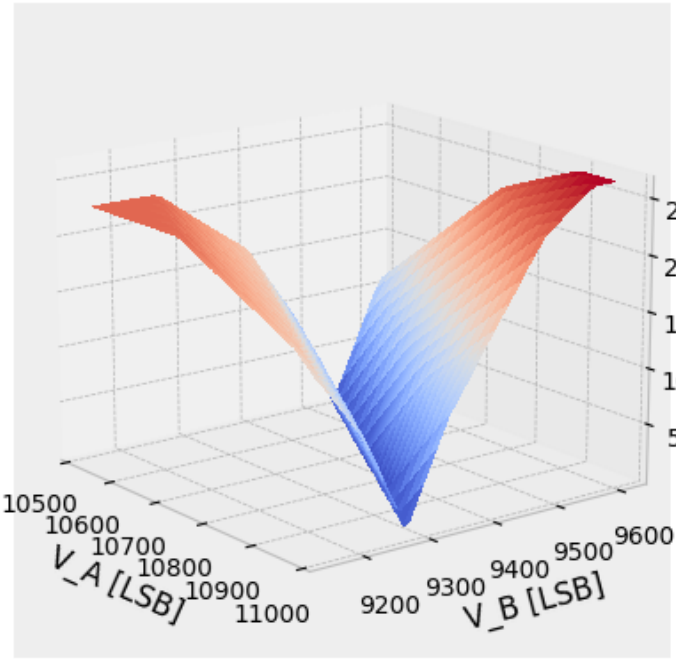


Figure 13: Effect of voltages V_A, V_B on infrared sensor output.

In the next step, the ideal ADC value for the BB temperature of 100 $°C$ was calculated from the RBO function, with the R1 settings parameters. Using this value, the matrix M_{idela} was constructed, as an all-ones matrix multiplied by the ideal value.

The matrix M_{idela} was subtracted from $M_{measured}$ which creates the matrix $M_{deviation}$ showing the deviation from the ideal value. From the matrix $M_{deviation}$, the error was calculated using the absolute value for each element. The error is shown in Figure 13. It can be seen from the figure that the lowest error follows a trend of a straight line. However, the error is never equal to zero, which is caused by the fact that the parameters V_B and V_A are discrete.

The line is defined by two points with coordinates C_A and C_B. To allow the curve to have an error equal to zero, the coordinates are decimal. The coordinates C_A of these two points were chosen as $C1_A$ = 10521.0 and $C2_A$ = 10980.0, their distance being the maximum possible. The C_B coordinators must be calculated. The column of matrix $M_{deviation}$ corresponding to V_A = 10521 LSB for $C1_A$ and V_A = 10980 LSB for $C2_A$ is taken and approximated with a line using least squares methods. The value in which the approximated line crosses the zero deviation is the coordinator C_B.

The same procedure was repeated for BB temperature 200 $°C$ and 15 $°C$. The low temperature 15 $°C$ was chosen because it is above the Dew point, which ensures that the measurement is not affected by the condense humidity. For a lower ambient temperature, the measurement has to be done in a short time, or humid air has to be removed from the BB. With this procedure, three lines were obtained, one for each BB temperature. These lines are drawn in Figure 14. The blue points A and B define the line f_{T100}, which corresponds to the temperature of the BB 15 $°C$, similarly, the points C, D define the line f_{T200} and last but not least, E and F define the line f_{T15}.

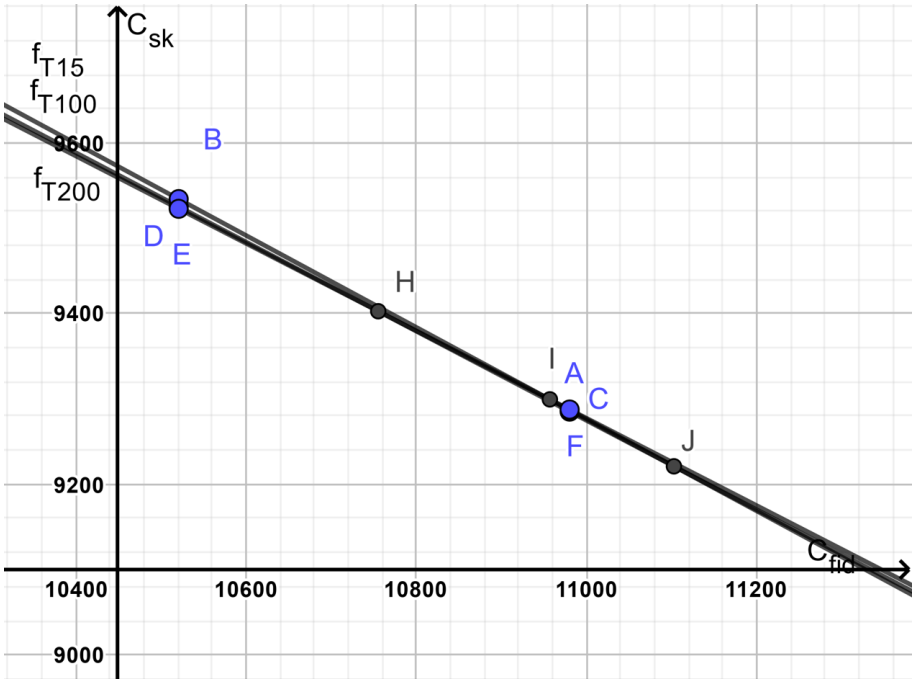


Figure 14: C_B x C_A vector space, demonstrating the relations between effect of V_B and V_A on ADC sensor output. The line f_{T15} shows how to configure the sensor to obtain the 2898 LSB ADC value when camera is capturing BB 15 $°C$. The line f_{T100} shows how to configure the sensor to obtain the 6467 LSB ADC value when camera is capturing BB 100 $°C$ and the last line f_{T200} corresponds to BB temperature 200 $°C$, which is converted as 13390 LSB. The gray points denotes the intersections.

As can be seen, the lines are nearly parallel. The gray point H marks the intersection of f_{T15} and f_{T100}, I of f_{T15} and f_{T200} and J denotes the intersection of f_{T200} and f_{T200}. The ideal state will be that if the points H, I, and J are identical, then all of the BB temperatures correspond to the required ADC values. However, this is not the case. As can be seen in Figure 14, the distance between them is not negligible. This may be due to the fact that the required RBO parameters are probably not reachable.

2.7.1 Time efficient V_B and V_A findings for infrared detector configuration

These data can be represented and used for ambient temperature compensation in two ways. The first way is to work with all measured data and find a point in which the distance between all three lines is minimal, which optimizes the error in these temperatures. The second way is to take into account only f_{T15} and f_{T200}, which ensures the range and offset, but the RBO parameters may be different and must be found when radiometry is performed. The range is important in this thesis, and the RBO parameters are only a tool to preserve the measure data. Thus, only point I is used for ambient temperature compensation. It can be seen that point I is within the H, J interval and the error for BB temperature of 100 $°C$ will probably not be large.

The configuration of the IR detector at 0 $°C$ and 50 $°C$ is still needed to create the two second-order polynomials presented in Section 2.6.3. The configuration is found using the BB of temperature 15 $°C$ and 200 $°C$. The measurement procedure is similar to the ambient temperature 25 $°C$. The number of measurements may be reduced to save time. In theory, only four measurements

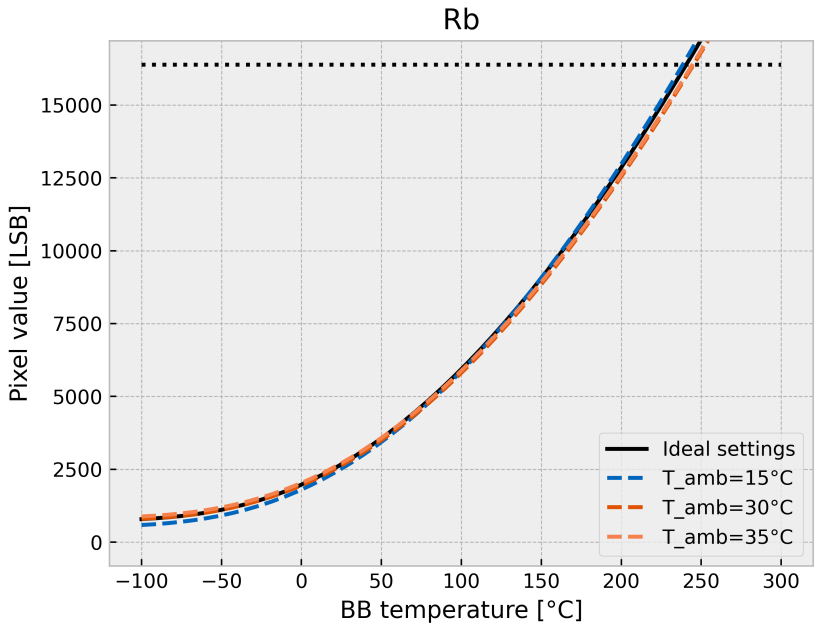


Figure 15: Ambient temperature compensation using time efficient configuration finding. The ideal settings use RBO parameters, which has been set to fulfill the assignment range.

are needed at each ambient temperature. The thermal compensation using these polynomials was evaluated at ambient temperatures 15 $°C$, 30 $°C$, 35 $°C$ and the result is shown in Figure 15. Because the RBO parameters were not measured during the configuration finding phase (at ambient temperatures 0 $°C$, 25 $°C$, 50 $°C$) it cannot be plotted in Figure 15. To enable comparison, the reference RBO parameters, which were used to determine the ADC values, were plotted by the black line and are denoted as ideal settings. As can be seen, the curves for 30 $°C$ and 35 $°C$ are very similar. Standard deviations for these three RBO parameters were calculated in a way similar to that presented in Section 2.6.3. The maximum standard deviation of 116 bits was found at BB temperature -50 $°C$, a minimum of 31 bits at BB temperature 93 $°C$, and the average standard deviation is 71 bits. The maximum standard deviation is 0.7 $\%$ of the ADC range. This result is slightly better than the result presented in Section 2.6.3, but less evaluation measurements were made.

2.8 Result of the ambient drift compensation

This section describes the measured behavior of the microbolometer array infrared detector. First, the measured ADC values corresponding to the measured BB temperature were interpolated with the line. However, despite the usage of linear approximation in the literature, the line does not fit the measured data. The reason why the linear approximation cannot be used in this thesis but other works used it may be caused by usage of a different sensor, or they have tested less BB temperatures and thus does not see the non-linearity. The special function presented in Equation 10 is used instead of the linear line. This function is usually used for radiometric temperature measurements. Based on this measurement, two ambient temperature drift compensation methods have been developed. The first is easy to find because it uses only the sensor parameter V_B, but does not support the temperature calculation. This method also suffers from changing sensitivity as the ambient temperature changes. The second method uses the parameters V_B and V_A. Due to

the difficult and time-consuming process of these configuration findings, the new faster method of finding the setting was introduced, which simplifies that process to 4 measurements on two BBs at three ambient temperatures. This method was tested at three ambient temperatures. The measured variance between the three measurement is only 0.7 % of the ADC range.

Chapter 3
Infrared camera system auto-focus

The goal of the automatic focus system is to automatically adjust the distance between the lens and the image sensor. An object of interest has to be well-focused in order to obtain all the details, which are essential for visual evaluation or data processing. Camera focusing is an important problem in both computer vision and microscopy. A sharp image is essential for thermal measurement. The fuzzy image does not look good, but the blur also increases the inaccuracies in the radiometry calculations. When the temperature is computed from a blurred image, the temperature seems to be lower because the IR flux is not concentrated in the proprietary pixels.

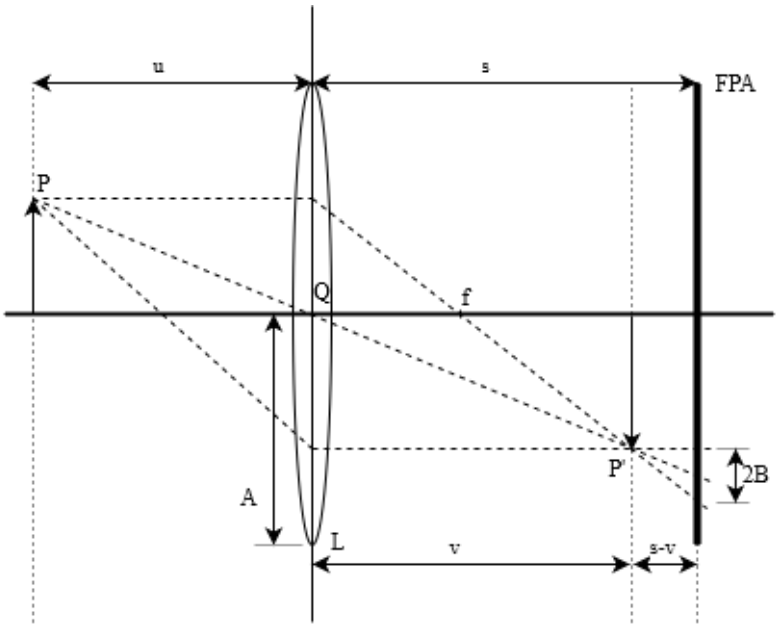


Figure 16: Lens model, where L is the lens, P object, P' focused point, A Aperture radius, Q Principal point, f focal length, FPA focal-plane array(sensor), B Blur circle radius.

The thin-lens camera model was presented in [32] and is illustrated in Figure 16. The thin lens approximation model neglects the lens thickness, which means that the first and second principal planes coincide in the lens. The lens thickness is the distance along the optical axis between the anterior and posterior faces of the lens. The more realistic model, which uses the thickness of the lens, is called a thick-lens model.

The thin convex lens is shown as an ellipse with center Q. Q is the principal point, which is the point at which the principal planes cross the optical axis. This point is identical to the nodal points, which are the points with angular magnification +1. (The ray aimed at this point will be reflected by the lens at the same angle with respect to the optical axis.)

The focal length f is the inverse of the system's optical power, which is a measure of how strongly the system converges or diverges light.

Let P be a point in the scene projected as a sharp point P' in an image plane. The relation between the position of P and P' is given by the lens formula:

$$\frac{1}{f} = \frac{1}{u} + \frac{1}{v} \tag{13}$$

where u is the distance between the lens center and the object plane and v is the distance between the lens center and the image plane. Because the focal plane array (FPA) is not placed in the image plane, the image will be blurred. The blurred effect has the shape of the aperture but is scaled by a factor. When the aperture is a circle with radius A, it will be seen as a circle with radius B on the sensor. Let q be the scaling factor defined as a ratio of those two radiuses. Using the similarity of the triangle and the substitution from 13, we obtain the following equation.

$$q = \frac{2B}{2A} = \frac{s-v}{v} = s\left(\frac{1}{v} - \frac{1}{s}\right) = s\left(\frac{1}{f} - \frac{1}{u} - \frac{1}{s}\right) \tag{14}$$

Equation 14 can be used to calculate the blurry radius:

$$B = A \cdot q = A \cdot s\left(\frac{1}{f} - \frac{1}{u} - \frac{1}{s}\right) \tag{15}$$

The diameter can be positive when the distance v is greater than s and negative otherwise. The camera system can be considered as focused on an object P, when the diameter 2B is smaller than the distance between two neighbouring pixels. This phenomenon also causes the depth of field, which is the distance between the nearest and farthest objects that are in acceptably sharp focus in an image.

When some blur radius is accepted (discussed later), the corresponding focus error **s-v** can be calculated from Equation 14. According to equation 13 *v* was replaced by $\frac{fu}{u-f}$. Then the f-number can be introduced as $f\sharp = \frac{f}{2A}$. The f-number is famous to the photographer as a lens property.

$$s - v = \frac{2B}{2A}v = 2B \cdot f\sharp\frac{u}{u-f} \approx 2Bf\sharp \quad , when\ u \gg f \tag{16}$$

In normal user case (not macro-photography), the distance between the camera and capturing subject is much larger than focal length. In that case, the result can be satisfied.

Ordinarily, the image is focused by screwing the lens closer or farther to the bolometer. However, when the camera is located in place, which is difficult to achieve or mount on an unmanned aerial vehicle, the motor must be used for focusing. For this reason, the Workswell Infrared Camera equipped with a motor was developed. This part of the thesis focuses on firmware related to the motor-focus camera version.

In regular Workswell Infrared Cameras with manual focus, the lens is screwed into a thread in the front of the microbolometer array. Workswell Infrared Cameras equipped with a motor can be equipped with a bayonet, which enables fast lens changes or the same screw thread front end as the manual focus camera. The bayonet does not allow the lens to change the focusing distance. If the non-bayonet lenses are used with motor focus, the lens has to be fully screwed down. Thus, only the motor can change the focusing distance.

Unlike visible cameras, the lens is permanently fixed and cannot change the focusing distance. Because the lens does not move, the only way to focus is to move with the microbolometer. The microbolometer is mounted on a metal support (in the figure shown in red). Movement is caused

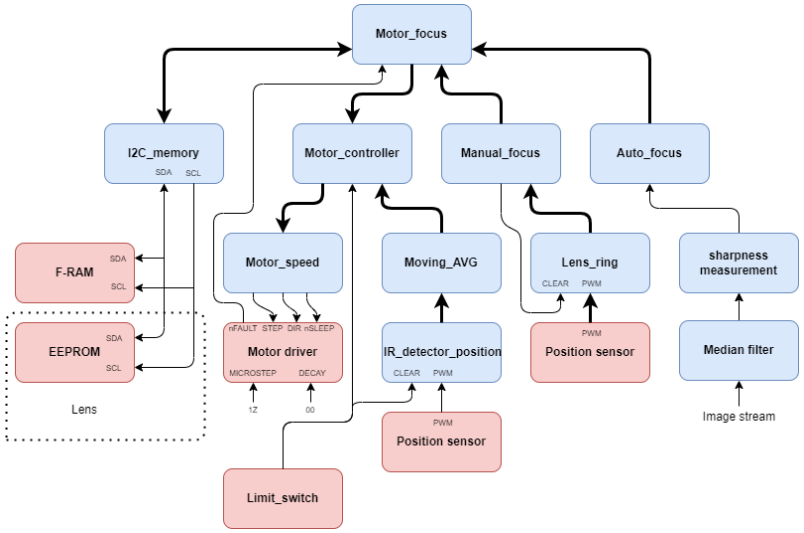


Figure 17: Diagram showing the motor focus firmware. Blue color shows the Verilog modules and red color stands for a hardware.

by a step motor located at the back of the camera. The motor is connected to the threaded rod, which rotates in a nut that is mounted on the bolometer support. To ensure stability and smooth operation, the sensor support is connected to linear bearings, which are mounted on the camera body.

The position of the microbolometer array is read using the magnetic linear encoder, which is located in the sensor support construction. The encoder reads a magnetic field from the hall elements, which is produced by a multipole magnetic strip located on the camera body. This magnetic field is converted in the linear encoder to pulse width modulation (PWM), which is read by the FPGA. The end switch is needed to determine the origin because the encoder measures the relative distance. Otherwise, the origin will be located at the place where the sensor was during the start-up. The same encoder is used to read the rotation of a special ring on the bayonet by which the user can set the focus. The bayonet also contains a connector, which can connect to the lens. The lenses sold with Workswell Infrared Cameras equipped with a bayonet have a built-in thermometer and electrically erasable programmable read-only memory (EEPROM). This memory can be used to determine which lens is connected to the camera and to store information connected to the lens.

The firmware that controls the motor and allows manual motor focusing (setting the focus distance with the bayonet ring) was implemented in Verilog [4].

The diagram in Figure 17 shows the structure of the motor focus firmware. The main module named Motor focus implements the interface with the rest of the firmware and decides which strategy of focusing should be used. A user can choose a focus strategy from the eight possibilities shown in the table2. The focus strategy can be changed by the existing communication protocol.

The first two focus modes set the sensor distance, which is sent via the communication protocol. They differ only in the range because the user mode enables us to set the distance only in the range from the Nearest Focus Distance (NFD) to the Infinite Focus Distance (IFD) for better user usability. In contrast, the Administrator mode allows to set any possible distance, which should be used to find the NFD and IFD for a new lens.

[4] Verilog is a hardware description language defined in IEEE standard 1364-2005. [33]

Name	Note	Range
Administrator	The distance is set via the communication protocol	full range
User	The distance is set via the communication protocol	limited range
Manual motor focus	The IR sensor position is set from the bayonet ring	limited range
Position calibration	The IR sensor presses the end switch and returns to the previous position	
Infinite focus distance		fix sensor position
Nearest focus distance		fix sensor position
Macro focus distance		fix sensor position
Autofocus	The camera computes the focusing distance.	limited range

Table 2: List of all possible focussing strategies

The next focusing strategy is manual motor focus. This strategy allows the user to manually control the focus distance by rotating the bayonet ring. The bayonet ring uses the same magnetic linear encoder to read the position of the bolometer, which allows the use of the same code to calculate the distance. However, the code is used in a different way. Each time, when new data are processed, they are added to the target sensor position and the clear signal is triggered (the trigger set actual position as the origin). The target is limited to the space between the infinite focus distance and nearest focus distance. If the new target is above that value, it is set to those boundaries. This has two consequences: The user can rotate the ring over the limit position, and when he decides to change the direction, the target position will change immediately, and the target position register never overflows.

It is necessary to calibrate the sensor origin because the sensor position is measured as a relative shift from the previous position. This task can be performed in the position-calibration mode. When this mode is activated, the sensor goes as far as possible from the lens, where the end switch is located. The distance in which the switch was pressed is stored as the maximum sensor position. When the switch is pressed, the original focus strategy is re-entry to action.

There are special modes for setting special sensor positions (infinite focus distance, nearest focus distance, and macro focus distance), which are determined by the lens or camera manufacturer.

Last but not least, is the autofocus. This focus mode is used to find the best sensor position. The principle used is described in Section 3.2.5.

The chosen focus strategy is stored in FRAM (Ferroelectric Random Access Memory)[5] in order to be active when the next camera is on. There is a separate Verilog module (named FRAM) that implements the communication protocol with the external FRAM through I2C. This module creates three ports to communicate with a single FRAM hardware. Each port enables both reading and writing operations. Writing requests are stored in First in First Out (FIFO) memory. Thus, the writing request can appear all at once, in contrast to the reading request. The module has a busy signal because the currently active reading request needs to be processed before a new reading request is started. When the busy signal is high, other read requests have to wait.

As mentioned previously, the bayonet lenses are equipped with EEPROM. Because EEPROM and FRAM use a similar communication protocol, the Verilog code was parameterized, and both EEPROM and FRAM share the code.

[5]FRAM is non-volatile random-access memory with similar functionality as flash memory, but much greater maximum read/write endurance.

Another important block of the firmware is the motor controller. This module regulates the motor speed based on the actual position of the sensor and the target position, with one exception: If the target distance is set to zero, the motor will rotate unless the end switch is pressed, regardless of the actual position of the bolometer. This function enables the position sensor to be calibrated and find the origin at start-up. Another feature of this module is the slow increase in motor speed to achieve the maximum possible speed. If the speed changes quickly, the rotor stops following the stator magnetic field. This module uses two submodules, the motor speed, which translates the speed register into the signals, which controls the dual-H-bridge current control motor driver, and the IR detector position module, which reads the magnetic linear encoder. The linear encoder produces a PWM signal directly proportional to the distance in one dipole. The module computes the distance in micrometres from the data and integrates the distance to produce the absolute position. When the clear signal is set to high, the absolute position is set to zero. After smoothing the output of this module with the moving average filter, the data are used as the actual sensor position.

3.1 Known auto-focus principals

When a camera system is equipped with a motor that can change the distance between the lens and the infrared sensor, the automatic focus procedure can be used. Automatic focus prevents the user from tediously manually trimming the optics until a satisfactory sharpness is reached. Manual focus depends on the user, which reveals threat such as insufficient qualification or sight problems.

The user is not the sole source of errors that cause the blurred image. The viewfinder screen is essential when manual focus is in action. The display has to have at least the same resolution and sufficient size to enable the user to see all the details.

On the other hand, a human operator may decide which object should be in the focus, which is essential when mere objects in different positions are presented in the scene.

In general, there are two possible metrics to determine the position of the sensor. The first is to directly determine how sharp the image is. The second possibility is to measure the distance between the camera and the subject in the scene. The distance between the lens and the subject can be calculated according to Equation 13, which can be used to change the problem of measuring the sharpness of the image to measure the distance from the subject. The distance of the subject can be measured in many ways, for example, using a measuring tape. Some smarter and frequently used methods in optics are presented below.

If the dimension of the subject in front of the camera is known, trigonometry can be used to calculate the distance. This principle is used in the rifle scope to determine the distance to the target, which is important for long-range shooting. The shooter knows the physical dimension of the target and can measure the angle the subject covers in the image using marks on the reticle.

The old film camera uses this principle as well. The camera was equipped with two windows, a viewfinder and a rangefinder. The image of a rangefinder is reflected by the mirror and combined with the viewfinder image. The mirror can rotate and is mechanically connected to the lens optic system. Focus shifts the rangefinder image to the left or right against the viewfinder image. When the image from the rangefinder and viewfinder overlaps, the scene is focused.

Distance can be measured using the time-of-flight principle. The rangefinder sends a pulse of light or ultrasound and tracks the time for which the reflected pulse is received. The measured time is multiplied by the propagation speed and divided by two to obtain the distance. Based on this distance, the sensor position is set. Cameras with such a rangefinder work even if a poor quality image is available. Automatic focus systems that need an energy-transmitting device to determine the focus are called Active Autofocus systems.

Modern digital single-lens reflex cameras use phase-detect autofocus. It is essential to see light as a ray to understand the principle of phase detection autofocus. The light beams from the subject hit the lens, which bands them and concentrates them into the sensor. Before hitting the sensor, the light is reflected by a semipermeable mirror. Behind this semi-permeable mirror, a prism is located within the same optical distance as a sensor. The prism bends the light (but, unlike the lens, does not focus), which makes only one specific ray pass through. The filtrated ray hits the one-dimensional photodiode array. If the camera is focused, the light beam hits the array in the middle. When the camera is not focused, the light beam is offset. The offset is related to the sharpness and enables the calculation of both the direction and the amount of sensor movement, which means that the camera can focus on one single measurement. The modern camera typically contains more sensors to choose the focal point.

Unfortunately, none of those methods can be used in this thesis because only the image can be used.

3.2 Image sharpness measures

The Workswell Infrared Camera performs automatic focus only based on the information provided by the microbolometer array, which is the image. The image is a two-dimensional array of 14-bit gray lavel. The following part assumes that the scene is constant, contains only one object, and the camera is fixed. The gray level of a certain pixel at coordinates i, j in the image (N x M array) and the focus distance z is denoted *G(i,j,z)*. The aim of this section is to find a sharpness function $F_{algorithm}(z)$ of the distance between the camera lens and its imaging sensor, which converts this 2D array into a number for each focus distance. There are several demands on the focus function presented in [34]:

- **Unimodality:** The focus function must be uinmodal, which means that it should have only one extreme (minimum or maximum). This prevents errors in the local extreme.
- **Reproducibility:** The result of the function should give the same results for a data set regardless of the time it was captured and computed. The sharp extreme of a function ensures that the extreme is always found.
- **Accuracy:** The extreme described above must be presented when the system is in focus.
- **Range:** The function should be used in all areas where the image could be focused.
- **General applicability:** The focus function should not be limited to a special type of images.
- **Insensitivity to other parameters:** The autofocus process should be resistant to changes in the parameters, which does not influence the focus position (e.g. mean brightness).
- **Video signal compatibility:** The automatic focus function usually works at the same time with the same data as the functions to process and display the image. It is important that those systems do not negatively influence each other. When the focus detector is different from the image detector, the distance has to be adjusted in such a way that the image detector is in focus, which avoids the systematic focus error.
- **Implementation:** The system must be easy to implement(in this thesis developed on FPGA). Moreover, in this case, there is a requirement for real time oration.
- **Noise robustness:** The function should be robust to noise. Even if the non uniformity correction (NUC) is performed, there is still present residual fix pattern noise. Moreover, during live time, new dead pixels may appear.

The problem of automatic focusing is well-known in visible photography and microscopy, and there is a wide range of literature available. In the field of infrared imaging, only a few works have been presented. In [35] the authors present five different functions using a TESTO 880-3 equipped uncooled 160x120 pixels detector sensitive to the range of 8 to 14 μm. They find out that the method of computing sharpness form image is possible for infrared cameras and recommends using three algorithms: energy of gradient, energy of Laplacian, and sum of modified Laplacian. They also claim that the Tenegrad operator does not work properly for low-resolution images.

In [36] the possibility of focusing for face recognition using the FLIR thermoVision A40M camera with a spectral range of 7.5 to 13 μm and a resolution of 320 x 240 pixels. Their conclusion is that the proper Tenegrad algorithm shows reasonably good focusing results.

Because the found scientific articles focusing on infrared implement only a few algorithms, more algorithms are presented, which were used in visible light automatic focusing solutions. As the implementation in FPGA is time-consuming, the algorithms were implemented in Python with the usage of publicly available libraries, mainly OpenCV [37], numpy [30], SciPy [38], and matplotlib [31]) for visualization. This script can be seen in the Appendix A.2. These algorithms were tested on five data sets captured by the Workswell Infrared Camera with different lenses and a visible control data set from [39]. The IR data sets were obtained when the IR camera was stationary and only the focus was changing. The sensor was moved within the entire area limited only by mechanics (0 - 3200 μm) with the largest step of 100 μm, but when the image began looking focused, the step was reduced to 50 μm, 10 μm or even 5 μm.

All photos of all data sets were manually evaluated, and the subjectively sharpest image was chosen. This image is shown in Figure 33. For better image contrast, histogram equalization was performed on this image. The equalization of the histogram changes the distribution of the histogram to the full range of intensities. The data sets are labeled with letters A to F. The first data set was captured with the f=14 mm lens. The object is the BB with temperature 100 $°C$ covered by a special layer, which increases the number of edges in the image. The distance between the camera and the BB was 70 cm. The second data set was captured with the same sensor configuration, the same lens, and the same BB. Compared to data set A, the BB temperature is lower (75 $°C$), the layer that adds edges does not apply, and the distance is smaller (50 cm). Data sets C and D use the same BB covered by the 1951 USAF resolution test chart [6]. The black body has a temperature of 90 $°C$ in both cases. Data set C was captured from this BB from distance 300 cm with the f=25 mm lens. The distance between the camera and the BB was 200 cm and the lens f=7.5 mm was used for the data set D. Data set E shows the author of the thesis sitting in a chair approximately 100 cm from the camera. To capture this data set, an f=25 mm lens was used. This data set contains the lowest contrast. Moreover, the data sets contain invalid pixels in the upper right corner. The data set can be affected by the movement of the subject. The last is the visible data set, which was used only to test the correctness of the algorithm implementation.

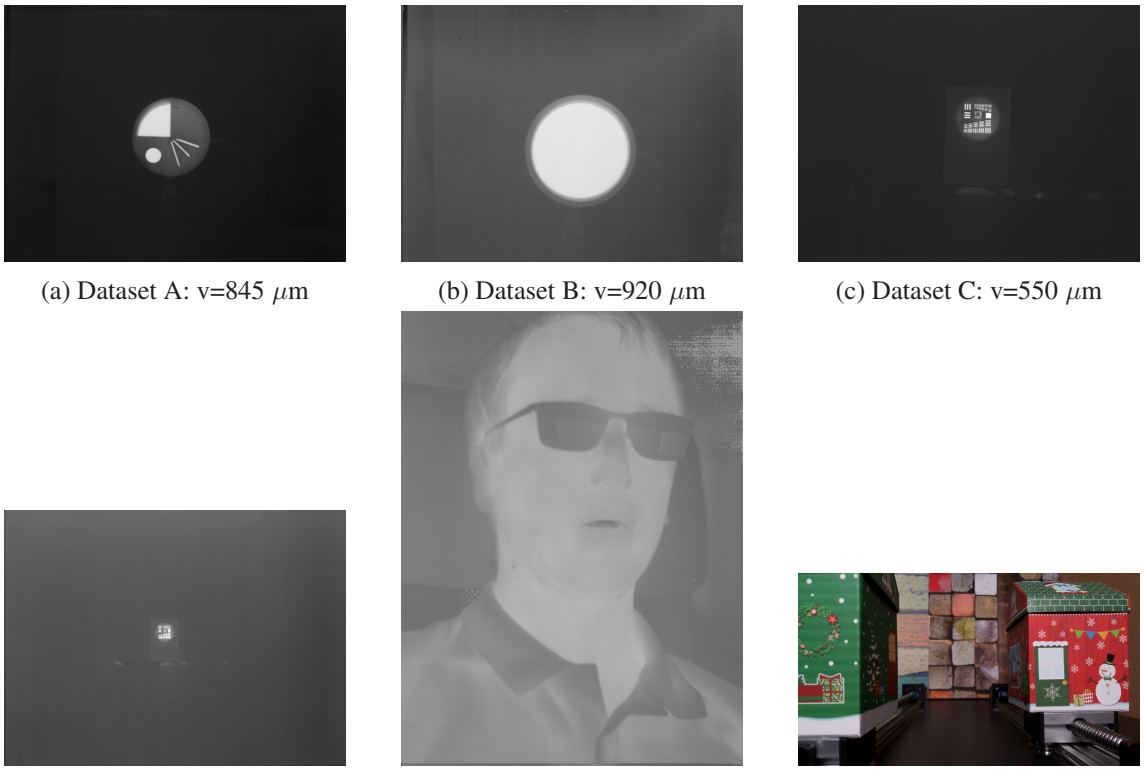
In the following three sections 3.2.1, 3.2.2, and 3.2.3 the sharpness measure function is described.

3.2.1 Statistic based focus metrics function

Firstly, the simpler methods will be described. These methods are based on the image statistic.

Mortimer L. Mendelsohn and Brian H. Mayall [40] presented in 1971 the image threshold counter. They assume that poor focus causes the removal of optical density. Their method integrates the amount of grayness that exceeds a preset reference Ψ for all points in the image. A higher value of the function 17 means a sharper image.

[6] The 1951 USAF resolution test chart is used for resolving the power tests. The test target was constructed according to the MIL-STD-150A standard.



(a) Dataset A: v=845 μm

(b) Dataset B: v=920 μm

(c) Dataset C: v=550 μm

(d) Dataset D: v=740 μm

(e) Dataset E: v=1050 μm

(f) Dataset F

Figure 18: One figure from each data set showing how the scene looks, when the image is in focus. The sensors position v is listed in the description for each infrared image.

$$F_{Meldelson}(z) = \sum_{j=0}^{M}\sum_{i=0}^{N}(G(i,j,z) - \Psi) \qquad , when\ G(i,j,z) > \Psi \tag{17}$$

The authors recommend setting the threshold Ψ somewhere in the mid range of the optical density of the image. In this thesis, Ψ was set at a value of 40 (255 is the maximum value in the Python test procedure), which is slightly above the median of all frames.

In [41] the authors sum the squared values of all the pixel intensities with the same motivation.

$$F_{ImagePower}(z) = \sum_{j=0}^{M}\sum_{i=0}^{N}G(i,j,z)^2 \tag{18}$$

In [32] the closely related focus measure is presented. The measure is gray-level variance, which is a linear and monotonic function of image energy computed as:

$$F_{Variance}(z) = \frac{1}{NM}\sum_{j=0}^{M}\sum_{i=0}^{N}(G(i,j,z) - \mu(z))^2 \tag{19}$$

,where $\mu(z)$ is the mean value:

$$\mu(z) = \frac{1}{NM}\sum_{j=0}^{M}\sum_{i=0}^{N}(G(i,j,z))\,. \tag{20}$$

Variance measures how far a set of numbers is spread out from its mean value. The sharp image should contain more information, which should cause greater energy variance. In contrast to previously presented methods, bright and dark pixels have the same influence. Because the division requires a significant amount of hardware resources, it may be useful to omit it when it is implemented in FPGA. Because the number of all pixels is constant, the division performs only a scaling and does not affect the ideal focus position.

The energy of variance can be further improved by dividing the sum by the mean value of the image, as presented in [41]. This should compensate for the difference in mean image brightness among different images.

$$F_{NormVar}(z) = \frac{1}{NM\mu(z)}\sum_{j=0}^{M}\sum_{i=0}^{N}(G(i,j,z) - \mu(z))^2 \tag{21}$$

Another approach was presented in 1988 by Vollath [42]. The contrast measurement is achieved by taking an autocorrelation of the object scene. When the edge appears in the image, the auto correlation creates a peak. According to the author, this approach should perform well in the presence of noise. The sharpness function is defined as

$$F_{Autocoleration}(z) = \sum_{j=0}^{M-1}\sum_{i=0}^{N}G(i,j,z)\cdot G(i+1,j,z) - \sum_{j=0}^{M-2}\sum_{i}^{N}G(i,j,z)\cdot G(i+2,j,z). \tag{22}$$

The same author [42] also describes the standard deviation based on autocorrelation.

$$F_{StdCorrelation}(z) = \sum_{j=0}^{M}\sum_{i=0}^{N-1}G(i,j,z)G(i,j+1,z) - \mu. \tag{23}$$

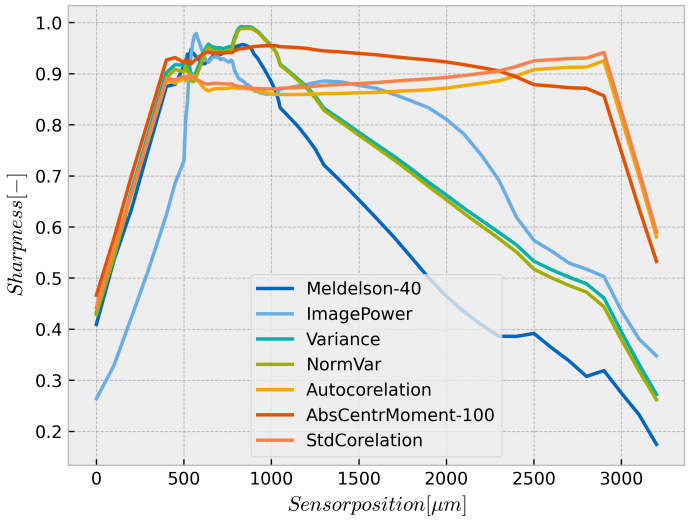


Figure 19: Result of the statistic based focus metrics function.

The last methods presented in this section are based on the histogram H(z). If the image is completely out of focus, all pixels have a very similar value. On the contrary, the sharp image consists of many different values of pixels. In [43] the difference between the maximum gray level and the minimum gray level was presented. This method does not fit to infrared cameras because new dead pixels (both at maximum and minimum) can appear during the live time of the camera. Those dead pixels will ruin this automatic focus mechanism. [44] proposed an absolute central moment:

$$F_{AbsCentrMoment}(z) = \sum_{k=1}^{L} |k - \mu| \, P_k \tag{24}$$

where μ is the mean intensity of the histogram H(z), L is the number of gray levels in the image and P_k the relative frequency of the gray level k (bin). The number of bins was chosen to be 100.

All metrics presented in this subsection were implemented in Python and tested on the data sets. The result of the first data set can be seen in Figure 19. Because the different functions have different maximal values, all were normalized to have maximum sharpness at 1. Because all curves were noisy, which can cause problems with local extrema, the moving mean of window size eight was used to smooth out the data. The image of the first data set is considered sharp when the focal distance is in the range of 780 to 940 μm, which was determined by subjective human criteria. The best result can be seen for the energy variance ($F_{variance}$) and the normalized energy variance ($F_{NormVar}$) with a peak in the range of 810 to 890 μm. Next, the Meldelson algorithm found the sharpness point correctly, but local maximum at 530 μm. The image power has a peak at 570 μm, which is out of focus. The remaining methods based on autocorlation and histogram were found to be not suitable for infrared cameras because they do not contain any peak.

3.2.2 Gradient based sharpness measure methods

Sharp image consists of sharp edges. Edge is a rapid change in image density within a small region. The edge in the figure may be caused by a depth discontinuity (object boundaries), a different material (emissivity), a change in surface temperature, or variations in scene illumination. Edge

detection is a fundamental tool for computer vision because it can be used for image segmentation, data extraction object boundary detection, and feature detection and extraction.

In 1976 the first gradient method was published [45]. This simple method measures the mean change in the gray level between pairs of points separated by *n* pixels. The maximum change is when focus is reached.

$$f_{Brenner}(z) = \sum_{j=0}^{M} \sum_{i=0}^{N-n} \left(G(i,j,z) - G(i+n,y,z)\right)^2 \tag{25}$$

where *n* is a small integer. As can be seen, this method computes the gradient only with respect to one axis. However, nowadays the computational power enables us to compute the gradient with respect to both axis. This is presented in [32] :

$$f_{GradientAVG}(z) = \sum_{j=0}^{M-1} \sum_{i=0}^{N-1} \left(g_x(i,j,z)^2 + g_y(i,j,z)^2\right) \tag{26}$$

, where the gradient with respect to the x axis g_x and the y axis g_y is calculated:

$$g_x(i,j,z) = \left(G(i,j,z) - G(i+1,j,z)\right) \tag{27}$$

$$g_y(i,j,z) = \left(G(i,j,z) - G(i,j+1,z)\right) \tag{28}$$

. The Tenengrad algorithm [46] showed that the gradient may be computed as a convolution with a Sobel operator listed in equation 31. Furthermore, they proposed a way to evaluate the gradient matrix:

$$f_{grad-threshold} = \sum_{j}^{O} \sum_{i}^{P} S(i,j,z) \qquad for\ S(i,j,z) > T \tag{29}$$

Here, the gradient matrix S(i,j,z) is computed as

$$S(i,j,z) = \left(s_x(i,j,z)^2 + s_y(i,j,z)^2\right) \tag{30}$$

where $s_x(i,j,z)$ and $s_y(i,j,z)$ are elements from convolution of the image and the convolution kernel S_{Sobel_x} and S_{Sobel_y}.

$$S_{Sobel3_x} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}; \qquad S_{Sobel3_y} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{31}$$

The Tenegrad algorithm sums all pixels that are above a certain threshold. The effect on the resulting curve is shown in Figure 20. If the threshold is equivalent to zero, all pixels are summed, which is the energy of the gradient. The low threshold is more sensitive to noise. The higher threshold causes more pixels to be filtered out and thus the peak is sharper. The disadvantage of a high threshold is that the value of the function when the image is not sharp equals zero. On the basis of this graph, the value of 10 was chosen because it creates a smooth function with a sharp peak.

The second method presented [46], how to evaluate the Sobel matrix, is by using standard variation. Because the operator highlights the edges, a high standard deviation means high focus and vice versa. The gradient operator is computed according to this equation.

$$f_{grad-variance}(z) = \sum_{j=0}^{O} \sum_{i=0}^{P} \left(S(i,j,z) - \bar{S}\right)^2 \tag{32}$$

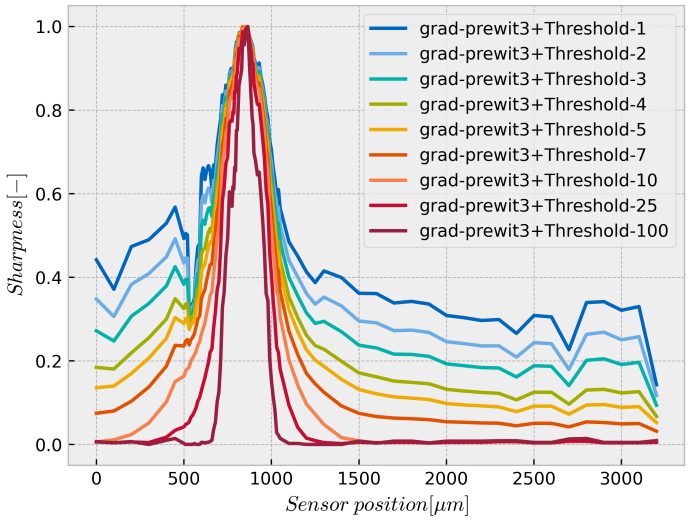


Figure 20: Threshold effect on Tenegrad algorithm.

, where O is the number of lines in the Soble matrix S (when the kernel size is 3x3: O = M-2) and P is the number of columns (P = N-2 for the 3x3 kernel). The $\bar{S}$ is the mean value of the matrix S calculated as

$$\bar{S} = \frac{1}{OP} \sum_{j=0}^{O} \sum_{i=0}^{P} S(i,j,z) \tag{33}$$

.

The Sobel operator performs a two-dimensional spatial gradient computation. The value, which is proportional to the magnitude of the gradient, was used in the two last methods presented. The magnitude can be computed by extracting the root of the equation 30. However, the Sobel operator can also determine the direction of the gradient.

$$\theta = atan2((s_y(i,j,z), s_x(i,j,z)) \tag{34}$$

This can be used for filtering, if an edge with a specific angle is found.

The convolution kernel that performs the derivation with respect to the y-axis is the transposed kernel, which performs the derivation with respect to the x-axis, as can be seen in Equation 31.

There are also other matrices that approximate the derivation when they are convoluted with the image. In this work, the convolution kernels from [47], [48], [49] are presented and tested.

The smallest convolution mask used is **Robert operator**. It is a 2x2 matrix, which approximates the gradient of an image through discrete differentiation.

$$S_{Roberts_x} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{35}$$

The 3x3 matrix is often used to calculate the gradient. The **Prewitt operator** takes the three elements that are on the right from the actual position in the picture and subtracts from them the three elements that are on the left to compute the horizontal derivation. The current position column is not used in this gradient computation. The **Sobel operator** is very similar, with one difference, Sobel operators put more weight to the actual row, where the elements are two times bigger. The third operator based on the same principle but with different weights is **Scharr**. Unlike

the previous 3x3 operators, the **Kirsch** matrix is not symmetric. It can be used to find the edge in eight compass directions by rotating the kernel in a 45-degree increment. The 3x3 kernels can be seen in Equation 31 and Equation 36.

$$S_{Prewitt3_x} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}; \quad S_{scharr3_x} = \begin{bmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{bmatrix}; \quad S_{Kirsch3_x} = \begin{bmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{bmatrix} \tag{36}$$

These kernels can be extended to size 5x5. The **extended Prewitt** operator has the same weight for all pixels in the same columns. The farther columns have more weight. The **extended Sobel** is modified Prewitt operator. The coefficients of the Prewitt kernel are divided by the distance from the center. Distance is calculated as the sum of the squared index difference. For example, the greatest distance from the center is at the corners, which can be computed as $2^2 + 2^2 = 8$. The distance of the next element is $2^2 + 2^1 = 5$, etc. For a more efficient implementation, the entire kernel is multiplied by the least common multiple (20). In [48] the extended Kirsch and Scharr operators are presented. The 5x5 kernels are shown in Equation 37.

$$\begin{aligned} S_{Prewitt5_x} &= \begin{bmatrix} -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \end{bmatrix}; \quad S_{Sobel5_x} = \begin{bmatrix} -5 & -4 & 0 & 4 & 5 \\ -8 & -10 & 0 & 8 & 10 \\ -10 & -20 & 0 & 20 & 10 \\ -8 & -10 & 0 & 8 & 10 \\ -5 & -4 & 0 & 4 & 5 \end{bmatrix} \\ S_{Scharr5_x} &= \begin{bmatrix} -1 & -1 & 0 & 1 & 1 \\ -2 & -2 & 0 & 2 & 2 \\ -3 & -6 & 0 & 6 & 3 \\ -2 & -2 & 0 & 2 & 2 \\ -1 & -1 & 0 & 1 & 1 \end{bmatrix}; \quad S_{Kirsch5_x} = \begin{bmatrix} -7 & -7 & -7 & 9 & 9 \\ -7 & -3 & -3 & 5 & 9 \\ -7 & -3 & 0 & 5 & 9 \\ -7 & -3 & -3 & 5 & 9 \\ -7 & -7 & -7 & 9 & 9 \end{bmatrix} \end{aligned} \tag{37}$$

Another approach to enlarge the convolution kernel in order to benefit from a higher neighborhood is dilating. The newly added positions of the dilated kernel are considered as gaps and set to zero. This principle is demonstrated in Equation 38 using a generic matrix.

$$S_{generic} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}; \quad S_{Dilated} = \begin{bmatrix} a & 0 & b & 0 & c \\ 0 & 0 & 0 & 0 & 0 \\ d & 0 & e & 0 & f \\ 0 & 0 & 0 & 0 & 0 \\ g & 0 & h & 0 & i \end{bmatrix} \tag{38}$$

The dilatation, in contrast to the extension, does not add mathematical operations when increasing the kernel.

All described kernels were used to calculate the edge maps. In Figure 21 the first edge maps of the data set are presented. The first image 21a shows the black body with special mask capture using the Workswell Infrared Camera at sensor position 845 μm, where the camera system is considered focused. For better image contrast, histogram equalization was performed on this image. All other images in Figure 21 were calculated from this image without histogram equalization. In those images, the edges are shown in gray level; darker color means more significant edge.

The last image in the first row presents the effect of the Roberts operator. The image looks completely white, but when the image is zoomed in, the very light and narrow contour of the black body mask can be seen. Roberts is the smallest kernel used (only 2x2) and therefore has the lowest sensitivity.

The rest of the figure consists of three columns. The images in the first column were created by convolution with the original 3x3 gradient kernels. All convolution kernels presented in the first column can reveal the edge; however, the edge is light. In the second column, the kernels were dilated. This causes a small magnification in the intensity of the edges compared to the original kernel. The disadvantage may be that the edge is stretched over more pixels. The third column uses extended kernels. Extension causes a rapid increase in sensitivity. It turns out that all methods look similarly. The best edge map is obtained using extended kernels.

All edge maps were evaluated using the methods presented at the beginning of this section. For all Tenengrad algorithms, the same threshold (10) is used for a better comparison of kernels. To allow plotting more graphs into one figure, all data were normalized by dividing by the maximal value. The result of the evaluation can be seen in Figures 22, 23, and 24. The smoothing filter is not used, in contrast to statistical methods. All curves contain a peak at a similar distance. The image captured in those peaks can be considered sharp.

The original 3x3 kernels and the Roberts 2x2 operator are used in the first set of graphs shown in Figure 22. In contrast to the edge maps, which look very similar, the convolution kernel plays an important role in the quality of the method. The graph on the left shows the variation computed from the derivation. The difference between the 3x3 kernels is the ratio between the peak and the rest of the image, where the camera is not in focus. The only 2x2 kernel shown is the Roberts gradient operator marked by a yellow color. The variance of the gradient computed by the Roberts operator has low sensitivity and some local maxima, which makes focusing very difficult. On the contrary, when the threshold in the Tenengra algorithm is used, the main peak is magnified, and the resulting curve is smooth. Surprisingly, the Tenegrad using the Roberts operator is very similar to the Prewit derivation evaluated with variation, even though Roberts uses less than half an element for the edge-map computation.

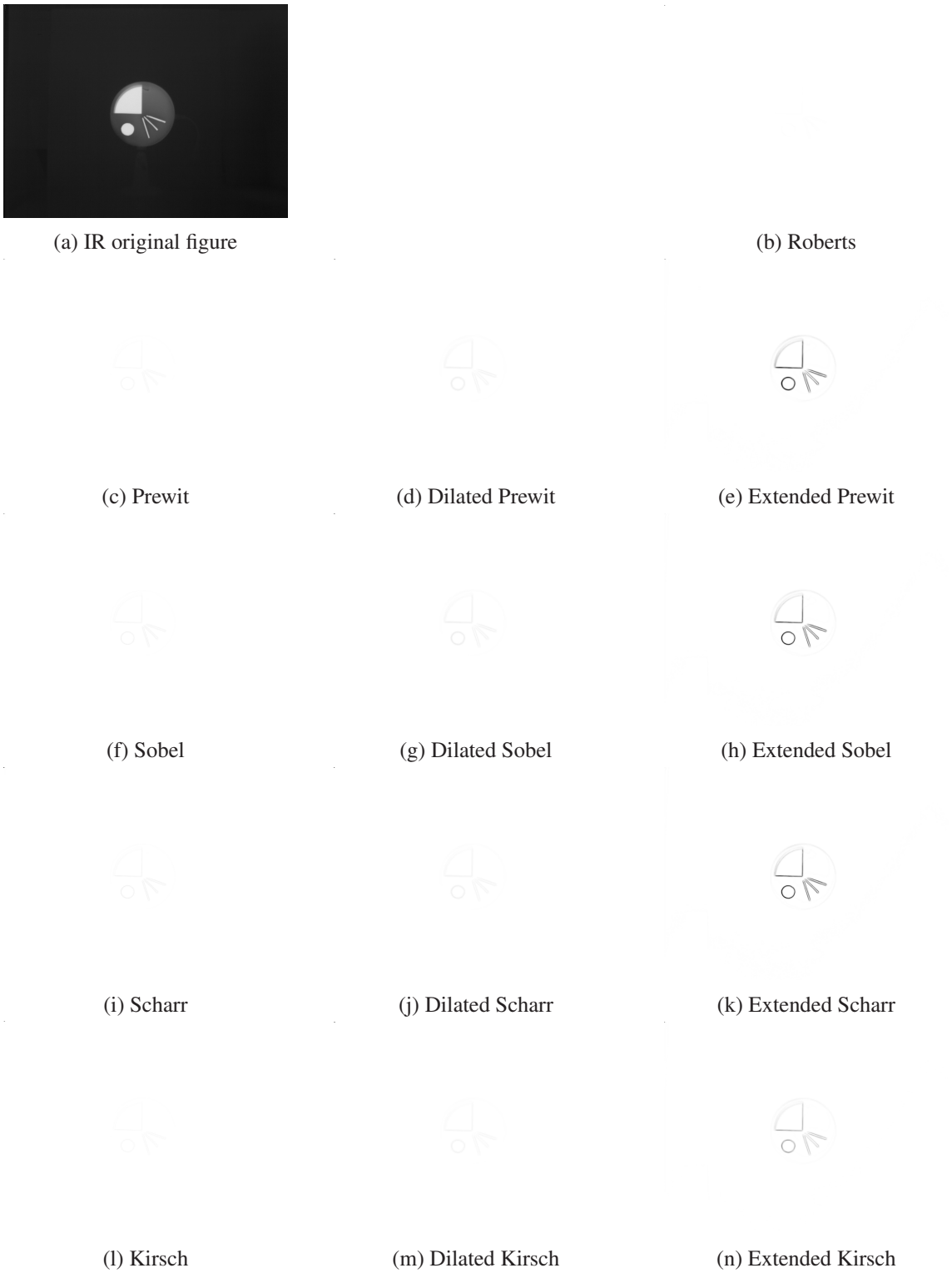
Figure 22b shows a considerable difference between the curve shape depending on the convolution kernel when the Tenengrad algorithm is used. The best curve is obtained using the Prewit kernel (dark blue), followed by the Sobel operator, which can be used on a wider area but is not that smooth. On the other hand, Scharr and Kirsch operators are not as good as 2x2 Roberts kernel, even if they use the 3x3 kernel.

In the next Figure 23 the kernel used was dilated. The dilatation kernels should be less sensitive to noise, which results in smoother curves in both images. The difference between the kernels is less significant. Only Scharr and Kirsch used by Tenegrad contain local maxima, which complicate the focus point finding.

Figure 24 presents the result when extended kernels are used. All convolution-extended kernels result in very similar curves.

All methods were further evaluated in the table 3. The table is organized as follows. The first column contains the convolution kernels. Then the table is divided into two parts. In the first part, the edge map obtained by the convolution is evaluated by the variation algorithm, and in the second by the Tenengrad algorithm. Each part consists of three columns. In the first column, the main peak is presented. This is the distance in which the image is in focus according to the algorithm used. This corresponds to the accuracy demand listed at the beginning of this section. The correct value is difficult to predict because the image looks sharp in a wide range around these values listed in the table and, therefore, all methods are considered accurate enough.

The second column presents the low range, which is the range in which the focusing criteria can be used. The range in which the function is not monotonous was selected to enable more efficient peak search algorithms. The dilated Sobel operator used in the Tenengrad algorithm has the widest range of operations. The standard 3x3 Sobel kernel has a similar performance. However, the variance evaluation criteria have a limited range. Especially when 3x3 kernels are



(a) IR original figure

(b) Roberts

(c) Prewit

(d) Dilated Prewit

(e) Extended Prewit

(f) Sobel

(g) Dilated Sobel

(h) Extended Sobel

(i) Scharr

(j) Dilated Scharr

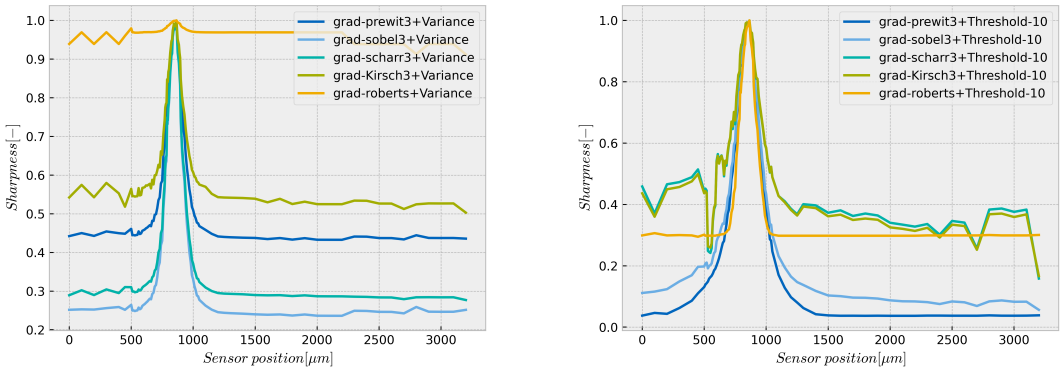
(k) Extended Scharr

(l) Kirsch

(m) Dilated Kirsch

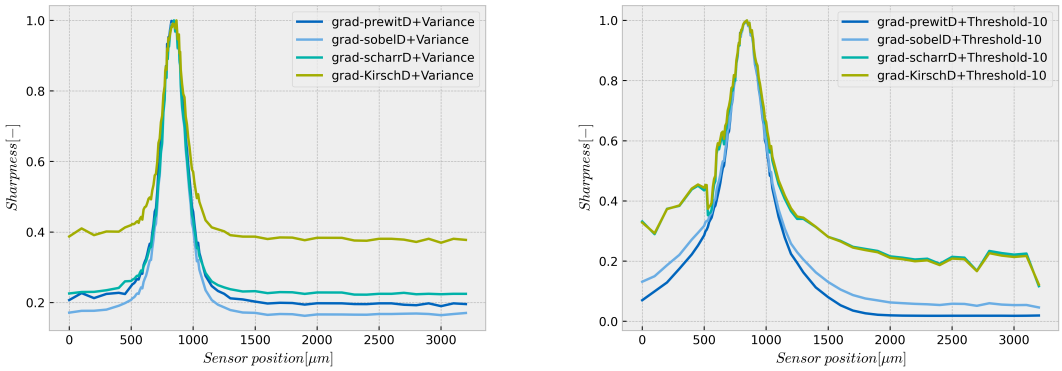
(n) Extended Kirsch

Figure 21: The result of gradient convolution operation with different kernels. The first image shows the photograph from which all edge maps were computed. The second image in the first row shows the 2x2 Roberts operator. The rest of the figure is organize as follows. In the first column, the 3x3 convolution filter is presented. In the second column, the filter from the first column is dilated and the third column consist of convolution with the extended filter.



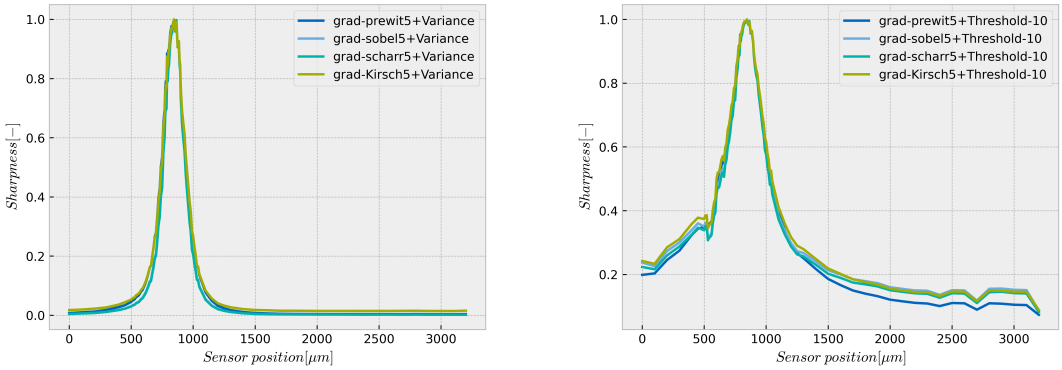
(a) Gradient variance (b) Tenegrad

Figure 22: Gradient methods with kernel size 3x3



(a) Gradient variance with dilated kernel (b) Tenegrad with dilated kernel

Figure 23: Gradient methods with dilated kernel.



(a) Gradient variance with Extended kernel (b) Tenegrad with Extended kernel

Figure 24: Gradient methods with extended kernel.

used, the range size is limited to approximately 220 μm.

The last column is named "high range" and presents how sharp the peak is. This parameter corresponds to the repeatability demand. This presents the distances at which the sharpness value is higher than ninety-five percent from the maximum value. The smaller interval means higher reproducibility. The smallest range has variance computed from 3x3 kernels. As can be seen, the variance criterion always has a smaller value than the Tenengrad presented. However, if a higher threshold is used, the range can be reduced.

Derivation kernel	Variance			Tenengrad		
	Peak [μm]	**Low range[μm]**	**High range[μm]**	**Peak [μm]**	**Low range[μm]**	**High range[μm]**
Roberts	865	830 - 980	830 - 980	865	750 - 960	835 - 865
Prewit	865	750 - 960	835 - 865	865	590 - 1050	825 - 875
Sobel	865	740 - 970	835 - 865	865	510 - 2400	825 - 875
Scharr	865	740 - 970	835 - 865	865	700 - 1250	815 - 885
Kirsch	865	740 - 970	835 - 865	865	700 - 1250	825 - 885
Dilated Prewit	845	640 - 1050	800 - 885	845	0 - 1700	790 - 885
Dilated Sobel	845	650 - 1040	815 - 865	845	0 - 2000	790 - 885
Dilated Scharr	845	670 - 1020	815 - 865	845	550 - 2000	790 - 885
Dilated Kirsch	865	680 - 1010	815- 885	845	550 - 2000	790 - 885
Extended Prewit	845	650 - 1040	815 - 865	845	550 - 2400	790 - 885
Extended Sobel	845	680 - 1020	825 - 865	845	550 - 2400	800 - 885
Extended Scharr	845	680 - 1010	825 - 865	845	550 - 2400	805 - 885
Extended Kirsch	845	660 - 1040	825 -865	845	550 - 2400	790 - 885

Table 3: Comparison of gradient methods

Another important aspect of the sharpness measurement function is the complexity of implementation. The variance is more complicated to calculate because the entire image has to be evaluated twice. First, the mean value of the pixels is calculated, which is used in the second run to compute the standard deviation of each pixel, which is then summed. Even though real-time estimation is used, the methods still need division, which cost a lot of resources. The Tenegrad algorithm returns the value in the single image processing and consists of one comparator and a sum. Thus, it is more suitable for real-time operations.

The next aspect is the convolution kernel. When the convolution kernel is 2x2, the result is computed on the basis of two rows. If the kernel is 3x3, one more row of the image has to be stored and used when the convolution is computed, and so on. The smaller convolution kernel needs less memory. Furthermore, when convolution is not implemented generally, but rather only one selected kernel is used, fewer resources are needed. Obviously, the Roberts operator needs fewer resources in order to implement the convolution with this kernel, only one operation is performed (from one element another element is subtracted). The next easy-to-implement operator is Prewit. The advantage of the Prewit operator is that it does not require multiplication because it performs only tree subtractions and additions of elements. The convolution with the Sobel operator can be effectively implemented without multiplication as well, but with one more subtractions and additions operation or a bit shift.

The dilated kernels only need more memory, but use the same computation mathematical operations as original kernels. Extended kernels are the most difficult to implement because they require the highest number of mathematical operations and the same amount of memory as dilated kernels.

3.2.3 Laplacian based sharpness measure methods

The previous section shows the approach of the first-order derivative. This section goes further and introduces the second-order derivative usage in the sharpness measure. The sharp image contains high frequency caused by sharp edges. The second-order derivation can be used as a high-pass filter. When the low frequency is removed, the data may be evaluated using the techniques presented above.

The Laplacian can be approximated in several ways. Similarly to gradient methods, convolution with various convolution kernels can be used. In [46] the highest weight is placed on the current pixel (the center of the convolution kernel). This element is multiplied by 4 (which can be implemented using a double left bit shift), and the 4-neighbors are subtracted from it. This can be implemented as a convolution with the kernel L_1 presented in Equation 42. This Laplacian approximation is referred to as Laplacian1 in this thesis. In [32] the Laplacian uses the 8-neighbors. The 4-neighbors have weights four times higher than the elements in the corners, and all the neighbors are subtracted from the current element, which is multiplied by twenty. When all pixels have the same value, the edge is not present and the result will be zero. This is represented as the convolution kernel L_2 in Equation 42. This Laplacian approximation is referred as Laplacian2 in this thesis.

$$L_1 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}; \qquad L_2 = \begin{bmatrix} -1 & -4 & -1 \\ -4 & 20 & -4 \\ -1 & -4 & -1 \end{bmatrix} \tag{39}$$

The Laplacian can result in a low value when two perpendicular edges with similar intensity are presented. For this reason, [50] modified the Laplacian as follows:

$$\begin{aligned} f_{ModifiedLaplacian1}(x,y,z) = &|2G(x,y,z) - G(x-1,y,z) - G(x-1,y,z)| + \\ &|2G(x,y,z) - G(x,y-1,z) - G(x,y-1,z)| \end{aligned} \tag{40}$$

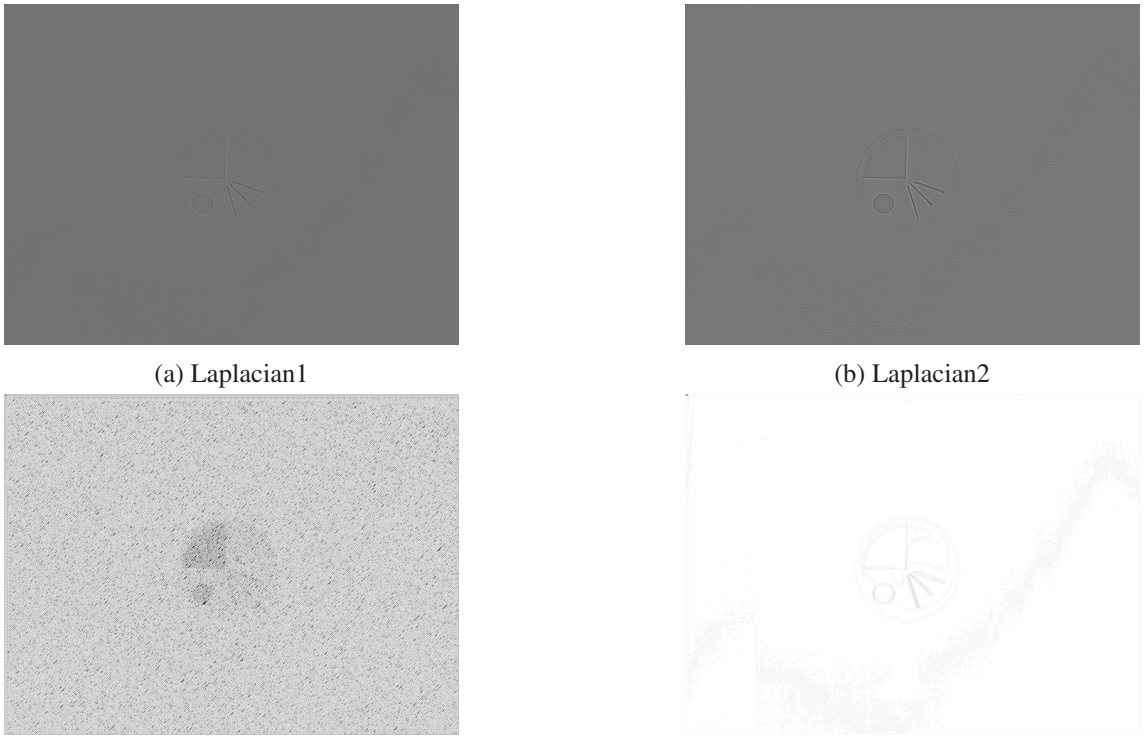
The modified Laplacian is always equal to or greater than the Laplacian computed by convolution with the kernel $L1$. This Laplacian approximation is referred to as ModifiedLaplacian1 in this thesis.

This method was further extended to the eight neighbors by [51].

$$\begin{aligned} f_{ModifiedLaplacian2}(x,y,z) = &|2G(x,y,z) - G(x+1,y,z) - G(x-1,y,z)| \\ &+ |2G(x,y,z) - G(x,y+1,z) - G(x,y-1,z)| \\ &+ \frac{1}{\sqrt{2}} |2G(x,y,z) - G(x+1,y+1,z) - G(x-1,y-1,z)| \\ &+ \frac{1}{\sqrt{2}} |2G(x,y,z) - G(x-1,y+1,z) - G(x+1,y-1,z)| \end{aligned} \tag{41}$$

The authors used weights that are difficult to implement in an FPGA. If this method yields good results, the weights will be modified as follows. The first two absolute values of the parts of equation 41, which correspond to the row and column, will be multiplied by four, and the second two correspond to the diagonal by square root. This editing scales up the result, and a slightly change the weights. When the center element is large enough to ensure that all absolute values are positive, the equation is identical to the convolution with the matrix L_1. However, when implemented in Python, the original method is used. This Laplacian approximation is referred to as ModifiedLaplacian2 in this thesis.

All four of these methods have been tested in the first data set, and the image computed at a distance of 845 μm is shown in Figure 25.



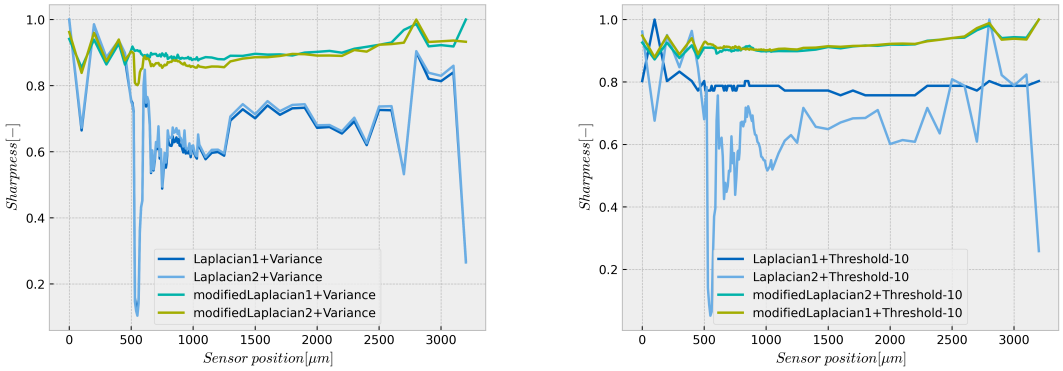
(a) Laplacian1

(b) Laplacian2

(c) ModifiedLaplacian1

(d) ModifiedLaplacian2

Figure 25: The result of Laplacian computation using different approximations. In the first column, the approximation uses only 4-neighbors and in the second column the diagonal elements are added to the computation. The first row was computed using convolution and the second row displays the extended Laplacian computation with the absolute value.



(a) Variance

(b) Threshold

Figure 26: Laplacian methods.

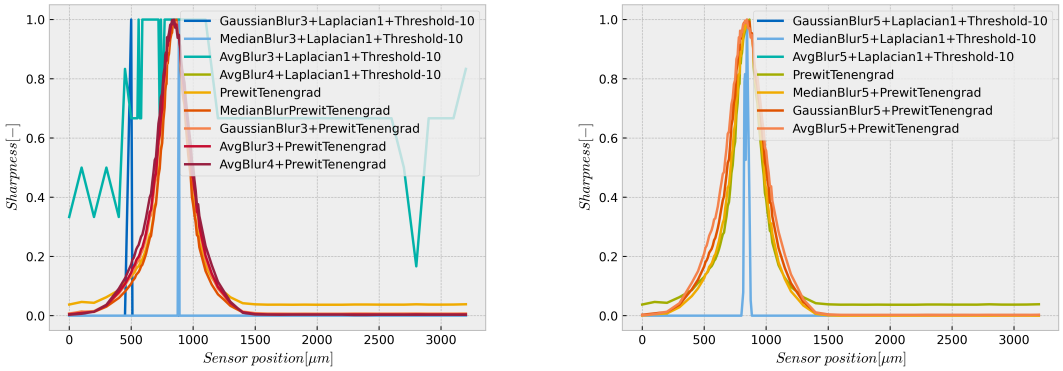
The first two images (25a and 25b) have a gray background. This gray is equivalent to zero value. The lighter pixels are negative, and the darker pixels are positive. The second two images (25c and 25d) are always positive, and thus the white is equivalent to zero. The Laplacian 2 has better highlighted the blackbody, which indicates more sensitivity. Modified Lapalcian 1 contains high-contrast noise.

Similarly to the gradient, the Laplacian can be evaluated using the variance [46] and the threshold. The results are shown in Figure 29. This result shows that none of the Laplacian-based methods is suitable for infrared focusing. Although more different threshold values including zero, which correspond to compute the energy, were tested, the ideal curve shape was not found. This may be caused by the noise because the IR image is never homogeneous, even if the nonuniformity correction is performed, there is still some residual noise pattern.

3.2.4 Image filtering

As mentioned, the residual noise after the nonuniformity correction is still present in the image, which can decrease the performance of the sharpness measure. For that reason, the blurring filter can improve the results of the metrics. Filter usage has previously been used in the literature; for example, the Canny edge detection algorithm uses the Gauss smoothing filter [52]. The three filters are presented in this thesis. The most simple is the mean filter, which replaces the pixel value with the mean value of its neighbors. It can be implemented as a convolution with an all-ones matrix and divided by the size of the kernel elements. The favorable kernels have a size of power of two, because when implementing such a kernel, the division can be replaced by bit shift, which is easy to implement. The second is the median filter, which replaces the pixel value with the median of the set window. This non-linear filter is well known for removing salt and pepper noise. The last filter used is the Gaussian blur filter. This filter can be approximated as a convolution with the following kernels [53]:

$$B_{gauss3} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}; \qquad B_{gauss5} = \frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix} \tag{42}$$



(a) Smaller filter kernel size

(b) 5x5 filter kernel size

Figure 27: Effect image filtering on the sharpness measure.

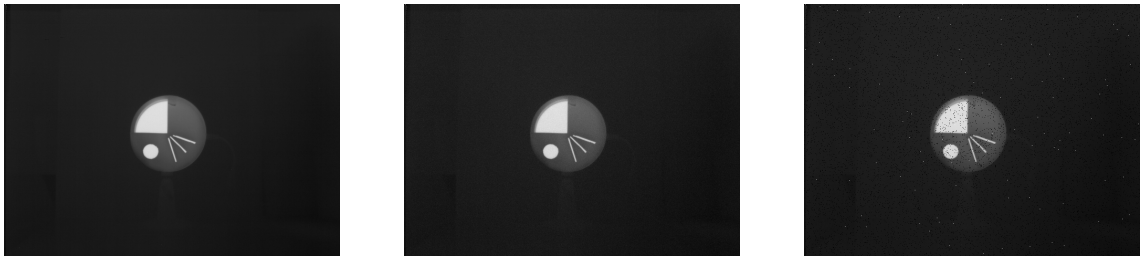
The effect of all filters has been tested in Python. The effect on Tenengrad obtained from the Prewit gradient and the first Laplacian method is presented in Figure 27. If the sensor is in an out-of-focus position and any of the presented filters are not applied, the sharpness measure is equal to zero with the exception of Laplacian and a filter with smaller kernel. However, the filter may increase the width of the main peak. If no filter is used, the main high-peak range is 50 μm in width. The same values hold for the 3x3 kernel size of all three filters. For the mean filter of kernel size 5x5 the range of peaks is 75 μm and for the 4x4 kernel size the range is 60 μm. The Gauss filter of the 5x5 kernel size has the biggest high range 85 μm. On the other hand, the median filter with the 5x5 kernel has a smaller range (40 μm) compared to the gradient computation of the original image.

When the image is filtered with the median filter of size 5x5, the Laplacian base method starts to work, as can be seen in Figure 27b. Even if a filter is used, the Laplacian methods do not perform better than the gradient-based sharpness measurement methods.

The filter can increase the noise resistance. The first noise tested is the salt and pepper noise. Salt and pepper noise is an impulse type of noise caused by a certain number of pixels stuck at a high or low value. This type of noise can be caused by dead pixels. This noise was simulated in Python by setting randomly chosen pixels to maximal or minimal values. The image with this noise can be seen in Figure 28c.

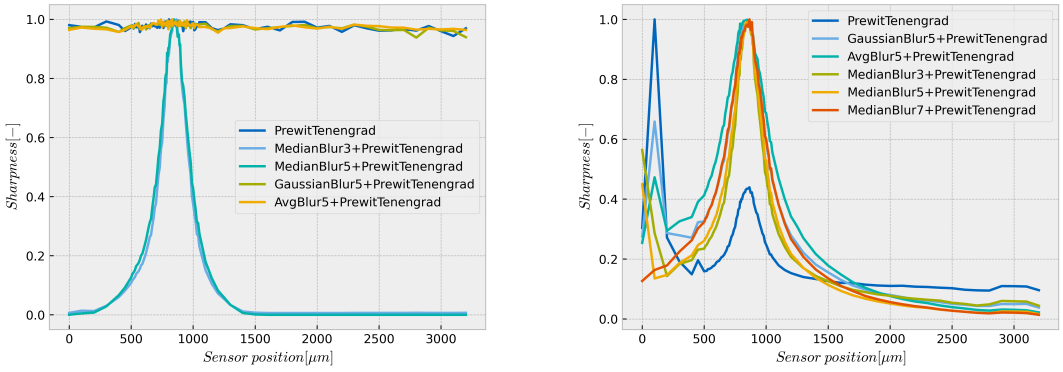
The image with added salt and pepper noise was used to test all filters. The filtered image is evaluated using the Tenengrad algorithm based on the Pewit operator. The result can be seen in Figure 29a. Only median filters can filter out the noise in such a way that sharpness can be measured. The edges caused by the salt and pepper noise have a higher value than the edges caused by the image for the Gaussian and also the mean filters.

The second added noise is Gaussian noise, which has a probability density function equal to the normal distribution. This noise was implemented as a sum of the original image and the noise matrix of the same shape as the image. The noise matrix was computed using the NumPy library [30]. Sharpness values were computed on the images with added Gaussian noise using all three filters and the Tenengrad algorithm computed from the Prewit operator. The result that can be seen in Figure 29b shows that if the image is used without filtering, the image will be considered focused at position 100 μm, which is far from the ideal 845 μm. When filters are used, the highest peak is always in 865 μm, which means automatic focusing is possible. If the median filter of kernel size 7 is used, then the function has only one maximum.



(a) Original image (b) Gaussian noise (c) Salt and pepper noise

Figure 28: Image with added noise.



(a) Salt and pepper noise (b) Gaussian noise

Figure 29: Added noise.

3.2.5 Auto focus implementation

The automatic focus described in this thesis finds the optimal sensor position once per trigger. This trigger can be run through the communication protocol. Modern visible DSLRs always provide a sharp image because the automatic focus procedure is always running. This can be achieved only because a special sensor is used for automatic focusing. Because the automatic focus described in this section uses the image from the infrared detector, the image goes blurred during the focusing. For this reason, the automatic focus procedure runs only on an user request. The automatic focus consists of three parts. First, the focus area has to be chosen. In this thesis, the whole image is used. This is typically done on processors to save computation time and to enable real-time operation, which is not a problem on FPGAs. However, memory resources can be saved when the area is reduced. The user can choose the area that should be in focus, which is useful, when more objects at different distances are presented in the scene.

The second part is the sharpness measurement. The greatest attention was paid to this part. Gradient-based sharpness measurement methods described in Section 3.2.2 show the best results. In general, when the gradient is evaluated by the threshold, sharpness metrics can be used in a wider range of sensor position. The second advantage compared to variance evaluation is the complexity of the calculation. On the other hand, the peak is wider, which could decrease repeatability. This is not a problem because the eye considers the image to be sharp in wider range. The Sobel and Prewit operators show the best results. Due to easier implementation, the Prewit kernel of 3x3 size was chosen.

Median filtering turns out to be effective in making the sharpness measurement immune to noise and should be applied before the sharpness measurement. The median filter with kernel size 3x3 performs well for salt and pepper noise and can also filter out Gaussian noise. Larger kernel sizes can eliminate small edges [54] and therefore the 3x3 kernel size was chosen.

The automatic focus module is connected to the video stream after the dead pixels are replaced and non-uniformity is corrected. The video stream consists of a two-byte video data register, a video clock register, and a *startNewFrame* signal. Each rising edge at the video clock new pixel data is valid. The pixels start from the upper left corner and continue to the right. After sending the first row, the second row starts from the left again, and so on.

Both the median filter and the sharpness measure use the 3x3 kernel. This kernel uses nine elements, three from each of the three following rows. First, those elements are separated from the image. This part is common for these two operations.

The schematic of these operations is shown in Figure 30. The process starts with the rising edge of the *startNewFrame* signal, when the column and row counter is set to zero. Then the row counter is enlarged by one each time new video data are received. When the counter equals 639 it is set to zero and one is added to column counter.

Each time the new pixel is received, it is stored in the FIFO1 memory. When the first line is saved in the FIFO1 memory, the read request is activated, and the line is saved into the FIFO2 memory. This memory is filled with this line similarly to the FIFO1 memory. Both FIFO memories used in this design are 16 bits wide and 1024 words wide implemented in one M10K block. When the third row starts, the read request for FIFO2 memory is enabled. Since then, all three rows can be used.

The output of the two FIFO memories is stored in the cascade of two registers. This cascade ensures the delay in the columns. The nine elements are ready to be used after 1282 pixels are received. At this time, the first element of the edge map is computed.

Before sharpness is computed, the image is smoothed by the median filter. The effective implementation of the median is presented in [54]. The median filter is built from a basic node that can sort two elements. This basic node is shown in Figure 31. The node consists of one

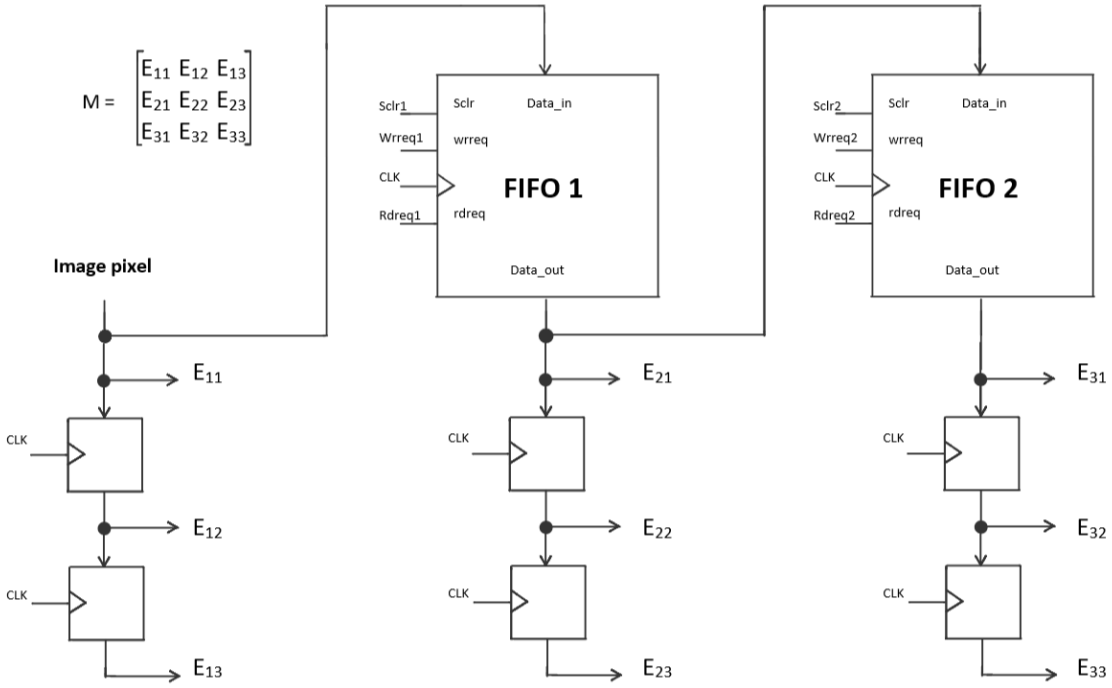


Figure 30: Implementation of element preparation for median filter and sharpness measure. This schematic spare 3x3 matrix M from the image.

two-byte comparator and two 2:1 multiplexers. Multiplexers are controlled by the output of the comparator.

The median value is the center value of the ascending-order array. The schema presented in [54] and shown in the figure 32 uses only 19 basic nodes instead of 41, which is required for the full sorted array. Eight of those elements can be implemented with one single multiplexer because the second output is not used.

The output of this filter is sent to the sharpness measurement function. This function implements the Prewitt operator in a simple way:

$$A_{Sharpness} = E_{13} - E_{11} + E_{23} - E_{21} + E_{33} - E_{31} \tag{43}$$

and uses the same structure for accessing the needed 3 x 3 matrix.

3.2.6 Peak search algorithms

The last part is the decision about which sensor position to set. This position has two categories, temporal and ideal. Temporal positions are used during the automatic focus procedure. The ideal sensor position is the position in which the sensor is the most sharp.

Some methods have been presented in the literature. The most intuitive search algorithm is the full sweep search (sometimes referred to as a global search). This algorithm visits all possible distances, measures the sharpness of this distance, and then returns to the distance with the highest sharpness. This method always finds the best sensor position. On the other hand, this algorithm is very time-consuming.

Krotkov [55] solves this problem using the Fibonacci sequence. The Fibonacci sequence can be defined as a recurrence relation $F_k = F_{k-1} + F_{k-2}$ for k $>$ 1 and the boundary condition

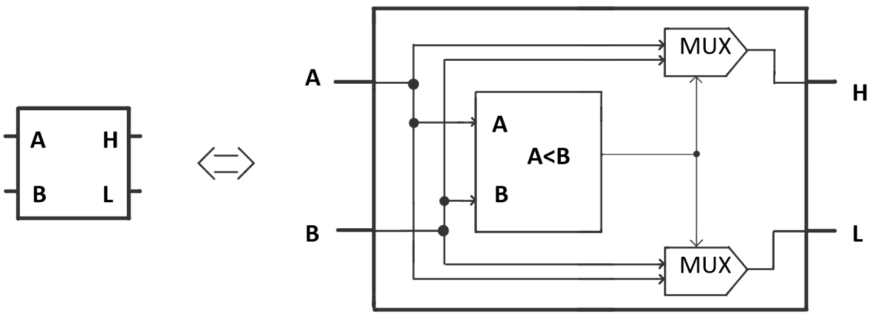


Figure 31: Scheme for each basic node used in the Median filter.

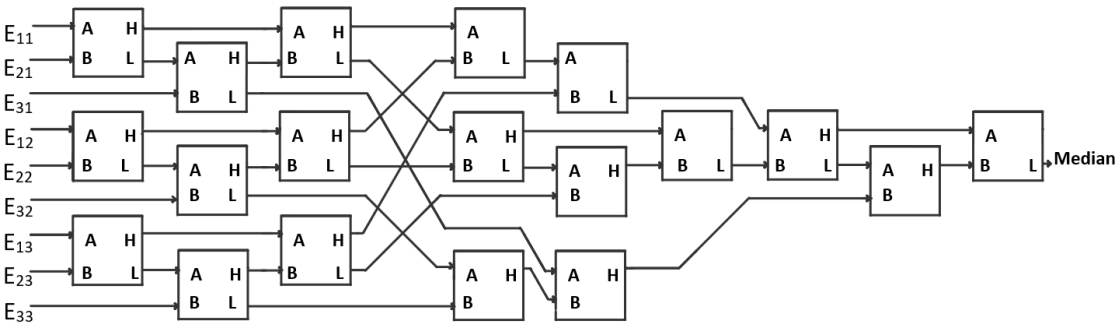


Figure 32: Implementation of Median filter.

$F_1 = F_2 = 1$. Then the interval L_1, in which the sharpness should be found, has to be known in advance, in this case 3200 μm. This range determines the number of iterations required to find the optimal sensor position as the least integer N, such that $F_N > L_1$. In this case N = 19, because $F_19 = 4181$ μm and $F_18 = 2584$ μm. In the first iteration step, the sharpness has to be evaluated at two points, the distance from the boundaries of L_1 is $\frac{F_{N-1}}{F_N}$. The interval L_1 is now divided into three subintervals, and the sharpness value at each boundary distance is known. The two subintervals neighboring the boundary with a higher sharpness value are used in the next step. The sub-interval that was neighboring to the last removed sub-interval is divided by a point at a distance of $L_K = L_{K-1}\frac{F_{N-(K-1)}}{F_{N-(K-2)}}$ from the outside boundary. The sharpness value is measured at this point and used to divide the next interval. This process is used iteratively, unless the maximum iteration k = N is reached. Then the distance at which the highest sharpness value was measured is the optimal sensor position. This process has the disadvantage of frequent changes in the direction of sensor movement.

The number of direction changes can be reduced using the Hill-climbing search. This algorithm uses two stages. In the first stage, the largest constant step is used to move the sensor. When the first search stage found the peak (the sharpness value decreases with increasing sensor position), the direction changes, and the second stage begins. In the second phase, a smaller step is used to find the exact peak position. The hill climbing algorithm is improved by adaptive step size in the second stage [56]. The step size is based on the following principle: When the image sharpness changes relatively small, the next moving step should be increased. If the difference in the sharpness measurement of two consecutive images is relatively large, the step size should be reduced, because the peak may be close to this position.

The rule-based peak searching algorithm presented in [57] divides the focus range into four types of search areas: Initial, Fine, Mid, and Coarse. The fine search is applied to areas where the presence of the peak is highly probable. In this area, the smallest possible step is chosen. The mid-areas have a lower probability; thus, larger step (three or four times) is used in those areas. The biggest moving step (seven to ten times greater than the smallest possible) is used in the coarse search areas because there is a low possibility of containing the global peak. All approaches require an initial setup area to move the sensor to the initial position before the search begins. The first five steps are performed by the smallest step. Then the area is classified as follows: If the current sharpness value is less than one-quarter of the maximal measured sharpness, then the area is classified as coarse. Otherwise, if the difference in sharpness between the current and previous sharpness is greater than one quarter of the previous sharpness, the area is classified as fine. When none of the above-disscribed conditions holds, the area is classified as mid. The fine area can change to the mid area when the currently measured sharpness is smaller than the previously measured sharpness for three times.

All these focus-finding methods target the absolutely focused image. In such an image, the radius of the blur circle 2B in Figure 16 is zero. However, this requirement is an unnecessarily strict requirement, because of the depth of field. If the blur circle is smaller than one pixel, the image is still in focus. Moreover, Jingqiang Li [58] introduces a photographic term circle of confusion. This term is used to define the depth of field of an optical system by a psychological way. He presented that the needed minimum resolution is 0.2 mm for 8 x 10-in printed photography at normal viewing distance. The photoprint is linearly scaled from the image sensor (film, CCD or, in our case, microbolometer array). Therefore, the circle of confusion in the image plane will depend on the size of the image sensor. The size of the image sensor is the diagonal length, which means that the size of the 8 x 10 in print is 325.3 mm. The microbolometer used has a sensitive area of 12.2 x 9.8 mm, which means that the sensor size is 15.6 mm. Using Equation 16 the tolerable sensor error can be calculated:

$$s - v = \frac{2B}{2A}v = \frac{0.2 \cdot 15.6}{325.3} \cdot f\sharp\frac{u}{u-f} \approx \frac{0.2 \cdot 15.6}{325.3}f\sharp \quad , when\ u \gg f \tag{44}$$

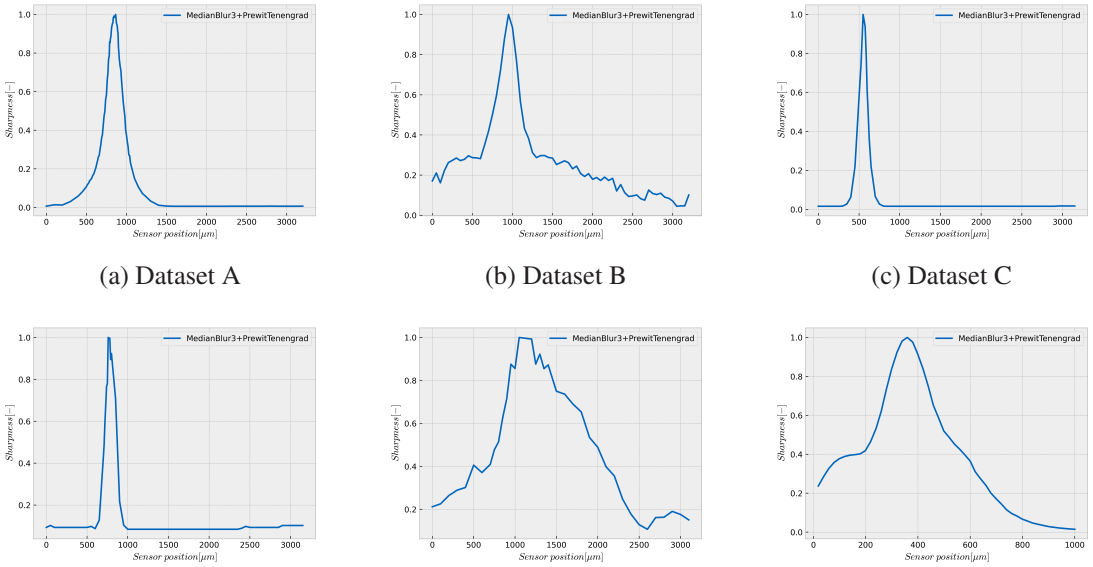
The camera is intended to be used with lenses with $f\sharp$ 1.1 or 1.2. To be on the safe side, $f\sharp$ 1.0 is used in the calculation of the distance, in which the blur effect is apparent. From Equation 44 the distance of 9.6 μm was obtained. This distance may be multiplied by a factor of two, because the blur can be accepted both behind and after the focal point; thus, the minimal step is 19 μm. The whole space in which the sensor can be moved has to be not whole searched, because the image cannot be at focus anywhere between the lens and the point at which the camera focused to infinity). The remaining space can be divided into a normal focus area and a macrofocus area. This reduces the sensor movement area in which the image should be focused to 0.64 mm (for f = 25 mm lens) to maximum from all the lenses which are intended to be used with this camera.This distance depends on the lens used. This maximum distance can be searched through 34 steps. The camera supports up to 60 frames per second (on one channel), which means that all frames are captured in half of second (the time needed for the sensors to move is neglected).

The Hill climbing algorithm was chosen to be used because the sharpnes measure function is unimodal. First, the larger step of size 19 μm is used. This step should not cause observable blurring when the image is once focused. The smaller step of 2 μm is applied in the second stage, where the interval of 19 μm around the maximum found sharpness value is searched through. This search should not have an eye-invisible effect on the image, but ensures the maximal focus for the image processing.

This method can be accelerated by changing the starting point to the last distance, when the camera was focused [58]. Then the direction is chosen on the basis of a larger sensor move. When the sharpness value increases, the direction is correct. If the sharpness value decreases, the direction has to be changed.

3.3 Results of the automatic focusing

This section presents the automatic focus procedure. First, all possible methods of automatic focus are described. From them the image sharpness measured was chosen because it does not need any additional hardware equipment or additional user information. This method uses a function that calculates the sharpness of the image. The spatial sharpness measure was tested on the five IR photo-detests captured for this purpose. First, the five different functions based on the raw image were processed. Second, the derivative methods were evaluated. The gradient was approximated by thirteen convolution kernels. The edges are highlighted in the result of that convolution and, thus, it is denoted as an edge map. The best edge map is obtained by the 5x5 convolution kernel. The threshold sum or variance can be used to obtain the sharpness measure from the edge map, which adds in total twenty-six sharpness measure functions. The last method was based on the second-order derivative. Two different kernels were used for the approximation and then each of them was modified. The modified kernels shows the edges always positive. The result was evaluated with the same two approaches as the derivation. In total, thirty-nine sharpness measures were evaluated. From them, the best result was obtained using the gradient method. Due to the complexity of the implementation, the 3x3 Prewit kernel evaluated with the threshold sum was chosen. Then, three image filters were tested. Each filter was used with multiple kernel sizes. This filter was applied on all presented methods, but any result does not exceed the derivation methods. Because the filter improves the noise resistance, additional Gaussian and salt and pepper noise was added to the images. It turns out that only the median filter is capable of filtering out the salt and pepper noise and enabling the focusing. The larger filter performs better for Gaussian blur, but needs more resources when implemented. Finlay, the median filter was chosen to be used before



(a) Dataset A (b) Dataset B (c) Dataset C

(d) Dataset D (e) Dataset E (f) Dataset F

Figure 33: Graph showing the sharpness over the sensor position computed by the selected method.

the sharpness measure to ensure that the automatic focusing will work even when dead pixels are present in the image. Lastly, the implementation of the module that decides which position should be selected is described. Based on the literature, four peak search algorithms are described. From them, the hill search algorithm is chosen, with the big step of 19 μm and small step of 2 μm. When the big step crosses the peak, the algorithm switches to the small steps. The small steps should not be seen by the user. To find the sharp point faster, this algorithm starts at the point at which the image was in focus last time. Unfortunately, this module was not implemented due to the lack of time.

Chapter 4
Conclusion

This thesis focuses on uncooled infrared cameras and is divided into three parts. The first part of this thesis provided the necessary theoretic background that was used in later parts. Fundamental information about the thermal camera, its parts, and principles of functionality was presented. This part also touched on the topic of infrared light, which is the operating space for thermal cameras.

The second part of the thesis dived into the hearth of an uncooled infrared camera, the microbolometr array. At first, general electrical schema was used to describe the working principle. All the parameters which can be changed to the sensor functions were presented, defined, and explained. Setting the microbolometer array is essential for image quality. Before the calibration itself, the behavior of the sensor was measured. First, the measured data were approximated by the linear line, as following the process outlined in the literature, but then the research proved that a special function is required. After experiments with different functions, the RBO function was chosen. This function achieved a mean fit error smaller than 0.2 percent of the ADC range. Using this function, the influence of ambient temperature on the measured data was described, as well as the effect of all parameters that can be measured. On the basis of the measurement, the two methods of drift compensation at ambient temperature were suggested and tested in a climate chamber. The first methods change only one parameter (the rest of the parameters set to constant values), which has the advantage of easy findings and not changing the gain of the fix pattern noise of the detector. This range is suitable for all applications where object temperature is not important. This application could be, for example, surveillance, hunting, or water pipes faults detection. It was experimentally proved in the environmental chamber that this methods manage to display BB range from -15 $°C$ to 150 $°C$ at any tested ambient temperature.

On the other hand, many applications, mainly in the industry, need a precise temperature measurement. For this purpose, the second ambient temperature drift compensation method was developed. This method was built on the basis of the time-consuming measurement. A special method was presented to find the compensation model without the time-consuming measurement. The new method found the coefficients based on the two measurements on the two black bodies at three ambient temperatures instead of measuring all parameter sweep. This method was also tested in the climatic chamber. The maximal difference between two RBF curves (each at different ambient temperatures) was 71 bits. This deviation will cause an error of approximately 1.2 $°C$, when the radiometry is calculated. Further research should investigate whether the methods presented are suitable for other cameras and sensors.

The third part of the thesis provided an overview of automatic focus methods. In the beginning, methods known from visible cameras were presented. All presented methods except one cannot be used in this thesis because of requiring additional hardware or user interaction. The only possible method used the sharpness computed from the image. This method was discussed in more detail. In total, thirty-nine sharpness measurement techniques were evaluated, whose can be divided into three main groups. The first group of techniques calculated the sharpness of the raw image. These techniques were found to be poorly sensitive. The second group evaluated the derivation of the image. This group provided the best results. The third group used the second-order derivative. The techniques in this group were very sensitive to noise in the image. All described techniques were tested on five infrared data sets, which were specially captured for this thesis. One data

set captured the author's face, all other data sets captured the BB of various temperatures, and consisted of approximately 100 frames. To increase the diversity, the BBs were covered with two special masks, adding more edges. Additionally, various lenses were used. On the basis of the performance of all the techniques on the test data sets, it was determined that the Prewit derivation evaluated with the threshold sum provided the best results. Other convolution kernels also provided good results. The main benefit of Prewit is the easy implementation in the FPGA and the resistance to noise. This method was improved by using the median filter, which is capable of preventing errors caused by dead pixels and even enhancing the noise resistance. This capability was tested in data sets with added noise.

In conclusion, the following result was by achieved post implementation of all selected methods. Every tested IR scene can be focused using the threshold sum of the convolution of the image with the Prewit convolution kernel. In addition, the median filter makes this method robust against dead pixels in the image. Data sets with higher contrast and more edges are easier to focus with all the tested methods. The camera is focused at the point where the sharpness measure is maximal. The peak search algorithm is responsible for finding the maximum value. This algorithm sets the sensor position and, on the basis of the sharpness measure, decides the next step. This method was not implemented in the FPGA due to a lack of time, but four different algorithms are discussed in the thesis. On the basis of the literature, one was chosen, and the appropriate step size was computed for the presented IR camera. The results presented are promising and should be validated by further data sets with different IR cameras.

References

[1] How do thermal cameras work? Accessed: 2022-06-16. [Online]. Available: https://www.flir.com/discover/rd-science/how-do-thermal-cameras-work/

[2] A. A. Sarawade and N. N. Charniya, "Infrared thermography and its applications: A review," in *2018 3rd International Conference on Communication and Electronics Systems (ICCES)*, 2018, pp. 280–285.

[3] Z. Ma, H. Li, W. Fang, Q. Liu, B. Zhou, and Z. Bu, "A cloud-edge-terminal collaborative system for temperature measurement in covid-19 prevention," in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2021, pp. 1–6.

[4] R. Brzezinski, N. Rabin, N. Lewis, R. Peled, A. Kerpel, A. Tsur, O. Gendelman, N. Naftali-Shani, I. Gringauz, H. Amital, A. Leibowitz, H. Mayan, I. Ben-Zvi, E. Heler, L. Shechtman, O. Rogovski, S. Shenhar-Tsarfaty, E. Konen, E. Marom, and O. Hoffer, "Automated processing of thermal imaging to detect covid-19," 12 2020.

[5] J. Head, C. Lipari, and R. Elliott, "Comparison of mammography and breast infrared imaging: sensitivity, specificity, false negatives, false positives, positive predictive value and negative predictive value," in *Proceedings of the First Joint BMES/EMBS Conference. 1999 IEEE Engineering in Medicine and Biology 21st Annual Conference and the 1999 Annual Fall Meeting of the Biomedical Engineering Society (Cat. N*, vol. 2, 1999, pp. 1116 vol.2–.

[6] S. Mishra, A. Prakash, S. K. Roy, P. Sharan, and N. Mathur, "Breast cancer detection using thermal images and deep learning," in *2020 7th International Conference on Computing for Sustainable Global Development (INDIACom)*, 2020, pp. 211–216.

[7] G. Steiner, *Handbook of Spectroscopy*. John Wiley and Sons, Ltd, 2003.

[8] Conrad electronic. Accessed: 2021-11-30. [Online]. Available: https://www.conrad.cz/p/voltcraft-wb-80-termokamera-20-do-600-c-32-x-32-pixel-9-hz-integrovana-digitalni-kamera-2362843

[9] R. Richwine, R. Balcerak, C. Rapach, K. Freyvogel, and A. Sood, "A comprehensive model for bolometer element and uncooled array design and imaging sensor performance prediction," in *Infrared and Photoelectronic Imagers and Detector Devices II*, R. E. Longshore and A. Sood, Eds., vol. 6294, International Society for Optics and Photonics. SPIE, 2006, pp. 118 – 128. [Online]. Available: https://doi.org/10.1117/12.682836

[10] P. A. Bell, C. W. H. Jr., and S. J. P. Jr., "Standard NETD test procedure for FLIR systems with video outputs," in *Infrared Imaging Systems: Design, Analysis, Modeling, and Testing IV*, G. C. Holst, Ed., vol. 1969, International Society for Optics and Photonics. SPIE, 1993, pp. 194 – 205. [Online]. Available: https://doi.org/10.1117/12.154715

[11] Netd explained. Accessed: 2021-12-02. [Online]. Available: https://movitherm.com/knowledgebase/netd-thermal-camera/

[12] M. Planck, *The theory of heat radiation*. Blakiston, 1914.

[13] Infrared detection basics. Accessed: 2021-10-20. [Online]. Available: http://www.lynred.com/e-learning/infrared-detection-basics

[14] C. B. Kaynak, A. Göritz, Y. Yamamoto, M. Wietstruck, M. Stocchi, K. E. Unal, M. B. Ozdemir, Y. Ozsoy, Y. Gurbuz, and M. Kaynak, "Mechanical and thermal modeling of an uncooled microbolometer," in *2019 European Microwave Conference in Central Europe (EuMCE)*, 2019, pp. 339–342.

[15] J. Lv, H. Zhong, Y. Zhou, B. Liao, J. Wang, and Y. Jiang, "Model-based low-noise readout integrated circuit design for uncooled microbolometers," *IEEE Sensors Journal*, vol. 13, no. 4, pp. 1207–1215, 2013.

[16] J. Lv, L. Que, L. Wei, Y. Zhou, and Y. Jiang, "Cmos readout integrated circuit for uncooled microbolometers without substrate temperature stabilization," in *Proceedings 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC)*, 2013, pp. 493–496.

[17] G. Bieszczad and M. Kastek, "Measurement of thermal behavior of detector array surface with the use of microscopic thermal camera," *Metrology and Measurement Systems*, vol. 18, pp. 679–690, 01 2011.

[18] M. Abdel-Rahman, A. Odebowale, N. Alkhalli, and M. R. Alshareef, "Ultra-thin film metallic absorbers for amophous silicon microbolometers: A comparative study," in *2021 International Conference on Electromagnetics in Advanced Applications (ICEAA)*, 2021, pp. 247–248.

[19] H. Fuentes, D. García-Romeo, N. Medrano, B. Calvo, P. A. Martínez, C. Gimeno, M. T. Sanz, M. Moreno, and A. Torres, "A sige microbolometer interface for low-power microcontrolled applications," in *2012 IEEE 3rd Latin American Symposium on Circuits and Systems (LASCAS)*, 2012, pp. 1–4.

[20] Y. S. Kim, T. H. Kim, G. T. Kim, B. T. Lim, S. K. Lim, H. D. Lee, and G. W. Lee, "Uncooled microbolometer arrays with high responsivity using meshed leg structure," *IEEE Photonics Technology Letters*, vol. 25, no. 21, pp. 2108–2110, 2013.

[21] S. Eminoglu, M. Y. Tanrikulu, and T. Akin, "A low-cost 128 $\times$ 128 uncooled infrared detector array in cmos process," *Journal of Microelectromechanical Systems*, vol. 17, no. 1, pp. 20–30, 2008.

[22] H. Jerominek, M. Renaud, N. R. Swart, F. Picard, T. D. Pope, M. Levesque, M. Lehoux, G. Bilodeau, M. Pelletier, D. Audet, and P. Lambert, "Micromachined VO_2-based uncooled IR bolometric detector arrays with integrated CMOS readout electronics," in *Micromachined Devices and Components II*, ser. Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, K. Chau and R. M. Roop, Eds., vol. 2882, Sep. 1996, pp. 111–121.

[23] Y. Shinde and A. Banerjee, "Design and calibration approach for shutter-less thermal imaging camera without thermal control," in *2012 Sixth International Conference on Sensing Technology (ICST)*, 2012, pp. 259–264.

[24] J. Lv, L. Que, L. Wei, Y. Zhou, and Y. Jiang, "Cmos readout integrated circuit for uncooled microbolometers without substrate temperature stabilization," in *Proceedings 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC)*, 2013, pp. 493–496.

[25] J. Čech and M. Rozkovec, "Selection of ideal operating point for infrared camera system," in *2018 International Conference on Applied Electronics (AE)*, 2018, pp. 1–4.

[26] P. W. Nugent, J. A. Shaw, and N. J. Pust, "Correcting for focal-plane-array temperature dependence in microbolometer infrared cameras lacking thermal stabilization," *Optical Engineering*, vol. 52, no. 6, pp. 1 – 8, 2013. [Online]. Available: https://doi.org/10.1117/1.OE.52.6.061304

[27] G. Kim, S. Lim, Y. Kim, H. Kim, B. Lim, T. Kim, J. H. Park, H. Lee, H. Kim, J. Seo, K. Lim, C. Seok, and H. Ko, "High-uniformity post-cmos uncooled microbolometer focal plane array integrated with active matrix circuit," in *2013 Transducers Eurosensors XXVII: The 17th International Conference on Solid-State Sensors, Actuators and Microsystems (TRANSDUCERS EUROSENSORS XXVII)*, 2013, pp. 2361–2364.

[28] S.-L. Liu, Y.-C. Zhang, X.-Y. Meng, W.-G. Lu, and Z.-J. Chen, "A design of readout circuit for 384×288 uncooled microbolometer infrared focal plane array," in *2012 IEEE 11th International Conference on Solid-State and Integrated Circuit Technology*, 2012, pp. 1–3.

[29] N. Horny, "Fpa camera standardisation," *Infrared Physics Technology*, vol. 44, no. 2, pp. 109–119, 2003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1350449502001834

[30] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[31] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[32] M. Subbarao, T.-S. Choi, and A. Nikzad, "Focusing techniques," *Optical Engineering*, vol. 32, no. 11, pp. 2824 – 2836, 1993. [Online]. Available: https://doi.org/10.1117/12.147706

[33] "Ieee standard for verilog hardware description language," *IEEE Std 1364-2005 (Revision of IEEE Std 1364-2001)*, pp. 1–590, 2006.

[34] F. C. A. Groen, I. T. Young, and G. Ligthart, "A comparison of different focus functions for use in autofocus algorithms," *Cytometry*, vol. 6, no. 2, pp. 81–91, 1985. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cyto.990060202

[35] M. Faundez-Zanuy, J. Mekyska, and V. Espinosa-Duró, "On the focusing of thermal images," *Pattern Recognition Letters*, vol. 32, no. 11, pp. 1548–1557, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167865511001358

[36] M. G. Chun and S. G. Kong, "Focusing in thermal imagery using morphological gradient operator," *Pattern Recognition Letters*, vol. 38, pp. 20–25, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167865513004157

[37] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[38] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[39] A. Abuolaim, A. Punnappurath, and M. S. Brown, "Revisiting autofocus for smartphone cameras," in *European Conference on Computer Vision (ECCV)*. Springer, 2018, pp. 523–537.

[40] M. L. Mendelsohn and B. H. Mayall, "Computer-oriented analysis of human chromosomes—iii. focus," *Computers in Biology and Medicine*, vol. 2, no. 2, pp. 137–150, 1972, chromosome Analysis. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0010482572900443

[41] A. SANTOS, C. ORTIZ DE SOLÓRZANO, J. J. VAQUERO, J. M. PEÑA, N. MALPICA, and F. DEL POZO, "Evaluation of autofocus functions in molecular cytogenetic analysis," *Journal of Microscopy*, vol. 188, no. 3, pp. 264–272, 1997. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1046/j.1365-2818.1997.2630819.x

[42] D. Vollath, "The influence of the scene parameters and of noise on the behaviour of automatic focusing algorithms," *Journal of Microscopy*, vol. 151, no. 2, pp. 133–146, 1988. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1365-2818.1988.tb04620.x

[43] L. Firestone, K. Cook, K. Culp, N. Talsania, and K. Preston Jr., "Comparison of autofocus methods for automated microscopy," *Cytometry*, vol. 12, no. 3, pp. 195–206, 1991. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cyto.990120302

[44] M. Shirvaikar, "An optimal measure for camera focus and exposure," in *Thirty-Sixth Southeastern Symposium on System Theory, 2004. Proceedings of the*, 2004, pp. 472–475.

[45] J. F. Brenner, B. S. Dew, J. B. Horton, T. King, P. W. Neurath, and W. D. Selles, "An automated microscope for cytologic research a preliminary evaluation." *Journal of Histochemistry & Cytochemistry*, vol. 24, no. 1, pp. 100–111, 1976, pMID: 1254907. [Online]. Available: https://doi.org/10.1177/24.1.1254907

[46] J. L. Pech-Pacheco, G. Cristóbal, J. Chamorro-Martínez, and J. Fernández-Valdivia, "Diatom autofocusing in brightfield microscopy: a comparative study," *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, vol. 3, pp. 314–317 vol.3, 2000.

[47] H. Kekre and S. Gharge, "Image segmentation using extended edge operator for mammographic images," *International Journal on Computer Science and Engineering*, vol. 2, 07 2010.

[48] C. Orhei, V. Bogdan, and C. Bonchiş, "Dilated filters for edge detection algorithms," 06 2021.

[49] G. N. Chaple, R. D. Daruwala, and M. S. Gofane, "Comparisions of robert, prewitt, sobel operator based edge detection methods for real time uses on fpga," in *2015 International Conference on Technologies for Sustainable Development (ICTSD)*, 2015, pp. 1–4.

[50] S. K. Nayar and Y. Nakagawa, "Shape from focus," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, pp. 824–831, 1994.

[51] A. Thelen, S. Frey, S. Hirsch, and P. Hering, "Improvements in shape-from-focus for holographic reconstructions with regard to focus operators, neighborhood-size, and height value interpolation," *IEEE Transactions on Image Processing*, vol. 18, no. 1, pp. 151–157, 2009.

[52] Z. Xu, X. Baojie, and W. Guoxin, "Canny edge detection based on open cv," in *2017 13th IEEE International Conference on Electronic Measurement Instruments (ICEMI)*, 2017, pp. 53–56.

[53] O. Shipitko and A. Grigoryev, "Gaussian filtering for fpga based image processing with high-level synthesis tools," 06 2018.

[54] M. A. Vega-Rodríguez, J. Sánchez-Pérez, and J. A. Gomez-Pulido, "An fpga-based implementation for median filter meeting the real-time requirements of automated visual inspection systems," 01 2002.

[55] E. Krotkov, "Focusing," *International Journal of Computer Vision*, vol. 1, pp. 223–237, 1988.

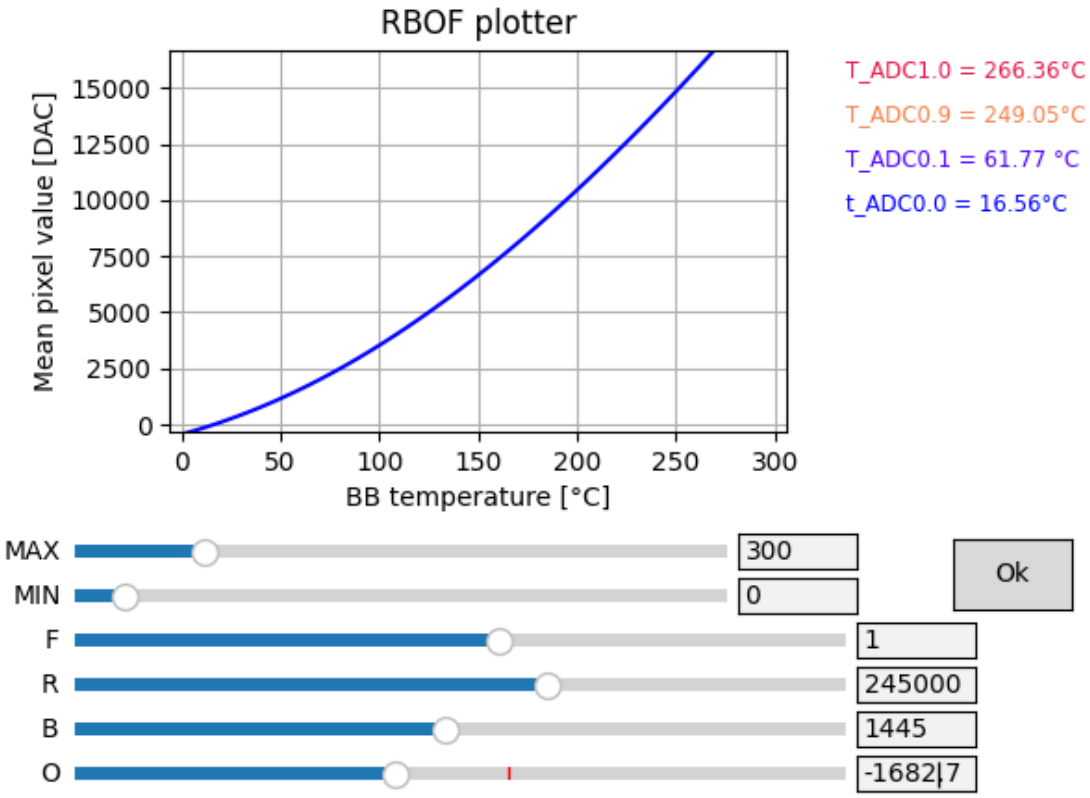
[56] J. He, R. Zhou, and Z. Hong, "Modified fast climbing search auto-focus algorithm with adaptive step size searching technique for digital camera," *IEEE Transactions on Consumer Electronics*, vol. 49, no. 2, pp. 257–262, 2003.

[57] N. Kehtarnavaz and H.-J. Oh, "Development and real-time implementation of a rule-based auto-focus algorithm," *Real-Time Imaging*, vol. 9, no. 3, pp. 197–203, 2003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1077201403000378

[58] J. Li, "Autofocus searching algorithm considering human visual system limitations," *Optical Engineering*, vol. 44, no. 11, pp. 1 – 4, 2005. [Online]. Available: https://doi.org/10.1117/1.2130725

Appendix A
Scripts and programs

A.1 Program for plotting RBFO function




```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.widgets import Slider, Button, TextBox
import os

def RBFO_c(R,B,F,O,T):
    try:
        return RBFO_k(R,B,F,O,np.array(T)+np.array([273.15]*len(T)))
    except:
        return RBFO_k(R,B,F,O,T+273.15)

def RBFO_k(R,B,F,O,T):
    return np.array(R)/(np.exp((np.array(B)/np.array(T)))-np.array(F)) + np.
    array(O)

def RBO_c(R,B,O,T):
```

```
    F = 1
    return RBFO_c(R,B,F,O,T)

def inv_RBFO_k(R,B,F,O,DAC):
    if (DAC-O) == 0:
        return -1000
    if (R/ (DAC-O)) + F <= 0:
        return -1000
    else:
        return B / np.log( (R/ (DAC-O)) + F)

def inv_RBFO_c(R,B,F,O,DAC):
    T = inv_RBFO_k(R,B,F,O,DAC)
    if T == -1000:
        return -1000
    else:
        return  T- 273.15

class RBFO_gui:
    def __init__(self,BB = [[],[]],bigger_img = 0.02, R_init=245000,B_init
    =1445,F_init=1,O_init=1261,t_min = 0, t_max = 300):
        self.t_min = t_min
        self.t_max = t_max
        self.R = R_init
        self.B = B_init
        self.F = F_init
        self.O = O_init
        self.bigger_img = bigger_img
        self.x_lim = [t_min-(t_max-t_min)*bigger_img,t_max+(t_max-t_min)*
    bigger_img]
        self.y_lim = [0-(2**14*bigger_img),2**14*(1+bigger_img)]
        t = np.linspace(self.x_lim[0],self.x_lim[1])

        self.fig, self.ax = plt.subplots()
        self.ax.set_ylim(self.y_lim)
        self.ax.set_xlim(self.x_lim)
        plt.grid(True)
        plt.title('RBOF plotter')

        plt.plot(BB[0],BB[1],linestyle='none', marker='o',color='#0091FF')
        self.line, = plt.plot(t, RBO_c(self.R, self.B, self.O,t),color='#0000ff
    ')
        self.ax.set_xlabel('BB temperature [ C ]')
        self.ax.set_ylabel('Mean pixel value [DAC]')
        plt.subplots_adjust(left=0.18, right= 0.7, bottom=0.45)
        self.btn = Button(plt.axes([0.84, 0.25, 0.10, 0.08]),'Ok')

        t_100 = self.inv_RBFO_c(2**14)
        self.m_text = plt.gcf().text(0.75, 0.85, 't_ADC1.0 = %.2f  C' %t_100,
    color='#e6194b',fontsize=8.5)
        t_90 = self.inv_RBFO_c(2**14*0.9)
        self.text_90 = plt.gcf().text(0.75, 0.80, 't_ADC0.9 = %.2f  C' %t_90,
    color='#F5824C',fontsize=8.5)
        t_10 = self.inv_RBFO_c(2**14*0.1)
        if t_10 == -1000:
            self.text_10 = plt.gcf().text(0.75, 0.75, 't_ADC0.1 = NA',color='#4
    c01ff',fontsize=8.5)
        else:
            self.text_10 = plt.gcf().text(0.75, 0.75, 'T_ADC0.1 = %.2f  C' %
    t_10,color='#4c01ff',fontsize=8.5)
```

```
        t_0 = self.inv_RBFO_c(0)
        if t_0 == -1000:
            self.text_0 = plt.gcf().text(0.75, 0.7, 'T_ADC0.0 = NA',color='
    #0000FF',fontsize=8.5)
        else:
            self.text_0 = plt.gcf().text(0.75, 0.7, 'T_ADC0.0 = %.2f  C' %t_0,
    color='#0000FF',fontsize=8.5)

        axcolor = 'lightgoldenrodyellow'

        self.max_slider = Slider(
            ax=plt.axes([0.1, 0.30, 0.55, 0.03], facecolor=axcolor),
            label='MAX',
            valmin= 0,
            valmax= 1500,
            valinit=t_max,
        )
        self.min_slider = Slider(
            ax=plt.axes([0.1, 0.25, 0.55, 0.03], facecolor=axcolor),
            label='MIN',
            valmin=-50,
            valmax= 600,
            valinit=t_min,
        )
        self.F_slider = Slider(
            ax=plt.axes([0.1, 0.20, 0.65, 0.03], facecolor=axcolor),
            label='F',
            valmin=-10,
            valmax= 10,
            valinit=self.F,
        )
        self.R_slider = Slider(
            ax=plt.axes([0.1, 0.15, 0.65, 0.03], facecolor=axcolor),
            label='R',
            valmin=0.1,
            valmax=400000,
            valinit=self.R,
        )
        self.B_slider = Slider(
            ax=plt.axes([0.1, 0.1, 0.65, 0.03], facecolor=axcolor),
            label='B',
            valmin=0.1,
            valmax=3000,
            valinit=self.B,
        )
        self.O_slider = Slider(
            ax=plt.axes([0.1, 0.05, 0.65, 0.03], facecolor=axcolor),
            label='O',
            valmin=-10000,
            valmax= 10000,
            valinit=self.O,
        )

        self.max_text_box = TextBox( plt.axes([0.66, 0.295, 0.1, 0.04]),"",
    initial=t_max)
        self.min_text_box = TextBox(plt.axes([0.66, 0.245, 0.1, 0.04]),"",
    initial=t_min)
        self.R_text_box = TextBox(plt.axes([0.76, 0.145, 0.1, 0.04]),"",initial
    =self.R)
        self.B_text_box = TextBox(plt.axes([0.76, 0.095, 0.1, 0.04]),"",initial
```

```
    =self.B)
124         self.O_text_box = TextBox(plt.axes([0.76, 0.045, 0.1, 0.04]),"",initial
    =self.O)
125         self.F_text_box = TextBox(plt.axes([0.76, 0.195, 0.1, 0.04]),"",initial
    =self.F)
126
127
128         self.R_slider.on_changed(self.R_sliders_changed)
129         self.B_slider.on_changed(self.B_sliders_changed)
130         self.O_slider.on_changed(self.O_sliders_changed)
131         self.F_slider.on_changed(self.F_sliders_changed)
132         self.max_slider.on_changed(self.max_sliders_changed)
133         self.min_slider.on_changed(self.min_sliders_changed)
134
135         self.R_text_box.on_submit(self.R_boxes_submit)
136         self.B_text_box.on_submit(self.B_boxes_submit)
137         self.O_text_box.on_submit(self.O_boxes_submit)
138         self.F_text_box.on_submit(self.F_boxes_submit)
139         self.min_text_box.on_submit(self.min_boxes_submit)
140         self.max_text_box.on_submit(self.max_boxes_submit)
141
142         self.btn.on_clicked(self.export_potential_configurations)
143         plt.gcf().canvas.manager.set_window_title('RBFO plotter')
144         plt.show()
145
146     def max_sliders_changed(self,value):
147         if self.t_max != value:
148             self.t_max = value
149             self.max_text_box.set_val(round(value,1))
150             self.update(value)
151
152     def min_sliders_changed(self,value):
153         if self.t_min != value:
154             self.t_min = value
155             self.min_text_box.set_val(round(self.t_min,1))
156             self.update(value)
157
158     def R_sliders_changed(self,value):
159         if self.R != self.R_slider.val:
160             self.R = self.R_slider.val
161             self.R_text_box.set_val(round(self.R,1))
162             self.update(value)
163
164     def B_sliders_changed(self,value):
165         if self.B != self.B_slider.val:
166             self.B = self.B_slider.val
167             self.B_text_box.set_val(round(self.B,1))
168             self.update(value)
169
170     def O_sliders_changed(self,value):
171         if self.O != self.O_slider.val:
172             self.O = self.O_slider.val
173             self.O_text_box.set_val(round(self.O,1))
174             self.update(value)
175
176     def F_sliders_changed(self,value):
177         if self.F != self.F_slider.val:
178             self.F = self.F_slider.val
179             self.F_text_box.set_val(round(self.F,1))
180             self.update(value)
```

```
181
182     def max_boxes_submit(self,value):
183         new_value = float(value.replace(",","."))
184         if self.t_max != new_value:
185             self.t_max = new_value
186             self.max_slider.set_val(self.t_max)
187             self.update(value)
188
189     def min_boxes_submit(self,value):
190         new_value = float(value.replace(",","."))
191         if self.t_min != new_value:
192             self.t_min = new_value
193             self.min_slider.set_val(self.t_min)
194             self.update(value)
195
196     def R_boxes_submit(self,value):
197         new_value = float(value.replace(",","."))
198         if self.R != new_value:
199             self.R = new_value
200             self.R_slider.set_val(self.R)
201             self.update(value)
202
203     def B_boxes_submit(self,value):
204         new_value = float(value.replace(",","."))
205         if self.B != new_value:
206             self.B = new_value
207             self.B_slider.set_val(self.B)
208             self.update(value)
209
210     def O_boxes_submit(self,value):
211         new_value = float(value.replace(",","."))
212         if self.O != new_value:
213             self.O = new_value
214             self.O_slider.set_val(self.O)
215             self.update(value)
216
217     def F_boxes_submit(self,value):
218         new_value = float(value.replace(",","."))
219         if self.F != new_value:
220             self.F = new_value
221             self.F_slider.set_val(self.F)
222             self.update(value)
223
224     def update(self,value):
225
226         self.x_lim = [self.t_min-(self.t_max-self.t_min)*self.bigger_img,self.
    t_max+(self.t_max-self.t_min)*self.bigger_img]
227         self.ax.set_xlim(self.x_lim)
228         t = np.linspace(self.x_lim[0],self.x_lim[1])
229
230
231         max_t = self.inv_RBFO_c(2**14)
232         t_90 = self.inv_RBFO_c(2**14*0.9)
233
234         t_10 = self.inv_RBFO_c(2**14*0.1)
235         if t_10 == -1000:
236             self.text_10.set_text('t_ADC0.1 = NA')
237
238         t_0 = self.inv_RBFO_c(2**14*0)
239         if t_0 == -1000:
```

```
240             self.text_0.set_text('t_ADC0.0 = NA')
241         else:
242             self.text_0.set_text('t_ADC0.0 = %.2f  C'%t_0)
243 
244 
245         print("ADC value of 15 C = "+str(self.RBFO_c(15))+" ;100 C  = "+str(
    self.RBFO_c(100))+" ;200 C  = "+str(self.RBFO_c(200))+"")
246 
247 
248         self.line.set_data(t,RBFO_c(self.R,self.B, self.F,self.O,t))
249 
250         #line_90.set_data([t_90,t_90],y_lim)
251         #line_h100.set_data(t,[2**14]*len(t))
252         #line_h90.set_data(t,[2**14*0.9]*len(t))
253 
254         self.fig.canvas.draw_idle()
255         self.is_needet_uppdate = 0
256 
257     def inv_RBFO_c(self,DAC):
258         return inv_RBFO_c(self.R, self.B, self.F, self.O,DAC)
259 
260     def RBFO_c(self,T):
261         return RBFO_c(self.R,self.B,self.F,self.O,T)
262 
263     def export_potential_configurations(self,event):
264 
265         #path = '../Vysledky/to_work_with/'
266         path=""
267         f = open(path+"Find_RBOF.txt", "w")
268         f.write("Saved constants: ")
269         f.write("\nR: ")
270         f.write(str(self.R))
271         f.write("\nB: ")
272         f.write(str(self.B))
273         f.write("\nO: ")
274         f.write(str(self.O))
275         f.write("\nF: ")
276         f.write(str(self.F))
277 
278         f.close()
279 
280         plt.close()
281         quit()
282 
283 if __name__ == "__main__":
284     #filename = "Target_BB.txt"
285     filename = ""
286     BB_T = []
287     BB_v = []
288     if os.path.isfile(filename):
289         f = open(filename, "r")
290         f.readline()
291         for x in f:
292             data = x.replace(",",".").split(" ")
293             BB_T.append(float(data[0]))
294             BB_v.append(float(data[1]))
295         f.close()
296 
297     gui = RBFO_gui(BB=[BB_T,BB_v])
```

A.2 Evaluation Sharpness measure algorithm framework

```
import os
from unicodedata import name
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation
import matplotlib.cm as cm
import cv2
from scipy.ndimage import gaussian_filter
from scipy import signal
import math
import pywt
from scipy.stats import entropy
import random



my_colours = ['#0065BD', '#6AADE4', '#00B2A9',   '#A2AD00',
        '#F0AB00', '#E05200', '#F5824C' ,'#C60C30', '#981F40']

class Edge_detector:
    def __init__(self, alg):
        self.alg = alg
        self.frames = []
        self.distances = []
        self.max_values = []
        self.mean_values = []
        self.sharpness = []
        self.frame_columns = 480
        self.frame_rows = 640
        self.max_sharpness = self.frame_columns*self.frame_rows
        self.max_pixel_value = 255

    def add_frame(self,dist,frame):
        algo = self.alg.split("+")
        sharpness_value = 0
        for alg in algo:
            ##Filters
            alg_split = alg.split('-')
            filter = alg_split[0]
            if len(alg_split) > 1:
                param = alg_split[1]


            if filter == 'GaussianBlur':
                frame = cv2.GaussianBlur(frame,(int(param),int(param)),0)
            elif filter == 'AvgBlur':
                frame = cv2.blur(frame,(int(param),int(param)))
            elif filter == 'MedianBlur':
                frame = cv2.medianBlur(frame,int(param))

            elif filter == 'modifiedLaplacian1':
                frame = self.modifiedLaplacian1(frame)
            elif filter == 'laplacian':
                frame = self.laplacian(frame)
            elif filter == 'grad':
                frame = self.sobel(frame,param)
                sharpness_value = np.mean(frame)

```

```
59              elif filter == 'sobelx':
60                  frame = cv2.Sobel(frame,cv2.CV_64F,1,0,ksize=int(param))
61                  sharpness_value = np.max(frame)
62              elif filter == 'sobely':
63                  frame = cv2.Sobel(frame,cv2.CV_64F,0,1,ksize=int(param))
64              elif filter == 'sobelxy':
65                  frame = cv2.Sobel(frame,cv2.CV_64F,1,1,ksize=int(param))
66                  sharpness_value = np.mean(frame)
67              elif filter == 'MySobel':
68                  mySobelKer =np.array([[-1, -2, -1],
69                                        [-2, 12, -2],
70                                        [-1, -2, -1]])
71                  #mySobelKer =np.array([[-1, -1, -1],
72                #                        [-1, 8, -1],
73                #                        [-1, -1, -1]])
74                  frame = cv2.filter2D(src=frame,ddepth=-1, kernel=mySobelKer)
75              elif filter == 'None':
76                  frame = frame
77              elif filter == 'BrennerGradient':
78                  frame = self.Brenner_gradient(frame,int(param))
79                  sharpness_value = np.sum(np.sum(frame))
80              elif filter == 'Energy_variance':
81                  frame = self.Energy_variance(frame)
82                  sharpness_value = np.mean(frame)
83              elif filter == 'Energy_gradient':
84                  frame = self.Energy_gradient(frame)
85                  sharpness_value = np.mean(frame)
86              elif filter == 'Energy_of_Laplacian':
87                  frame = self.Energy_of_Laplacian(frame)
88                  sharpness_value = np.sum(frame)
89              elif filter == 'Gaussian_derivative':
90                  frame = self.Gaussian_derivative(frame,int(param))
91                  sharpness_value = np.sum(frame)
92              elif filter == 'AbsCentrMoment':
93                  sharpness_value = self.Absolute_Central_Moment(frame,int(param)
    )
94              elif filter == 'modifiedLaplacian2':
95                  frame = self.modifiedLaplacian2(frame)
96              elif filter == 'modifiedLaplacian3':
97                  frame = self.modifiedLaplacian3(frame)
98              elif filter == 'Laplacian1':
99                  frame = self.Laplacian1(frame)
100             elif filter == 'Laplacian2':
101                 frame = self.Laplacian2(frame)
102             elif filter == 'sumOfWavelets':
103                 sharpness_value = self.wavelets1(frame,param)
104             elif filter == 'VarianceOfWavelets':
105                 sharpness_value = self.wavelets2(frame,param)
106             elif filter == 'DifferenceOfGaussians':
107                 L1 = cv2.GaussianBlur(frame,(5,5),1)
108                 L2 = cv2.GaussianBlur(frame,(5,5),2)
109                 frame = L1-L2
110
111             elif filter == 'Threshold_count':
112                 sharpness_value = self.threshold_count(frame,int(param))
113             elif filter == 'Threshold':
114                 #print(np.mean(frame))
115                 sharpness_value = self.threshold(frame,int(param))
116                 self.max_sharpness = self.frame_columns*self.frame_rows
117             elif filter == 'Variance':
```

```
                sharpness_value = self.variance(frame)
            elif filter == 'Energy':
                sharpness_value = np.sum(frame)
            elif filter == 'ImagePower':
                self.max_sharpness = self.max_pixel_value**2
                                    * self.frame_columns
                                    * self.frame_rows
                sharpness_value = self.image_power(frame)
            elif filter == 'Mean':
                self.max_sharpness = self.max_pixel_value**2
                                    * self.frame_columns
                                    * self.frame_rows
                sharpness_value = np.mean(frame)
            elif filter == 'Entropy':
                sharpness_value = entropy(frame.flatten())
                print(sharpness_value)
            elif filter == 'Range':
                sharpness_value = np.max(frame) - np.min(frame)
                print(sharpness_value)
            elif filter == 'Standar_deviation':
                sharpness_value = np.std(frame)
            elif filter == 'Autocorelation':
                sharpness_value = self.autocoralation(frame)
            elif filter == 'NormVar':
                sharpness_value = np.var(frame)/np.mean(frame)
            elif filter == 'Meldelson':
                sharpness_value = self.Meldelson(frame,value=int(param))
            elif filter == 'StdCorelation':
                sharpness_value = self.StdCorelation(frame)

            elif filter == 'SPNoise':
                frame = self.AddSPNoise(frame)
            elif filter == 'GaussNoise':
                frame = self.AddGaussNoise(frame)

            else:
                print("ERROR: Unknown filter: "+filter)


        self.frames.append(frame)
        self.distances.append(dist)
        self.max_values.append(np.amax(frame))
        self.mean_values.append(np.mean(frame))
        self.sharpness.append(sharpness_value)

    def plot_sharpness(self,color_index,names = []):
        self.sharpness = self.sharpness/np.max(self.sharpness)
        if names == []:
            plt.plot(self.distances,self.sharpness,label=self.alg,
                color=my_colours[color_index])
        else:
            plt.plot(self.distances,self.sharpness,label=names[color_index],
                color=my_colours[color_index])
        kernel_size = 8
        kernel = np.ones(kernel_size)/kernel_size
        smoth_data = np.convolve(self.sharpness,kernel,'same')

    def plot_max_values(self):
        plt.plot(self.distances,self.max_values,label="MaxValue of " + self.alg
    )
```

```
177
178     def plot_avg_values(self):
179         plt.plot(self.distances,self.mean_values,label="MeanValue of " + self.
    alg)
180
181     def animate(self):
182         fig, ax = plt.subplots()
183         plt.axis('off')
184         plt.title(self.alg)
185         ims = []
186         for count,img in enumerate(self.frames):
187             im_plot = plt.imshow(img, cmap=cm.Greys_r,animated=True)
188             ims.append([im_plot,dist,avg,max])
189
190         ani = animation.ArtistAnimation(fig, ims, interval=60, blit=False,
191                                         repeat_delay=1000)
192         plt.show()
193
194     def modifiedLaplacian1(self,img):
195         img = np.uintc(img)
196         for j in range(self.frame_columns-2):
197             for i in range(self.frame_rows-2):
198                 img[j+1][i+1] = abs(2*img[j+1][i+1]-img[j+2][i+1]-img[j][i+1])
199                                 +abs(2*img[j+1][i+1]-img[j+1][i+2]-img[j+1][i])
200         return img
201
202
203     def laplacian(self,img):
204         return cv2.Laplacian(img, ddepth=cv2.CV_64F)
205
206     def Brenner_gradient(self,img,n=2):
207         img = np.intc(img)
208         for j in range(self.frame_columns):
209             for i in range(self.frame_rows-n):
210                 img[j][i] = int((int(img[j][i]) - int(img[j][i+n]))^2)
211         return img
212
213     def Energy_variance(self,img):
214         img = np.intc(img)
215         mu = np.mean(img)
216         return np.power(img-mu,2)
217
218     def Energy_gradient(self,img):
219         img = np.intc(img)
220         for j in range(self.frame_columns-1):
221             for i in range(self.frame_rows-1):
222                 gx = int((int(img[j][i]) - int(img[j][i+1])))
223                 gy = int((int(img[j][i]) - int(img[j][i+1])))
224                 img[j][i] = gx^2 + gy^2
225         return img
226
227     def Energy_of_Laplacian(self,img):
228         img = np.intc(img)
229         for j in range(self.frame_columns-1):
230             for i in range(self.frame_rows-1):
231                 img[j][i] = (-(int(img[j-1][i-1]))
232                              -4*-(int(img[j-1][i]))
233                              -(int(img[j-1][i+1]))
234                              -4*(int(img[j][i-1]))
235                              +20*(int(img[j][i]))
```

```
                        -4*(int(img[j][i+1]))
                        -(int(img[j+1][i-1]))
                        -4*(int(img[j+1][i]))
                        -(int(img[j+1][i+1])))^2
        return img

    def Gaussian_derivative(self,img,sigma = 1):
        img = np.intc(img)
        return gaussian_filter(img,(sigma,sigma),(0,1))
            + gaussian_filter(img,(sigma,sigma),(1,0))

    def Absolute_Central_Moment(self,img,bins=100):
        img = np.intc(img)
        H = np.histogram(img, bins=bins)
        mu = np.mean(H[1])
        suma = 0
        P = H[0]/(640*480)
        for k in range(len(H[0])):
            suma = suma + abs((H[1][k]+H[1][k+1])/2-mu)*P[k]
        return suma

    def image_power(self,img):
        img = np.intc(img)
        return np.sum(img**2)#sum

    def sobel(self,img, param):
        img = np.intc(img)
        print("Grad: "+param)
        if param == 'roberts':
            my_ker_X=np.array([[ 0, 1],
                               [-1, 0]])
            my_ker_Y = np.transpose(my_ker_X)
        elif param == 'prewit3':
            my_ker_X=np.array([[-1, 0, 1],
                               [-1, 0, 1],
                               [-1, 0, 1]])
            my_ker_Y = np.transpose(my_ker_X)
        elif param == 'prewit5':
            my_ker_X=np.array([[-2,-1, 0, 1,2],
                                [-2,-1, 0, 1,2],
                                [-2,-1, 0, 1,2],
                                [-2,-1, 0, 1,2],
                                [-2,-1, 0, 1,2]])
            my_ker_Y = np.transpose(my_ker_X)

        elif param == 'prewitD':
            my_ker_X=np.array([[-1,0, 0, 0,1],
                               [ 0,0, 0, 0,0],
                               [-1,0, 0, 0,1],
                               [ 0,0, 0, 0,0],
                               [-1,0, 0, 0,1]])
            my_ker_Y = np.transpose(my_ker_X)

        elif param == 'prewit7':
            my_ker_X=np.array([[-3,-2,-1, 0, 1,2,3],
                               [-3,-2,-1, 0, 1,2,3],
                               [-3,-2,-1, 0, 1,2,3],
                               [-3,-2,-1, 0, 1,2,3],
                               [-3,-2,-1, 0, 1,2,3],
                               [-3,-2,-1, 0, 1,2,3],
```

```
                                [-3,-2,-1, 0, 1,2,3]])
            my_ker_Y = np.transpose(my_ker_X)

        elif param == 'prewit9':
            my_ker_X=np.array([[-4,-3,-2,-1, 0, 1,2,3,4],
                               [-4,-3,-2,-1, 0, 1,2,3,4],
                               [-4,-3,-2,-1, 0, 1,2,3,4],
                               [-4,-3,-2,-1, 0, 1,2,3,4],
                               [-4,-3,-2,-1, 0, 1,2,3,4],
                               [-4,-3,-2,-1, 0, 1,2,3,4],
                               [-4,-3,-2,-1, 0, 1,2,3,4],
                               [-4,-3,-2,-1, 0, 1,2,3,4],
                               [-4,-3,-2,-1, 0, 1,2,3,4]])
            my_ker_Y = np.transpose(my_ker_X)

        elif param == 'sobel3':
            my_ker_X=np.array([[-1, 0, 1],
                               [-2, 0, 2],
                               [-1, 0, 1]])
            my_ker_Y = np.transpose(my_ker_X)

        elif param == 'sobelD':
            my_ker_X=np.array([[-1, 0, 0, 0, 1],
                               [ 0, 0, 0, 0, 0],
                               [-2, 0, 0, 0, 2],
                               [ 0, 0, 0, 0, 0],
                               [-1, 0, 0, 0, 1]])
            my_ker_Y = np.transpose(my_ker_X)

        elif param == 'sobel5':
            my_ker_X=np.array([[ -5,  -4, 0,  4, 5],
                               [ -8, -10, 0, 10, 8],
                               [-10, -20, 0, 20,10],
                               [ -8, -10, 0, 10, 8],
                               [ -5,  -4, 0,  4, 5]])
            my_ker_Y = np.transpose(my_ker_X)

        elif param == 'sobel7':
            my_ker_X=np.array([[  -780,  -720,  -468, 0,  468,  720,  780],
                               [ -1080, -1170,  -936, 0,  936, 1170, 1080],
                               [ -1404, -1872, -2340, 0, 2340, 1872, 1404],
                               [ -1560, -2340, -4680, 0, 4680, 2340, 1560],
                               [ -1404, -1872, -2340, 0, 2340, 1872, 1404],
                               [ -1080, -1170,  -936, 0,  936, 1170, 1080],
                               [  -780,  -720,  -468, 0,  468,  720,  780]])
            my_ker_Y = np.transpose(my_ker_X)

        elif param == 'scharr3':
            my_ker_X=np.array([[-3, 0, 3],
                               [-10, 0, 10],
                               [-3, 0, 3]])
            my_ker_Y = np.transpose(my_ker_X)

        elif param == 'scharrD':
            my_ker_X=np.array([[-3,0, 0,0, 3],
                               [ 0, 0, 0, 0, 0],
                               [-10,0, 0,0, 10],
                               [ 0, 0, 0, 0, 0],
                               [-3, 0,0, 0,3]])
            my_ker_Y = np.transpose(my_ker_X)
```

```
356
357         elif param == 'scharr5':
358             my_ker_X=np.array([[ -1,  -1, 0,  1, 1],
359                                [ -2,  -2, 0,  2, 2],
360                                  [ -3,  -6, 0,  6, 3],
361                                  [ -2,  -2, 0,  2, 2],
362                                [ -1,  -1, 0,  1, 1]])
363             my_ker_Y = np.transpose(my_ker_X)
364
365         elif param == 'Kirsch3':
366             my_ker_X=np.array([[-3, -3, 5],
367                                [ -3, 0, 5],
368                                [ -3, -3, 5]])
369             my_ker_Y = np.transpose(my_ker_X)
370
371         elif param == 'KirschD':
372             my_ker_X=np.array([[-3, 0,-3,0, 5],
373             [0, 0,0,0, 0],
374                                [ -3,0, 0,0, 5],
375                                  [0, 0,0,0, 0],
376                                [ -3,0, -3,0, 5]])
377             my_ker_Y = np.transpose(my_ker_X)
378
379         elif param == 'Kirsch5':
380             my_ker_X=np.array([[ -7,  -7, -7,  9, 9],
381                                [ -7,  -3, -3,  5, 9],
382                                  [ -7,  -3, 0,  5, 9],
383                                  [ -7,  -3, -3, 5, 9],
384                                [ -7,  -7, -7,  9, 9]])
385             my_ker_Y = np.transpose(my_ker_X)
386
387         else:
388             print("Unknow parameter")
389
390         fx = signal.convolve2d(img,my_ker_X,mode='valid')
391         fy = signal.convolve2d(img,my_ker_Y,mode='valid')
392         return np.array(fx)**2 + np.array(fy)**2
393
394     def Laplacian2(self,img):
395         img = np.intc(img)
396         mySobelKer =np.array([[-1, -4, -1],
397                               [-4, 20, -4],
398                               [-1, -4, -1]])
399         return signal.convolve2d(img,mySobelKer,mode='valid')
400
401     def AddSPNoise(self,img):
402         for i in range(100):
403             x = random.randint(0,self.frame_columns-1)
404             y = random.randint(0,self.frame_rows-1)
405             img[x][y] = 0
406         for i in range(100):
407             x = random.randint(0,self.frame_columns-1)
408             y = random.randint(0,self.frame_rows-1)
409             img[x][y] = 255
410         im = Image.fromarray(img)
411         return img
412
413     def AddGaussNoise(self,img):
414         noise = np.random.normal(0,3,(self.frame_columns,self.frame_rows))
415         noise = noise.astype(np.uint8)
```

```
416         img = noise + img
417         return img
418 
419     def Laplacian1(self,img):
420         img = np.intc(img)
421         myKer =np.array([[0, -1, 0],
422                          [-1, 4,-1],
423                          [0, -1, 0]])
424         return signal.convolve2d(img,myKer,mode='valid')
425 
426     def modifiedLaplacian2(self,img):
427         imgA = np.double(img)
428         for j in range(self.frame_columns-2):
429             for i in range(self.frame_rows-2):
430                 imgA[j+1][i+1] = float(abs(2*img[j+1][i+1]-img[j+2][i+1]-img[j
    ][i+1])
431                                 + abs(2*img[j+1][i+1]-img[j+1][i+2]-img[j+1][i])
432                                 + 1/math.sqrt(2)*abs(2*img[j+1][i+1]-img[j+2][i
    +2]-img[j][i])
433                                 + 1/math.sqrt(2) *abs(2*img[j+1][i+1]-img[j+2][i
    ]-img[j][i+2]))
434         return imgA
435 
436     def modifiedLaplacian3(self,img):
437         imgA = np.intc(img)
438         for j in range(self.frame_columns-2):
439             for i in range(self.frame_rows-2):
440                 imgA[j+1][i+1] = 4*float(abs(2*img[j+1][i+1]-img[j+2][i+1]-img[
    j][i+1])
441                                 + 4*abs(2*img[j+1][i+1]-img[j+1][i+2]-img[j+1][i
    ])
442                                 + abs(2*img[j+1][i+1]-img[j+2][i+2]-img[j][i])
443                                 + abs(2*img[j+1][i+1]-img[j+2][i]-img[j][i+2]))
444         return imgA
445 
446 
447     def threshold_count(self,img,value):
448         count = 0
449         print(np.max(img))
450         for j in range(len(img)-1):
451             for i in range(len(img[1])-1):
452                 if img[j][i] > value:
453                     count = count + 1
454         return count
455 
456     def threshold(self,img,value):
457         sum = 0
458         for j in range(len(img)-1):
459             for i in range(len(img[1])-1):
460                 if img[j][i] > value:
461                     sum = sum + img[j][i] - value
462         return sum
463 
464     def variance(self,img):
465         return np.var(img)
466 
467     def wavelets1(self,img,wavelet = 'bior1.3'):
468         img = np.intc(img)
469         LL, (LH, HL, HH) = pywt.dwt2(img, wavelet)
470         return np.mean(np.abs(LH)) + np.mean(np.abs(HL)) + np.mean(np.abs(HH))
```

```
471
472     def wavelets2(self,img,wavelet = 'bior1.3'):
473         img = np.intc(img)
474         LL, (LH, HL, HH) = pywt.dwt2(img, wavelet)
475         return np.var(LH) + np.var(HL) + np.var(HH)
476
477     def autocoralation(self,img):
478         sumA = 0
479         sumB = 0
480         for j in range(self.frame_columns-1):
481             for i in range(self.frame_rows-1):
482                 if i != self.frame_rows-2:
483                     sumB = sumB +  int(img[j][i]) * int(img[j][i+2])
484                 sumA = sumA +  int(img[j][i]) * int(img[j][i+1])
485         return sumA - sumB
486
487     def Meldelson(self,img,value):
488         img = np.intc(img)
489         sum = 0
490         print(np.mean(img))
491         for j in range(self.frame_columns-1):
492             for i in range(self.frame_rows-1):
493                 if img[j][i] > value:
494                     sum = sum + img[j][i] - value
495         return sum
496
497     def StdCorelation(self,img):
498         sum = 0
499         mu = np.mean(img)
500         for j in range(self.frame_columns-1):
501             for i in range(self.frame_rows):
502                 sum = sum + int(img[j][i])*int(img[j+1][i]) - mu
503         return sum
504
505     def saveFrame(self,name,dist,srcpath):
506         plt.imsave( srcpath + '/'+str(name)+"--"+self.alg+ ".png" ,
507             self.frames[self.distances.index(dist)],cmap ='Greys')
508
509 class all_detectors:
510     def __init__(self,used_alg = [],names=[] ):
511         self.used_alg = used_alg
512         self.edge_detectors = []
513         self.names = names
514         for alg in used_alg:
515             self.edge_detectors.append(Edge_detector(alg))
516
517
518     def uppdate(self,dist,img):
519         for detector in self.edge_detectors:
520             detector.add_frame(dist,img)
521
522     def animate(self,to_animate = "all"):
523         if to_animate == "all":
524             for detector in self.edge_detectors:
525                 detector.animate()
526         for count,alg in enumerate(self.used_alg):
527             if(to_animate == alg):
528                 self.edge_detectors[count].animate()
529
530     def plot_max_values(self,to_plot="all"):
```

```
531         fig = plt.figure()
532         if to_plot == "all":
533             for detector in self.edge_detectors:
534                 detector.plot_max_values()
535         for count,alg in enumerate(self.used_alg):
536             if(to_plot == alg):
537                 self.edge_detectors[count].plot_max_values()
538         plt.xlabel('Distance')
539         plt.ylabel('Sharpness')
540         plt.title('Max value')
541         plt.grid(True)
542         plt.legend()
543         plt.show()
544 
545     def plot_mean_values(self,to_plot="all"):
546         fig = plt.figure()
547         if to_plot == "all":
548             for detector in self.edge_detectors:
549                 detector.plot_avg_values()
550         for count,alg in enumerate(self.used_alg):
551             if(to_plot == alg):
552                 self.edge_detectors[count].plot_avg_values()
553         plt.xlabel('Distance')
554         plt.ylabel('Sharpness')
555         plt.title('Mean value')
556         plt.grid(True)
557         plt.legend()
558         plt.show()
559 
560     def plot_sharpness_values(self,to_plot="all"):
561         fig = plt.figure()
562         color_index = 0
563         if to_plot == "all":
564             print(self.names)
565             for detector in self.edge_detectors:
566                 detector.plot_sharpness(color_index=color_index,names = self.
    names)
567                 color_index = color_index +1
568         for count,alg in enumerate(self.used_alg):
569             if(to_plot == alg):
570                 self.edge_detectors[count].plot_sharpness(color_index=
    color_index)
571                 color_index = color_index +1
572         plt.xlabel('Distance')
573         plt.ylabel('Sharpness')
574         plt.grid(True)
575         plt.legend()
576         plt.show()
577 
578     def save_sharpness_graph(self,name,path="",to_plot="all"):
579         fig = plt.figure()
580         color_index = 0
581         if to_plot == "all":
582             for detector in self.edge_detectors:
583                 detector.plot_sharpness(color_index=color_index,names = self.
    names)
584                 color_index = color_index +1
585         for count,alg in enumerate(self.used_alg):
586             if(to_plot == alg):
587                 self.edge_detectors[count].plot_sharpness(color_index=
```

```
    color_index)
588                 color_index = color_index +1
589         plt.xlabel(r'$Sensor\;position [\mu m]$')
590         plt.ylabel(r'$Sharpness [-]$')
591         plt.grid(True)
592         plt.legend()
593         plt.savefig(path+"/results/"+name+".png",dpi=500)
594 
595     def save_frame(self,name,dist,srcpath):
596         for detector in self.edge_detectors:
597             detector.saveFrame(name,dist,srcpath+"/results/")
598 
599     def printcurveInfo(self,name,srcpath):
600         for detector in self.edge_detectors:
601             print("New detector: "+str(detector.alg))
602             detector.printcurveInfo(name,srcpath+"/results/")
603 
604 def getFiletype(filename) -> str() :
605     dot = 0
606     for i in range(len(filename)-1, 0, -1):
607         if filename[i] == '.':
608             dot = i+1
609             break
610     return filename[dot:]
611 
612 def getFileName(filename) -> str() :
613     dot = 0
614     for i in range(len(filename)-1, 0, -1):
615         if filename[i] == '.':
616             dot = i+1
617             break
618     return filename[:dot-1]
619 
620 
621 if __name__ == "__main__":
622     paths = [
623         "H:/Autofocus/Datasets/5_dataset"
624     ]
625 
626     for srcpath in paths:
627         print("Start: "+srcpath)
628         directories = os.listdir( srcpath )
629         file_name_ints = np.array([])
630         for filename in directories:
631             if getFiletype(filename) != "png":
632                 continue
633             file_name_int = int(getFileName(filename))
634             file_name_ints = np.append(file_name_ints,file_name_int)
635         file_name_ints.sort()
636 
637         test = []
638         test_name = []
639         # Image statistic
640 
641         test_name.append('Used')
642         my_detectors = all_detectors(['MedianBlur-3+grad-prewit3+Threshold-10'
    ],
643             names=['MedianBlur3+PrewitTenengrad'])
644         test.append(my_detectors)
645 
```

```
646         for i in range(len(test)):
647             print("Iterace "+str(i)+"/"+str(len(test)) + " - " + test_name[i])
648             for filename in file_name_ints:
649                 fileName = str(int(filename)) + ".png"
650                 img = cv2.imread(srcpath + '/'+fileName,cv2.IMREAD_GRAYSCALE)
651                 test[i].uppdate(filename,img)
652 
653             with plt.style.context('bmh'):
654                 test[i].printcurveInfo("DATA"+test_name[i],srcpath)
655                 test[i].save_sharpness_graph(test_name[i],srcpath)
```