BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INTELLIGENT SYSTEMS
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

REAL-TIME LIQUID LENS FOCUSING

ZAOSTROVANIE TEKUTEJ ŠOŠOVKY V REÁLNOM ČASE

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **MARTIN ZAŤOVIČ**
AUTOR PRÁCE

SUPERVISOR **Ing. ŠTĚPÁN RYDLO**
VEDOUCÍ PRÁCE

BRNO 2022

Brno University of Technology
Faculty of Information Technology

Department of Intelligent Systems (DITS) Academic year 2021/2022

Bachelor's Thesis Specification

Student: **Zaťovič Martin**
Programme: Information Technology
Title: **Real-Time Liquid Lens Focusing**
Category: Embedded Systems

Assignment:

1. Study the literature regarding liquid lenses, their applications in image focusing, and the means of image sharpness detection.
2. Propose a method to evaluate the image sharpness and control the liquid lens for image focusing.
3. Implement an algorithm that evaluates the image sharpness using a method proposed in the previous point.
4. Implement an algorithm that controls the liquid lens, which allows focusing on the data acquired from the methods in previous points.
5. Discuss the achieved results, measure the time required to focus and propose methods to improve the application.

Recommended literature:

- HECHT, Eugene a A. R. GANESAN. *Optics*. Fifth edition. PEARSON INDIA, 2019. ISBN 978-9353439590.
- HONGRUI, Jiang a Zeng XUEFENG. *Microlenses: Properties, Fabrication and Liquid Lenses*. University of Wisconsin-Madison: Taylor & Francis Group, 2020. ISBN 9780367576493.

Requirements for the first semester:

- Items 1 to 3.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor: **Rydlo Štěpán, Ing.**
Head of Department: Hanáček Petr, doc. Dr. Ing.
Beginning of work: November 1, 2021
Submission deadline: May 11, 2022
Approval date: January 3, 2022

Abstract

Speed of traditional focusing algorithms is often limited by a bottleneck which is a mechanical movement of stepper motor, that moves a lens towards or away from cameras image sensor. The liquid lens technology promises to solve that issue by eliminating the need for mechanical movement, speeding up systems using it significantly. The main purpose of this thesis is to confirm or disprove that claim and to state the exact figures obtained from testing. To put the liquid lens to a practical test, an image-based auto-focus algorithm will be implemented and executed on camera systems using liquid and electromechanical lenses. An image-based auto-focus algorithm captures and processes the images at different focus values in order to calculate score of their sharpness and find its maximum.

Abstrakt

Rýchlosť zaužívaného prístupu k zaostrovaniu kamerových systémov naráža na úzke hrdlo - mechanický pohyb motora, ktorý mení vzdialenosť šošovky od obrazového snímača kamery. Technológia tekutej šošovky sľubuje riešenie tohto problému elimináciou potreby mechanického pohybu. Systémy ktoré ju využívajú sú teda podstatne zrýchlené. Cieľom tejto práce je potvrdenie, alebo vyvrátenie týchto tvrdení a v prípade ich potvrdenia zistiť do akej miery je tekutá šošovka schopná zrýchliť systémy, ktoré ju využívajú. Pre testovanie zrýchlenia implementujeme automatické zaostrenie na základe spracovania obrazu a spustíme ho na kamerových systémoch, ktoré využívajú tekutú a elektromechanickú šošovku. Automatické zaostrovanie na základe spracovania obrazu počíta skóre ostrosti zhotovených obrázkov a hľadá jeho maximum pri rôznych hodnotách zaostrenia.

Keywords

focusing, liquid lens, camera, image processing, opencv

Klíčová slova

zaostrovanie, tekutá šošovka, kamera, spracovanie obrazu, opencv

Reference

ZAŤOVIČ, Martin. *Real-time Liquid Lens Focusing.* Brno, 2022. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Štěpán Rydlo

Rozšířený abstrakt

Šošovky sa v kamerových systémoch využívajú na sústredenie obrazu na obrazový senzor kamery. Svetelné lúče prechádzajú šošovkou, ktorá mení ich smer na základe vlastností šošovky. Typicky sú šošovky vytvorené zakriveným sklom. Takéto šošovky majú fixnú fokálnu vzdialenosť, čo znamená, že ich miera sústredenia, alebo rozptýlenia svetelných lúčov je vždy rovnaká (2.1).

Jednou zo základných operácií kamerových systémov je ostrenie. Ostrením rozumieme nastavenie ostrosti obrazu. Zaužívané kamerové systémy so sklenenými šošovkami na ostrenie využívajú mechanický pohyb šošovky smerom od, alebo k obrazovému senzoru kamery. Mechnický pohyb šošovky zaisťuje typicky krokový motor. Táto operácia je však časovo náročná a tak kamerové systémy pri ostrení narážajú na úzke hrdlo (2.1).

Riešenie sľubuje technológia tekutej šošovky, ktorá pre ostrenie využíva zmenu jej zakrivenia. So zmenou zakrivenia sa mení fokálna vzdialenosť šošovky a tak je eliminovaná potreba mechnického pohybu pre dosiahnutie ostrenia. Tekutú šošovku vytvára rozhranie dvoch nezmiešateľných tekutín - typicky vody a oleja, ktoré sú uzatvorené v priestore medzi dvoma sklenenými oknami a dvoma železnými časťami. Aplikovanie striedavého napätia na na železné časti spôsobí zmenu v ich hydrofóbií. Čím väčšie napätie aplikujeme, tých viac vody k sebe pripustia a zmenia tak zakrivenie rozhrania vody a oleja viz. 1. Zmena zakrivenia nastáva takmer okamžite so zmenou napätia 2.2.

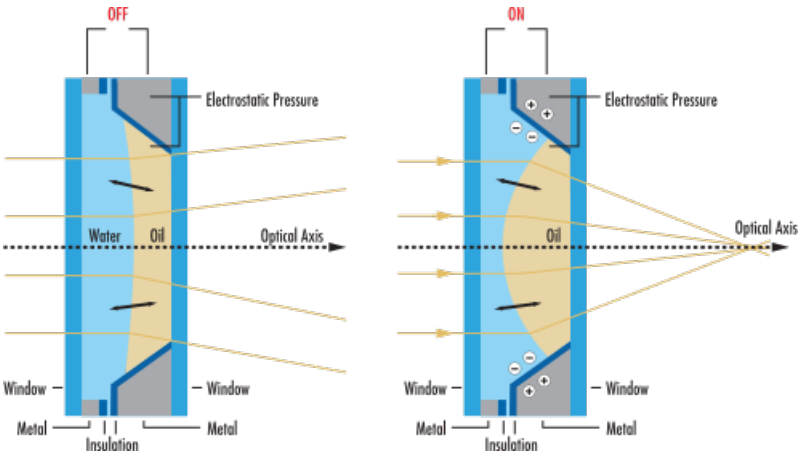

Figure 1: Princíp ostrenia tekutou šošovkou. Obrázok bol prevzatý [7].

Úlohou tejto práce je overiť, že tento prítup k ostreniu je rýchlejší, ako prístup, ktorý využívajú elektromechanické šošovky. Pre overenie budeme merať čas potrebný na zmenu napätia na železných častiach šošovky a porovnáme ho s časom potrebným na preostrenie elektromechnickej šošovky cez rovnakú vzdialenosť (3.4).

Kamerové systémy využívajú rôzne prístupy pre automatické zaostrovanie - jedným zo základných je automatické zaostrenie na základe spracovania obrazu. Výpočetná jednotka kamerového systému počíta skóre ostrosti obrázkov zhotovených pri rôznych ostrostiach obrazu a hľadá ostrosť, pri ktorej skóre dosahuje maximum. Existujú rôzne metódy pre určenie skóre ostrosti obrazu, pričom sú často založené na detekcií hrán - čím viac hrán je na obrázku, tým je ostrejší. Pre porovnanie tekutej a elektromechanickej šošovky sme implementovali algoritmus automatického zaostrenia a spustili sme zostrojených kamerových systémoch (5). Dva kamerové systémy, ktoré využívame porovnávame využívajú tekutú šošovku a jeden využíva elektromechnickú šošovku. Rozdiel v kamerových systémoch s

tekutou šošovkou je v ich výpočetnej jednotke - jeden využíva osobný laptop, druhý je prisposobený potrebám vstavaného zariadenia a využíva tak Raspberry Pi 4B. Rýchlosť ostrenia tekutej šošovky závisí na rýchlosti zmeny napätia na železných častiach. Toto napätie ovláda ovládač tekutej šošovky. Vstavaný kamerový systém využíva ovládač ADM00931 (3.1.1), zatiaľčo nevstavaný kamerový systém využíva Flexiboard (3.2.1). Pre zmeranie času potrebného na jediné preostrenie sme na osciloskope sledovali ustálenie napätia na výstupe ovládačov.

Pre testovanie algoritmu automatického zaostrenia bez závislosti na kamerovom systéme sme implementovali testovaciu suitu. Tátu suita vytvorí testovaciu sadu obrázkov zhotovených pri rôznych ostrostiach obrazu. Algoritmus automatického ostrenia potom načítava obrazky z pamäti a stáva sa tak nezávislým na spomaleniach daných kamerovým systémom(načítanie obrázku, preostrenie).

Výsledky:

-	**1/25**	**1/5**	**2/5**	**1/2**	**3/5**	**4/5**	**1**
ADM00931	20	75	90	100	105	130	150
Flexiboard	15	80	95	105	115	125	140
EM systém	200	740	1440	1780	2115	2820	3485
Average difference	182,5	662.5	1347.5	1677.5	2005	2,692.5	3,340

Table 1: Porovnanie preostrení kamerových systémov cez zlomky maximálnej vzdialenosti, na ktorú dokážu zaostriť. EM - kamerový systém, ktorý využíva elektromechanickú šošovku.

Automatické zaostrenie:

-	Celkový čas
KSTŠ1	3,48
KSTŠ2	2,06
KSEM	17,48

Table 2: KSTŠ1 - vstavaný kamerový systém s tekutou šošovkou, KSTŠ2 - nevstavaný kamerový systém s tekutou šošovkou, KSEM - kamerový systém s elektromechnickou šošovkou. Časy sú udávané v sekundách.

Testovacia suita, nám pomohla určiť úzke hrdlá algortmu automatického zaostrenia. Sú nimi načítanie obrázkov z kamery a výpočet skóre ostrosti obrázku. Výpočet skóre ostrosti môže byť optimalizovaný výpočtom na grafickej karte, alebo použitím výkonnejšej výpočetnej jednotky.

Real-time Liquid Lens Focusing

Declaration

I hereby declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of Ing. Štepán Rydlo. The supplementary information was provided by Ing. Adam Trhoň, Ing. Jan Váňa, Ing. Vilém Jelen and Ing. Radim Pavlik. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

.....
Martin Zaťovič
May 11, 2022

Acknowledgements

I would first like to thank my supervisor, Ing. Štěpán Rydlo, whose expertise was invaluable in formulating the research questions and methodology. Your insightful feedback pushed me to sharpen my thinking and brought my work to a higher level.

I would also like to acknowledge my colleagues at Touchless Biometrics for providing core insights into the technicalities of this work. I would particularly like to thank Ing. Adam Trhoň for his patient support and for all of the opportunities I was given to further my research.

My final thanks go to my partner, family and friends who provided stimulating discussions as well as happy distractions to rest my mind outside of my research.

Contents

Chapter 1

Introduction

Camera systems in various devices achieve focusing by moving the lens towards or away from the camera's image sensor. This movement is controlled by a precise stepper motor [6]. This approach brings a disadvantage of being considerably slow, as the movement of mechanical parts takes a significant amount of time.

A liquid lens eliminates the need for mechanical movement by a unique solution - making a flexible lens using liquids. Such a lens can change its curvature depending on the voltage applied to it. Many sources claim the supremacy of liquid lens technology over its electro-mechanical alternative in terms of performance. These claims are often not supported by any exact figures comparing the two. A part of this thesis is about finding these figures and deciding whether the liquid lenses really are faster and by how much. We will also discuss the technology, advantages and disadvantages of the liquid lens and its real-world applications.

One of the most common actions imaging systems undertake is auto-focus. By auto-focus, we understand the action of bringing a manually or automatically selected object to focus. While there are several approaches to achieving auto-focus, this thesis is mainly concerned with the one that does not require any additional sensors besides the camera's image sensor - an image-based auto-focus. It requires the lens to refocus several times, making it a suitable comparison operation. In order to test and compare the performance of liquid lenses, we will assemble three camera systems and make them perform the same auto-focus algorithm.

A fundamental part of image-based auto-focus methods is sharpness score assessment. The score is usually calculated from the number of visible edges on the image. More edges in an image usually mean it is sharper. The edges are separated from the image using edge detectors like the Sobel operator or Canny edge detector. Each edge detector yields different results, so we will compare four of the most well-known ones and pick one for our auto-focus implementation.

Chapter 2

Lenses in technology

The first chapter is dedicated to explaining the principles of how lenses work and the technology behind liquid lenses. Liquid lens is a fascinating technology that deserves plenty of attention to make sure it is understood well. The knowledge we gather will be used to make the most of its application later. We will explain the differences between the standard lenses used in most modern-day camera systems and the liquid lenses. These differences make liquid lenses stand out in more than one way and make up the basis of their increasing popularity. Then we will explain the principles of making liquids into lenses and controlling their curvatures using electrical energy.

2.1 Lenses

A lens is an optical device that changes the direction of light beams. Cameras use it to concentrate captured light onto a small image sensor. Image sensors in cameras measure the attenuation of waves hitting the sensor - the gradual loss of flux intensity [1] through a medium. An example of attenuation is dark glasses that attenuate sunlight [20]. It converts the variable attenuation of light waves (as they pass through or reflect off objects) into signals, small bursts of current that convey the information [21]. This data is then plotted onto a virtual frame resulting in a photo.

The light beams reflected from objects within the lens's sight enter the lens in parallel. The lens changes the light beam's direction so that they all converge in a single point called the focal point. Negative focal length means that the focal point is located on the same side of the lens as the objects that are being captured. It means that the lens is concave, and when the parallel beams hit it, they are dispersed on its opposite side. The dispersion angle is inversely proportional to the focal length - the smaller the focal length, the stronger the dispersion. Image 2.1 demonstrates these principles.

The focal length measures how strongly a lens concentrates or disperses the captured light. In cameras, it stands for the distance from the image sensor to the optical centre of the lens[3]. Positive focal length means that the focal point is located on the lens's opposite side as the objects that are being captured. Such lens is convexly shaped.

[1] Radiant flux (also known as radiant power) is the radiant energy emitted, reflected, transmitted or received per unit time. The SI unit of radiant flux is the watt (W), which is the joule per second (J/s) in SI base units. Luminous flux, in contrast, takes into account the varying sensitivity of the human eye to different wavelengths of light - is photometrically weighted radiant flux (power). The SI unit of luminous flux is the lumen (lm). One lumen is defined as the luminous flux of light produced by a light source that emits 1 Candela of luminous intensity over a solid angle of $\omega = 1sr$ [4].

Different lens shapes are used for capturing different types of images. A concave lens is typically used in macro photography in combination with a convex lens to achieve optimal image properties.

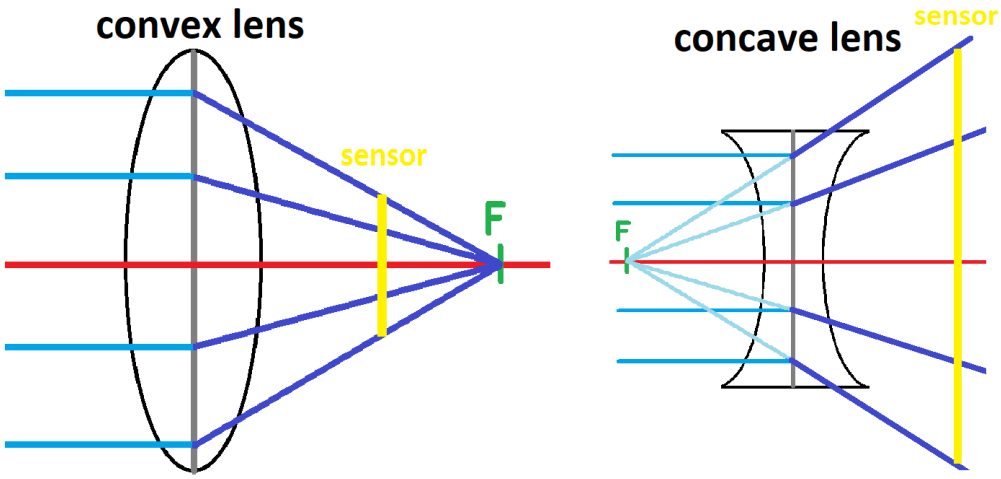

Figure 2.1: Principle of light reflection of convex and concave lens

2.1.1 The principle of focusing with a lens

The traditional lenses have a fixed focal length. Such lenses always concentrate the light with the same strength. Focusing is then achieved by moving the lens towards or away from the image sensor [**?**]. Cameras usually use a stepper motor to execute such movements. Any physical movement of a device's component is costly, especially when the movement must be precise.

On the other hand, liquid lenses take advantage of their variable focal length, which changes with respect to their curvature. It is possible to bend a liquid lens into convex and concave shapes, allowing for a broad focus range.

2.2 Liquid lenses

A liquid lens is a technology that allows for rapid focusing - quickly adjusting focus to accommodate objects located at various working distances (WDs). The changes in focus occur within milliseconds, allowing auto-focusing or scanning of various WDs in minimal time. The lens is also much more durable, as there are no mechanical parts that would wear over time. These properties make liquid lenses superior to traditional lenses in almost any application.

Nowadays, many businesses fund research on possible applications of liquid lenses in their products. From microscopes, telescopes and biometric devices to depth scanners and smartphones, they prove effective in almost any camera device. Although liquid lenses are slowly building a reputation and gaining popularity among technology producers, many are still reluctant to use them. One of the main reasons for it is the price, which is higher than traditional lenses. Its price, however, has experienced a steady decline since its discovery.

We have examined this technology thoroughly to find the extent of its advantages, and we have looked for its negatives. This chapter gives insight into these advantages and disadvantages, history, principles and applications of liquid lenses.

2.2.1 Technology

A liquid lens is formed by two immiscible liquids placed inside the same container. The container has two glass windows to let the light pass through. One of the liquids is typically water or an aqueous solution, and the other is a transparent oil. They stay separated, forming an interface in between. Two metal nodes entrap the liquids from the top and bottom of the container. They are cut under an angle, and they touch both liquids.

This interface forms a curvature resembling a typical lens when electrical energy is applied to the nodes - increasing the electrical energy results in the metal nodes becoming less hydrophobic. Consequently, more water flows towards their surface, and the oil forms a greater curvature. This process is called electrowetting[2] The change of curvature of a liquid lens changes its focal length. This behaviour is used to change the focus.

The image 2.2 illustrates the working principle of the Corning® Varioptic® Liquid Lens. The metal nodes become electrically charged, making them less water repellent, allowing more water to touch their surface [11].

Using liquids to form a lens was first introduced by Prof Dr Werner B. Schneider in 1988. He has successfully built his idea into a working prototype, demonstrating it to his students. Despite its potential, the idea remained unstudied and was only used in 2002.

That year, American scientist Tom Krupenkin and his colleagues built a liquid lens as we know it today. Nowadays, most fundamental patents (e.g. on the chamber shape and centring of the drop) are held by the company Corning® Varioptic®, and they continue the main line of research [19].

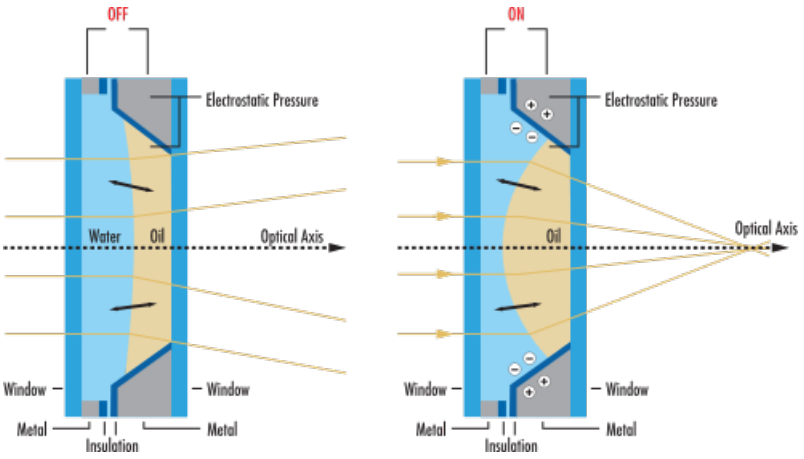

Figure 2.2: Liquid lens principle, image taken from [7]. Electrical field applied to the metal nodes touching both liquids changes the shape of their interface.

The first smartphone to use a liquid lens is `Xiaomi Mi Mix Fold`, released in 2021. It uses multiple cameras, one of which has a liquid lens mounted. The possibility of bending a liquid lens into a concave shape is beneficial for capturing macro images.

2.2.2 Applications

As liquid lenses gained popularity, various companies started to find them helpful in their produced devices. Today, we can find liquid lenses in digital photography, biometric devices,

[2]Electrowetting describes the change of the wetting properties of the materials when an electric field is applied to them.

telescopes, industrial data capture, barcode reading devices, and many more devices across various industry sectors.

These lenses are often used in applications where fast focusing is needed. That includes various product quality checks, where the products are being photographed on a production line, and they need to be focused on individually.

Biometric companies utilize their capabilities in devices based on imaging, such as touchless fingerprint scanners. It is often required of such sensors to perform an accurate auto-focus within a short period. No matter the auto-focus method, in a vast amount of cases, liquid lenses outperform conventional solutions.

Most modern-day smartphones use several cameras with different lenses integrated into a single camera system to compensate for the inability to capture macro and wide-angle images by a single lens. A single camera with a liquid lens could one day replace these systems. The liquid lens's ability to bend into a concave shape is mainly utilized in macro photography.

2.2.3 Strengths, weaknesses and accuracy

The main strength of liquid lenses is the short time it takes to change its focal length and, therefore, refocus. There is no need for mechanical changes within the lens, as its curvature changes almost instantly by changing the voltage applied to the metal nodes inside it.

Most liquid lenses can perform the change of voltage on the nodes within tens of milliseconds. The shape of the interface of the liquids changes almost instantly as the voltage changes.

The rate of change of the focal length depends on the change of voltage applied to it. Refocusing speed is thus limited by the rate of change of the voltage outputted by a controller - the so-called driver. If this device is slow, the whole refocusing cycle will be slow.

The size of a liquid lens is limited, as increasing its diameter lowers its resolution. It is likely due to the optical properties of the liquids - the possibility of forming optical errors in the resulting images increases with a larger diameter.

The accuracy of a liquid lens depends on external factors. These factors include the environment's temperature or the pressure under which the lens operates. A liquid lens is not affected by a movement such as that of a CCTV camera.

Another critical factor to the accuracy of the lens is the number of cycles of refocusing it has undergone. Most liquid lenses can withstand around 50 000 000 cycles of refocusing. That is around 500 times more than what the typical lens can withstand. It is mainly thanks to the absence of mechanical parts that would wear over time. This property makes the liquid lens a perfect gadget for industrial purposes.

2.3 Liquid lens controls

It is a common misunderstanding that liquid lenses are controlled by pulse-width modulation of the alternating voltage. Instead, liquid lenses are generally controlled by increasing or decreasing the alternating voltage amplitude applied to its metallic nodes. The liquid lens driver applies the alternating voltage. It is typically measured in V_{RMS}. The drivers typically output an alternating voltage in a range from $10V_{RMS}$ to $60V_{RMS}$. The exact numbers depend on the lens used. The controlling signal is analyzed using an oscilloscope in the section 3.4.

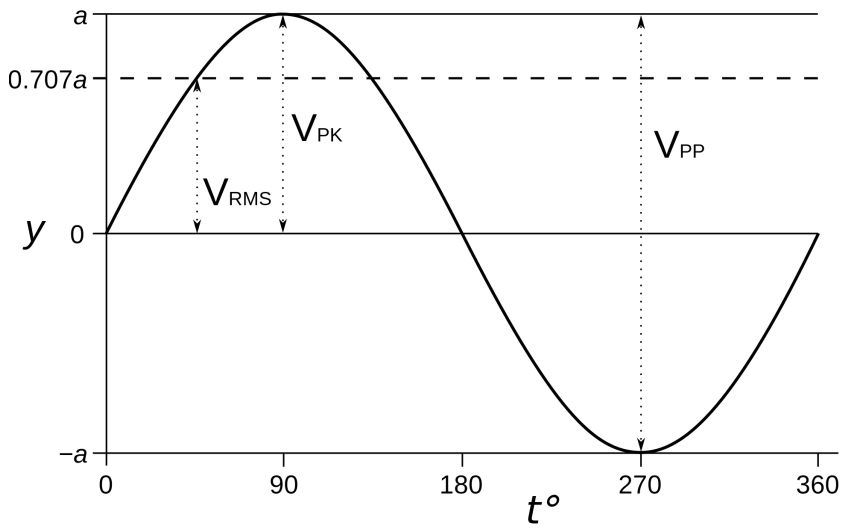

Figure 2.3: An ideal alternating voltage graph showcasing meaning of RMS voltage. Image taken from https://www.allaboutcircuits.com/tools/rms-voltage-calculator/. A real graph often looks significantly more distorted.

The driver's alternating output voltage is set by writing a value into its focus value register.

2.4 Focus value and focus distance of a camera

A driver device usually controls a liquid lens's focus. The driver accepts a value sent by the MCU and accordingly sets the voltage on its output, ultimately changing the lens's focus. Suppose sending the value 0 makes the camera focus on the furthest point the lens can focus on, and sending the maximal value sets the lens to focus on the closest point it can focus on. The drivers are usually 8, 10 or 16-bit, meaning their maximal values are 255, 1023, and 65535, respectively. As this value sets the focus of the lens, we will call it the focus value further on.

Focus distance refers to the distance from the lens to the objects that are in focus at a given focus value. The maximal distance a camera can focus on depends on the distance of the lens from the camera's image sensor.

Suppose a lens is focused on a point in the maximal focus distance of a camera(focus value 0). By writing the focus value 1 to the driver, the lens refocuses on a point at a different focus distance. The following formula gives the distance between these two points:

$$(FD\_max - FD\_min)/FR\_max \tag{2.1}$$

where

- FD_max is the maximal focus distance
- FD_min is the minimal focus distance
- FR_max is the maximal value writable to the driver's focus value register

Chapter 3

Tested camera systems

This chapter will discuss the hardware of the imaging devices that we will build. We will build 3 cameras - two of them will use a liquid lens, and one, used for comparison, will use a mechanical lens. One of the liquid lens cameras will be implemented as an embedded system, while the others MCU will be a personal laptop. A proper assembly of the device is essential to its functionality. We will use images and schemes to help explain it in detail. We will also link the datasheets of individual components used in this project in the attachments.

The last part of this chapter summarizes the testing of the liquid lenses they underwent. We will measure the exact time needed for the liquid lens to refocus and compare it to the time taken by the mechanical lens.

3.1 Embedded liquid lens camera

The first camera we will assemble is a camera that might be used in embedded devices, as it uses Raspberry Pi as an MCU.

3.1.1 Hardware

We will discuss the specifications of each part to be used.

- Edmund Optics M12 16mm Liquid Imaging Lens
- ADM00931 Inductorless Liquid Lens Driver Board
- DMM37UX252ML monochrome USB 3.1 camera 3.2MP
- Raspberry Pi 4 Model B

Edmund Optics M12 16mm Liquid Imaging Lens

The producer of this lens states that it is designed for fast electronic focus, superior image performance and quick auto-focus and is applicable in high-speed machine vision. The lens's 16mm focal length version supports cameras with a 1/1.8" maximal chip format. It is controlled via a four-channel flexible cable.[7]

Liquid lens parameters:

- Focal length: 16mm

- Maximal camera sensor format: 1,8”
- Working distance: 220mm - infinity
- Wavelength range: 400nm - 700nm
- IR Cut filter: No
- Field of view at maximal sensor format: horizontal - 109mm - 25.4°, vertical - 81.7mm - 19.2°, diagonal - 136.2mm - 31.5°

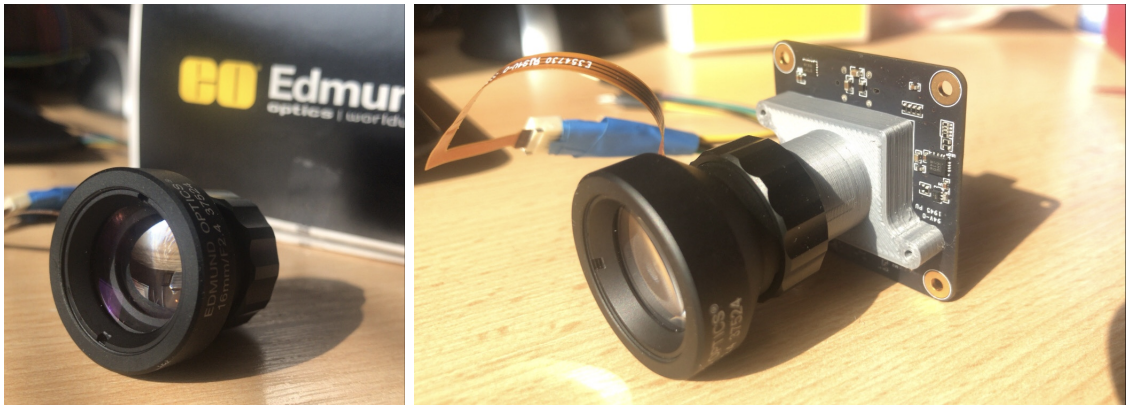

Figure 3.1: The lens is sold either as an individual component or along with a mount system. The first picuture shows the lens with a mount system and the second image shows the lens mounted onto a camera.

ADM00931 Inductorless Liquid Lens Driver Board

ADM00931 is used to drive several liquid lens models; Edmund Optics 16mm Liquid Imaging Lens is one of them. It contains A HV892DB1 chip, controlled via an I2C interface. The driver board processes a piece of single-byte information (focus value) sent by an MCU via I2C and generates the respective voltage on the output. It is a straightforward driver board to operate, as there are no additional registers.

The board uses these pins:

- `VIN` - input voltage(2,65 - 5,5V)
- `SCL` - I2C clock(max 400kHz)
- `SDA` - I2C data
- `VDD` - reference voltage(1,7 - 2,95V)
- `GND` - ground
- `OUT1/OUT2` - output pins

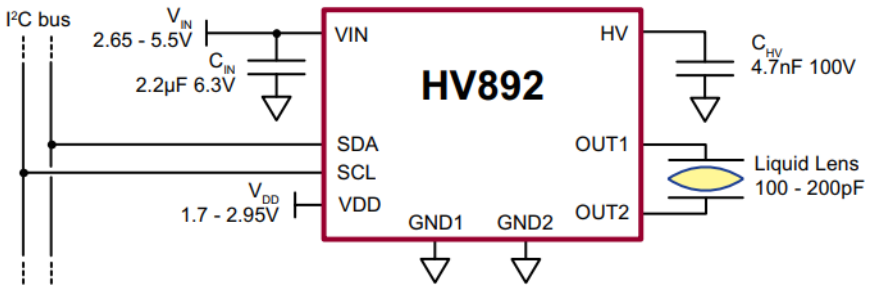

Figure 3.2: Driver board schematic

Since the focus register is 8 bits, the driver board can set 256 different focus values. The MCU sets the focus value using an I2C communication at a 400kHz rate. There is no need to send the address of a register as the driver board only has one. It is not possible to read any data from the driver board.

Focus value legend[9]:

- `0`: stand-by mode, the chip is inactive, output voltage is close to 0 V_{RMS}
- `1`: initiates the chip(exits stand-by mode), the output voltage is ramped up to 9.3 V_{RMS}
- `2 - 255`: setting these focus values sets the output RMS voltage to $9,511 - 63V_{RMS}$, incrementing the focus value by one adds 0.211 V_{RMS} to the current output RMS voltage

To set an output voltage, the driver accepts an 8-bit focus value and sets the RMS voltage calculated by:

$$V_{RMS} = N * 0.208 + 9.6Volts \tag{3.1}$$

where N is a focus value from 0x00($9.6V_{RMS}$) to 0xFF($62.64V_{RMS}$).

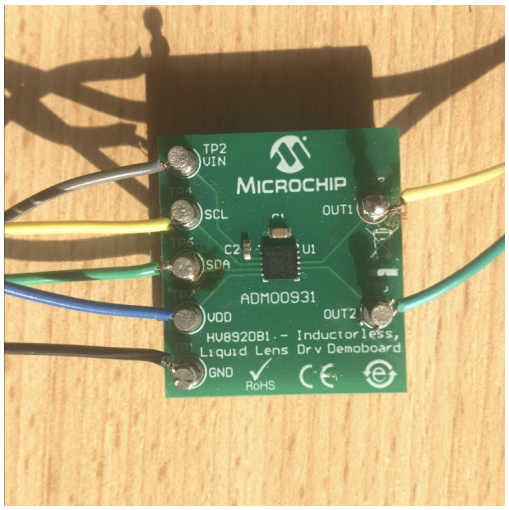

Figure 3.3: ADM00931 Inductorless Liquid Lend Driver Board

DMM37UX252ML monochrome USB 3.1 camera 3.2MP

DMM37UX252ML is a camera system using the Sony CMOS Pregius sensor(IMX252). It uses an M12x0.5 lens mount, compatible with Edmund Optics 16mm Liquid Imaging Lens. It uses USB 3.1 technology since a high data throughput is required to support its high framerate and resolution. High framerate is especially important for image-based auto-focusing applications, as many images need to be captured and analyzed. The camera can be operated using three different APIs - UVC/V4L2(Linux), IC Imaging Control(Windows) and USB3 Vision.

Camera parameters:

- Resolution: 2,048×1,536 (3.1 MP)
- Framerate using maximal resolution: 119 fps
- Video output formats: 8-Bit Monochrome, 16-Bit Monochrome
- Sensor format: 1/1.8”
- Manual trigger: yes

A manual trigger is used to capture frames at specific points in time. The camera’s exposure needs to be synchronized with the lighting to implement a camera flash. For that purpose, the strobe pin is used. The camera uses dedicated wires for the trigger and strobe signals, as shown in picture 3.4.

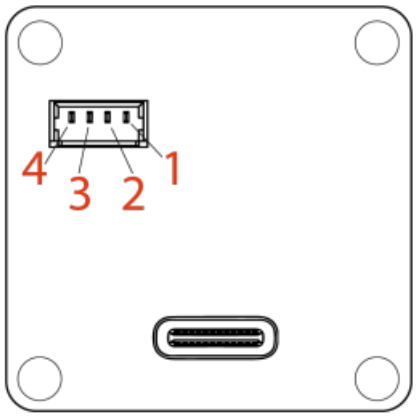

Figure 3.4: Trigger and strobe signals connection

Image 3.4 legend:

1. `TRIGGER_IN(+)` - Optocoupler signal $(3, 3^2 - 24.0^2 V)$
2. `TRIGGER_IN(-)` - Optocoupler ground
3. `STROBE_OUT` - Open drain
4. `GND_I/O` - External ground

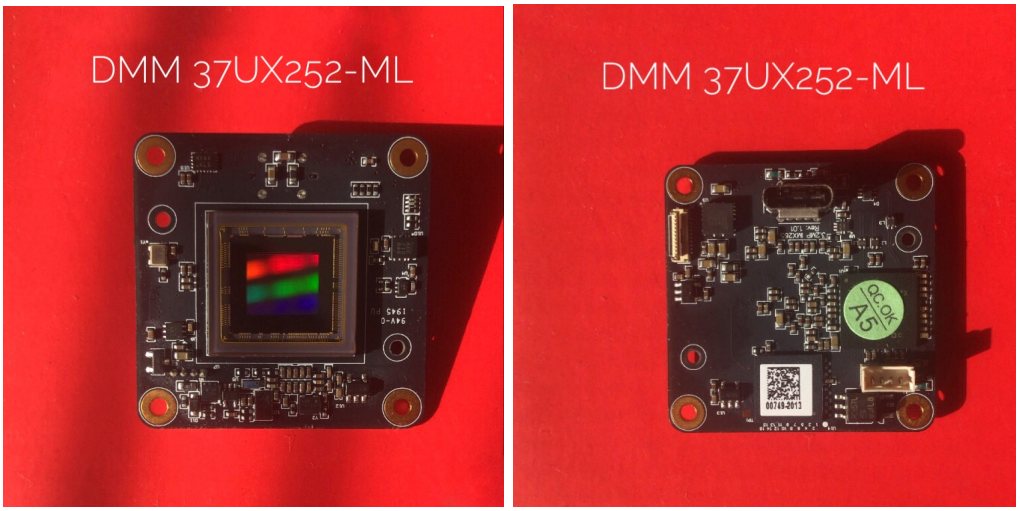

Figure 3.5: Camera - front and back

Raspberry Pi 4 Model B

The Raspberry Pi 4 computer is used as an MCU. It will communicate with all the other parts to execute an auto-focus algorithm. It will obtain frames from the camera via USB 3.1 interface, control the lens's focus using I2C communication with the driver board and send trigger signals using the GPIO interface.

Raspberry Pi 4 Model B specifications:

- `Processor:` Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- `RAM:` 4GB LPDDR4-3200 SDRAM
- `USB:` 2x USB 3.0 port; 2x USB 2.0 port
- `I/O:` Raspberry Pi standard 40 pin GPIO header(I2C included)
- `Memory:` Micro SD card slot
- `Operating system:` Raspberry Pi OS (version raspios_armhf-2022-01-28)

Raspberry Pi will provide the computational power for image sharpness detection of the camera, which could be challenging to its ARM processor. It was chosen because it is one of the most popular, accessible embedded platforms available. Installing the Raspberry Pi OS provides easy access to variable camera APIs, image processing libraries and I2C interface control. The operating system will be installed on and booted from a micro-SD card.

3.1.2 Assembly

The assembly of a camera

Parts compatibility

Different parts of the camera system must be compatible with one another. Specifications of parts that need to be compatible:

- Camera sensor format with lens's maximal sensor format
- Cameras data-transfer interface (USB 3.1) has to be supported by an MCU
- Driver board's communication interface(I2C) has to be supported by an MCU
- Trigger signal interface needs to be supported by an MCU

Camera connection

The camera is connected using USB3.1 - USB C cable. The maximal amount of data outputted by the camera is calculated using its maximal resolution, maximal framerate and biggest image format:

$$\text{maximal\_resolution} * \text{maximal\_framerate} * \text{image\_format} \tag{3.2}$$

In the case of DMM37UX252ML:

$$(2048 * 1536)\text{px} * 119\text{frames} * 2\text{bytes/pixel} = 748,683,264\text{B/s} \ = 748.68\text{Mb/s} \tag{3.3}$$

It is essential to use a USB cable that supports the USB3 technology as the camera requires a connection with minimal data throughput of 748,68Mb/s. USB2 technology supports data throughput up to 480Mb/s, which is not sufficient.

Trigger is connected using two wires - one connects the `TRIGGER_IN(+)` pin of the camera(3.4) to Raspberry's GPIO17 pin and the other connects `TRIGGER_IN(-)` pin to one of Raspberry's ground pins.

Driver board connection

The driver board is connected using an I2C interface. I2C uses separate wires for data(`SDA`) and clock(`SCL`) channels. The input voltage pin(`VIN`) is connected to one of Raspberry Pi's 5V output pins. The reference voltage needs to be within the 1,7V - 2,95V range; however, Raspberry Pi's lowest voltage output supplies 3,3V. Instead, we will use a 5V output pin and lower the voltage using a voltage divider composed of two 110 Ohm resistors. Finally, the ground pin(`GND`) is connected to one of the ground pins. The connection scheme is described in picture 3.6.

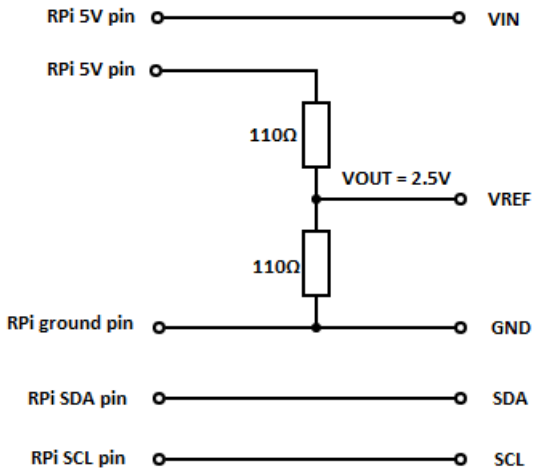

Figure 3.6: Driver board connection scheme

Trigger connection

The trigger is sent using a simple GPIO signal. The cameras `TRIGGER_IN(+)` pin is connected to Raspberry Pi's GPIO 17 pin and the `TRIGGER_IN(-)` pin is connected to one of the ground pins. The complete assembly of the camera system, depicted in image 3.7 shows all of the connections.

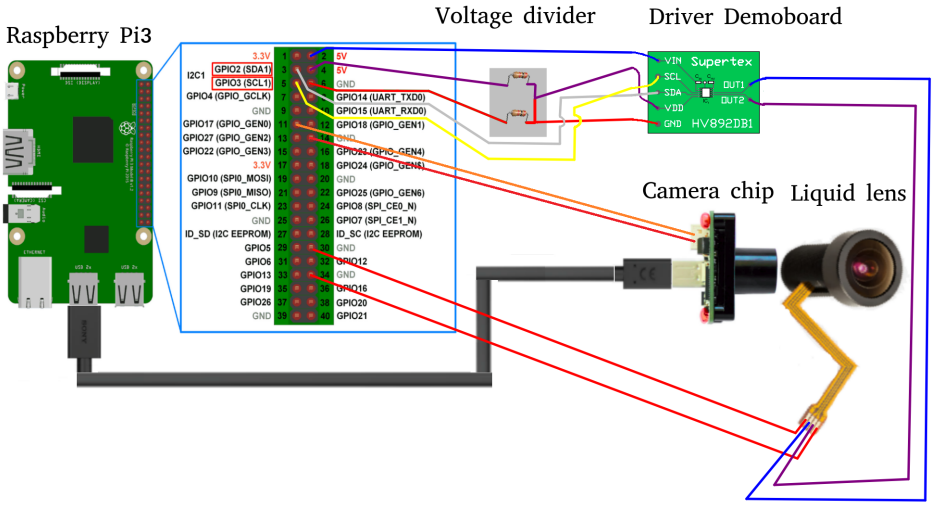

Figure 3.7: Assembly of the optical device - scheme

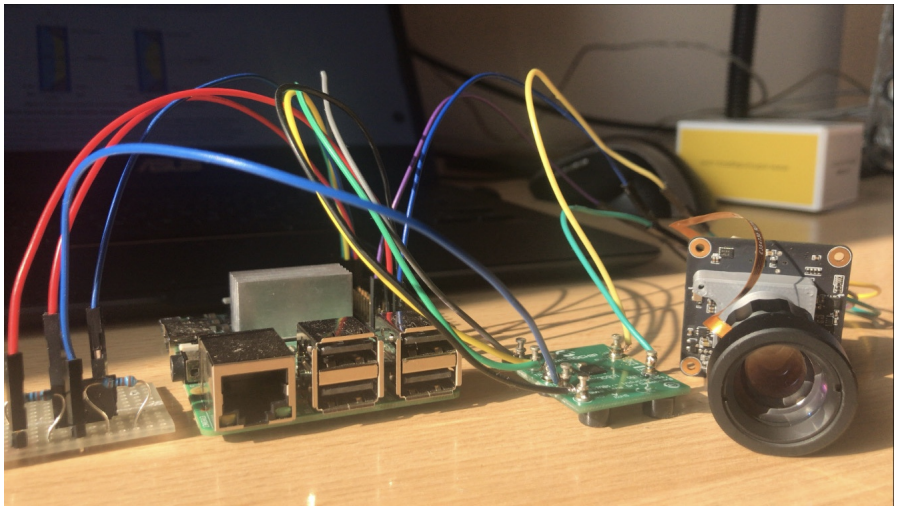

Figure 3.8: Assembly of the optical device - photo

3.1.3 Camera system controls

After assembling the hardware parts, it is essential to establish communication with the external devices. We have implemented several libraries to access these peripherals easily.

VideoCapture library

This library is used to communicate with the camera and to set its properties such as resolution, image format and maximal framerate. The library also implements a function to retrieve newly captured frames from the camera. It is based on the V4l2 API.

Important library functions:

- `VideoCapture()` - constructor, initializes the communication with the camera and sets its properties
- `VideoCapture()` - destructor, closes the communication with the camera
- `cv::Mat getImage()` - returns OpenCV's Mat type containing the captured image

AMD00931 library

The ADM00931 driver board is controlled via the I2C interface, which detects the connected device on address `0x23`. We have implemented a library called `ADM00931` to support the communication with the driver board. It is a simple library since the driver only contains a single register for setting the focus value.

The ADM00931 library initially established the I2C communication using the library `i2c-dev`. It has, however, caused a problem when a value would not be written under certain conditions. The library was then removed, and the communication was established the kernel way - via ioctl calls and writing values to system files. The `i2c-dev` was therefor replaced by these libraries: `sys/ioctl` and `fstream`.

Library functions:

- `ADM00931()` - constructor, opens the I2C interface and establishes connection with a device on address 0x23

- `~ADM00931()` - destructor, closes the I2C communication
- `setValue(unsigned int data)` - writes the focus value `data` into the drivers focus value register

Trigger library

The camera recognizes the trigger signal as a short impulse of 3,3V up to 24V. The Raspberry Pi's GPIO pins output 3,3V, so they can be used to create the signal. We have implemented the Trigger library to access the camera's trigger easily.

Library functions:

- `Trigger()` - constructor, opens the GPIO communication on Raspberry Pi's GPIO pin 17
- `~Trigger()` - destructor, closes the communication
- `sendTrigger(unsigned int delay)` - sends a trigger impulse, delay argument controls length of the impulse

3.2 Non-embedded liquid lens camera

We will build one more liquid lens camera system to see the difference in performance using a non-ARM processor. As the evaluation of image sharpness should take less time using a more powerful MCU, we expect the execution of image-based auto-focus or similar applications to be faster.

3.2.1 Hardware

The non-embedded camera system consists of the following parts:

- Edmund Optics 16mm Cx Series Fixed Focal Length Liquid Lens
- Corning® Varioptic® Flexiboard with MAX14574 driver
- IDS UI-3860CP-M-GL Rev. 2

Edmund Optics 16mm Cx Series Fixed Focal Length Liquid Lens

This lens is applicable in the same ways as the previous one. The difference is in their focal length ranges, mounting mechanism and controls. It is a bit more robust, and thanks to its greater maximal focal length, it can focus on greater distances. It is connected to an MCU by a 4-wire flexible cable.

Edmund Optics Cx Series Fixed Focal Length Lens parameters:

- Focal length: 16mm
- Maximal camera sensor format: 2/3”
- Working distance: 100mm - infinity
- IR Cut filter: No
- Field of view at maximal sensor format: horizontal: 59.9mm - 31°, vertical: 44.7mm - 23.4°, diagonal: 75.2mm - 38.3°

Corning® Varioptic® Flexiboard with MAX14574 driver

The Corning® Varioptic® Flexiboard driver controls the Cx Series liquid lens. It is an advanced driver that can be used to drive up to 4 liquid lenses simultaneously. It uses 11 registers, 6 of which are accessible. They control the output and inform MCU about the current status. To set an output voltage, Flexiboard accepts a 16-bit focus value and sets the RMS voltage calculated by:

$$V_{RMS} = N * 0.001 + 24Volts \tag{3.4}$$

where N is a focus value from 0x0000($24V_{RMS}$) to 0xB3B0($70V_{RMS}$).

List of Flexiboard's 8-bit registers:

- `FOCUS_LSB[0x00]` - 8 LSB bits of focus value
- `FOCUS_MSB[0x01]` - 8 MSB bits of focus value, the output is updated after writing to this register
- `CONTROL[0x02]` - setting this register saves the Focus and Mode registers to EEPROM memory to be loaded after reset
- `MODE[0x03]` - setting the zeroth bit(LSB) sets the flexiboard into stand-by mode, setting the first bit sets the driver to be controlled by analog input
- `SW_VERSION[0x05]` - USB-M firmware version
- `FAULT[0x0A]` - if the zeroth bit(LSB) is set, it indicates the driver is overloaded(could not reach desired output voltage), the second bit indicates that the driver is not responding to I2C requests and the third bit indicates the driver is in thermal shutdown

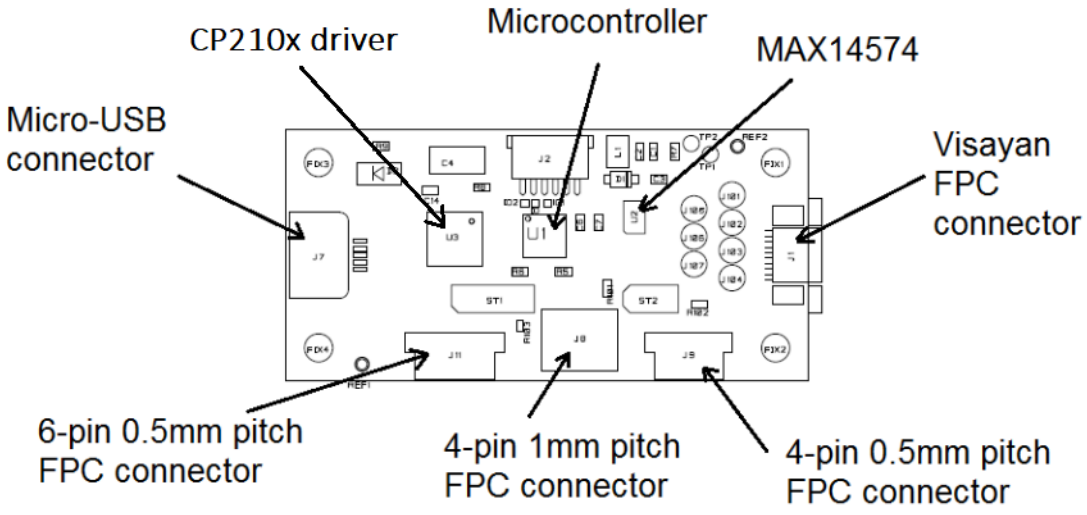

Figure 3.9: Flexiboard description

Flexiboard is connected to the MCU via USB. The kernel recognizes the USB to UART serial CP210x adapter and loads its driver. The serial communication is then translated into an I2C message by the microcontroller, and that message is sent to the MAX14574 chip. To properly establish the connection, the following Serial Bus settings are used:

- Baud-rate: 57 600 bauds
- Parity: no parity
- Data length: 8 bits
- Amount of stop-bits: 1

The Serial Bus communication follows a UART protocol described below:
Writing frame

STX	Command	Add	Nb_data	Data_1	Data_2	...	Data_n	CRC

- `STX` = 0x02
- `Command` = 0x27 (write)
- `Add` = Address of first register to be written
- `Nb_data` = number of registers to be written
- `Data_1,... Data_n` = data to be written to registers
- `CRC` = 1-byte sum of all bytes(STX, Command, Add, Nb_data, Data)

If the transaction was successful, flexiboard replies with an ACK(0x06). If it did not succeed, flexiboard sends NACK(0x15). The format of flexiboards reply message:

STX	0x37	ACK/NACK	CRC

Reading frame

STX	Command	Add	Nb_data	CRC

- `STX` = 0x02
- `Command` = 0x38(read)
- `Add` = address of the first register to be read
- `Nb_data` = number of registers to be read
- `CRC` = 1-byte sum of all bytes(STX, Command, Add, Nb_data)

Response of the board if transmission is successful:

0x02	0x38	Data_1	Data_2	...	Data_n	CRC

Response of the board if transmission is not successful:

0x02	0x38	NACK	CRC

The microcontroller translates the UART messages to appropriate I2C messages following the MAX14574 driver protocol found in its datasheet. The driver then acts on the message by setting the output voltage or writing to/reading from its registers.

IDS UI-3860CP-M-GL Rev. 2

This is a camera specialized for industrial use. It provides a framerate of up to 135 maximal resolution frames per second. It is often used in fast imaging lines in various factories for quality control or barcode reading.

Camera parameters:

- Resolution: 1936x1096px (2.1 MP)
- Framerate using maximal resolution: 135 fps
- Video output formats: 12-bit RGB
- Sensor: Sony IMX290LLR-C
- Sensor format: 1/3„ CMOS
- Manual trigger: yes

A manual trigger can be activated; however, the producer claims that the camera reaches half of the original maximal framerate in this mode. The V4L2 API does not support the camera; instead, the producer supplies its API. Registration on the producer's site is required in order to download it.

MCU

The non-embedded camera system is easily accessible by and compatible with a broad range of MCUs, as the liquid lens driver it uses communicates using USB instead of I2C. Since Raspberry Pi 4 includes several USB 3 ports(needed for camera connection), it is possible to use it as an MCU, making the system an embedded device. However, such an approach negates the benefits of a more powerful MCU of a proper non-embedded system.

3.2.2 Assembly

The assembly of the non-embedded camera system is much more user friendly. Using the provided flex cable, the liquid lens is easily attached to the driver. The driver and the camera are connected via the Universal Serial Bus(USB). The camera manufacturer provides a custom API to access it. After registration on the site, it can be downloaded on the link `https://en.ids-imaging.com/downloads.html`. The module taken from the API is called `CamControl.cpp`.

3.2.3 Camera system controls

This section describes the libraries implemented in order to establish the communication of an MCU with the camera and liquid lens driver.

Flexiboard library

The communication with the driver(Flexiboard) of the non-embedded liquid lens camera system is established via a serial bus. Flexiboard is connected to the USB port of the MCU, where it is recognized as a CP210x USB to serial adapter. To make the communication with the driver easier, we have implemented a custom library called `Flexiboard`, supported by a custom `SerialPort` library. The Flexiboard library provides a high-level abstraction of the Flexiboard UART communication protocol 3.2.1. We have also attempted to re-make this library into a kernel driver. This attempt has failed since a Flexiboard driver is recognized as a CP210x adapter, as the USB variables `VENDOR_ID` and `PRODUCT_ID` are set to those of one [5]. The kernel, therefore, mistakes the Flexiboard device for a CP210x adapter.

The base of the Flexiboard driver is the Flexiboard class. It implements the following functions:

- `Flexiboard(char* path)` - constructor, initializes Serial Port communication with a device associated with `path`
- `void setValue(unsigned short value)` - the function that sets the Flexiboard's focus value to `value`
- `bool isAnalogMode()` - returns true if Flexiboard is set to analog mode
- `bool isStandbyMode()` - returns true if Flexiboard is in standby mode
- `void setAnalogMode()` - sets Flexiboard to analog mode
- `void setStandbyMode()` - sets Flexiboard to standby mode
- `void clearModes()` - clears any mode set
- `void saveState()` - saves the focus value and mode to EEPROM memory to be loaded on next power on
- `int softwareVersion()` - returns the software version of Flexiboard
- `bool isNotResponding()` - returns true if Flexiboard is not responding
- `bool isThermalShutdown()` - returns true if Flexiboard is in thermal shutdown
- `bool isOverloaded()` - returns true if Flexiboard is overloaded

These functions are based reading and writing of the Flexiboards registers, so internally they call the following functions:

- `void writeFrame(unsigned char addr, unsigned short data, unsigned int dataLength)` - assembles a message accoding to the UART protocol 3.2.1
- `std::vector <unsigned char> readFrame()` - similar to writeFrame, except it assembles a message with read command
- `void writeRegister8(unsigned char addr, unsigned char data)` - writes 8 bits of `data` to register on address `addr` (calls writeFrame, to assemble a proper message)

- `void writeRegister16(unsigned char addr, unsigned short data)` - same as `writeRegister8`, except it writes to two registers at the same time (useful for setting the focus value, as it is stored in two registers)
- `unsigned char readRegister(unsigned char addr)` - reads a single register on address `addr`
- `void checkChecksum(std::vector<unsigned char> buffer)` - calculates the sum of all bytes of the assembled message, used as a checksum

SerialPort library

The SerialPort library is used to initiate the communication via serial bus. The SerialPort class contains the following functions:

- `SerialPort(char* path)` - constructor, opens the serial port associated with `path`
- `void closeSerialPort()` - closes the serial port, terminates communication
- `void configureSerialPort` - configures the serial port according to the requirements from Flexiboards datasheet
- `void SerialPort::writeMessage(std::vector <unsigned char> message)` - sends the message in `readBuf` onto the serial bus
- `unsigned int SerialPort::readMessage(std::vector <unsigned char> readBuf)` - reads a message from serial bus to the `readBuf`

3.3 Mechanical lens camera system

To provide a comparison of the liquid lenses, we will also assemble a mechanical lens camera system.

3.3.1 Hardware

This system uses a camera controlled by I2C, and the frames from the camera are transferred to the MCU using a MIPI cable. Therefore, the obvious choice for an MCU is a Raspberry Pi 4B, also used by the embedded liquid lens camera system.

Arducam Pan Tilt Zoom Camera

The Arducam Pan Tilt Zoom Camera uses a stepper motor to move the lens to achieve focusing. The motor is controlled using an I2C interface. It was designed for CCTV cameras, so another motor rotates the camera vertically and horizontally. This functionality will not be used, as we only need to measure its focusing speed.

Camera parameters:

- Maximal resolution: 2592x1944
- Framerate using maximal resolution: 15 fps
- Video-output formats: 8-bit RBG RAW, 10-bit RGB RAW

- Sensor: 1,4“ CMOS QSXGA (5MPx)
- Manual trigger: No

3.3.2 Assembly

The assembly of this camera system consists of connecting the camera and the focusing motor to the MCU. The camera is connected using a MIPI interface, while the motor is controlled using an I2C interface.

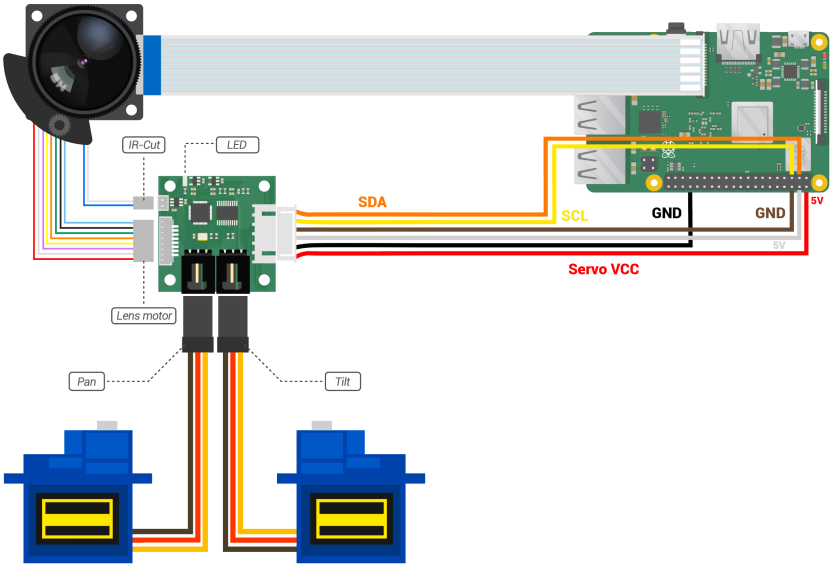

Figure 3.10: Connection of Arducam PTZ to Raspberry Pi. Image taken from Arducam website [1].

3.4 Lens testing

The main focus of this section is to figure out how fast focusing with a liquid lens is. We have already established that the shape of the liquid lens changes almost instantly after changing the electrical field applied to the nodes. The rate of change of the liquid lens focus is therefore limited by the rate of change of the voltage on its nodes. We have connected both camera systems to an oscilloscope to measure the exact time needed to change the voltage at the driver's output. The measurement starts with the driver boards receiving the focus value to be set and ends when the voltage on its output stabilizes.

Before the first measurement, we have set the ADM00931 to an initialized state by writing 0x01 to its focus register. Then we started the oscilloscope and wrote the maximal focus value(0xFF) to the focus register. The oscilloscope outputted the following graph:

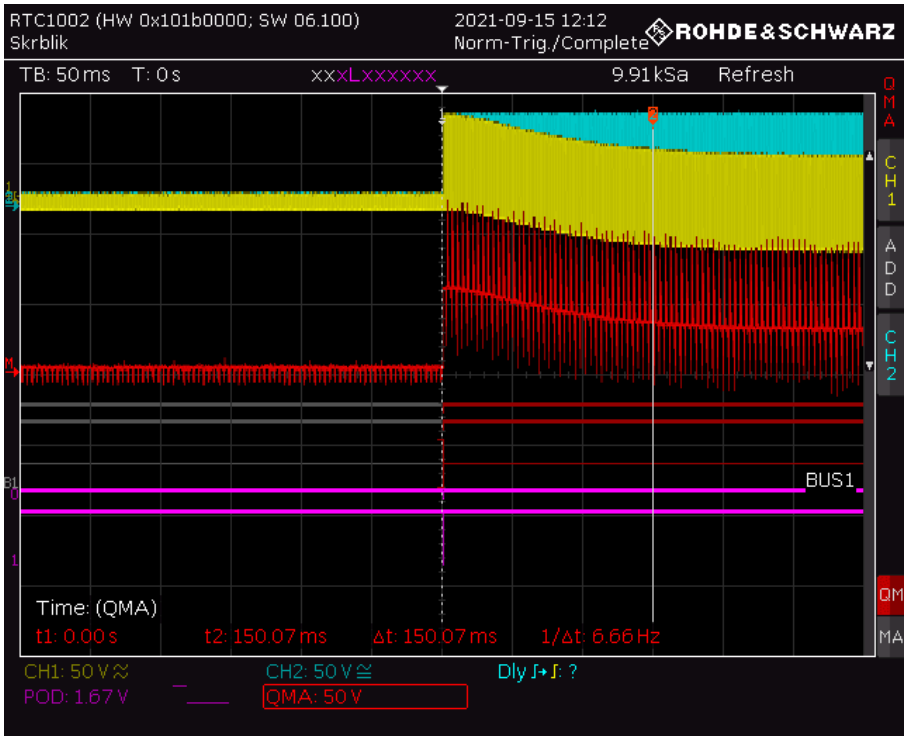

Figure 3.11: The oscilloscope output graph indicates that the voltage settles at the desired value after around 150ms.

After executing several tests, we have observed that the time needed to stabilize the output voltage varies depending on the difference between the current and the new focus value. The greater the difference, the longer the driver board takes to set the proper output voltage. The driver boards in the two mentioned camera systems use focus registers of different sizes, so to compare the two, we have performed the tests over several ranges of focus values. Fractions of the maximal focus value describe these ranges - e.g. 1/5 means that the difference between the new focus value and the current one is 1/5 of the maximal focus value(51 for 8-bit ADM00931 and 13107 for 16-bit Flexiboard).

Since the mechanical lens Arducam(todo) is controlled similarly, we have also performed the tests using a mechanical lens. The time taken was measured by an application, as the Arducam's API uses a blocking function to set the focus.

-	**1/25**	**1/5**	**2/5**	**1/2**	**3/5**	**4/5**	**1**
ADM00931	20	75	90	100	105	130	150
Flexiboard	15	80	95	105	115	125	140
Arducam PTZ	200	740	1440	1780	2115	2820	3485
Average difference	182,5	662.5	1347.5	1677.5	2005	2,692.5	3,340

Table 3.1: The table states the periods (in ms) needed to set the output voltage of a driver board over a fraction of the number of possible focus values. The times are stated in milliseconds. The average distance row calculates the difference between the mechanical camera system and the average time taken by both liquid lens camera systems.

We observe that the liquid lens camera systems are, on average, 17 times faster than the Arducam PTZ using a mechanical lens. The worst-case scenario is that the system refocuses from the closest to the furthest point. We have established that this operation takes 140ms-150ms to complete for the embedded and non-embedded systems using liquid lenses, respectively.

3.4.1 Possible optimizations

Measuring the settling time of the voltage gives insight into possible bottlenecks of the camera. Refocusing over a range in several steps is always slower than refocusing in a single step. An image-based auto-focusing algorithm needs to do just that, however. It needs to refocus and capture an image more than 20 times to complete in many cases. Thanks to the testing, we propose two ways to optimize the camera systems:

Dynamic waiting times

Testing proved that the more significant the difference between the current(initial) value in the demo board register and the new value written to the driver board, the longer the output amplitude takes to settle. The camera does not have to wait 150ms after refocusing over a smaller range of focus values. It only has to do so in the worst-case scenario. The period the camera needs to wait for a proper focus is usually lower than 150ms, and it depends on the difference in current and new focus value. Dynamic waiting times are therefore calculated using the initial and new values difference. The exact figures are found by observation via an oscilloscope.

Limiting the focus values range

As a result of limiting the focus values range, the differences between the initial value and the new value become smaller; therefore, it is not necessary to wait so long for the amplitude to settle. Combining this optimization with dynamic waiting times is possible to achieve even faster refocusing times.

Chapter 4

Image sharpness detection

This chapter summarizes the findings concerning image sharpness evaluation and discusses the methods used to find the score of image sharpness. Image sharpness refers to how clear or blurry an image is. Often edge detection algorithms are used to evaluate image sharpness. Usually, the more edges on an image, the sharper(more in focus) it is.

Image-based auto-focus methods analyze the image sharpness of several images. They usually capture the images using different focus values and save the sharpness scores of these images. After analyzing all images, the auto-focus algorithm sets the focus value at which the highest score was achieved.

Image sharpness detection is also used to sort blurry photos. Professional photographers often take large amounts of photos, some of which may be not in focus. They choose a minimally required sharpness score - a threshold and discard all photos the score of which is lower than the threshold. A threshold can also be used for an image-based auto-focus algorithm. When the algorithm finishes, it compares the maximal sharpness score found with a threshold, and if lower, it starts the algorithm over again.

Nowadays, there are various options to determine whether an image is clear. These methods differ in their accuracy under different conditions. One method may be more precise in determining whether an image is clear if the image is bright, while others may struggle to determine the sharpness score under such conditions. Another method may calculate the score precisely if there are many edges on the image, while others may take too long under such circumstances. The choice of the image sharpness evaluation method depends on the objects that the camera will most likely capture and the environment they are placed within.

The model situation for this project is that the camera will be focusing on a hand with proper lighting. We assume that the lighting conditions will be proper as we will use external lighting.

4.1 Ways of image sharpness score assessment

Selecting the proper method of image-sharpness detection is a crucial part of the task. The main objective is to find a precise method that takes as little time as possible to meet the near real-time requirements. We will use a camera triggered manually, achieving 30 to 40 frames per second in this mode. Various auto-focusing methods require different amounts of images to be taken to focus correctly. We will assume the algorithm requires taking and calculating the sharpness score of at least ten separate images for the early

stages. Ideally, a single auto-focusing cycle should finish under one-third of a second. Such a requirement is almost impossible to meet as no image-sharpness detection method can calculate the sharpness of 10 high-quality images in such a short time. That means the sharpness calculation will most likely limit the camera's framerate. Later on, we will discuss the methods to counteract this limitation.

4.2 Edge detection

An edge in an image is a boundary between two homogeneous areas. If only one image channel is considered, the areas are characterized by similar pixel intensities. Edge detection is an image-processing method of identifying sharp discontinuities in an image. It usually does so by calculating the differences in neighbouring pixel intensities.

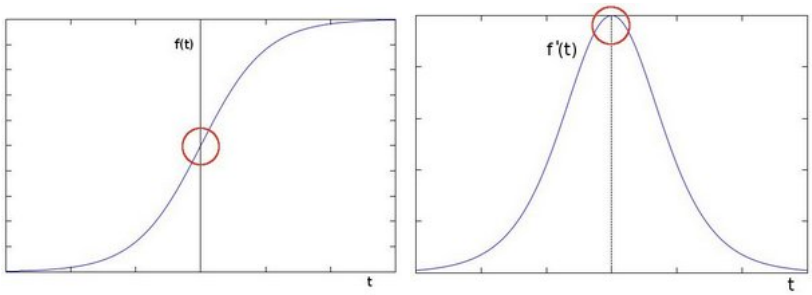

Figure 4.1: The image on the left is a graph representation of a one-dimensional image, also known as the intensity function. The image contains a single edge, as there is only one shift from low intensity to high intensity. The red circle is the point on a graph where the edge is detected. On the right, there is the first derivative of the intensity function of the image. Notice that the edge now appears as a local maximum of the function.[15]

Edge detection is a fundamental tool used in most image processing applications to obtain information from the frames as a precursor to feature extraction and object segmentation. Edge detection filters are also widely used to increase the appearance of blurred images. It is frequently used in camera vision applications [2].

Image gradient

An image gradient represents the directional change in the intensity of colour in an image. It is one of the fundamental parts of image processing. In mathematics, the gradient of a two-variable function(the image intensity function) at each image pixel is a 2D vector. The derivatives give the components of the vectors in both horizontal and vertical directions. The direction of the vector is equal to the direction of the most significant intensity increase, and the length represents the rate of intensity change. The mathematical denotation of an image gradient is

$$\nabla f = \begin{bmatrix} g_x \\ f_x \end{bmatrix} = \begin{bmatrix} \frac{\partial g}{\partial x} \\ \frac{\partial g}{\partial y} \end{bmatrix} \tag{4.1}$$

where

- $\frac{\partial f}{\partial x}$ is the derivative with respect to the x-axis(gradient in x direction)
- $\frac{\partial f}{\partial y}$ is the derivative with respect to the y-axis(gradient in y direction)

Finite differences in the pixel intensities can approximate the partial derivatives. They are calculated by the convolution of an image with a matrix. Different edge-detection methods use different convolution matrices. The fundamental one is:

$$\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

The convolution is then denoted as:

$$\frac{\partial f}{\partial y} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} * I \tag{4.2}$$

where

- I is the image matrix
- symbol * stands for 1-dimensional convolution operation

The output of such convolution is a gradient image, denoted as G. The pixels of the original image corresponding to pixels with a high gradient in the gradient image become possible edges. The edge direction is perpendicular to the direction of the gradient vector, which is calculated by the formula:

$$\alpha = \arctan \frac{g_y}{g_x} \tag{4.3}$$

The magnitude of the vector is given by

$$|\nabla f| = \sqrt{g_x^2 + g_y^2} \tag{4.4}$$

In terms of image gradient, an edge is defined as a discontinuity in an image gradient graph. Figure 4.2 shows 5 classifications of the these discontinuities.

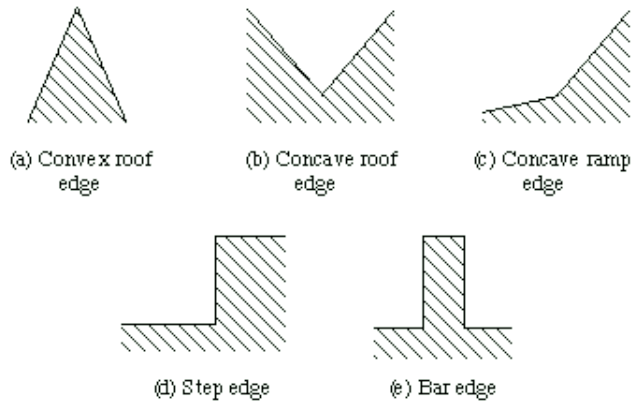

Figure 4.2: Types of discontinuities in image gradient graph. Image taken from Professor A. D. Marshall's website [**?**].

There are many edge detectors based on the convolution of the image with a mask, often using 3x3 maks sizes or even larger. A larger size mask reduces the errors caused by noise and by local averaging within the neighbouring pixels. The masks are usually of an odd size so that they can be centred and therefore provide an estimate that is biased towards a centre pixel.

4.2.1 Original image

In the following section, we will compare the outputs of different edge-detection methods. The photo that will be used is a high-resolution photo of a hand.

Figure 4.3: Original of the image that will be used to demonstrate edge-detection methods outputs.

4.2.2 Sobel Operator

A Sobel operator is a simple approximation of the image gradient concept. It uses different masks(convolution matrices) to detect edges in horizontal and vertical directions. One is obtained by rotating the other by 90 degrees. The input is convoluted with the mask to result in a Sobel approximation of the gradient image[15]. We only take one channel of an image into account - it represents image intensity. Convolution with integer values is not an expensive operation in terms of time, making it a relatively fast method. [17]

$$S_x = \begin{bmatrix} 21 & 0 & -1 \\ 2 & -0 & -2 \\ 1 & 0 & -1 \end{bmatrix} S_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{4.5}$$

Figure 4.4: Sobel 3x3 kernels

The following steps are taken to obtain a Sobel approximation of gradient image in both directions:

1. The original image is convoluted with the S_x mask to obtain an image of horizontal edges

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & -0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * I \tag{4.6}$$

2. The original image is convoluted with the S_y mask to obtain an image of vertical edges

$$G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * I \tag{4.7}$$

3. The final gradient image is obtained by combining the two partial gradient images:

$$G = \sqrt{G_x^2 + G_y^2} \tag{4.8}$$

or sometimes in a more straightforward manner:

$$G = |G_x| + |G_y| \tag{4.9}$$

The mask matrices can be larger; the only condition is that it has to be an odd number. Since the Sobel operator produces only an approximation of the gradient image, noticeable inaccuracies may occur using small masks such as the mentioned 3x3 metrices4.4. The problem is usually solved using a Scharr operator instead [10].

The outputs of the Sobel operator using a 3x3 matrix.

Figure 4.5: Original image, taken from [15]

Figure 4.6: Sobel derivative, image taken from [15]

4.2.3 Scharr operator

This operator, just like Sobel, calculates the approximate gradient image using matrix convolution. It uses the following matrices:

$$Sch_x = \begin{bmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{bmatrix} Sch_y = \begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{bmatrix} \tag{4.10}$$

Scharr operator is as fast as the original Sobel operator, achieving more accurate results.

4.2.4 LaPlacian Operator

The previous operators approximated the first derivatives of the image intensity function. The Laplacian operator instead approximates the second derivatives of the image intensity function to detect the edges.

$$LaPlacian(f) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{4.11}$$

The second partial derivatives of the image intensity function are noted as [8]:

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y) \tag{4.12}$$

$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y) \tag{4.13}$$

An edge in the second derivative function is represented by the value 0. The operator follows the same steps as the Sobel operator, calculating the second derivatives in both directions and combining them into a single gradient approximation image.

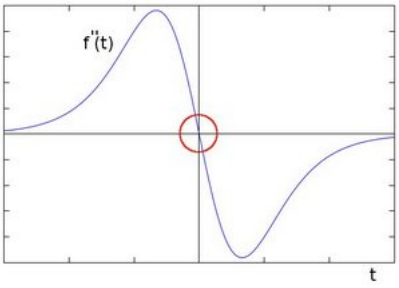

Figure 4.7: The graph of the second derivatives of a one-dimensional image containing a single edge[13]. Notice that the edge is now located at 0. This property is used to detect the edges.

The Laplacian operator approximates the values of the second derivative of the image intensity by convolution. Again, the matrices must be of an odd size; the smallest is a 3x3 kernel.

$$LP_x = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} LP_y = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{4.14}$$

Figure 4.8: The Sobel 3x3 mask matrices used by LaPlacian operator

Taking images derivatives accentuates its high frequencies, which leads to noise amplification since the proportion of noise to signal is more prominent at higher frequencies. Therefore, it is a common practice to use a low-pass filter prior to computing the image gradient to smooth the image. The only circularly symmetric filter - the Gaussian smoothing filter is used to make the response of the edge detector independent of the orientation of the edges [18].

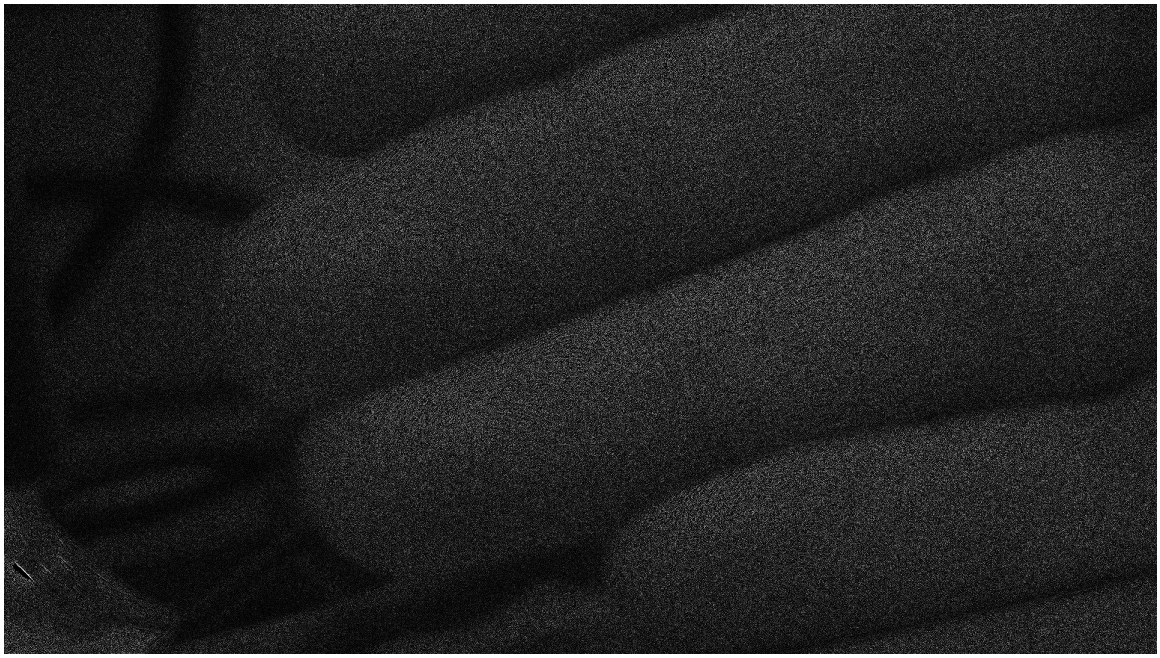

Figure 4.9: The output of a LaPlacian edge detector, the result shows the original image convoluted with a LaPlacian 3x3 kernel.

4.2.5 Canny edge detector

Canny edge detector uses multiple steps to detect various edges in an image. It was developed by John F. Canny in 1986, who set three main criteria for the detection:

- Accurate detection with a low error rate, meaning as many edges as possible need to be detected.
- If an edge is wider than one pixel, the detected edge pixel will be placed in the middle of the edge.
- An edge in the image should only be marked once, and where possible, image noise should not create false edges.

The Canny operator consists of 5 separate steps:

1. Applying the Gaussian smoothing in order to reduce noise
2. Convolution of the smoothed image with Sobel filters - finding edges in horizontal and vertical directions.

$$C_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} C_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{4.15}$$

Figure 4.10: The Sobel 3x3 mask matrices used by Canny edge detector.

Two additional matrices are used to find the edges in diagonal directions. All of the gradient images are then combined into one.

3. Thresholding or lower bound cut-off suppression is applied to the output of the previous step to get rid of spurious response to edge detection
4. Apply double threshold to determine potential edges.
5. Track edge by hysteresis: Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges.

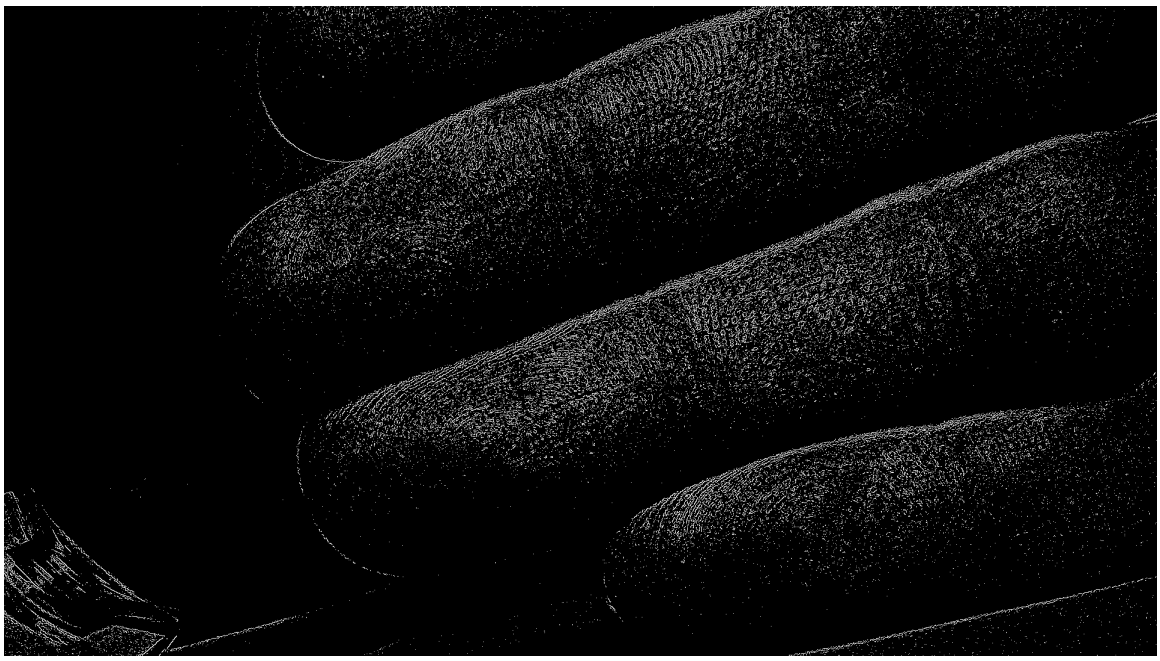

Figure 4.11: Canny derivative, image taken from [12]

4.2.6 Fourier Transform

An alternative to edge detectors that calculate derivatives of intensity image is the Fast Fourier Transform method. It is based on calculating the image data's Fast Fourier Transform(FFT). It first computes the FFT to find the frequency transform and then shifts the zero-frequency component (i.e., the DC component located at the top-left corner) to the centre, which will be easier to analyze. Then it zeros out the centre of the FFT shift (i.e., remove low frequencies), applies the inverse shift such that the DC component once again becomes the top-left, and applies the inverse FFT. After that, the magnitude spectrum of the reconstructed image and its mean value are calculated. This value is the score of image sharpness, meaning the sharper the image, the greater the value and vice versa.[16]

Minor manual tuning is required compared to LaPlacian methods. It is usually precise if the inputted images contain many objects with sharp edges - text is a good example. The individual letters are usually distinctly differentiated from the background, and every letter has very sharp edges. This method might thus struggle if the input images visualize a hand(model situation).

4.3 Comparing the methods

We have analyzed the performance of the individual image sharpness detection methods. A program was used to analyze the sharpness of 255 images, each taken at a different focus value. The images were taken by the embedded camera system 3.1, which we will describe later. The resolution of all the images is 2048x1536 px. We have used the same program to measure the times needed and only switched the image-sharpness detection methods. The scene of the image was a hand such as the one used as an example for image sharpness detection methods outputs 4.11.

-	**Lenovo IdeaPad 5 15ARE05**	**Raspberry Pi 4B**
Sobel operator	2,48	34,14
Laplace operator	2,70	38,16
Canny edge detector	2,36	21,66
FFT method	35,54	342,51

Table 4.1: Table of average times needed to analyze 255 high-resolution images by two different processing units. The averages were calculated from 10 runs. The times are stated in seconds. The differences in times between Sobel, Laplace and Canny are not really significant in case of using Lenovo IdeaPad as MCU. When Raspberry Pi 4B was used as MCU, the Canny ended up being the fastest. The FFT method proved to be quite slow for large images.

Graph of sharpness scores of images over the focus range

Image sharpness detection methods are often the basis of image-based auto-focus algorithms. These algorithms must find the maximum in the graph of image sharpness scores. A graph of sharpness scores over a range of all possible focus values 2.4 of the liquid lens helps us determine how hard it will be to find its maximum. The focus values are plotted on the x-axis, and the sharpness scores are plotted on the y-axis.

The following graphs were constructed by the embedded camera system 3.1 :

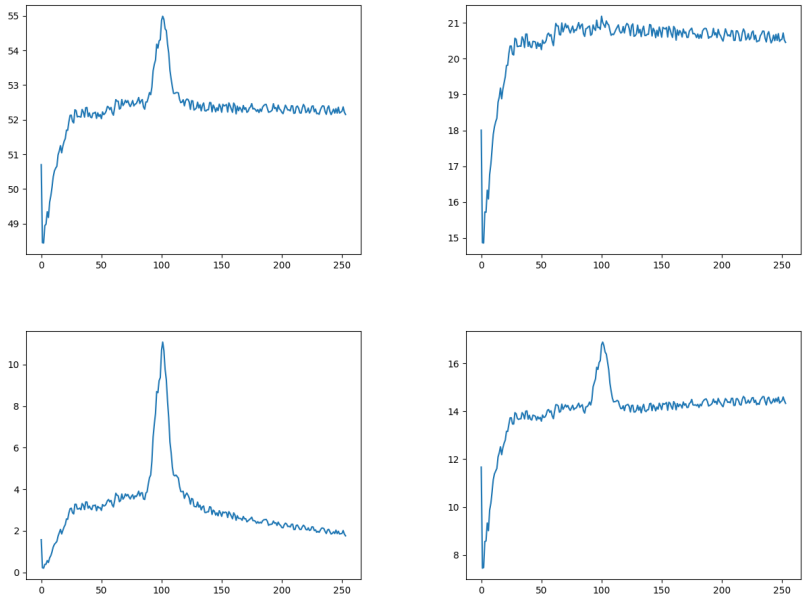

Figure 4.12: Top-left: Graph of image sharpness scores for Sobel operator.
Top-right: Graph of image sharpness scores for Laplace operator.
Bottom-left: Graph of image sharpness scores for Canny edge detector.
Bottom-right: Graph of image sharpness scores for FFT edge detection method.

To find the maximum in graphs like these by sweeping the range of focus values to focus the camera system, it is essential for the peak at which the maximum is located to be as wide as possible. We observe that the maximum in the Laplace operator graph is insignificant compared to the rest of the graph. The graphs for the Sobel operator and the FFT method look similar; however, the FFT method was significantly slower. Thus we will primarily be concerned with the Sobel operator and Canny edge detector.

Chapter 5

Implementation of an image-based auto-focus

An image-based auto-focus algorithm sets the focus values, captures and analyzes images and finds the focus value at which the image is the clearest. The image-sharpness detection method needs to be independent of the auto-focusing algorithm, meaning it can be switched, and the algorithm will still work.

This chapter will explain implementing an auto-focus program for both mentioned camera systems - the embedded and non-embedded liquid lens cameras. The main difference in implementation between these two systems is the different controls of the liquid lens. Different drivers control the two liquid lenses used. The embedded liquid lens camera driver is controlled via I2C, while the non-embedded camera driver is controlled via USB.

Another difference is that the embedded camera will utilize the manual trigger. In contrast to the non-embedded system, it will not be used to display the process of auto-focus. It means that the MCU does not have to obtain the frames taken by the camera while refocusing; therefore, it will only obtain the frames that the image-sharpness detection methods will analyze. A manual trigger (3.1.1) is used to make sure that the camera does not capture the frames while the lens is refocusing.

5.1 Language choice and libraries

The application will be implemented in the C++ programming language. It is low level enough to meet the real-time requirements. There is also the possibility of using a well-known OpenCV image-processing library. The first tests done in Python proved to be slower than those in C++. The unpredictability of memory management operations is also in contrast to the real-time needs.

In our implementation, we have used several open-source libraries. The libraries provide easy access to the I2C and GPIO busses and image processing. The library common for both embedded and non-embedded camera systems is `OpenCV`. `OpenCV` is a library for image processing. It was designed to be applicable in real-time applications [14].

5.2 Algorithm

A prototype is a slow, not optimized, very general usage application. It will undergo many optimizations and structural changes before becoming the final product. It is supposed

to be used for various testing cases, so keeping it as general as possible is good. In this case, it means not downsizing the focus values range and not using the other mentioned optimizations. The answer to these requirements is a sweep auto-focus algorithm. Its name is derived from sweeping the range of focus values and finding the value at which the sharpness score is the greatest. A must-do add-on for the final version is another similar loop, which finds the greatest sharpness score from images obtained from around the greatest found value from the first loop.

The algorithm for the embedded camera system assumes the mentioned components 3.1 (8-bit driver), making the highest writable value 255. It increments the value written to the driver board by 25; thus, it takes ten images to sweep through the entire focal length range.

The non-embedded camera system uses a 16-bit driver; thus, the increment is also greater. Considering the Sobel and Canny image sharpness scores graphs 4.12, an increment of 2621 was decided.

Algorithm 1 Sweep auto-focus algorithm pseudo-code for embedded camera system

```
increment = 25
score, maxScore, focusValue, maxFocusValue = 0
for i = 0; i <= 10; i++ do
    sendTrigger()
    image = loadImage()
    score = scoreCalculationMethod(image)
    if score > maxScore then
        maxScore = score
        maxFocusValue = focusValue
    end if
    focusValue += increment
    i2c_write(i2c_data)
end for
```

Algorithm 2 Sweep auto-focus algorithm pseudo-code for non-embedded camera system

```
increment = 2621
score, maxScore, focusValue, maxFocusValue = 0
for i = 0; i <= 10; i++ do
    image = loadImage()
    score = scoreCalculationMethod(image)
    if score > maxScore then
        maxScore = score
        maxFocusValue = focusValue
    end if
    focusValue += increment
    Flexiboard.setValue(focusValue)
    showImage(image)
end for
```

The code for image-based auto-focus implemented for the camera system using electromechnical lens was taken from https://github.com/ArduCAM/PTZ-Camera-Controller.

It was edited so that it matched the operations undertaken by the liquid lens camera systems.

5.2.1 What do the measurements of liquid lens speed mean for the algorithm

To implement the auto-focusing applications, we will use the knowledge acquired from the measurements in section 3.4. These measurements helped us to come up with the following actualities:

- The application must process the frames that are correctly focused according to the focus value; therefore, it needs to wait for the focusing to finish. Only then it should capture the next frame for processing. To achieve such behaviour, we will discard the frames that are not focused according to the focus value or display them as live capture using a separate thread. The latter approach only applies to the non-embedded system.
- Assume using the maximal possible framerate of the camera with the maximal resolution(119 fps for 2154x1536px). Then in the worst-case scenario(waiting 150ms), the embedded-camera application will acquire 18 images that are not focused as desired. The figure is similar for the non-embedded system, as its maximal framerate is similar.
- The settling times are not distributed proportionately to the fractions of focus range over which is being refocused. It means that refocusing over a range of focus values in any amount of steps will take longer than refocusing in one step. Refocusing from the closest point to the furthest point in 5 steps would take the embedded system 375ms of delay, while it would only take 150 if done in a single step.

5.2.2 Testing of embedded system

Prototype testing revealed the following actualities:

- The prototype only works properly if the application sleeps for a few milliseconds after every `i2c_write()`. Otherwise, images obtained will not be adequately focused according to focus values.
- Incrementing the value to be written to the driver by 25 often results in an inability to find a sharp image (the peak in the graph of sharpness scores is steep); therefore, we will use a smaller increment.
- Second loop for minor adjusting of the focus is needed. After finding the focus value at which the maximal sharpness score was obtained, it will also calculate the sharpness scores of images obtained at focus values around the one that is set. The increment of the focus value for the second loop will be much smaller, such as 1 or 2. It will ensure that the final image is the most in-focus one.
- The sharpness score calculation takes a significant amount of time - area of interest optimization is needed. It will select only a part of the image to be analyzed by the image sharpness detection method.

- Loading the image from the camera takes a significant amount of time. It can be solved by using lower-resolution images or a different image format; the optimal solution would be loading only the area of interest from the previous point and using separate threads to load the image and calculate the sharpness score.

5.2.3 Testing of non-embedded system

Testing of the prototype for the non-embedded system revealed these findings:

- The increment used is suitable; the algorithm performs auto-focus successfully if the scene is a hand at any distance it can focus on.
- It is needed to optimize the live projection of images using a separate thread. Otherwise, only 12 frames per second can be obtained. It is because the sharpness score calculation takes a significant amount of time. If a separate thread was used, the system could project all the images obtained while only analyzing the sharpness score of the correctly focused images according to the focus value.

5.2.4 Test suite for testing without camera

We have implemented a test suite for testing the auto-focus application without the need to use a camera. First, the `TestSetsGenerator` application makes the camera system capture an image at every possible focus value and saves those images. The scene must remain constant during the capture. These images, named after the focus value at which they were captured, make up a test set. The application `AutoFocusNoCam` does not capture any images using a camera. Suppose the auto-focus application wants to refocus according to a specific focus value. Then, instead of setting the focus value to the driver of the lens and capturing a new image, the application loads an image from the data set, the name of which contains the mentioned focus value. Thus we can test the auto-focus algorithm itself, minimizing the time needed to load images(from the camera in case of the original auto-focus application) and eliminating the time that the original application has to wait for a proper refocus.

The `AutoFocusNoCam` application takes mere 200 milliseconds to finish using Lenovo IdeaPad 5 15ARE05 as an MCU and around 1120 milliseconds running on Raspberry Pi 4B. It proves that the most significant bottleneck of the auto-focus application is the image-loading process.

5.2.5 First optimizations and add-ons

Adding a second loop

Adding a second loop to the prototype code ensures an image with the greatest possible sharpness score is found by sweeping the values around the value found by the sweep algorithm. There are various approaches to implementing this loop. It comes with a cost of more frames needed for the auto-focus loop; however, it is essential for wide ranges of focus values.

Adding the second loop proved to be essential for both systems. Otherwise, the algorithm would only approximate the focus value at which the most in-focus image is obtained. For the embedded system, an increment of 2 was used. For the non-embedded system, we have used the increment of 256.

Area of interest optimization

Limiting the area used to calculate the image sharpness may not be optimal for all applications. It means that the algorithm only processes a part of the obtained image, meaning it will try to find the focus value at which the selected area is most in-focus. This approach may not be ideal for systems that need the whole image in focus. This optimization proved to speed up the algorithms significantly.

Working distance optimization

Camera systems, especially those used industrially, rarely use the lens's entire range of focus distances. An example is an imaging line, which photographs boxes of different sizes as they pass beneath the camera system. Such a system will only have a working distance equal to the height of the highest box that will pass through the line. We can thus optimize the range of focus values sent to the lens. The smaller the focus values range, the faster the refocusing will be, speeding up the system even more.

5.2.6 Comparing the image-based auto-focus performance across systems

We have optimized all systems using the second loop and area of interest optimizations. All the systems were made to capture and analyze the same amount of images. We observed the times taken by all three camera systems to finish the optimized auto-focusing algorithms.

The embedded camera system uses an 8-bit driver and Raspberry Pi as an MCU. The non-embedded system uses a 16-bit driver, and its MCU is a personal notebook Lenovo IdeaPad 5 15ARE05. The assumption is that it will be able to process images much faster than the embedded system as it does not use an ARM processor. The operating system is booted from an SSD drive instead of an SD card, which should make memory operations faster.

Lenovo IdeaPad 5 15ARE05 specifications:

- `Processor:` AMD Ryzen 7 4700U, 2.00 GHz, 8 cores
- `RAM:` 16GB
- `Memory:` 512GB SSD drive
- `USB:` 2x USB 3.0 port, USB-C port
- `Operating system:` Ubuntu 18.04

The fastest one was the non-embedded camera system, achieving an average of 2,06 seconds. The embedded system performed as expected for an ARM architecture processor system - it was a little slower, taking 3,48 seconds. The camera system with a mechanical lens finished the auto-focus algorithm after 16.73 seconds from its start.

The non-embedded system was 6.08 times faster than the system using a mechanical lens. The embedded system was 4,80 times faster than the system using a mechanical lens. These results prove that the speed of a liquid lens is unmatched by their mechanical alternatives.

5.2.7 Alternative approach

The performance of image-based auto-focus methods is often correlated with the quality of the input image, the image's scene, lighting and the number of edges on the picture. Consequently, it may perform poorly in an environment different from the one it was tested on. This phenomenon can be counteracted to a certain degree. It, however, often requires a significant amount of manual tuning or adding camera lighting.

Adding a distance sensor to the device would limit the number of refocusing cycles the lens must undergo dramatically. The MCU would keep a table of ranges of distances associated with focus values that make the lens focus on these distances. Following this approach, ideally, a single refocusing would be required to achieve the auto-focus. Then the performance comparison of the liquid lenses and the mechanical lens is summarized by 3.1. The liquid lenses were on average faster by 1,7 seconds than the mechanical lens in a test of 7 refocusing cycles across different focus distance ranges.

Chapter 6

Summary

The liquid lens technology promises focusing capabilities unmatched by their electromechanical alternative. This thesis has established that liquid lenses are, in fact, faster in both individual and repeated focusing over any distance. In the case of a single refocusing, the liquid lenses performed on average 17 times faster. Repeated focusing performance was tested by image-based auto-focus. The embedded system using a liquid lens performed the auto-focus loop in 3,48 seconds, and the non-embedded system using a liquid lens achieved an average of 2,06 seconds. That is, on average, 6,03 times faster than the camera system using a liquid lens.

In the case of the non-embedded system, the image-based auto-focus took approximately 57ms to load and process each frame, while the embedded system took 96ms to do so. Most of the time was spent loading images from the camera and calculating the sharpness score by the Sobel operator. It is unlikely that the bandwidth of USB3 interface limits the image-loading process, as it can handle up to 5Mb/ms and the images captured are usually of size lower than 5Mb. The limitation possibly lies in allocation of memory for the image and the conversion of raw image data into an OpenCV `Mat` datatype. These actions are not easy to optimize, so instead, we will focus on optimizing the time needed to calculate the sharpness score of an image.

It is possible to optimize the image-processing calculations in 2 ways - using a more powerful MCU, running the calculations on a GPU. The first option assumes that a more powerful MCU can perform the calculations faster. Running the calculations on a GPU offers the advantage of parallelization. It can be utilized by segmenting the image being processed into several parts. The GPU would then calculate the score of all the segments simultaneously. Finally, the segment scores would be simply summed up to obtain the final sharpness score of the image.

The extent to which these optimizations would improve the speed of the auto-focus loop is not clear. It is due to the need to wait for the driver to refocus the lens properly. The algorithm knows in advance, what focus value will be sent to the driver for the subsequent image capture, so it sends it to the driver before calculating the image sharpness score. This way, the driver's output voltage settles while the score is being calculated. If the order was reversed, the auto-focus loop would have to wait tens of milliseconds for the voltage to settle before capturing the following image. It means that if the sharpness score calculations were faster than the time needed to settle the voltage at the driver's output, the algorithm would have to wait in order to take a picture adequately focused according to the focus value.

A possibly more valuable optimization would be the one that adds a distance sensor. This way, the auto-focusing could theoretically bring an object to focus by a single refocus. However, the algorithm would be limited by the speed and accuracy of the distance sensor. A possibly low accuracy could be compensated by using a greater depth of field.

Bibliography

[1] *Hardware Connection of the Pan Tilt Zoom Raspberry Pi Camera* [online]. Arducam, may 2021 [cit. 2022-1-2]. Available at: https://www.arducam.com/docs/cameras-for-raspberry-pi/ptz-camera/hardware/.

[2] Amer, G. M. H. and Abushaala, A. M. Edge detection methods. In: *2015 2nd World Symposium on Web Applications and Networking (WSWAN)*. 2015, p. 1–7. DOI: 10.1109/WSWAN.2015.7210349.

[3] Banks, M., Cooper, E. and Piazza, E. Camera Focal Length and the Perception of Pictures. may 2014, vol. 26, p. 30–46. DOI: 10.1080/10407413.2014.877284.

[4] Choudhury, A. K. R. *Principles of Colour and Appearance Measurement*. Woodhead Publishing, 2014 [cit. 2021-3-2]. 1-52 p. ISBN 978-0857092298. Available at: https://www.sciencedirect.com/science/article/pii/B9780857092298500012.

[5] Corbet, J., Rubini, A. and Kroah Hartman, G. *Linux Device Drivers*. 3rd ed. O'Reilly Media, Inc., february 2005. ISBN 978-0596005900.

[6] Deshayes, Y. and Béchou, L. 1 - State-of-the-Art of Infrared Technology. In: Deshayes, Y. and Béchou, L., ed. *Reliability, Robustness and Failure Mechanisms of LED Devices*. Elsevier, 2016, p. 1–44. DOI: https://doi.org/10.1016/B978-1-78548-152-9.50001-8. ISBN 978-1785481529. Available at: https://www.sciencedirect.com/science/article/pii/B9781785481529500018.

[7] *Liquid Lens Features, Applications, and Technology* [online]. [cit. 2021-1-2]. Available at: https://www.edmundoptics.eu/knowledge-center/application-notes/imaging/liquid-lenses-in-imaging/.

[8] Gonzalez, R. C. and Woods, R. E. *Digital Image Processing*. 2nd ed. Tom Robbins, 2001. ISBN 978-0201180756.

[9] inc., S. *HV892DB1 - Inductorless Liquid Lens Driver Demoboard Datasheet*. https://cz.mouser.com/datasheet/2/268/hv892db1-1022670.pdf.

[10] Kaehler, A. and Bradski, G. *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library*. 1st ed. O'Reilly Media, Inc., 2017. ISBN 978-1491937969.

[11] Knight, W. *Liquid lens promises cheap gadget optics* [online]. March 2004 [cit. 2022-1-2]. Available at: https://www.newscientist.com/article/dn4751-liquid-lens-promises-cheap-gadget-optics/.

[12] OpenCV. *Canny Edge Detection* [online]. January 2022 [cit. 2022-1-2]. Available at: https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html.

[13] OpenCV. *Laplace Operator* [online]. January 2022 [cit. 2022-1-2]. Available at: https://docs.opencv.org/4.x/d5/db5/tutorial_laplace_operator.html.

[14] OpenCV. *OpenCV Documentation* [online]. January 2022 [cit. 2022-1-2]. Available at: https://docs.opencv.org/4.x/index.html.

[15] OpenCV. *OpenCV: Sobel Derivatives* [online]. January 2022 [cit. 2022-1-2]. Available at: https://docs.opencv.org/4.x/d2/d2c/tutorial_sobel_derivatives.html.

[16] Rosebrock, A. *OpenCV Fast Fourier Transform (FFT) for blur detection in images and video streams* [online]. June 2020 [cit. 2022-1-2]. Available at: https://www.pyimagesearch.com/2020/06/15/opencv-fast-fourier-transform-fft-for-blur-detection-in-images-and-video-streams/.

[17] Rosebrock, A. *Image Gradients with OpenCV (Sobel and Scharr)* [online]. May 2021 [cit. 2022-1-2]. Available at: https://www.pyimagesearch.com/2021/05/12/image-gradients-with-opencv-sobel-and-scharr/.

[18] Szeliski, R. *Computer Vision: Algorithms and Applications*. Springer, 2011. ISBN 978-1848829343.

[19] Wiki, S. *Liquid lens - Flüssiglinse* [online]. November 2021 [cit. 2022-1-2]. Available at: https://second.wiki/wiki/flc3bcssiglinse.

[20] Wikipedia. *Attenuation* [online]. [cit. 2021-3-2]. Available at: https://en.wikipedia.org/wiki/Attenuation.

[21] Wikipedia. *Image sensor* [online]. [cit. 2021-3-2]. Available at: https://en.wikipedia.org/wiki/Image_sensor.

Appendix A

Contents of the Included Storage Media

- `AutoFocusEmbedded/src` - source files for embedded camera system image-based autofocus application
- `AutoFocusEmbedded/include` - header files for embedded camera system image-based autofocus application
- `AutoFocusEmbedded/README.md` - readme for embedded camera system image-based autofocus application
- `AutoFocusNonEmbedded/src` - source files for non-embedded camera system image-based autofocus application
- `AutoFocusNonEmbedded/include` - header files for non-embedded camera system image-based autofocus application
- `AutoFocusNonEmbedded/README.md` - readme file for non-embedded camera system image-based autofocus application
- `TestSuite/src` - source files for test suite
- `TestSuite/include` - header files for test suite
- `TestSuite/README.md` - readme file for test suite