TAMPERE UNIVERSITY OF TECHNOLOGY

HEIKKI JANTUNEN

SOFTWARE IMPLEMENTATION OF CONTRAST-BASED AUTOFOCUS IN MOBILE CAMERA SYSTEM

Master's thesis

Examiners:
Professor Hannu-Matti Järvinen
Markus Vartiainen
Examiner and topic approved by
the Faculty Council of Computing and
Electrical Engineering on 05.03.2014

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Information Technology
JANTUNEN, HEIKKI: Software Implementation of Contrast-Based Autofocus in Mobile Camera System
Master of Science Thesis, 55 pages
April 2014
Major: Software Engineering
Examiners: Professor Hannu-Matti Järvinen, Markus Vartiainen
Keywords: Autofocus, Contrast, NEON, Optimization, Digital camera, Low-light autofocusing, Mobile camera

Autofocus is an important part of a modern digital camera system. Lens of the camera redirects light onto the surface of the imaging sensor. The distance between the lens and the sensor is in direct relation to the distance at which the scene appears sharp in the captured image. The purpose of focusing is to move the camera lens so that the region of interest in the image is sharp. Autofocus aims to do this automatically, without user interaction. Contrast-based autofocus algorithm works as a part of the image processing pipeline and uses metrics provided by the image signal processor (ISP) to analyse the sharpness and moves the based on this analysis.

In this thesis, the target was to create an autofocus system that works independent of the ISP and calculates the metrics on the CPU of the target device, Nokia Lumia 1520. The benefit of a pure software implementation is that it will remove the need for the ISP hardware for autofocus and adds flexibility to the metrics calculation process because configurability is not limited by the particular hardware implementation of the ISP. By preprocessing the image data before metrics calculation, it is possible to enhance the low-light performance of the system. However, the challenge of replacing a dedicated piece of hardware with software processing lies in creating an implementation that is efficient enough to be practical.

An autofocus framework was implemented. It provides a background processing system for calculating the metrics and possible preprocessing. Threading is utilised as means of optimization so that the image is processed in parts. The metrics are processed during the exposure of the next frame, which leads to latency in the availability of the metrics. To take this into account, also a simple sweep-based single pass autofocus algorithm was implemented.

For calculating the metrics, three focus operators and three preprocessing methods were implemented and evaluated. The techniques varied in the heaviness of calculation and they were optimized using NEON which is a single instruction multiple data (SIMD) extension of ARM instruction set architecture. MATLAB simulation was used to evaluate the output of the implemented methods.

While all of the focus operators produced very similar results, using median filter for preprocessing provided a significant improvement for low-light focusing. The autofocus system was also run on the target device with combinations of the implemented metrics processing techniques. Processing times were measured and the framework was proved to be applicable with any combination of the techniques.

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Tietotekniikan koulutusohjelma
JANTUNEN, HEIKKI: Kontrastipohjaisen automaattitarkennuksen ohjelmistototeutus mobiilikamerajärjestelmässä
Diplomityö, 55 sivua
Huhtikuu 2014
Pääaine: Ohjelmistotuotanto
Tarkastajat: Professori Hannu-Matti Järvinen, Markus Vartiainen
Avainsanat: Automaattitarkennus, kontrasti, NEON, optimointi, digitaalikamera, automaattitarkennus hämärässä, mobiilikamera

Automaattitarkennus on tärkeä ominaisuus modernissa digitaalisessa kamerassa. Kameran linssi ohjaa valonsäteitä kuvasensorin pinnalle. Linssin ja sensorin välinen matka on suoraan suhteessa etäisyyteen, jolla näkymä muodostuu terävänä tallennettuun kuvaan. Tarkentamisen tarkoitus on liikuttaa kameran linssiä siten, että haluttu alue kuvasta on terävä. Automaattitarkennuksen tehtävä on suorittaa tarkentaminen automaattisesti, ilman käyttäjän apua. Kontrastipohjainen automaattitarkennusalgoritmi on osa kuvaprosessointiputkea, jossa se hyödyntää kuvasignaaliprosessorin (Image Signal Processor, ISP) laskemia metriikoita. Niistä algoritmi arvioi kuvan terävyyttä ja liikuttaa linssiä tehdyn analyysin perusteella.

Tämän työn tarkoitus oli luoda automaattitarkennusjärjestelmä, joka toimii ilman ISP:tä, laskien vastaavat metriikat kohdelaitteen, Nokia Lumia 1520, keskusprosessorilla. Täysin ohjelmistopohjaisen toteutuksen ansiosta ISP voidaan automaattitarkennuksen puolesta jättää kokonaan pois laitteistokokoonpanosta. Metriikoiden laskemisesta tulee joustavampaa, kun ISP:n säädettävyys ei rajoita sitä. Lisäksi esiprosessoimalla kuvadataa ennen metriikoiden laskemista on mahdollista parantaa tarkennustarkkuutta hämärässä. Haasteena laitteiston korvaamisessa ohjelmistolla oli se, että toteutuksen on oltava riittävän tehokas ollakseen käytännöllinen.

Työssä toteutettiin automaattitarkennuskehys. Se sisältää taustaprosessoinnin, jota voidaan käyttää metriikoiden laskemiseen sekä mahdolliseen esiprosessointiin. Kuva on mahdollista prosessoida osissa, joten taustaprosessoinnin optimointikeinona käytettiin säikeistämistä. Automaattitarkennuksen metriikat lasketaan seuraavan kuvan valotuksen aikana, joka aiheuttaa viiveen niiden saatavuuteen. Siksi kehykseen toteutettiin myös yksinkertainen pyyhkäisyyn perustuva tarkennusalgoritmi, joka osaa ottaa viiveen huomioon.

Metriikoiden laskentaa varten toteutettiin kolme tarkennusoperaattoria sekä kolme esiprosessointimenetelmää. Tekniikoiden laskennallinen vaativuus oli vaihteleva, ja ne optimoitiin käyttäen ARM-käskykannan laajennusta, NEON, joka suorittaa saman laskuoperaation rinnakkaisesti usealle muuttujalle. MATLAB-simulaation avulla arvioitiin eri tekniikoiden ulostuloja.

Tarkennusoperaattoreiden välillä ei havaittu merkittäviä eroja, mutta esiprosessointitekniikoista mediaanisuodin osoittautui tehokkaaksi menetelmäksi silmälläpitäen automaattitarkennuksen tarkkuutta hämärässä. Toteutettua automaattitarkennusjärjestelmää ajettiin myös kohdelaitteessa erilaisilla prosessointimenetelmäyhdistelmillä. Suoritusajat mitattiin, ja jokainen yhdistelmä osoittautui käyttökelpoiseksi.

PREFACE

This Master of Science thesis, Software Implementation of Contrast-Based Autofocus in Mobile Camera System, was carried out while working in Algorithm and Middleware team of Nokia Smart Devices, Imaging organization in Tampere, Finland.

I would like to thank all my colleagues in the Imaging team and especially in Algorithm and Middleware team, for creating an excellent and enjoyable working environment. Thanks to Hannu-Matti Järvinen who was the examiner of the work. Special thanks to Markus Vartiainen who was my supervisor on behalf of Nokia providing guidance when needed.

I would also like to thank my family and girlfriend for all the support and words of encouragement, which helped me to carry out this whole process.

Tampere, April 28th, 2014

Heikki Jantunen

CONTENTS

TERMS AND ABBREVIATIONS

A/D	Analogue to Digital conversion
AdobeRGB	A colour space developed by Adobe Systems Inc.
AEC	Auto Exposure Control
AF	Autofocus
ARM	A family of instruction set architectures
AWB	Auto White Balance
BSI	Backside Illumination
CCD	Charge-Coupled Device
CFA	Colour Filter Array
CMOS	Complementary Metal-Oxide Semiconductor
CPU	Central Processing Unit
DLP	Data-Level Parallelism
DOF	Depth Of Field
DSC	Digital Still Camera
DSLR	Digital Single-Lens Reflect
DSP	Digital Signal Processor
FF	Fixed Focus
FOV	Field Of View
FPN	Fixed-Pattern Noise
fps	Frames per second
FSI	Frontside Illumination
HD	High Definition
IPP	Image Processing Pipeline
IR	Infra Red

ISP	Image Signal Processor
NEON	SIMD extension of ARM instruction sets
NV12	YCbCr format where Y plane is first in buffer. Cb and Cr are interleaved after Y plane.
QE	Quantum Efficiency
ROI	Region Of Interest
SIMD	Single Instruction Multiple Data
SNR	Signal to Noise Ratio
sRGB	Standard RGB, commonly used colour space in digital imaging
VCM	Voice Coil Motor
YCbCr	Colour space where Y is the luma component, Cb is the blue-difference component and Cr is the red-difference component

1. INTRODUCTION

Popularity of digital cameras has been increasing for several years. This is true especially in mobile phones where it may be difficult to find one without embedded camera, even in low-end. In high-end smart phones have already somewhat replaced compact cameras and are beginning to challenge more expensive cameras in image quality. An advanced digital camera system is always a complex combination of hardware and software, with the target of capturing the scene as we see it.

One very important piece of the puzzle on the lights journey to a digital image is directing the light beams to the imaging sensor. Only a small portion of the scene depth will appear sharp in the produced image. The distance is adjusted by tuning the gap between the lens and the imaging sensor. The system performing this adjustment automatically in a digital camera is called autofocus (AF). A typical technique used is contrast-based consisting of hardware capable of moving the lens and software making the movement decisions. In smart phones there is usually a dedicated image signal processor (ISP) that calculates metrics information for the autofocus algorithm according to which the decisions are made.

In this thesis, an attempt is made to create autofocus system where calculation of the needed metrics is performed on the general purpose CPU of the Nokia Lumia 1520 smart phone. In success this would remove the need for an ISP in the device for autofocus part. Also going around the restrictions in configuration of the ISP would add flexibility to the autofocus system and open exciting possibilities for challenging shooting conditions such as low-light scenes. The challenge lies in finding a software-based solution that produces high quality metrics, but is efficient enough to be practical at the same time.

To achieve the goal, an autofocus framework is created. It provides a flexible environment for metrics processing to be performed in the background to allocate enough time for the calculations to complete. The framework also implements a simple focusing algorithm that takes the shortcomings of the software calculation process into account. A few optimized processing and preprocessing methods were implemented and tried out in the created framework to evaluate its performance.

This thesis consist of two theory chapters, one implementing chapter, one implementing and analysing chapter and conclusion. Chapter 2 explains the basics of digital camera systems to provide background information about the environment,

a part of which the autofocus system is. The typical pieces of hardware and image processing techniques used in digital cameras are explained. Chapter 3 introduces the reader to the details of contrast-based autofocus, which is a method that has various implementation approaches. Also an overview of other autofocus techniques is presented. The implemented autofocus framework is described in Chapter 4. A detailed explanation of the implementation methods and their reasoning is explained for each part of the system. In Chapter 5 the implemented preprocessing and processing methods are introduced. The quality of the metrics obtained is analysed by capturing authentic input data and using MathWorks MATLAB to simulate the processing. This chapter also presents the processing results of the autofocus system run with the implemented metrics processing. Chapter 6 consist of the conclusions and evaluation of the work carried out.

2. INTRODUCTION TO DIGITAL CAMERA SYSTEMS

It is important for the reader to understand the domain in which the topic of this thesis belongs to. This chapter is meant for an introductory of digital camera systems, the components they consist of and how they operate. First, the most important components are presented and explained how the data from the scene is obtained. Second part focuses on processing of that data.

2.1 Overview

A digital camera system is a combination of hardware and software components. The purpose of which is to capture an image of a scene by converting the information into signals that can be stored and afterwards reproduced on a display device. Image can be defined to be "variation of light intensity or rate of reflection as a function of position on a plane" [1, p.2].

Early concepts of digital still cameras (DSCs) date back to 1970s [1, p.3]. Nowadays there are multiple digital camera types which differ in features and applied technologies. Compact cameras are relatively small, portable and intended for casual photo shooting. Digital single-lens reflex (DSLR) cameras are an example of a type for more serious photographers. They are larger in size, but provide higher quality images and also versatility trough interchangeable lenses. Mobile phone cameras are the most interesting type in the scope of this thesis and even though many digital cameras are capable of video recording, the focus is set to still imaging.

2.2 Hardware

Typical components of a digital still camera (DSC) are illustrated in Figure 2.1. Many of the depicted components contain multiple parts. First, object needs to be illuminated. If there is not enough light in the scene, a flash may be used. Object will then reflect some of the light towards the camera system. Beams will pass trough the optics system and form an image of the object on the image sensor. Energy of light beams then gets converted into electrical signals and sensor outputs raw data onwards. The processor utilizes information about the capturing device and generates image from the raw data. Reproduced image is finally shown on the

display and stored. [2]

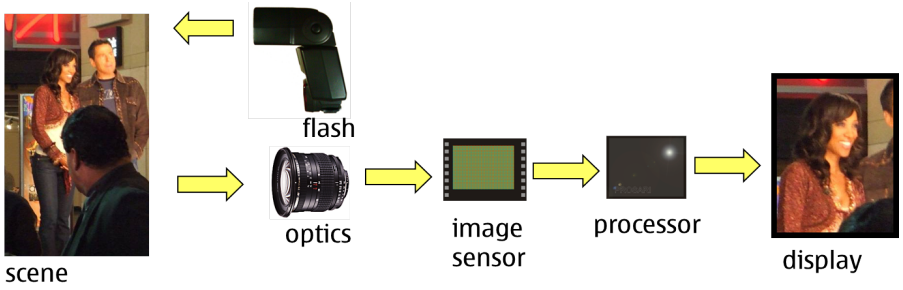

Figure 2.1: Simplified schematic of typical digital still camera components [2].

The processing component usually consists of digital signal processors (DSPs) which in case of a camera device are often referred as *image signal processors* (ISPs). It is also possible to use a microprocessor or combination of both for the task. Information about the image data is gathered in this phase to be used in *automatic exposure control* (AEC), *automatic white balance* (AWB) and *autofocus* (AF), see Section 2.3.3, Section 2.3.2 and Chapter 3. System control is responsible for reading these metrics sequentially and adjusting the system so that the actual capture has the best possible outcome. [1, p.17] The display component is not strictly for showing the final outcome, but a preview is shown on it before capture so it is used as *viewfinder*.

2.2.1 Optics

Optics is one of the most important parts in DSC. Its function is to control light rays entering the imaging system by using the refraction and reflection properties of light. When talking about camera optics, the whole system is often referenced simply as "lens". However, optics consists of multiple lens elements, excluding some very simple devices. Target is to create as good image as possible to the image sensor located behind the lens configuration. Even though the terms are explained here using a single lens example, it is important to note that a configuration of multiple lenses can be used to modify the properties and thus the set of lenses may be considered a single lens.

Figure 2.2 illustrates how the light coming from the object forms image behind the lens. Here objects at distance S_1 are said to be in focus if viewed behind the lens at distance S_2 which forms a *focal plane*. All rays of light passing trough the lens parallel to it will cross at distance f. This is a lens property called *focal length*. For thin lenses the following equation holds:

$$\frac{1}{S_1} + \frac{1}{S_2} = \frac{1}{f}. \tag{2.1}$$

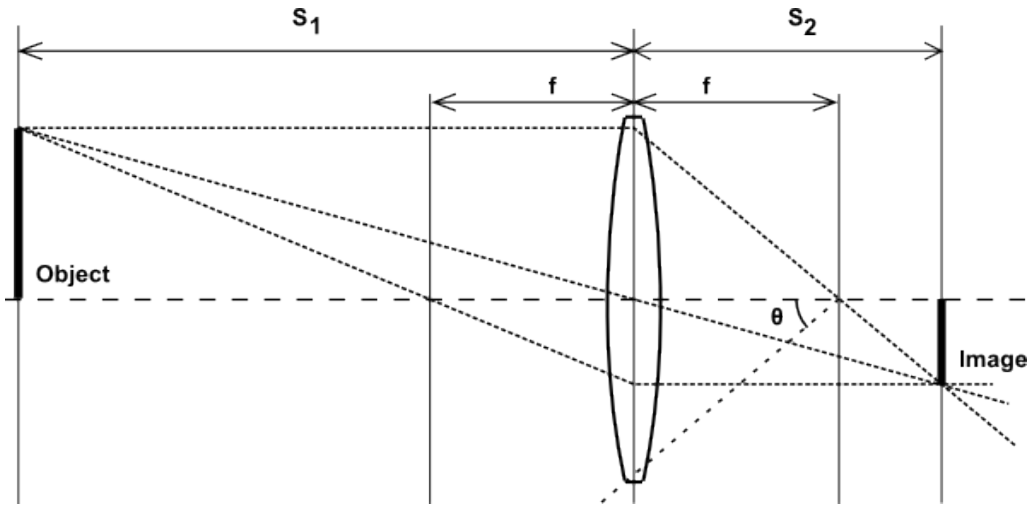

Figure 2.2: Schematic diagram of image forming behind a single lens.

Focal length also determines the *field of view* (FOV) of a lens. This property describes how wide is the scene seen by the camera system. FOV is defined as twice the angle θ seen in Figure 2.1. Lenses may be categorized as telephoto or wide-angle by their field of view. There is no clear definition between the two, but generally lenses with FOV of 65° or greater are regarded as wide-angle and 25° or less telephoto, respectively. Optical zooming is achieved by manipulating the focal length by adjusting the lens configuration. [1, pp.24-32] Due to size constraints mobile phones do not generally have optical zoom.

In front of the optics there is an opening that determines the amount of light entering the optics system. The larger this opening called *aperture* is, the more light beams it allows to pass trough the lens. Aperture is commonly reported as *f-number*. It might be confusing that large aperture size actually has small f-number and vice versa.

In strict sense, only the plane formed at a certain distance may be perfectly in focus. In Figure 2.1 this would be distance S_1. However, in the image the transition from sharp to out-of-focus is not necessarily sudden. The range at which the scene appears acceptably sharp in the image is called *depth of field* (DOF). It is determined by two factors: aperture size and *circle of confusion*. Larger aperture size results in shallower depth of field.

Circle of confusion is a subjective measure describing acceptable sharpness. A dot-like source of light will show as dot on the imaging plane when the lens is focused. When the source is moved closer to the lens or farther away from it, the dot formed on the imaging plane will grow, but it is unnoticeable by the human eye at first. At some point the growing becomes noticeable and focus is no longer considered sharp. The dot forming on the imaging plane is called circle of confusion and the range, at which the source can be moved so that the growing of the circle remains unnoticed,

corresponds to the depth of field. A distance where the farthest edge of DOF is at infinity is called *hyperfocal distance*.

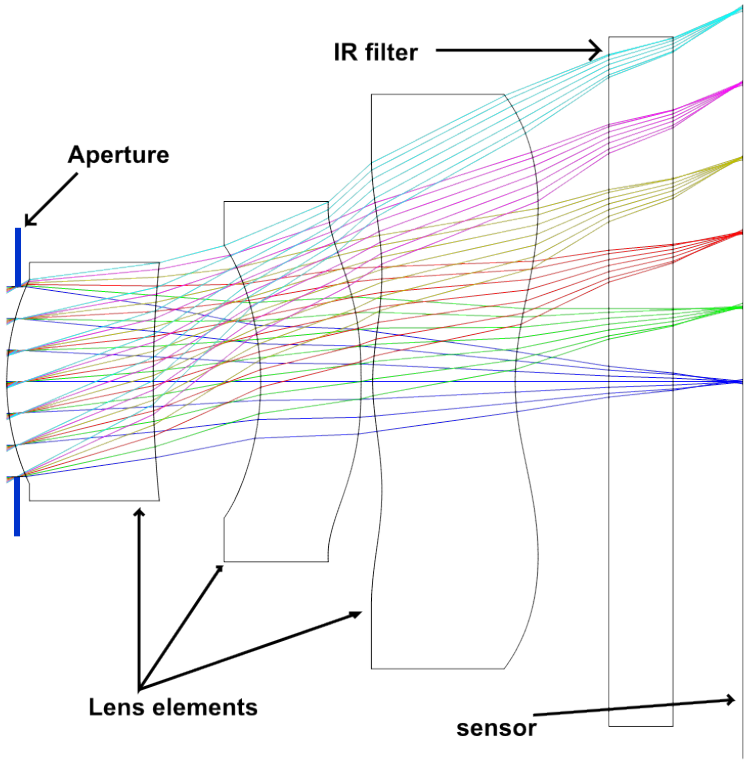

Figure 2.3: Typical phone camera lens with three elements [3].

Figure 2.3 illustrates the course of light trough a typical phone camera lens consisting of three lens elements. The light beams allowed to enter the system by aperture first travel trough the lens configuration, then pass the infra red filter and finally end up to the imaging sensor. The IR filter is placed in front of the sensor to cut out the unwanted infra red wavelengths as imaging sensors are naturally sensitive to it [1, p.32].

2.2.2 Image Sensor

Image sensor, or "imager", is the correspondence of film in a DSC and its purpose is to capture the image formed on it. Sensors consist of image elements, or pixels, arranged to rows an columns. These elements convert the photons landing on them into electric signals. Each pixel corresponds to a point in image and thus the amount must be sufficient in order to reproduce the image with acceptable resolution. [1, pp.54-55]

Shutter is a mechanism that is used to control whether light is allowed to reach the imaging sensor or not. The time between shutter opening and closing is called *exposure time*. During exposure the energy of photons, that light consists of, is

accumulated on pixels and they get charged. After exposure these charges need to be measured and read out from the sensor. *Quantum efficiency* (QE) is the percentage of photons converted into energy and can be used to compare sensors. Charge-coupled device (CCD) and complementary metal-oxide semiconductor (CMOS) are the two sensor technologies used in digital still cameras. They differ in they way the charge of pixels is converted into voltage and how it is read from the sensor. [4, pp.9-10]

Even though CMOS sensors are more complex in design and fabrication, the manufacturing costs are evened by the fact that they are fabricated using the same processes as for other digital and analog circuits. Moreover, CCD sensors also require a more complicated overall system around them. Complexity also affects the *fill factor* which is the percentage of light sensitive area, photodiode, out of entire pixel area. CCD sensor are more sensitive and can achieve higher dynamic range due to their greater fill factor. Respectively, CMOS sensors consume much less power, are capable of being integrated with the whole system on a single die and have the possibility of addressing pixels separately enabling windowing. [4, pp.10-11] CMOS sensors are used in mobile phones because of these advantages.

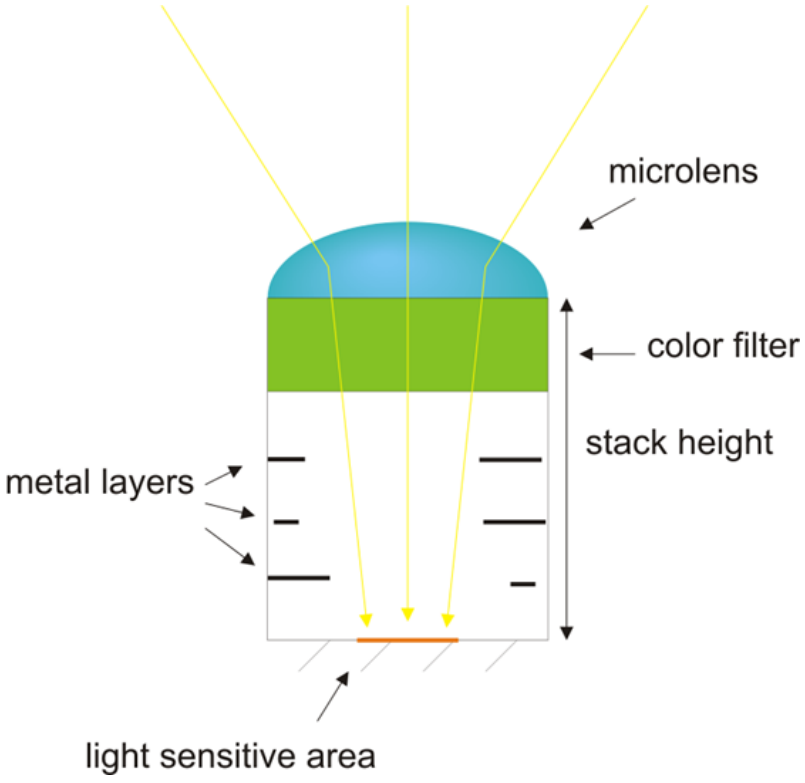

Figure 2.4: Simplified cross-section diagram of CMOS FSI pixel structure [2].

The structure of a pixel is highly important in attempt to reproduce an image as precisely as possible. "The "perfect pixel" accepts all light, classifies it by color content, preserves its spatial information, and produces an output representative of

its intensity – all in a lossless manner. Yet, like all things perfect, such a pixel cannot be achieved." [5] One of the imperfections regarding pixels is *crosstalk* caused by optics and electrical components [1, p.83]. It means that some photons are received by the wrong pixel and will affect the image colours [5]. Colour information capturing is explained in 2.2.3.

Structure of a frontside illumination (FSI) CMOS pixel is illustrated in Figure 2.4. This is the dominant structure because of the technology maturity and performance-cost ratio [5]. Microlens on top of the pixel is used to direct light beams to the centre of the light sensitive area reducing crosstalk [1, p.84]. Sensor wiring is in the depicted metal layers which will reflect some of the light away from the sensing area causing loss of photons. Also this effect is reduced by microlens.

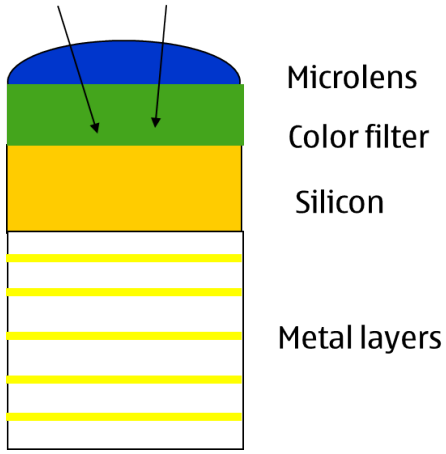

Figure 2.5: Simplified cross-section diagram of CMOS BSI pixel structure [2].

Backside illumination (BSI) pixel is another CMOS sensor technology that tries to overcome shortcomings of FSI. The basic idea is that sensor is turned around allowing light to reach the silicon layer containing the sensitive area from the backside. As can be seen in Figure 2.5, microlens and colour filter are also switched to the other side. This way the interconnections in metal layers are no longer in the lights way. With BSI the optical and electronic part of the sensor can be designed and optimized separately as they do not affect each other. This is an especially important attribute in attempts to further reduce pixel size, but the fabrication process is more complicated due to more steps. It will take some time for this technology to mature before it can challenge FSI in lower price points. [5]

2.2.3 Colour Imaging

Light is electromagnetic radiation that causes a physical stimulus on the visible spectrum. In air and vacuum the said range is between wavelengths 360 nm and 830 nm. On this range the receptors of the eye get stimulated and result in vision and

perception of colour. [6, p.3] The scope of this thesis also falls into the visible part of spectrum. Three-component additive colour model describes colour information in amounts of three primary colours: red (R), green (G) and blue (B). Each of them have a wavelength range of their own in the visible spectrum [6, p.16]. [4, p.11] Colour separating technologies used in DSCs are based on that fact.

As such an image sensor is monochrome and cannot detect the colour of light as it simply measures the light energy. Thus the colours need to be somehow separated in order to reproduce the scene in colour. [1, p.62] A high end solution is to use three separate sensors, one of each primary colour. Light is split into components using a prism and then directed to the sensors. This method provides good colour fidelity, but is also relatively big in size and expensive. A company named Foveon has introduced another technology based on special hardware, Foveon® [7]. The idea is to capture the colour information with a sensor that has three light detecting layers. Different wavelengths of light penetrate on different depths into the silicon and thus all colour channels can be separated. [4, p.11-12]

Another way to separate colour channels is to use a filter to cut away unwanted wavelengths, allowing only light of specific colour to reach the sensor. The simple solution is to capture three separate images, one with each primary colour filter for the whole sensor, and then combine them. However, this method is not very useful due to requiring multiple exposures. [4, p.11]

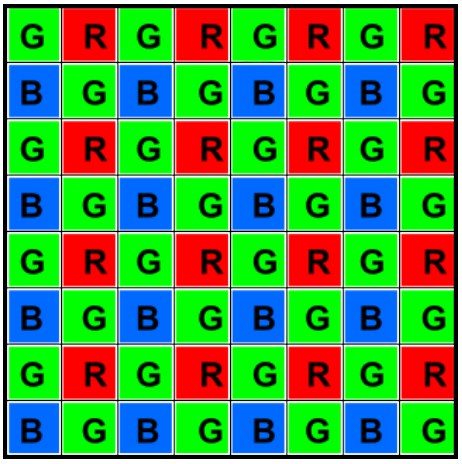

Figure 2.6: The Bayer pattern based colour filter array.

The most common method is to use a *colour filter array* (CFA) on top of the sensor. It is a mosaic-like filter with colours mixed in a specific pattern. On pixel level this is depicted in Figure 2.4 and in Figure 2.5. This is the smallest, simplest and cheapest technology. Several patterns using primary colours have been studied and even some which use complementary colours: cyan (C), magenta (M) and yellow (Y) [8, pp.39-41]. CMY patterns achieve higher sensitivity compared to primary colour versions, but in the end their colour reproduction is not as accurate [1, p.63].

Information about the used pattern is required to combine measurements for each pixel colour. This is done in the image processing pipe (See section 2.3).

Bayer pattern, illustrated in Figure 2.6, is the most used filter pattern in DSC CFAs. Half of the pixels are covered with green filters, which works well because human visual system extracts image details primarily from green segment of the visible spectrum. Green is thus associated with luminance differences. The remaining half of pixels is divided evenly to red and blue filter covered pixels so there is a quarter of both. Those colours are associated with colour perception, respectively. [1, p.63]

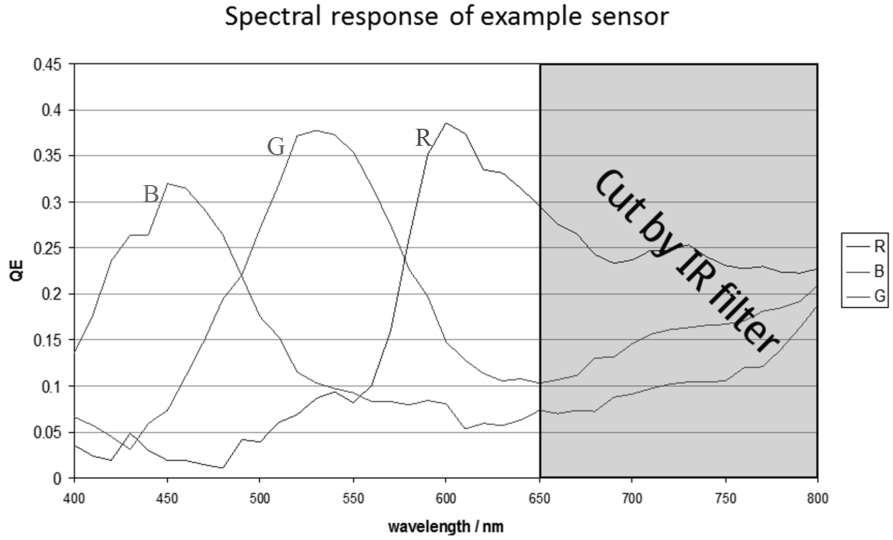

Figure 2.7: Example of measured spectral response from a sensor using Bayer pattern [2].

There is an example of Bayer pattern CFA produced spectral response depicted in Figure 2.7. As can be observed, the filters are not perfect and have some overlap in passing wavelengths. From this figure it is also easy to see the importance of an infra red filter for colour separation.

2.2.4 Lens Actuators

In order to focus the lens so that image forms sharply on the sensor, a mechanism is needed to move it. Simplified schematic diagram of the principle is represented in Figure 2.8. As explained in 2.2.1, the lens unit in most cases actually contains multiple lenses. Far and near ends are the actuator limitations for focusing as far and as near as possible. Camera modules without the actuator system for lens moving are called *fixed focus* (FF) cameras.

There are multiple factors that affect selection of the actuator system. From performance point of view, movement accuracy and time are important. Lens needs to be able to move very short distances. Moreover, repeatability of the movement must be precise. The faster the actuator responses and moves lens the faster focusing

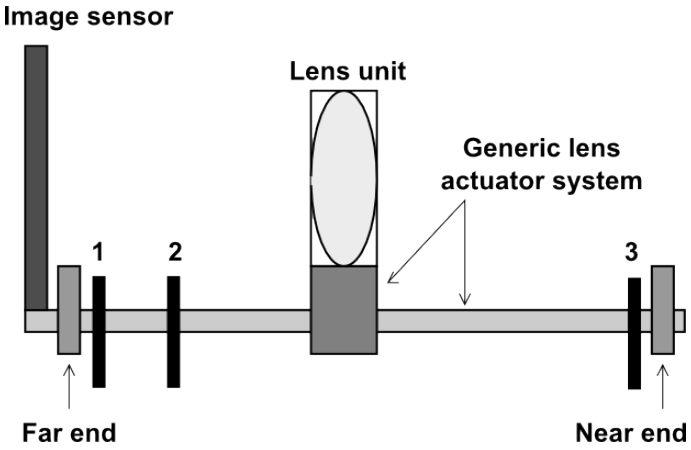

Figure 2.8: Principle schematic of lens actuator system. In position **1** lens is focused to infinity, in **2** to hyperfocal distance and in **3** as close as possible.

can be achieved. Also error caused by gravity when camera device posture changes should be minimal. [9] Of course it is possible to compensate this with control logic if the effect characteristics are known.

To achieve these attributes, the system size, power consumption and cost needs to be evaluated. Power is consumed during lens movement, but depending on the technology, even keeping the lens still in a certain position may require current. For a mobile phone camera the desired properties are: low cost, small size, low power consumption, rapid response time and high repeatability. [9]

A technology that meets the requirements set for mobile phone lens actuators is *voice coil motor* (VCM). The lens is moved with electromagnetic force induced by leading current into coils. Depending on the implementation, the need for holding current can be eliminated. [9] Another potential option to be used in phone cameras due to its low power consumption is the *piezoelectric motor* [8, p.278]. In this technology a crystal extends or contracts according to the electric field it is in.

Stepper motors are also a good choice because of their speed and preciseness. However, they are large in comparison and thus unsuitable to be used in phone cameras. [8, p.278] A totally different type of technology is to use a liquid lens. There are different types of implementation methods, but the lens usually comprises two immiscible liquids. The surface between those liquids is then manipulated e.g. using electricity or pressure. This way the optical properties of the lens change and focusing functionality is achieved. [10]

2.3 Image Processing Pipeline

An analogue to digital (A/D) conversion is performed to the image data when it is read from the imaging sensor. At this stage it is also possible to add gain, which is one of the parameters for AEC, but doing so will increase image noise. [4, pp.15-16]

Noise may be defined as any image signal variation that has deteriorating effect on an image [1, p.66] or something that was not present in the original scene, but appears on the captured image. [8, p.60]

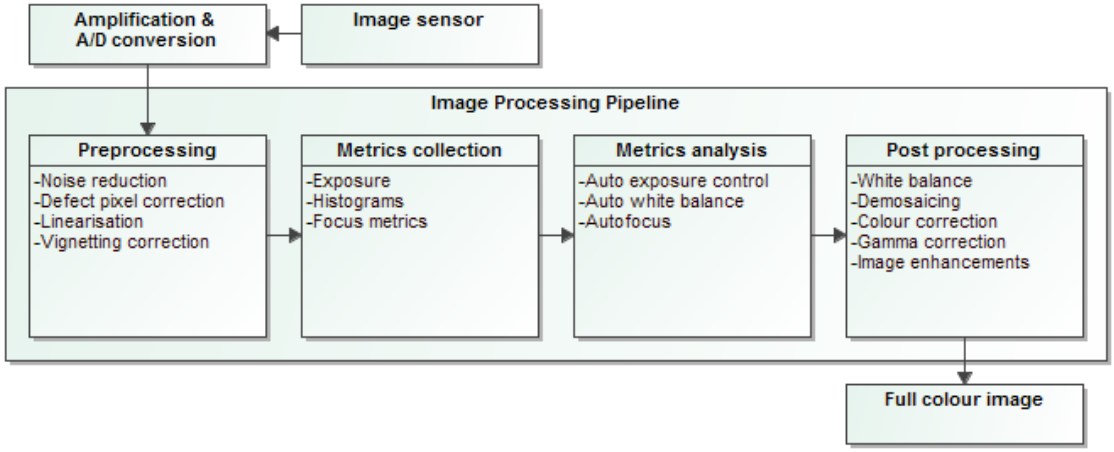

Figure 2.9: Example of an image processing pipeline in digital camera system.

As the raw image data has been read from the sensor, it still needs considerable amount of processing. At that point the data enters *image processing pipeline* (IPP) of the digital camera system. [4, p.15] The system presented in this thesis is merely an example as the amount of processing algorithms, their order and implementation is dependent on the camera manufacturer [11]. IPP may be divided into four steps: preprocessing, image metrics collection, metrics analysis and post processing. Figure 2.9 illustrates these steps and some algorithms typically performed during those steps to produce a full colour image. Metrics collection and analysis steps are related to AEC, AWB and AF. The last one is presented in detail in Chapter 3.

2.3.1 Preprocessing

There are multiple sources of image noise and defects in the camera system hardware preceding the image processing pipeline [1; 4; 11]. It is important to correct these errors and imperfections in an early stage of IPP as otherwise they may spread on a larger area of the image. These essential corrections are the reason for the operations in the preprocessing phase. Because of the popularity of Bayer pattern CFAs, preprocessing algorithms usually operate on mosaic-patterned data.

Noise reduction is one of the processing steps performed in the preprocessing phase. As a result the *signal to noise ratio* (SNR) is improved. Image noise may be classified into signal-level, temporal and spatial effects. The first one refers to statistical variations in the original signal, light. [8, p.60] Temporal effects are caused by random variation that changes over time and the largest source of this type of noise is thermal noise. In case of video the human eye filters out temporal noise

effectively. However, when an image is captured, temporal noise becomes spatial noise that changes from shot to shot deteriorating image quality. Noise that has fixed position in the image is called *fixed-pattern noise* (FPN) and can potentially be removed efficiently. [1, pp.66-67] Noise reduction algorithms attempt to recognize the noise pattern and then remove it. Various noise reduction methods have been studied and compared [12].

Defect pixel correction is needed to compensate for pixels responding differently from others. As a result the pixel may be a black or white spot or the characteristics of its response significantly differ from the expected. [1, pp.201-202] The reason for these defects may be in the individual pixels structure or caused by the sensor layout. They may be corrected e.g. with a small median filter or simple interpolation. [8, pp.230-240]

Linearisation is performed on the image data because it may be captured into a non-linear space and the algorithms after preprocessing require it to be linear. It is also possible to add offset, or pedestal, to individual pixel values. This may be done e.g. to compensate for dark current, which is a result of pixels getting charged even when the sensor is not exposed to light. It is caused by ambient temperature and the longer the exposure time, the more dark current will be generated. The applied pedestal may be received either by placing an opaque mask on the edge of the sensor and using the mean of the covered pixels to estimate dark current or by capturing a dark image with a corresponding exposure time to measure the offset needed for each pixel. [11] Example result of linearisation is presented in Figure 2.10.

Figure 2.10: Example of an image without (A) and with (B) linearisation [13].

Vignetting is caused by the fact that the amount of light reaching the image sensor is at its highest in the middle of the sensor and decreases towards the edges. As a result the image appears darker near edges. The whole vignetting phenomenon is a combination of lens, filter and sensor vignetting. Lens vignetting also causes sharpness degradation in a similar spherical pattern. In addition, filter and sensor

may also cause colour vignetting, which is very undesirable because of the complexity in correcting it. Lens and sensor vignetting may be compensated with gain, but it will also increase noise the component. [8, pp.130-145] Figure 2.11 demonstrates the effect of vignetting.

Figure 2.11: Example of an image without (A) and with (B) vignetting correction [13].

2.3.2 Automatic White Balance

Colour characteristics of the scene depend on the light source illuminating it. It is a result of lights of different spectrum composing of varying amounts of red, green and blue components. For example a white paper illuminated by day light under a clear blue sky radiates bluish. Under these conditions we would be able to see the paper as white because the human visual system is very good in adapting to different types of illuminants. However, a digital camera system only captures light as it is and thus needs to determine the reference white in order to adjust colours accordingly. This is what automatic white balance algorithms are for. [1; 4; 11] By looking at Figure 2.12, it is easy to see why AWB adjustment is needed to reproduce colours as they are seen by the human eye.

As illustrated in Figure 2.9, histograms are collected after preprocessing step and analysed in the step after by auto white balance algorithm. A very basic method for determining the needed adjustments is the *grey world* algorithm. It assumes that colours of an image will average out to grey and can be scaled according to the deviation from the average. Another similar approach is to assume that a white area must result in maximal response in all colour channels. Colours are therefore scaled by the maximum. Nonetheless, both of these algorithms are poor estimates for white balance adjustment as such. For example the grey world assumption fails if the image is a close-up of a brightly coloured object, such as flower, or any other scene containing uneven distribution of primary colours. This is also the case for

Figure 2.12: Example of an image without (A) and with (B) white balance adjustment [13].

the other algorithms presented. [1; 4; 11]

Another approach to the white balance problem is to determine a white point in the image and use it as a reference in adjusting the colours. The point may be chosen by selecting the brightest spot in the image. Unfortunately, this method fails if the chosen point has any tonality or has been overexposed. *Gamut mapping* is a statistical method based on database of images captured under different illuminants and compare colour gamuts of the scene and the database images. This way the light source can be estimated. [1; 4]

In practice, AWB algorithm of a modern DSC uses a mixture of these approaches. Some of them utilize *Retinex theory*, proposed by E. H. Land [14], that is based on how the human eye works. It is combined with grey world method by Lam in [15]. Chen et al. propose a method based on light source estimation using fuzzy neural networks [16] and Kehtarnavaz et al. present a technique called "scoring" that uses a database approach of colour reference points [17].

2.3.3 Automatic Exposure Control

Exposure means the amount of light reaching the sensor and controlling it is an important factor in terms of overall image quality. Too little light will result in *underexposed* image causing the dark areas become noisy, but there are techniques to improve quality in such cases. The opposite problem, that is too much light falling on the sensor, is called *overexposure*. It may render many light areas saturated, which is a greater problem than underexposure because there are no means of correcting it. [8, p.261] Figure 2.13 demonstrates the effect of overexposure.

Auto exposure control logic tries to determine the correct exposure by adjusting three parameters: f-number, gain and exposure time. The first one means modifying aperture size and the amount of incoming light, as explained in Section 2.2.1.

Figure 2.13: Example of an overexposed (A) and well-exposed (B) image [13].

However, some DSCs, e.g. phone cameras, have a fixed aperture, meaning that only the latter two parameters are left for exposure control. Gain may be analogue, added during A/D conversion, see Section 2.3, or digital added after conversion, respectively. The third parameter determines how long the shutter is kept open. Technically there is no definition for correct exposure, but the target is to utilise the full dynamic range of the imaging sensor. [1; 4; 11]

Metrics are collected for AEC algorithm from the image luminance information. When using a Bayer pattern CFA, the green component may be used to estimate it, see Section 2.2.3. The image area can be divided into blocks and the average is calculated for the area of each. These metrics may then be used to determine the wanted exposure level. A simple method is to measure the average luminance signal and compare it with a reference. Exposure is then adjusted to provide a constant scene luminance. If the scene is clearly backlit or frontlit, the block may be used to divide image into areas. In this case the exposure is controlled to maintain the difference in those areas. Same idea may be utilised to ensure correct exposure in the *region of interest* (ROI). Blocks on the ROI area are selected and the adjustments are made to keep the luminance constant, possibly sacrificing the rest of the image to under- or overexposure. [1; 4; 11]

2.3.4 Post-Processing

Post-processing is the last stage in the IPP before the captured image is ready to be shown or stored. Depending on the hardware of the digital camera system, there are usually a few very important steps in the beginning while the final ones may be considered to be fine-tuning in comparison.

Demosaicing is also known as CFA interpolation and it is a mandatory processing step when a CFA is used for capturing colours. Each pixel has only captured

information of one colour channel and the true value needs to be estimated by utilizing the information stored in the neighbouring pixels. Computationally this is a very intensive process and the used algorithm is expected to be efficient while also providing high quality results. It is common for this step to introduce distortions and artefacts into the image, e.g. zipper effect, blurring and false colours. These errors are a problem especially with the simplest algorithms and they may be needed to be removed in later processing steps. [4; 11; 18] Because of the popularity of Bayer CFA, several demosaicing algorithms have been developed. Some of the popular methods are outlined and compared in [18] and more possibilities are proposed in [19], [20] and [21].

Colour correction step needs to be performed because the colour characteristics in the image are affected by the camera system hardware. In other words, the image is in the colour space of the camera device and far from the colours perceived by human visual system. The correction is applied according to the known camera properties in order to produce the best possible output. Finally the image is transformed into an universal colour space model that is suitable for most output devices. Popular choices are sRGB and AdobeRGB. [4, pp.18-19]

Gamma correction is performed because the image data is in linear format, but the input voltage-to-emitted light intensity response of most display devices is not. Therefore, the data needs to be non-linearised for correct output. Also the human visual system responds nonlinearly to light intensities and distinguishes smaller differences in the dark. A simple correction may be presented as equation:

$$output = input^{1/\gamma}, \tag{2.2}$$

where γ is a system-dependent parameter. [4, pp.19-20]

Image enhancements are the final processing steps in the IPP before the image is ready for display and storage. Camera manufacturers use different sets of steps for enhancing, but the aim is at improving image appearance. Noise and artefacts generated by preceding steps may be attempted to be removed with denoising algorithms. Distortions caused by the lens can be corrected by knowing its properties. As the human eye is highly sensitive to sharp edges, it is also a good idea to apply image sharpening. Contrast and brightness are subject to user preference and the combination may be adjusted in post-processing stage. It is also reasonable to scale the image. [4; 11]

3. AUTOFOCUS

There are several ways to implement autofocus in digital camera system. In this chapter the most common technique, contrast-based autofocus, is explained in detail. Knowing the stages of the focusing sequence is necessary to understand the target of metrics calculation.

3.1 Basic Principle

Focusing is a crucial part of capturing a successful image. Even though it can be a matter of opinion whether the image is focused on a correct object in the scenery, it is easy to see when the image has not been focused correctly. This is true especially when the object to focus on is far from the camera system as even the smallest error in lens position will result in blurry image.

Autofocus is one of the key functions in a digital still camera [1, p.239]. Its purpose is to adjust the distance between lens and the imaging sensor in an intelligent way to find the in-focus position where the image appears sharp as described in Section 2.2.1. The autofocus system controls lens trough an actuator that may be implemented in various ways, see Section 2.2.4, but it is not dependent on the actual focusing techniques used. Different ways to implement autofocus functionality have been developed. On a general level the techniques can be classified as being active of passive. Methods that work by analysing the scene the camera sees are considered passive and methods using an auxiliary mean of measuring the distance between camera and the object are called active. [11; 8; 1]

The pros and cons of different autofocus techniques vary. Focusing speed, accuracy on different types of content and system costs are subjects to consider. Also size and weight of the hardware needed by the autofocus system are important factors especially in mobile devices. In general, autofocus systems are effective in finding the correct lens position. However, the option to use manual focus may prove to be useful in situations where autofocus fails to focus at all or constantly focuses on wrong object.

3.2 Contrast-Based Autofocus

Contrast based autofocus is a passive technique used in lower cost cameras such as smartphones [8; 22]. It requires no special hardware for it utilizes the analysis

of data collected from the image. The basic concept is based on the assumption that an in-focus image has more high frequency components than an out-of-focus one [1, p.240]. In other words, the image contrast is at its highest when the image is in focus. When the focusing sequence is started the algorithm will always make decisions on where to move the lens to. The logic is based on measuring the sharpness value, also known as focus value, for each lens position visited. By using the sample measurements, it is possible to form a focus value curve as a function of the lens position. The algorithm then concludes the point with the highest focus value. As a whole, the system can be presented as three major components: approximation of image gradient intensity values, interpreting the intensities into sharpness values and an algorithm that interprets them, as the focusing sequence progresses. There are multiple ways to implement each of them, and most of them can be combined as such or with minor modifications to output format. Therefore, the concept of contrast based autofocus is not unambiguous in terms of implementation.

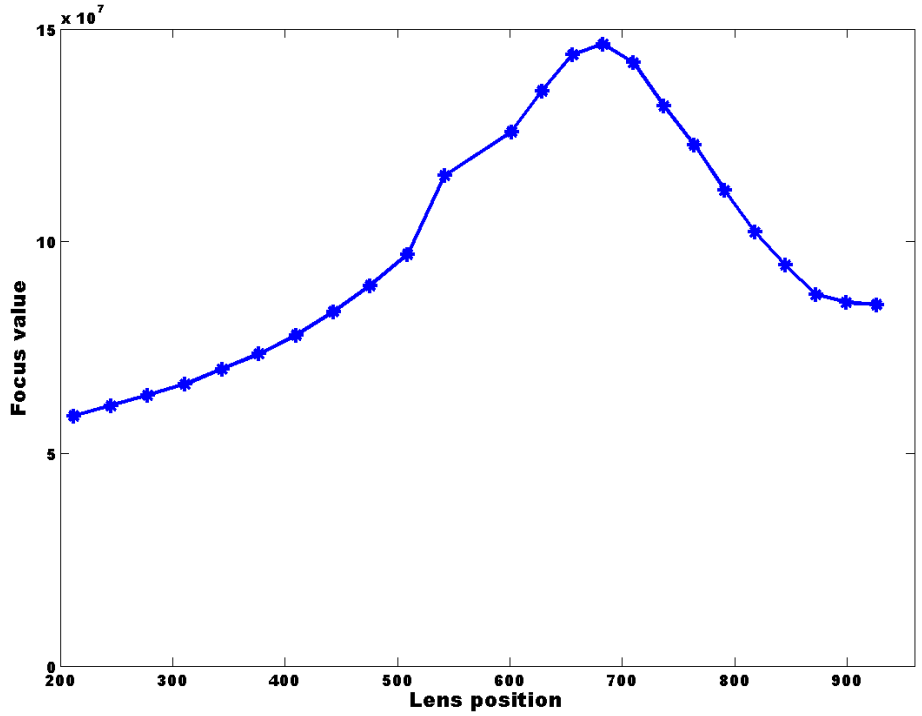

Figure 3.1: Example of frequently sampled focus value plotted against lens position. The actual focus values are not important, but the ratios between them. Lens position values presented are relative.

An example of focus value curve is illustrated in Figure 3.1. The asterisks depict sample points and the example has a frequent sampling rate throughout the lens movement range. This curve has a clear single peak that marks the in-focus position, but this is not always the case. The more and finer details the object has, the more distinct the difference is between an in-focus and an out-of-focus image in terms of focus value. It will naturally result in clearer and narrower peak in focus

curve. On the contrary, a flat surface with minimal texture may produce a near flat curve complicating the task of the autofocus algorithm. Similar problem also occurs in low-light situations as finer details are not as clearly visible as in good lighting conditions. This also makes the role of automatic exposure control very important for badly exposed images, especially overexposed ones, lack sufficient contrast information.

One of the downsides in contrast based autofocus, in addition to lack of contrast, is that it is sensitive to imperfections in the image data. Everything that causes changes in the content to focus, may cause an error in the focus value measurement resulting in skewed focus curve. Content changing can be caused by different types of movement: high contrast objects entering and exiting the focusing area, the whole camera system not staying steady or even the lens movement caused by the focusing process itself as it will change the field of view slightly. Motion not only changes the image content, but also causes blur, rendering the data unreliable. Therefore, the synchronization of exposure and lens movement must be paid attention to. Either the system has to wait for the lens to stop moving before starting the to expose the next frame or discard the surely corrupted data on following algorithm iteration. All in all, it is important that the measured focus values and the corresponding lens positions match. All motion related issues are further emphasized when the exposure time is increased in low-light situations.

Noise is another source of focus value corruption. Low-light conditions cause the exposure control to add gain to the image resulting in increase in noise. Noise may cause loss of detail in the image or add artificial contrast in some areas. The common property of the data imperfections is that the errors they cause will lead to distorted focus curve. The result may significantly differ from the lack of contrast case, where there is no clear peak, and have multiple focus peaks. Furthermore, depending on the situation, the in-focus lens position may not even have the highest peak.

To address problems regarding low-light situations, digital cameras may come with an autofocus assist light that is switched on for the time of focusing sequence. However, the range is very limited, easing only focusing on near objects. When photographing people or animals, the use of assist light is also likely cause a distraction, ruining the moment to capture.

3.2.1 Focusing Algorithm

The algorithm part in contrast based autofocus system is responsible for finding the peak in focus curve, the maximum of sharpness function. A single focus value alone does not give any information about the state of focus as the absolute maximum value is dependent on the image content. Furthermore, any conclusions cannot be drawn, on how far from the correct focus position the lens is. The only way for

finding the focus is to get multiple measurements in order to form at least a piece of content's focus curve [1, p.240]. This makes the contrast based autofocus slow compared to some other techniques presented in Section 3.3.

A basic hill climbing algorithm is able to find the maximum of the sharpness function. The search is initially started by moving the lens to either near or far end and then moved towards the other in steps. If the algorithm only does a single pass, the steps need to be fine in order to find an accurate focus position. The algorithm observes focus value measurements as the lens is moved. It should first notice the curve slope rising and then falling. Having passed the peak, it is possible to approximate the final in-focus lens position by using the samples around peak position. The accuracy may be improved by using a multipass method. It allows the use of coarser steps on first pass in order to find a rough estimate on the focus position. Lens is then moved back towards the assumed focus peak. Second pass uses finer steps to improve the accuracy of the estimation. Depending on the location of in-focus lens position, the two-pass sequence may achieve faster focusing time than single-pass. Furthermore, accuracy is likely to improve due to finer steps near focus function maximum.

The principle of two-pass method is depicted in Figure 3.2. The focusing sequence advances trough the measurement points in alphabetic order. The first six steps from **A** to **F** belong to the first pass with long step size. In point **E** the value has not yet dropped enough compared to the previous focus value and thus the sixth step is taken. The second pass consists of points from **G** to **I** and they point out a more precise estimation for the in-focus lens position.

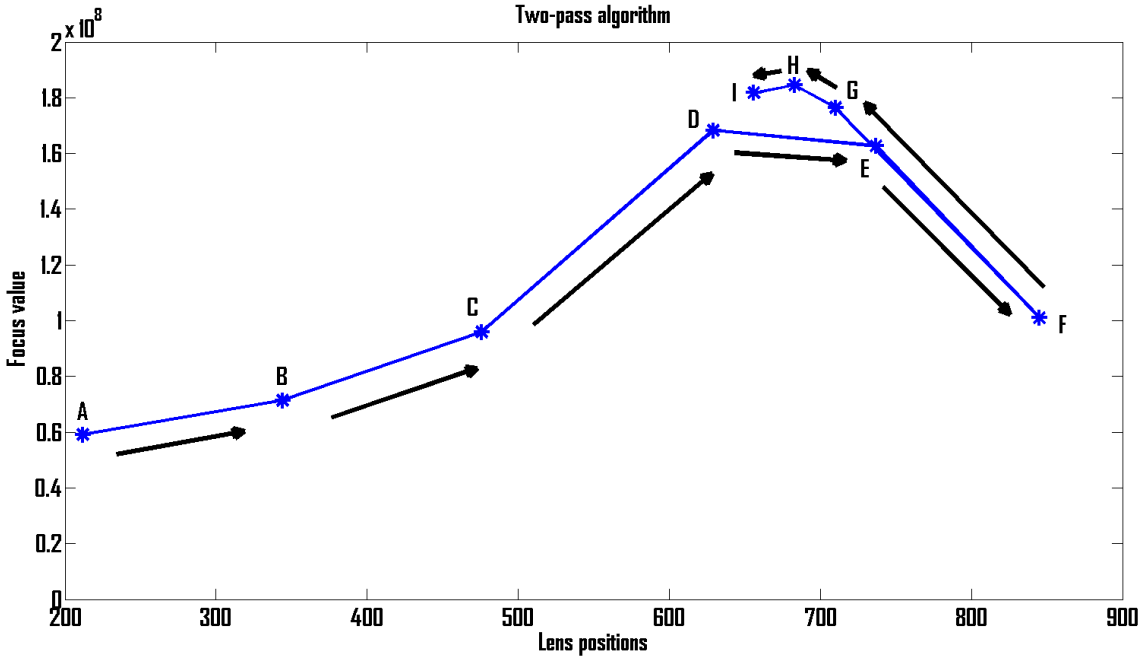

Figure 3.2: Example schematic of focusing sequence utilising two-pass method.

Choosing the area to focus on, the region of interest, is very flexible in a contrast based autofocus system. The size and location of the focusing area may be adjusted freely. The area used for focus value calculation is also called a focus window. It is possible to divide the region of interest into multiple focus windows [1; 22]. This naturally adds complexity to the autofocus algorithm. Figure 3.3 illustrates four different focus window configurations. Information from multiple windows may be combined in different ways. For example, a grid of sub-windows has been proven to be good for focusing on an object that does not fill the entire region of interest [22; 23] or on a moving object [24].

When using multiple focus windows, it is also possible to assign different weighting factors for them. The object to focus on is usually located in the centre of the window thus it is reasonable to use higher coefficient for middle windows than to those on the edges of the region of interest. This way the middle windows get more weight in the total process. In this method the focus value of each focus window is multiplied with the corresponding coefficient. [22]

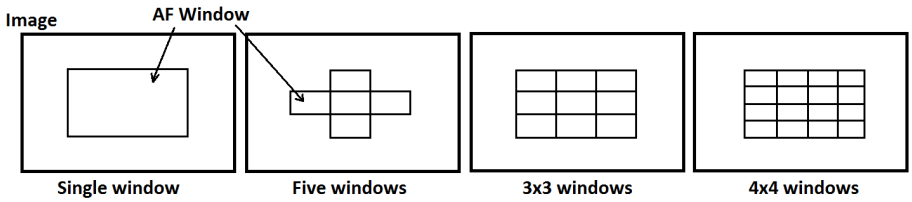

Figure 3.3: Example focus window configurations.

As explained in Section 3.2, the autofocus algorithm faces challenges with image imperfections that skew the focus curve. It is important not to make rushed decisions about focus peaks based on a single measurement as it may be erroneous and may lead to detection of a false peak. Thresholds need to be used when examining the behaviour of the curve. Using multiple focus sub-windows helps as unreliable areas can be discarded completely. In general, the problems caused by movement can be countered by performing the focusing sequence as quickly as possible. Although, in low-light conditions the longer exposure time inevitably prolongs the process.

More sophisticated autofocus algorithms use adaptive step sizes and make decisions on lens movement direction based on the measurements during the focusing sequence. They also attempt to overcome the challenges caused by imperfect focusing conditions. For more information on how these techniques operate, an interested reader is advised to explore proposed methods in [25], [26], [27] and [28].

One more complication to the task of the autofocus algorithm is the focusing mode. For single shot images the lens may be moved freely when searching for the in-focus position. If autofocus is to be utilized in video mode, the focusing must be continuous as the content will change and the in-focus position with it. The initial

sequence may be similar to that of single shot case, but excess lens movement should be minimized. Continuous autofocus mode introduces a whole new set of matters to consider in the algorithm. How to detect the loss of focus and how to search for it again are examples of those issues, but they are outside the scope of this thesis.

3.2.2 Gradient Intensity Approximation

Generally, focus values are based on the luminance gradients of an image. The absolute values get higher the better the image is focused. There are various methods to calculate the gradients. [29] One of the most common examples is the *Sobel operator* that consists of two kernels and for image **A** the computation is defined as:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * A \; and \; G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A$$

where * denotes a convolution operation. G_x is an image containing the horizontal derivative approximations and G_y the vertical ones, respectively. The gradient magnitude can then be calculated in each image point by using the following equation:

$$G = \sqrt{G_x^2 + G_y^2}. \tag{3.1}$$

It is also possible to calculate the direction of the gradient or the edge with

$$\theta = arctan(G_y/G_x), \tag{3.2}$$

but it is not needed for contrast based autofocus. Figure 3.4 illustrates well the properties of Sobel filters. Horizontal filter finds the vertical edges in the image and vertical filter finds the horizontal ones. In the said figure, this can be easily observed by looking at the colour checker chart or the photo frame. The edges of the colour patches and the frame are clearly detected differently depending on the filter.

A similar variation is the *Prewitt operator* that uses the following two filters:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix} * A \; and \; G_y = \begin{bmatrix} +1 & +1 & +1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} * A.$$

Even though the kernels are of same size than in those of Sobel operator, they are slightly lighter calculation-wise as the coefficients present no need for multiplication operations. Gradient magnitude and direction can be calculated in as with Sobel operator, using equation (3.1) and equation (3.2), respectively. This is also the case with the third example operator.

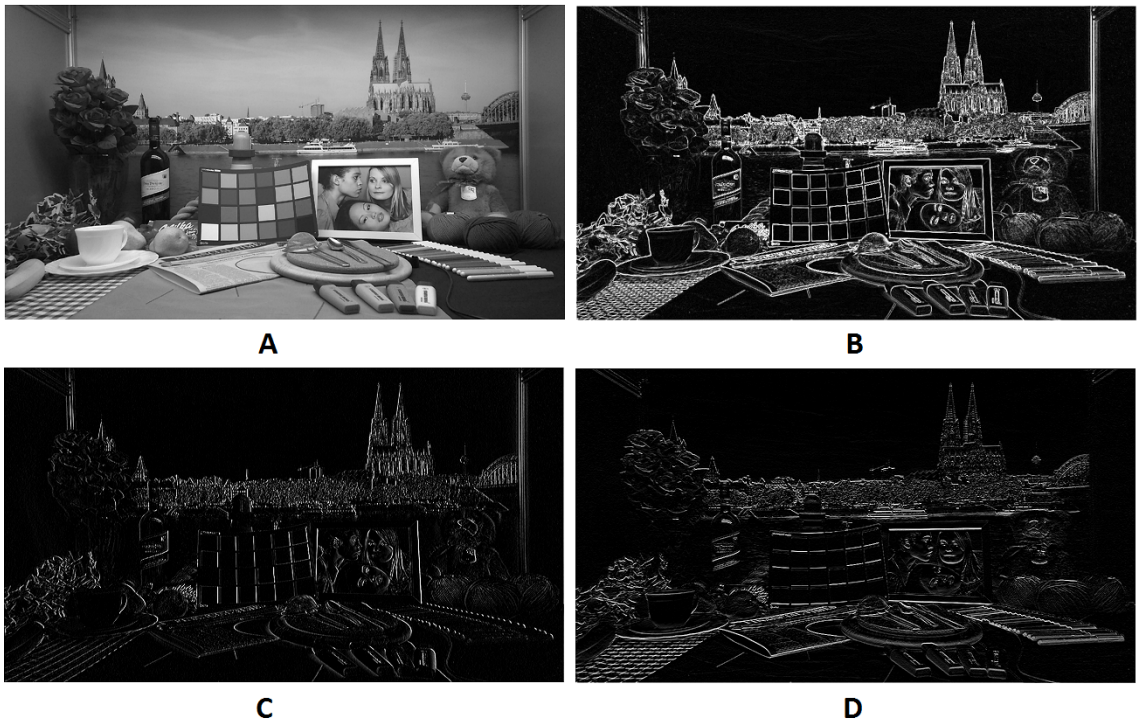

Figure 3.4: Gradient approximations of Sobel operator. (A) Shows the original image. (B) Has the combined magnitude of Sobel operator. (C) and (D) Are the result images of horizontal and vertical Sobel filters.

Compared to Sobel and Prewitt, *Roberts cross operator* has smaller kernels:

$$G_x = \begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix} * A \; and \; G_y = \begin{bmatrix} 0 & +1 \\ -1 & 0 \end{bmatrix} * A.$$

The notable difference is that the filters measure derivatives in diagonal directions instead of horizontal and vertical. By looking at Figure 3.4 and Figure 3.5, it can be observed that Sobel and Prewitt operators produce very similar results compared to Roberts cross. In this example the Roberts cross operator has picked up the scene details very well. Although, due to smaller kernels, the operator requires calculations than the other two. In fact, if the zero coefficients are ignored, the amount of calculations is only a third of what is required for Sobel and Prewitt.

3.2.3 Focus Value Calculation

The gradient intensities are approximated for each pixel in the entire region of the focus window. Nonetheless, the focusing algorithm works with focus values thus the intensity gradients need to be converted into a single value. The most simple solution would be to calculate the sum of the gradients.

As with the case of using multiple focus windows, it is possible to use weighting when combining the gradient values into a focus value. It is possible to achieve

Figure 3.5: Gradient magnitude approximations of Prewitt (A) and Roberts cross (B) operators.

a similar effect of emphasising the importance of objects located in the middle of the focus window. This is especially convenient when using only a single, relatively large, focus window as it is possible to achieve similar middle-centred focusing as with the use of multiple focus windows. Moreover, gradient weighting is simpler to implement as it adds no complexity to the focusing algorithm.

3.3 Other Autofocus Techniques

Contrast-based autofocus is not the only passive focusing technique. Other fundamental way to examine the incoming light is to use phase detection. The main principle is to have two separate images of the same scene and compare the matching points in them. These images are formed so that they are results of light passing trough the main lens from different sides. By detecting the amount and direction of the shift between images, it is possible to calculate the corresponding direction and distance that the lens needs to be moved for the image to get in-focus. [8, p.269] Figure 3.6 illustrates a simple schematic of the shift between light beams incoming from different sides of the lens. The direction of the shift depends on whether the lens is too near or too far from the imaging sensor, i.e. is the lens back focused or front focused.

A traditional method to implement phase detection is to place a mirror in front of the imaging sensor for the time of focusing. This mirror redirects the light trough prisms so that two images are produced and directed to separate autofocus sensor. The inputs of the sensors can then be compared and shift of individual points calculated. The AF sensors are small in size compared to the actual image sensor, but this system is a major modification to the hardware and occupies additional space. The mirror system may also be in place because of other differences in the camera system design and is not as big of a change in that case. [1; 8]

It is clear that phase detection enables faster focusing sequence than contrast-based autofocus because it is able to determine the in-focus lens position with a single

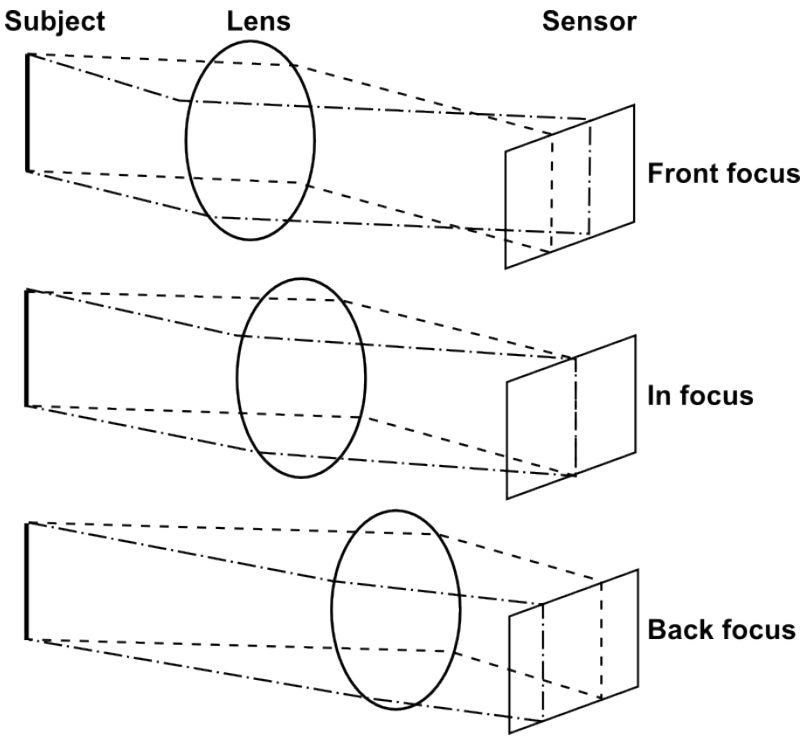

Figure 3.6: Schematic of light beams entering from different sides of the lens resulting in phase shift on the imaging sensor when the image is out of focus.

measurement. However, the additional hardware needed makes it an undesirable choice for small-sized and low-cost cameras. To overcome this issue, efforts have been made to implement phase detection in the imaging sensor itself. The idea is to replace some of the regular pixels of the sensor with special phase detection pixels, or focus pixels. The principle of comparing two image signals is applied by covering one half of the focus pixel with a plate, so that it only collects light entering from one side of the main lens. Additionally, focus pixels with the opposite half covered are needed to obtain the other image signal for phase comparison. The cost of this technology is more complex sensor manufacturing process. [8; 30]

Canon, a camera manufacturer, has developed a variation of the focus pixel idea. The focus pixel is implemented by simply splitting the light sensitive area of normal pixel into two separate photodiodes that are still under the same microlens. These photodiodes may be read-out independently, which enables them to be used for phase detection in a similar manner as the pixels with the other half covered. When capturing an image, the input may be combined and used like a regular pixel. [31] In practice, the implementation can be compared as to having twice as many pixels on the sensor, excluding the microlens [30].

In comparison to contrast-based AF, the low light performance of phase detection method is worse. Focus pixel techniques suffer from this even more because they only use half of the pixel area when focusing, thus collecting only half of the little

light available [30]. Furthermore, the points to focus on in the image are fixed for phase detection as the system can only focus on parts that the AF sensor or focus pixels are observing. With Canons implementation, this is not as big of an issue as it enables autofocus on 80 percent of the image width and height [31]

Active autofocus techniques do not rely on image data to determine the in-focus lens position. They aim at measuring the distance between the subject and the camera system, which enables calculation of the correct lens position. A signal transmitter and sensor are required for the hardware part. The former sends a signal that gets reflected back by the subject to capture. The latter picks up the reflected signal and interprets the changes to calculate the distance to the subject. The signal used may be e.g. ultrasound pulse, infra red light or laser. [1, p.240] This method enables focusing speed similar to phase detection as the in-focus position is also obtained with a single measurement, but performs better in low-light conditions. However, focusing will not work trough transparent objects, such as windows.

4. FRAMEWORK FOR SOFTWARE-BASED AUTOFOCUS

An autofocus framework is a system that gets the last exposed image as input and based on its characteristics decides where to move the lens in order to focus the region of interest in the scene. This chapter presents the software-based implementation that enables flexible autofocus metrics processing and an algorithm to find the in-focus lens position.

4.1 Target Environment

Replacing a dedicated piece of hardware with a software implementation is not an easy task. Especially when the subject to replace is a processor that consumes a lot of data. For this reason a target device was selected to be from the high end of smart phones with high processing resources, namely the Nokia Lumia 1520.

4.1.1 Hardware

The target hardware for the auto focus system in this work, Nokia Lumia 1520, is equipped with a Qualcomm Snapdragon 800 quad-core processor running at 2.2 GHz clock speed [32]. This is the most important piece of hardware as the attempt of replacing a dedicated ISP using a general purpose CPU is not an easy task. Other relevant specifications in the scope of this work are the 20 mega pixel camera and the full HD display of resolution 1920x1080 [32].

Parallelism is a driving force in computer design and data-level parallelism (DLP) is one of the types used in applications. A more specific category for DLP is single instruction multiple data (SIMD). SIMD instructions perform the same operation for multiple data in parallel as illustrated in Figure 4.1. [33, pp.9-10] Mobile devices are attractive targets for utilizing SIMD operations because of their image and sound processing needs and the potential energy efficiency resulted from less instruction fetches per data. Also one big advantage is the simplicity for the programmer who may continue to think sequentially while achieving parallelism. [33, p.262]

Snapdragon 800 implements the ARMv7 instruction set therefore also supporting the NEON extension [34]. ARM NEON is an SIMD instruction set extension technology that operates on 64-bit, doubleword, and 128-bit, quadword, data vectors. It

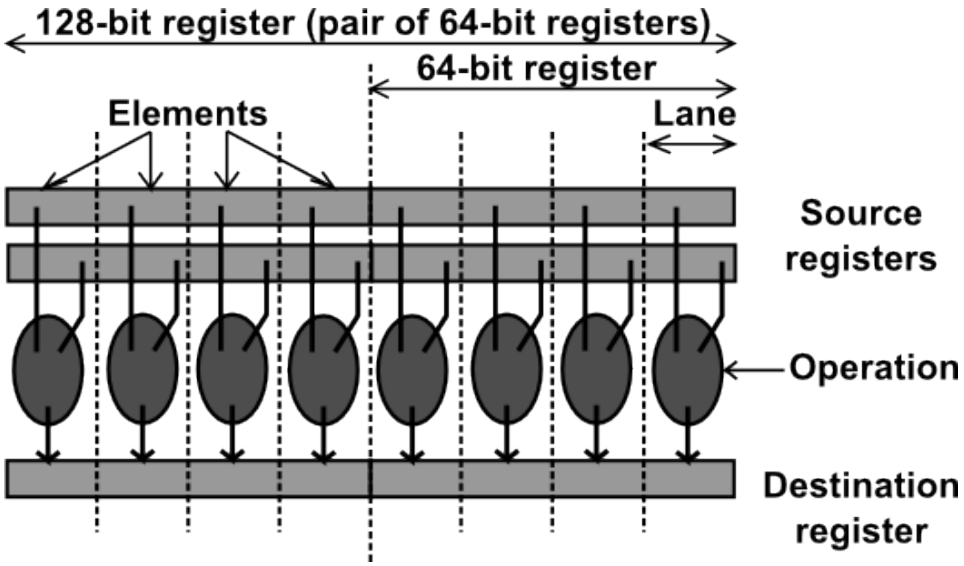

Figure 4.1: Neon instruction set architecture single operation diagram.

supports 8-bit, 16-bit, 32-bit and 64-bit signed and unsigned integer data types as well as 32-bit single-precision floating point elements and polynomials of 8-bit and 16-bit. NEON has a register bank consisting of 32 64-bit registers which can also be used as pairs resulting in 16 128-bit registers. [35]

The NEON instructions provide load, store and data processing operations that may be used in three different ways. The most straightforward method is to let the compiler optimize the code. This automatic vectorization simply needs to be switched on in the compiler. The code remains very portable using this method. However, in order to recognize the parts that can be parallelized, the compiler might need hints added to the code, such as implying that a loop is run a certain multiple of times. The most explicit way of utilizing NEON is to write inline assembly code. It has the highest potential optimization-wise, but the portability is reduced as assemblers may use different syntax. The third method, intrinsics, places between the former two in terms of complexity, portability and lines of code. NEON intrinsics appear in C or C++ code as simple functions and data types, but will be replaced with lower level instructions during compilation. Although the usage is similar to inline assembly, it leaves room for the compiler not only to further optimize the generated low level code. Compiler also handles the register allocation and interlock issues when intrinsics are used. [35]

4.1.2 Software

The Nokia Lumia 1520 runs Windows Phone 8 operating system [32]. As described in Section 2.2, the camera software is responsible for controlling the image capture process. For every frame exposed on the image sensor the ISP calculates statistics.

AWB, AEC and AF algorithms receive these statistics as inputs and decide the tunings for the next frame according to them.

The statistics for each frame are available after the exposure at the algorithm run time. The ISP provided AF metrics include pixel sums for each colour channel: red, green and blue. The input format is the image data reduced to a grid of size around ten times ten with a little room for configuration. One cell in the grid corresponds to a certain area in the image and pixel sum is simply a sum of pixel values on that area. Estimated image gradients are also input to the autofocus algorithm. They are in a similar format with a grid cell containing a sum on the corresponding area. Gradient intensities are calculated as described in 3.2.2 using a filter of which coefficients may be set, but has little configurability in dimensions.

The amount of frames per second used depends on the lighting conditions. With sufficient illumination, it is possible to keep the exposure time short and maintain 30 fps. In low-light situations it is necessary to increase the exposure time and drop the frame rate even down to 10, respectively. The frame rate does not affect the capability of the ISP to always calculate fresh metrics for each frame.

The autofocus system has only downscaled viewfinder frames available instead of the full sensor image. When the frame is available, the system will be notified and pointed the memory location of the frame data. In addition, the frame is in YC_BC_R colour space in NV12 format. YC_BC_R presents pixel as three values. Y component contains the luminance information, C_B the blue-difference and C_R the red-difference, respectively. NV12 is an image with 8-bit Y value plane followed by another plane of interleaved C_B and C_R values with 2x2 subsampling. From the autofocus point of view the luminance plane is the most interesting piece of data as it essentially contains the contrast information.

4.2 Requirements

A software-based autofocus framework is here defined as an implementation of an autofocus system that uses focus values to find the in-focus lens position. It does not define the process used for obtaining image gradient intensity values, but merely provides a setting for CPU calculation of the AF metrics. The methods for processing the gradient intensities and focus value calculation are no considered as part of the framework and should thus be modifiable with minimal effort. After all, configurability was one of the motivators for this work.

The challenge and the goal of the framework is to replace the ISP in autofocus metrics calculation. The only input data available for the framework, containing the contrast information, is the NV12-formatted viewfinder frame. Furthermore, it is rational to discard the C_B/C_R plane data for the contrast information is already isolated in the Y plane. As this radically differs from the input of ISP, it is not

reasonable to require the exact same autofocus metrics output either. The requirement for the output is set to be the pixel sums and gradient intensities calculated from the luminance plane. However, the format of the metrics is simple to match with the ISP. Output grid size is set to be an arbitrarily chosen 14 times 14 for both pixel sums and gradient intensities. These output requirements are specific for the metrics calculation unit. The framework's output consists of new lens position and the state of the system that is one of the following: idle, searching, focusing succeeded or focusing failed.

The framework will receive the frame data input only as a memory pointer. That data is not guaranteed to be valid for long as the memory will eventually be overwritten with following frames. Therefore, it is required that the data is copied to another memory location for use.

Even though the framework's calculation process has significantly less input data compared to the ISP, it is expected to be computationally very heavy operation. Nevertheless, it should not affect the performance of the rest of the camera system. This is a requirement of very high priority.

4.3 Implementation Overview

The size of the focus window is set to be half of the image height and half of the width resulting in a quarter of the image pixels. Focus window is located in the centre of the image by default. This not only further reduces the amount of input data, but is also an opportunity for optimization. There is no need to copy the entire input frame as most of it will not be used at all. However, ten extra pixels from every edge of the focus window will be copied. This is done to simplify the likely convolution operations to be performed on the data. This way the entire focus area may be processed in the same way without the need to behave differently near area edges as part of the filter is free to cross them.

Even though the new autofocus system has significantly less data to process compared to the ISP, the calculations were expected to take a lot of processing time. First, the possibility of calculating the metrics inside the autofocus algorithm was tried out with a simple prototype that convoluted the frame luminance plane using the Sobel operator within in the existing autofocus system. The prototype was successful and the algorithm managed find focus using these software calculated metrics. This also proved that the limited amount of viewfinder data was sufficient to be used in focusing. Nevertheless, already with this simple calculation method the run time of the autofocus algorithm had increased dramatically. The run time had also a lot of variation, probably due to execution of some non-camera related background process. Blocking the execution of autofocus and other camera tuning algorithms for too time long may cause undefined behaviour in the camera system

and thus it was realized that this approach was viable only when using very simple calculations.

One requirement set for the framework was that it should have minimal effect on the performance of the rest of the system thus the camera tuning algorithms should not get blocked. The metrics calculation process was decided to be done in the background on a separate thread that would merely be controlled by the focusing algorithm. Due to this approach the autofocus algorithm will not have the up-to-date metrics at its disposal as it is run only between frames. This lag of at least one frame in the availability of the metrics will have to be taken into account on autofocus algorithm side. This problem is discussed in more detail in section 4.4.

The background calculation thread will have the entire time between frames to process the data. As stated in 4.1.2, the viewfinder fps varies in correlation with the exposure time. At 30 fps it means that the metrics calculation process would have 33 ms time to complete the operation and at 10 fps the available time increases up to 100 ms, respectively. However, the quick prototype already showed that the time does not remain constant. Background thread also resolves that issue because the algorithm may check whether the calculation is complete or not. In case the results are not available the frame may simply be skipped. Moreover, this conditional skipping also virtually enables the metrics calculation to use as much time as needed. Though, as explained in 3.2 the unnecessary frame skipping is not advised.

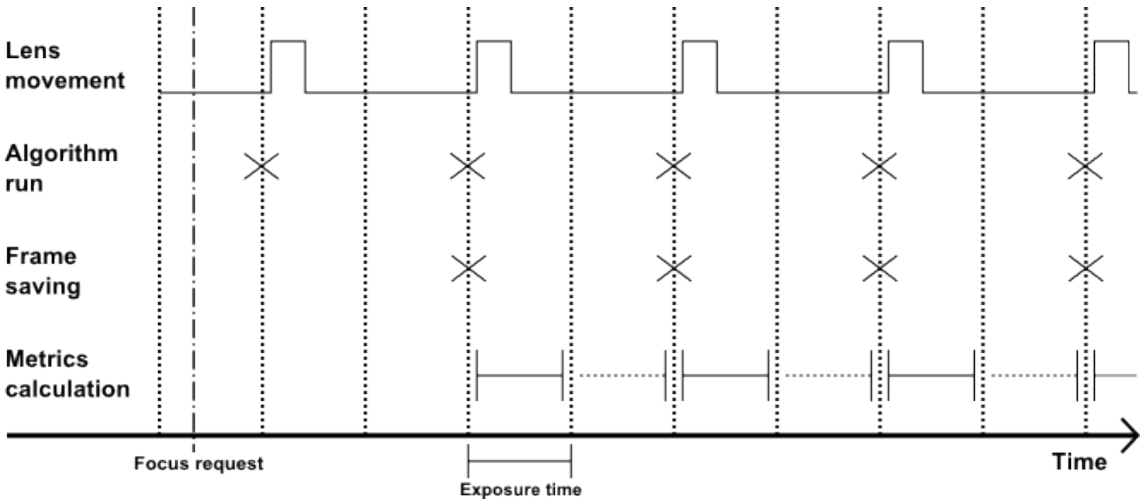

Figure 4.2: Autofocus framework timing diagram.

Timing model of the autofocus framework is illustrated in Figure 4.2. It depicts the ideal synchronization scheme between lens movement, algorithm execution, frame data saving and metrics calculation. The lens movement time is not in correct proportion compared to the exposure time as both may vary. More importantly, as can be seen in the figure, lens is moving during the exposure, especially with short exposure times, causing motion blur to the frame and possible mismatch between

lens position and focus value thus rendering it useless for metrics calculation. This mandatory frame skipping has a positive effect on the frame data processing as it doubles the time available. This extra time is depicted by the dashed line in the figure. It is also worth noting that the frame saving operation being placed in between exposures applies to the preceding frame.

The requirement about luminance pixel sums is easy to meet. It consist only of additions to a corresponding output grid cell. This operation was expectedly realized to be very light compared to the prototyped gradient calculations. As a result not much attention needs to be paid for the pixel sum requirement.

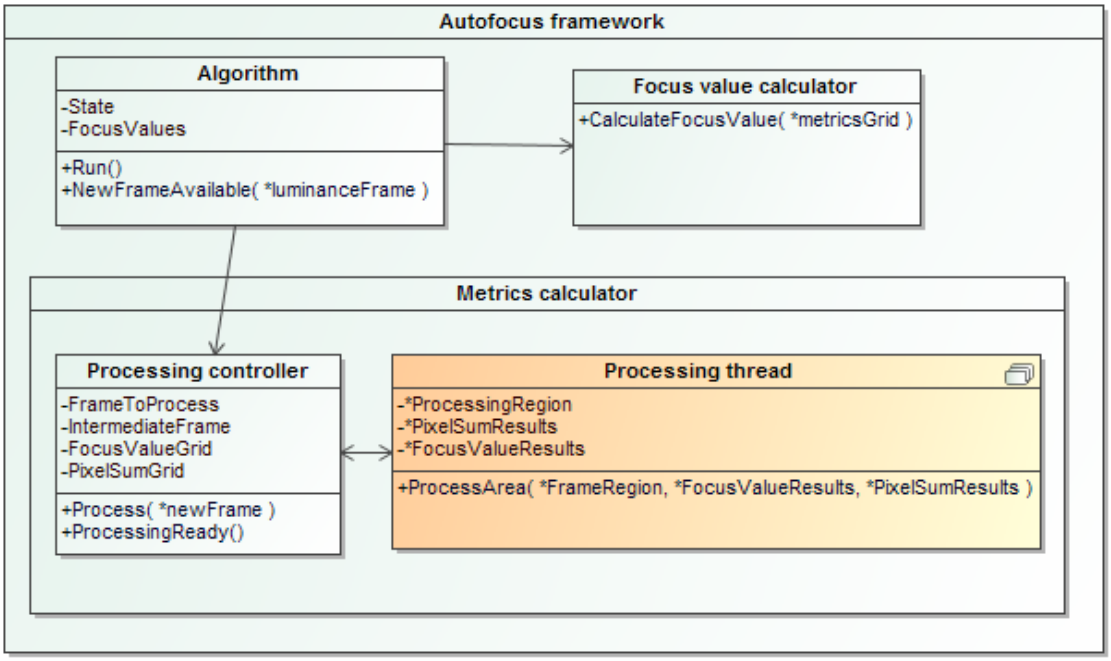

Figure 4.3: Architecture of the implemented autofocus framework.

A simplified architecture of the implemented autofocus frameworks is depicted in Figure 4.3. The components are explained in detail in the following sections. In general level, the system structure matches the contrast-based focusing steps presented in Section 3.2. That is, it is composed of three parts: algorithm, metrics calculation and focus value calculation.

4.4 New Autofocus Algorithm

The autofocus algorithm already in place in the target system utilizes the ISP calculated metrics. They enable it to make lens movement decisions according to the focus value after every frame. With the new calculation model presented in 4.3 the algorithm would need modification in order to handle the lag in the availability of the metrics. The simplest way to resolve that issue would be to skip a frame. However,

it would significantly lengthen the focusing sequence thus reducing its reliability as explained in 3.2.1.

To overcome the need for unnecessary frame skipping, a new autofocus algorithm was implemented. The target was not to create a comprehensive new system to compete with existing methods because the primary motivation lies in metrics calculation. Instead, the purpose was to make a very simple algorithm that would achieve the optimal timing scheme illustrated in Figure 4.2.

A sweep-based approach was selected. The lens movement range is divided into steps and then iterated trough, calculating the focus value at every step. This way during the sweep the algorithm always knows in advance where to move the lens next. The only situation, where the algorithm needs to wait for the results, is the last step as it will then have to decide the final in-focus position. However, it no longer has effect on the sequence for all the measurements are already done and the last one only needs to be processed.

The step with the greatest calculated focus value is assumed to be near the peak of the curve. After the sweep, focus curve is interpolated between the point preceding the greatest measured focus value and the point following it. Lens is then moved to a position with the greatest interpolated focus value as it is considered to be the peak of the curve. In case the greatest focus value is measured at the first or the last step, the interpolation part is skipped and lens moved directly to the corresponding step position. This is done because in such cases the focus peak appears to be outside the lens movement range thus the near or far end is closest to the correct position.

The logic of a single focusing sequence logic is illustrated in Figure 4.4. The algorithm waits for the focusing sequence request in idle mode. Each arrow in the figure represents an exposure, after of which there is a new frame available.

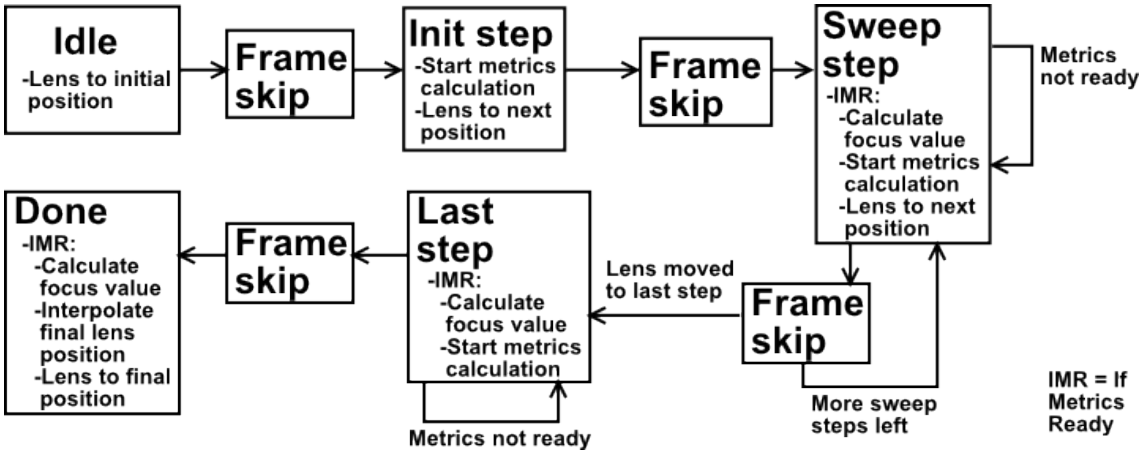

Figure 4.4: Simplified schematic of a single focusing sequence logic used in the algorithm.

Continuous autofocus mode was not considered at all. As such, the implementation only supports single-shot focusing mode. It would not even be meaningful to

use the sweeping method in continuous mode as it is slow in comparison and would be distracting in video due to the lens movement pattern. The amount of steps was set to fifteen, but the focusing sequence may be shortened by decreasing the amount. However, this will increasing the step size and reduce the accuracy.

The implemented algorithm is simple also in the sense that it does not utilize pixel sums at all. Furthermore, it is very sensitive to errors as even a single faulty measurement may cause the sequence to fail to focus properly. This happens if the error results in greatest focus value of the sweep causing a false peak to be chosen. Erroneous focus value on step preceding and following the highest value step does not result in total failure, but will affect the final position interpolation. The final accuracy depends on the used step size. There are no checks for detecting these types of failures and the algorithm will always report the sequence as a successful focusing.

4.5 Autofocus Metrics Calculation

The most important and the most challenging part of the entire autofocus framework is the metrics calculation process. As explained in 4.3 the task was decided to be run in the background, separated from the rest of the camera system. The purpose of this module is to take the focus window area of the image data as input and calculate the metrics. Results are reduced to a grid as described in 4.2.

Threading is the enabler for the metrics background processing scheme presented in 4.3. A worker thread was implemented to fill the input and output requirements of the metrics calculation. It was designed to be a simple data processing unit that simply takes the image luminance data, meta data about the plane and a pointer to result information structure as inputs. This worker then processes the data using a chosen method and saves results to the given location.

The worker thread unit was intentionally implemented as a simple data processor. This enables multiple worker threads to be used. The focus window area can be divided into smaller areas and each can be handed to a worker thread for processing. The input data is not copied separately for each thread, but remains as a whole in memory instead. The reasoning is same as for copying a larger area of the input luminance plane data than just the selected focus window. Threads may safely read image data outside of their input area an will not need to use spacial checks near edges. The target device has multiple cores and threading therefore better utilizes the hardware. Every core has a NEON unit of its own so vectorization will not suffer from the usage of multiple threads.

A controller unit for the worker threads was also implemented. It is responsible for saving the focus window area of the input luminance data and creating the metrics result structure. For processing it will initialize a number of worker threads,

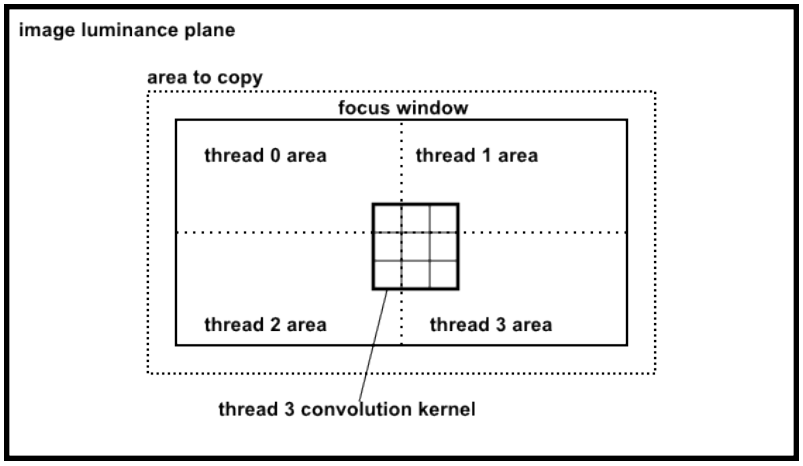

Figure 4.5: Diagram depicting the processing areas using four threads. Note that the relative area dimensions are out of proportion.

provide them addresses into the input data area and define the dimensions of the area to process. When configuration is ready, the controller will start the workers. Each worker initiates a callback when it has processed the given data. This way the controller can keep track on the whole process and reply to the algorithm querying the results availability.

Support was added for running one-, two- and four-threaded modes. The image was divided into two with a horizontal line in the middle of the image. For four-threaded processing the image was divided further with a vertical line. Figure 4.5 clarifies the relations of processing areas in four-threaded mode. It shows the area that is actually copied to the framework of the entire luminance plane input. The actual focus window is smaller than the copied area and divided for individual threads to process. Also an example of convolution kernel reading values outside the processable area is shown.

For example a convolution operation cannot store the results over the input image data. With a single processing operation it does not pose a problem as the outcome can be saved directly into the output grid. However, for multi-operation processing a secondary luminance plane buffer is needed for storing the intermediate results. This can be done by simply reserving another block of memory. Because the processing is expected to be able to read a few pixels outside its designated area, the threads need to be synchronized between processing operations. Otherwise it could happen that a thread would read values from the intermediate buffer from the area of another thread that has not yet finished the previous processing stage. Those values read would be incorrect in such case.

5. AUTOFOCUS METRICS PROCESSING AND RESULTS

The autofocus framework needs a metrics processing method. This chapter introduces the implemented techniques and the results they provided are evaluated by simulating the process. Also the optimization applied to them are explained. The combinations of implemented processing methods were run on a mobile device and the execution times were monitored. These results are presented in the end of this chapter.

5.1 Analysis Target

The implemented autofocus framework provides a very flexible setting for metrics calculation, but does not define a specific method to be used. The system is not complete without one and thus the processing steps need to be implemented. Several techniques exist for calculating the metrics. For a comprehensive analysis on what would be the most suitable method, an extensive study should be conducted with several processing methods, scene contents and real life use cases. However, it is outside the scope of this thesis. The goal is not to find the most suitable method to be used, but to explore the processing capabilities of the framework. This was done by implementing a few techniques that differ in the heaviness of the needed calculations.

The quality of the calculated metrics needs to be as good as possible. More precisely, the optimal metrics processing methods would provide a focus curve that has a single clear peak point. Focus values will inevitably be affected by noise resulting in jitter that needs to be ignored. Still, the relative differences between measured focus values should also be great enough not to be interpreted as jitter and for confident determination of the in-focus lens position.

On the other hand, the framework enables virtually unlimited calculation time due to the ability of skipping frames until the metrics are available as explained in Section 4.4. Yet, deliberately lengthening the focusing sequence is not practical as it is likely to cause errors in real life use cases. For that reason the processing time is considered to be a limiting factor. For optimal performance the calculations should not take longer than the exposure of two consecutive frames. That is, if the processing time does not exceed 66 ms, the system will work with optimal timing,

even with the maximum frame rate of 30.

Software-based metrics calculation also provides a lot more flexibility compared to hardware ISP that has limited configurability. Improving the focusing performance in low-light conditions with additional processing is an interesting subject. Longer exposure time will enable more processing to be done. This topic is addressed in Section 5.5.

5.2 Tools for Analysis

In order to test the autofocus framework and the implemented processing methods on the target device, a testing arrangement set-up in a studio was used. The object scenery was a layout of various objects placed into a cabinet. The testing scene is presented in Figure 5.1. The studio was fully darkened and the cabinet, where the scene was set-up, was equipped with adjustable lights. Lux meter is an apparatus that measures lumen, the amount of light, falling on a square metre. Such device was also placed into the cabinet to measure the amount of light illuminating the test scene.

The camera phone was attached to a stable stand to eliminate movement during capture. This very controlled environment also enabled testing in varying lighting conditions. Motionless and highly detailed scene combined with stationary camera system does not match well with real life use cases, but again it provides a good setting to evaluate the system as such, for motion-related random factors get eliminated.

Figure 5.1: The autofocus framework testing scene used in the studio.

The input data of the new autofocus framework significantly differs from the input of the ISP. A Windows Phone 8 application was modified to do a similar sweep across the lens movement range than the focusing algorithm described in Section 4.4 does. For every step of the sweep the application saves the luminance plane of the viewfinder image into the phones mass memory from where it can be copied on to a desktop computer. The data may then be used to better analyse the operation of the metrics processing part of the framework.

A single sweep recorded by the application consists of 24 images. Multiple sets were captured under varying lighting conditions using this application. Capture with the highest luminance was measured to be 2800 lx. The lowest lux reading achieved with the adjustable lights was 4.3 lx. One image set was also captured with the cabinet lights turned off and the darkening curtains slightly opened. This combination only allowed so little light to the scene that the lux meter was not able to get a reading. For comparison, Table 5.1 lists approximations of luminance values under different natural scenes.

Table 5.1: Approximate luminance values under natural scenes [36, p.70].

Sky Condition	**Illuminance - lux level (lx)**
Direct sunlight	1-1.3 x 10^5
Full daylight (Not direct sunlight)	1-2 x 10^4
Overcast day	10^3
Very dark day	10^2
Twilight	10
Deep twilight	1
Full moon	10^{-1}
Quarter moon	10^{-2}
Moonless, clear night sky	10^{-3}
Moonless, overcast night sky	10^{-4}

MathWorks® MATLAB® is a popular high-level programming language as well as an interactive environment. It can be used for data analysis and visualisation, numeric computation, algorithm development and creating models and applications. With the help of in-built functions and tools it is possible to try out different approaches and solutions faster than by using traditional programming languages. [37]

MATLAB was used to create a script that can be used to test the output of the autofocus framework metrics processing. The script takes the luminance planes captured by the phone application as input. Each image is processed and the focus value calculated. Finally the measurements are visualised as a focus curve. This allows easy modification of the processing methods and result comparison side by side on a desktop computer.

The phone may be connected to a debugging software run on PC. From the

autofocus framework it is possible to print messages to be shown on the debugger. Processing time of the steps was measured inside the metrics processor code by comparing the system time before beginning the processing and after it was completed. The measurements were printed out from the code to the debugger with accuracy of one microsecond.

5.3 Optimization

The most efficient way of optimizing the metrics processing is the NEON extension of the ARM instruction set implemented by the target device processor. This is because it increases parallelism in each of the processing threads. Jang et al. have studied the performance of NEON technology by applying optimization to several open source applications in [38]. They used the automatic vectorisation option of applying NEON code. The results for execution time improvements were varying and not very significant.

Image processing is an easily parallelized operation. Calculation of one pixel does not affect the calculation of others, which allows multiple pixels to be processed at the same time. Welch et al. studied how much bilinear interpolation and a distortion algorithm would benefit from NEON optimization and achieved very good results [39]. As a result, the execution speed of the algorithms was between two to three times higher than the original reference implementation. Similarly significant speed-ups has been obtained in digital image stabilisation on a mobile device [40]. Also video encoding and decoding has been shown to benefit from using NEON [41].

As processing time is the critical factor to be considered in the autofocus framework, the target was to optimize the operations for speed. Efficient use of the processor features may also introduce power saving as a by-product, which would be very beneficial. However, power usage was not monitored.

The metrics processing basically consists of applying one or more convolutions operations to the image data. These operations are easily parallelized an thus all the convolutions implemented were generally optimized in the same way. Pixels are presented in the input data as 8-bit unsigned integer values in consecutive memory addresses. For each coefficient in the convolution kernel, a separate NEON register was used. This enabled eight values to be read efficiently into a 64-bit register. The register was then expanded to 128-bit, in order to prevent the values from overflowing.

Zero coefficients were ignored completely and not even a register was allocated for them. Next the values in each NEON register were multiplied with the absolute value of the corresponding coefficient. For the result pixels, two separate NEON registers were used. Both of them were set to be 128-bit, containing four signed 32-bit values. The multiplied values were either added to the result vector or subtracted

from it according to the corresponding coefficient sign. Finally, the result vectors were written to the destination memory block.

This method enabled the theoretical speed-up for the reading and multiplication part to be eight because the signs were ignored at first, allowing more values to fit into the 128-bit register. For the addition and subtraction part of the convolution, the theoretical speed-up was four times. This is because the operations needed to be done in two parts, for only half of the multiplied values fit into to a single result register.

When the focus operator consists of two or more kernels, as do all of those presented in Section 3.2.2, the calculation of the final gradient values requires Equation (3.1). This involves a couple of heavy calculations and thus the following approximation is often used:

$$G = \sqrt{G_1^2 + G_2^2 + \cdots + G_n^2} \approx |G_1| + |G_2| + \cdots + |G_n|, \tag{5.1}$$

which eliminates raising to the power of two and square root operations by utilising absolute values. [42] This approximation equation was applied to every implemented focus operator.

5.4 Focus Value Calculation

Three focus operators with very different calculation needs were implemented. The lightest method was Roberts cross presented in Section 3.2.2. A bit more demanding operator is the Sobel, as it has more coefficients that also need multiplication operations. The heaviest method implemented was extended Sobel operator. In addition to horizontal and vertical directions, it also measures both diagonal directions like Roberts cross. The extension adds two more kernels to be used:

$$G_{d1} = \begin{bmatrix} 0 & +1 & +2 \\ -1 & 0 & +1 \\ -2 & -1 & 0 \end{bmatrix} * A \; and \; G_{d2} = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & +1 \\ 0 & +1 & +2 \end{bmatrix} * A$$

For calculating the focus value out of the gradient approximations the following

weighting matrix was used:

$$\begin{bmatrix} 0 & 0 & 0.5 & 1 & 0.5 & 0 & 0 \\ 0 & 0.5 & 1 & 2 & 1 & 0.5 & 0 \\ 0.5 & 1 & 2 & 3 & 2 & 1 & 0.5 \\ 1 & 2 & 3 & 3 & 3 & 2 & 1 \\ 0.5 & 1 & 2 & 3 & 2 & 1 & 0.5 \\ 0 & 0.5 & 1 & 2 & 1 & 0.5 & 0 \\ 0 & 0 & 0.5 & 1 & 0.5 & 0 & 0 \end{bmatrix}$$

It applies a very heavy emphasis on the centre of the focusing area. The actual result of the metrics processing is 14 times 14 grid, so this weight matrix needs to be used by using the same coefficient for a total of four values. Same weighting was used in all cases presented in this chapter.

With plenty of light, contrast-based focusing works very well. Therefore, it is more meaningful to study the results in low light. From the captured image sets, three were chosen to be presented here: **1700, 4.3 and <4.3 lx**. The situation with most illumination corresponds daylight. The second set had already relatively dark capturing conditions placing in twilight under natural scenes. The third set is the one captured with the cabinet lights turned off and opening the studio curtains slightly. See Table 5.1 for approximations of the luminance values under natural scenes.

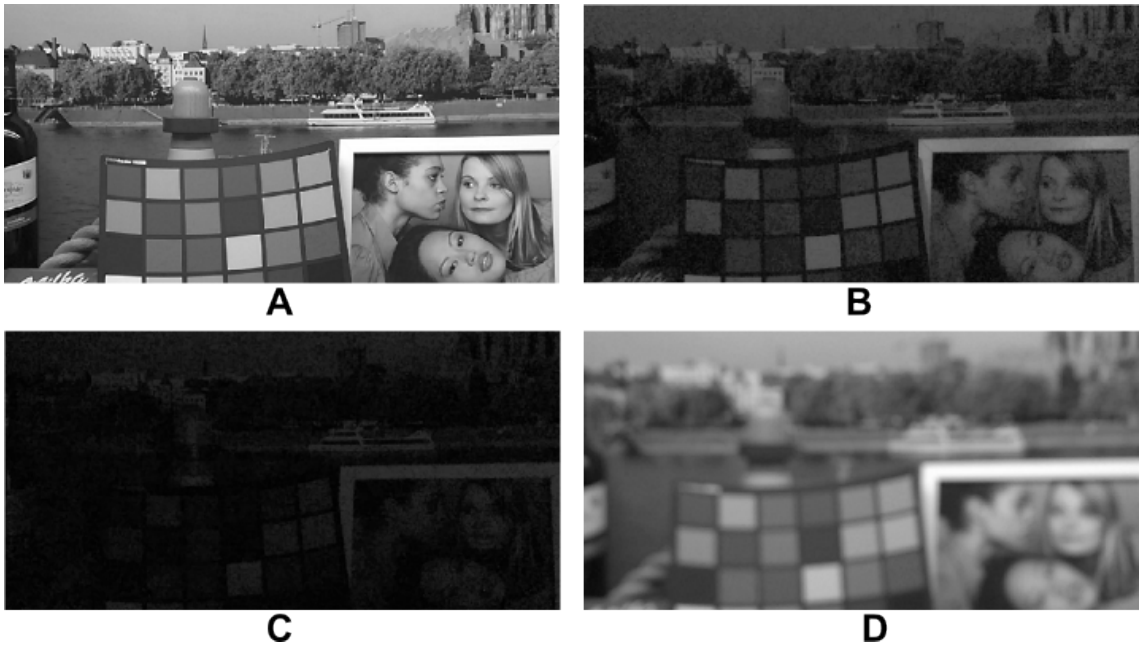

Figure 5.2: Original focusing area of the images near in-focus position under 1700 lx (A), 4.3 lx (B) and below 4.3 lx (C). (D) shows the 1700 lx focusing area with lens set to infinity.

The input to the metrics processing under varying illumination conditions is illus-

trated in Figure 5.2. Three of the images are selected from the image sets from near the in-focus position. From the fourth image one can see the amount of blur when the lens is very far from the correct position. While most details are still visible in **B**, the amount of noise has increased notably. In **C** some details are barely visible and noise is very dominant in some parts of the image.

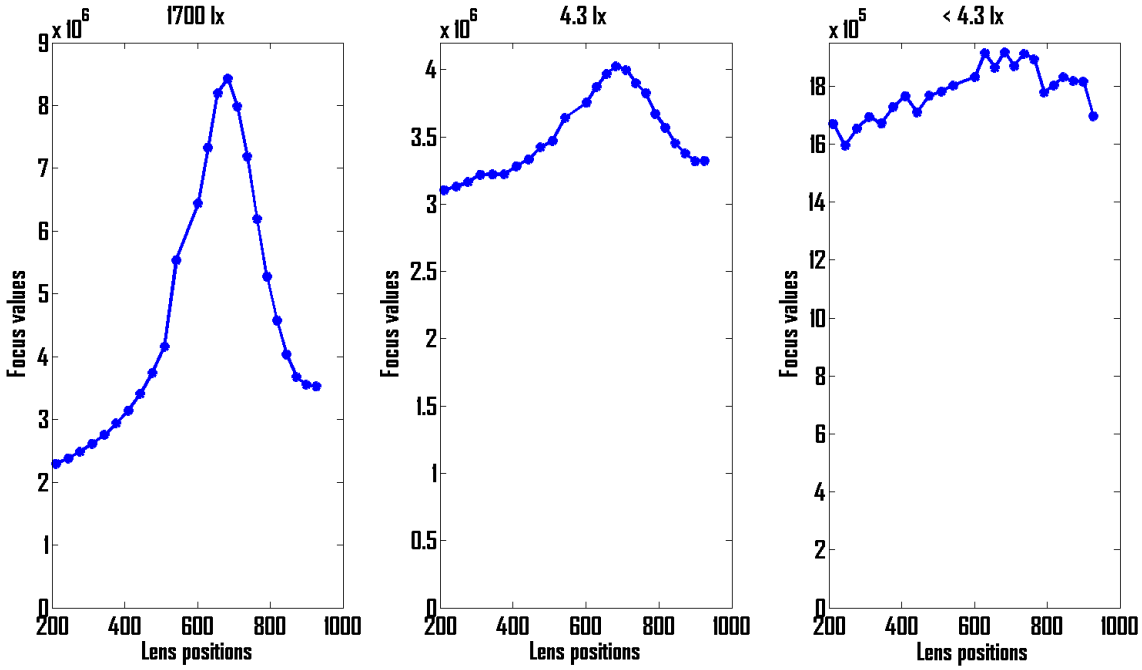

Figure 5.3: Focus curves processed with Roberts cross operator under different illumination conditions.

The MATLAB simulated results of the calculated focus values are presented as linearly interpolated focus curves in Figure 5.3, Figure 5.4 and Figure 5.5. It is worth noting that the scale of focus values on the y axis varies between each diagram. Moreover, the lens position values are relative and the scale represents the movement range.

Every operator seems to produce a very good focus curve under high illumination. The curves have a single clear peak pointing out the in-focus lens position. By looking at the results under 4.3 lx, the lack of contrast shows clearly in decreased differences of the absolute focus values. The effect of noise is also visible as the slope does not rise as steadily as under higher illumination. Nevertheless, each operator produced a curve that should result in successful focusing.

Under 4.3 lx both of the Sobel operators produced a slightly smoother focus curve than Roberts cross. However, in the sample input data captured under the least illumination noise is clearly an issue. Output focus curve of each operator is distorted with notable jitter. The general shape of the curve somewhat resembles those generated with better quality input data. Yet, by looking only the focus curves

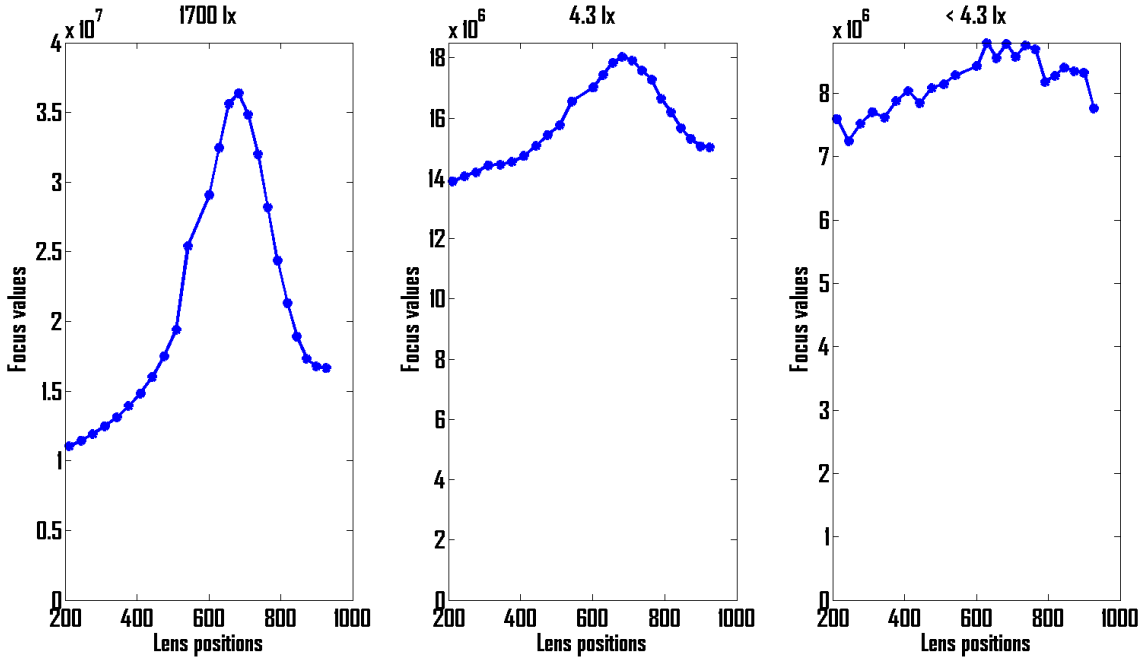

Figure 5.4: Focus curves processed with Sobel operator under different illumination conditions.

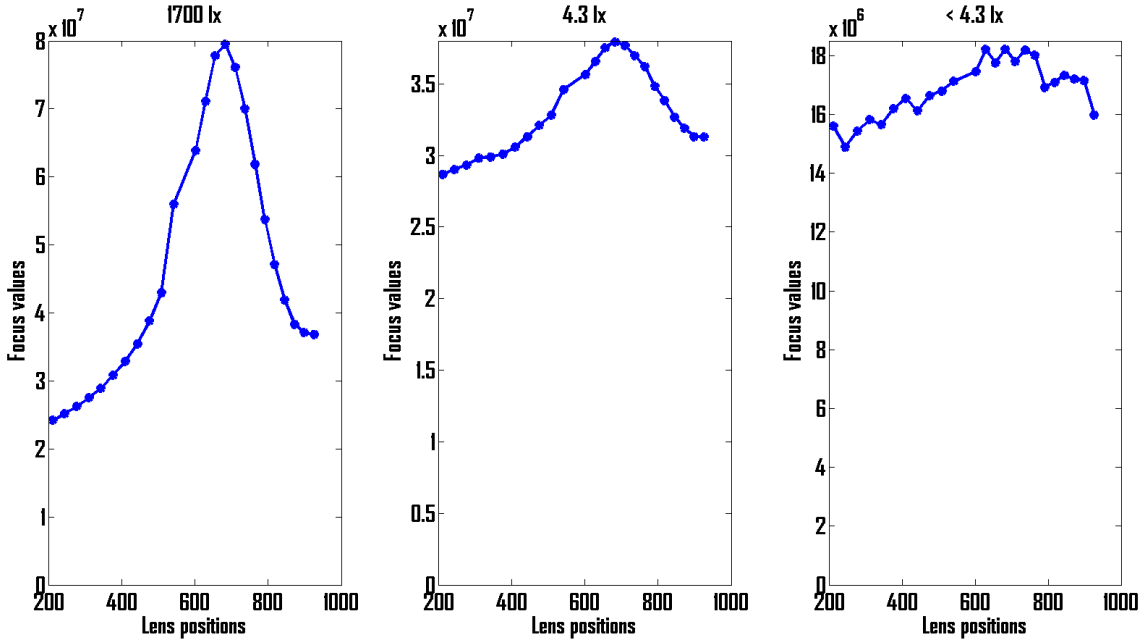

Figure 5.5: Focus curves processed with diagonal gradient detection extended Sobel operator under different illumination conditions.

calculated from low-light condition input, it is very hard to determine the in-focus lens position.

5.5 Image Preprocessing

In low-light conditions, the lack of contrast and the amount of noise becomes a problem for contrast-based autofocus system, i.e. the signal-to-noise ratios are low. Noise tolerance of existing focus operators have been studied and new ones have been proposed in [26] and [43]. These studies show that the output of focus operators vary under low-light situations and the proposed methods produce very good results in comparison. However, these techniques are outside the scope of this thesis.

Another approach to improving low-light performance of autofocus systems is that image data preprocessing should be applied before calculating the focus value. This has been studied in [44] and [45] with promising results. The goal is to improve the signal-to-noise ratio by attempting to reduce noise in the input data. To determine the best preprocessing method, a more comprehensive study would be needed, but in the scope of this thesis we are more interested in the performance of the autofocus framework. Three preprocessing techniques were implemented: **averaging filter**, **blurring low-pass filter** and **median filter**. The first two consist simply convolving the image with a filter kernel. Kernels of the two implemented methods are:

$$H_{Avg} = \begin{bmatrix} \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} \end{bmatrix} \; and \; H_{LP} = \begin{bmatrix} 0 & \frac{1}{8} & 0 \\ \frac{1}{8} & \frac{1}{2} & \frac{1}{8} \\ 0 & \frac{1}{8} & 0 \end{bmatrix}$$

The denominators are chosen to be powers of two because NEON does not support division, but the same arithmetic operation is achieved by shifting. Both filters were optimized in they way described in Section 5.3 as they are convolution operations.

Median filter fundamentally differs from the other two. It observes the values on the kernel area around the pixel to be processed and chooses the median value. This is fundamentally a heavy operation and the practical use has been limited to using small kernel sizes. There are different techniques to implement median filter and improving the efficiency by inventing better methods has been under several studies. [46] Weiss presents a viable solution in [46], but the kernel size is dependent on the implementation. More recently, Perreault et al. introduced a technique in [47] enabling median filtering in constant time, independent of the kernel size.

The median filtering method was implemented as described by Perreault et al. The optimization also differed from other implemented filters. The method utilises histograms in determining the median. Optimization methods proposed in the same study suggest the use of conditionally updated multilevel histograms and vectorization. These techniques were also applied to the implementation. A bit-depth of 8 was used in the histograms, which limits the kernel size to 16 times 16. The gain for this limitation is higher degree of vectorization that accelerates the histogram

operations. For testing the median filter kernel radius was set to 3, resulting in 7 times 7 kernel size.

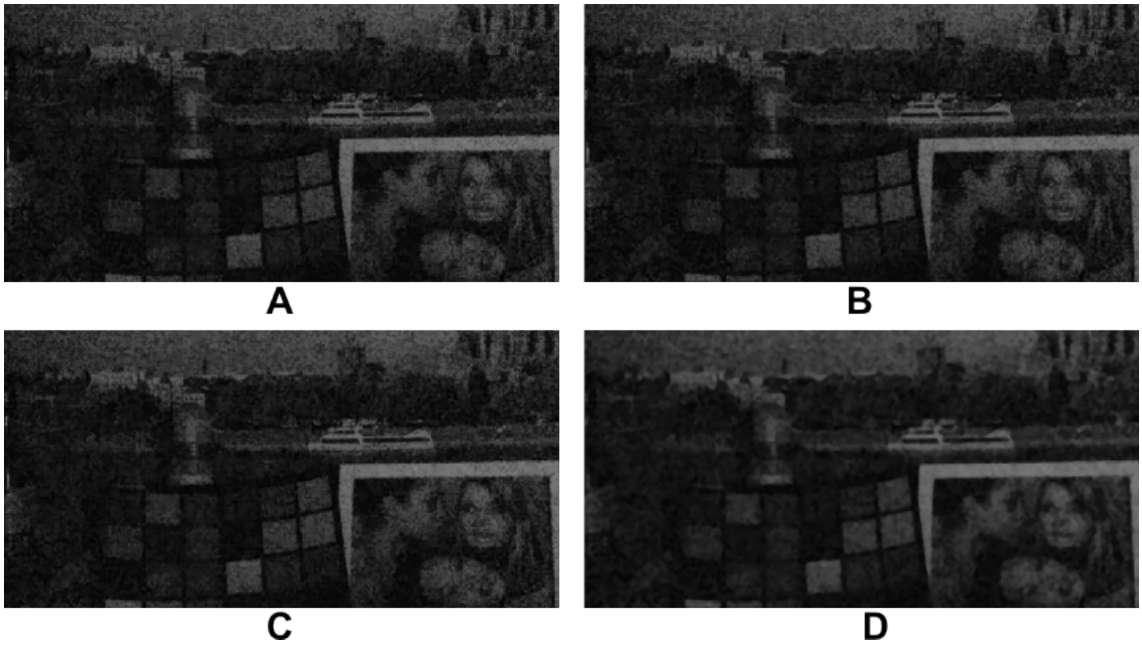

Figure 5.6: Focusing area of the image near in-focus position captured under illumination smaller than 4.3 lx. (A) is the original image, (B) has been processed with averaging filter, (C) with low-pass filter and (D) with median filter. Contrast of the images has been boosted for better visualisation of the differences.

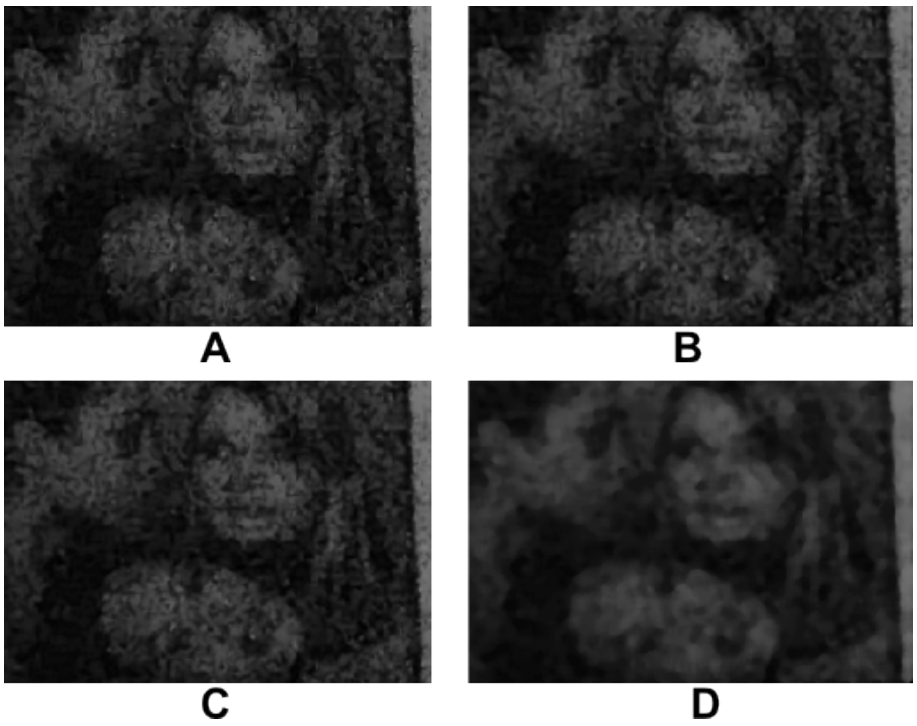

Figure 5.7: Cropped magnifications of the images presented in Figure 5.6 for closer examination of the processing technique differences.

Preprocessed input data is illustrated in Figure 5.6 alongside the original and their magnifications are presented in Figure 5.7 for closer examination. Averaging and low-pass filters have reduced noise most noticeably in smooth areas, such as the patches of the colour checker and photo frame edge. Median filter has smoothed the image the most and especially the strong edges stand out clearly.

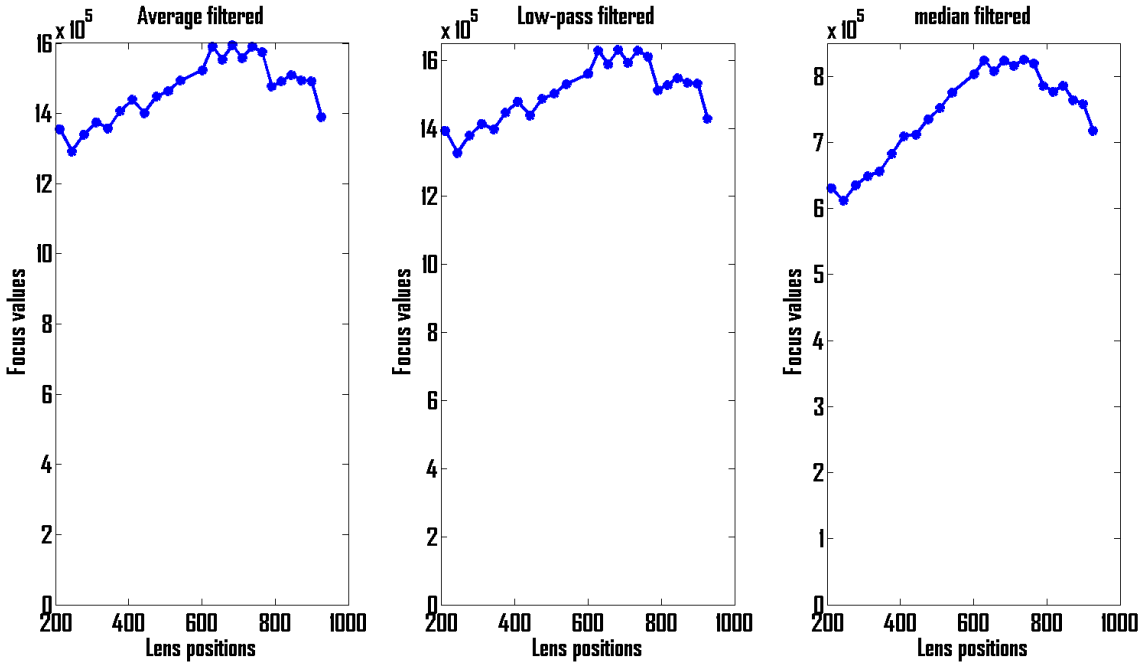

Figure 5.8: Focus curves preprocessed with averaging, low-pass and median filter and processed with Roberts cross operator under illumination smaller than 4.3 lx.

The effect of all three preprocessing methods implemented was simulated in MATLAB. Processing results of the image set captured under illumination below 4.3 lx are presented as linearly interpolated focus curves in Figure 5.8, Figure 5.9 and Figure 5.10.

There is no clear difference in the curves produced by different focusing operators, which was to be expected based on the results without preprocessing presented in Section 5.4. Averaging filter seems to have some smoothing effect on the focus curve when closely compared to its non-preprocessed counterpart. It has also increased the absolute value differences between the measure point slightly. Same applies to the result produced by the low-pass filter, but there is no noticeable difference between the two filters. It is likely that the amount of noise in the input data was too great for these two processing methods with relatively small kernels sizes to noticeably improve the result or they may perform better with different content.

Median filter produces very promising results with all three focus operators. It has significantly smoothed the focus curve and the absolute values clearly differ from each other. Even though the in-focus position still cannot be determined with certainty, the area is narrowed down to lens positions from 600 to 750. Compared to

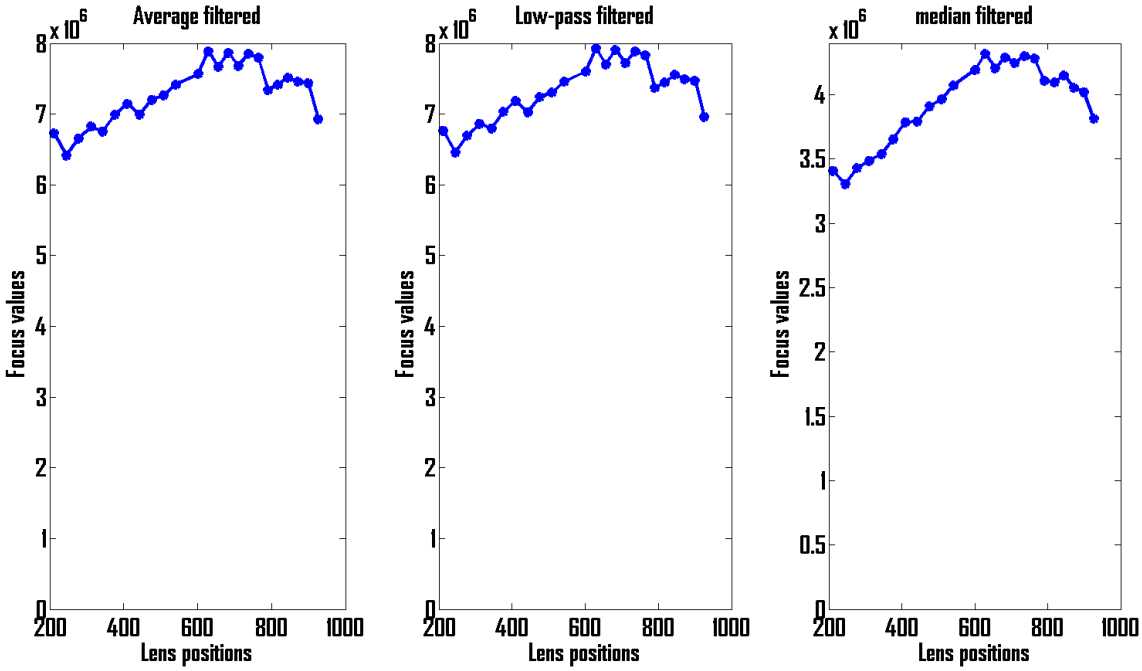

Figure 5.9: Focus curves preprocessed with averaging, low-pass and median filter and processed with Sobel operator under illumination smaller than 4.3 lx.

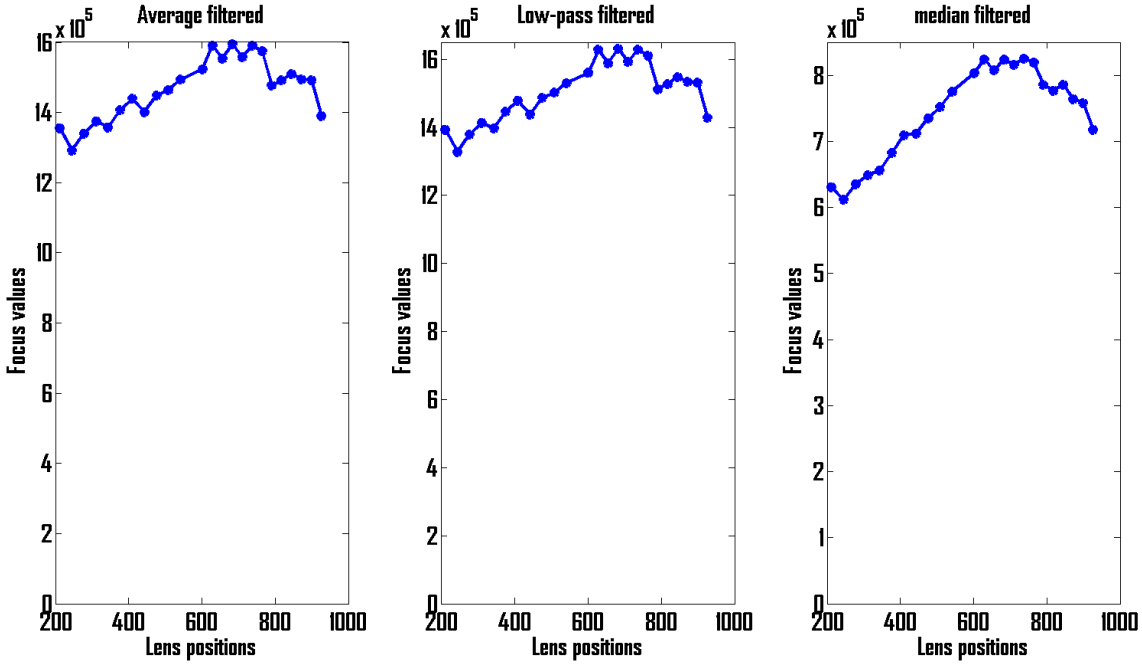

Figure 5.10: Focus curves preprocessed with averaging, low-pass and median filter and processed with diagonal gradient detection extended Sobel operator under illumination smaller than 4.3 lx.

the other two filters, this method is likely to benefit from the notably greater kernel size.

5.6 Processing Results

The implemented autofocus framework was run on the target device attached to the debugger to monitor processing times of the autofocus metrics. Every implemented focus operator was first tested on the device without preprocessing. Every combination of preprocessing filter and focus operator was also tested.

Table 5.2: Measured processing time intervals of the implemented focus operators and their combinations with preprocessing filters. The times are reported in milliseconds (ms).

Focus operator / **Preprocessing**	**Roberts cross**	**Sobel**	**Extended Sobel**
None	4..12	8..15	12..22
Averaging	7..15	11..19	13..25
Low-pass	8..16	12..20	14..27
Median	24..50	28..56	30..59

Processing time interval measurements of the focus operators and their combinations with preprocessing filters are presented in Table 5.2. As can be observed from the table, the processing time fluctuated significantly within the used method and was at worst even more than doubled compared to the fastest measured time. The fluctuation can be explained by the operating system running variable amount of other tasks in the middle of metrics processing. Also heating is likely to cause the device to lower the operating frequency of the processor at some point. However, no rise in processing times over time was noticed during measurements.

Despite the notable fluctuations in processing times, every combination managed to provide autofocus metrics within the set 66 ms time limit. Median filtering clearly takes the most time of the implemented methods and it is likely to exceed the time limit under some circumstances. Nevertheless, it is not an issue if occurring occasionally because the autofocus framework simply spends an extra frame for the sequence.

6. CONCLUSIONS

In this thesis a fully software-based contrast autofocus framework was introduced and implemented. It provided a very malleable setting for calculating the metrics used for finding the in-focus lens position on background running software instead of using the data provided by the ISP. The framework also included a simple focusing algorithm to utilise the calculated metrics. Three focus operators were implemented and their performance evaluated by simulating the output on a desktop PC. To truly test the performance of the framework, three preprocessing methods were also implemented with low-light focusing performance improvement in mind. The output of focus operators was also evaluated by simulation when using the preprocessing techniques. Finally the implemented methods were run on the autofocus framework on a real device and the processing times were monitored to determine whether the performance was efficient enough or not.

The conclusion on the autofocus framework performance was that the processing is possible to be done by the software run on a general purpose CPU. The downside is that the operation introduces a lag of at least one frame and thus the focusing algorithm needs to adapt to that. However, the technique allocates enough time for extra preprocessing to be done to the input data. With the help of preprocessing the low-light focusing performance can be improved, because the better quality of the metrics. All the implemented processing methods proved to be efficient enough to be practical in the framework. Although median filtering was increasing the processing time significantly and is likely to be unusable on devices equipped with more modest processors.

To further study the applicability of the framework, more test runs should be conducted on a broad range of devices. Also the processing methods need to be evaluated more thoroughly in order to determine the most suitable combination for achieving the best focusing result on the given input data. This study should include varying scenes in real life situations where cameras are used. This would also enable the possibility to switch the used method on-the-fly between focusing sequences to be more fit for the conditions.

The implemented autofocus framework has also room for improvement. The overall configurability, e.g. focus window and the output grid sizes, are fairly easy targets for development. Simplicity of the focusing algorithm is a definite weak spot

for the system to be reliable enough to be used in products. The algorithm would need features for analysing the measured focus values so that a single faulty value does not result in failure of the sequence. Another option would be to adapt an existing autofocus algorithm to understand the lag in the availability of the metrics.

Another interesting possibility is to add other types of analysing steps to the metrics processing. For example a bright spot detection can be used to modify the focus window size and shape so that unwanted regions are excluded from the calculations. All in all, the software-based implementation enables various opportunities for extracting information from the input data, which can be used to assist the focusing process.

BIBLIOGRAPHY

[1] Nakamura J. Image Sensors and Signal Processing for Digital Still Cameras. Boca Raton, FL, USA: CRC Press, Inc.; 2005.

[2] Kalevo O. Advanced Camera & Optics; 2008. Nokia internal training material.

[3] Nummela V. Camera Lenses; 2008. Nokia internal training material.

[4] Bazhyna A. Image Compression in Digital Cameras [Ph.D. dissertation]. Tampere University of Technology; 2009.

[5] Aptina Imaging Corporation. An Objective Look at FSI and BSI. Aptina Imaging Corporation; 2010.

[6] Sharma G. Digital Color Imaging Handbook. Boca Raton, FL, USA: CRC Press, Inc.; 2002.

[7] Foveon Inc. Website;. [WWW] Cited: March 18, 2014. Available from: `www.foveon.com`.

[8] Theuwissen A. Digital Camera Systems; 2013. Handout of CEI-Europe Corporate-Exclusive Course for Nokia.

[9] Liu CS, Ko SS, Lin PD. Experimental Characterization of High-Performance Miniature Auto-Focusing VCM Actuator. Magnetics, IEEE Transactions on. 2011 April;47(4):738–745.

[10] Ren L, Lee RH, Park HR, Ren H, Nah C, Yoo IS. A Liquid Lens Driven by Bubble Actuator. Microelectromechanical Systems, Journal of. 2013 Oct;22(5):1222–1228.

[11] Ramanath R, Snyder WE, Yoo Y, Drew MS. Color image processing pipeline. Signal Processing Magazine, IEEE. 2005 Jan;22(1):34–43.

[12] Chatterjee P, Milanfar P. Is Denoising Dead? Image Processing, IEEE Transactions on. 2010 April;19(4):895–911.

[13] Kalevo O. Introduction to Nokia Image Perfection System; 2005. Nokia internal training material.

[14] Land EH. The Retinex Theory of Color Vision. Scientific American. 1977 Dec;237(6):108–128.

[15] Lam EY. Combining gray world and retinex theory for automatic white balance in digital photography. In: Consumer Electronics, 2005. (ISCE 2005). Proceedings of the Ninth International Symposium on; 2005. p. 134–139.

[16] Chen CL, Lin SH. Automatic white balance based on estimation of light source using fuzzy neural network. In: Industrial Electronics and Applications, 2009. ICIEA 2009. 4th IEEE Conference on; 2009. p. 1905–1910.

[17] Kehtarnavaz N, Oh HJ, Yoo Y. Development and Real-time Implementation of Auto White Balancing Scoring Algorithm. Real-Time Imaging. 2002 Oct;8(5):379–386. Available from: http://dx.doi.org/10.1006/rtim.2001.0287.

[18] Ramanath R, Snyder WE, Bilbro GL, Iii WAS. Demosaicking methods for Bayer color arrays. Journal of Electronic Imaging. 2002;11:306–315.

[19] Gunturk BK, Altunbasak Y, Mersereau RM. Color plane interpolation using alternating projections. Image Processing, IEEE Transactions on. 2002 Sep;11(9):997–1013.

[20] Ramanath R, Snyder W. Adaptive Demosaicking. Journal of Electronic Imaging. 2003;12:633–642.

[21] Wang S, Lin F. Adaptive demosaicking with improved edge detection. In: Imaging Systems and Techniques, 2009. IST '09. IEEE International Workshop on; 2009. p. 198–201.

[22] Han JW, Kim JH, Lee HT, Ko SJ. A novel training based auto-focus for mobile-phone cameras. Consumer Electronics, IEEE Transactions on. 2011 February;57(1):232–238.

[23] Gamadia M, Kehtarnavaz N. A Real-time Continuous Automatic Focus Algorithm for Digital Cameras. In: Image Analysis and Interpretation, 2006 IEEE Southwest Symposium on; 2006. p. 163–167.

[24] Tsai TH, Lin CY. A new auto-focus method based on focal window searching and tracking approach for digital camera. In: Communications, Control and Signal Processing, 2008. ISCCSP 2008. 3rd International Symposium on; 2008. p. 650–653.

[25] Kehtarnavaz N, Oh HJ. Development and Real-time Implementation of a Rule-based Auto-focus Algorithm. Real-Time Imaging. 2003 Jun;9(3):197–203. Available from: http://dx.doi.org/10.1016/S1077-2014(03)00037-8.

[26] Zhao Q, Liu B, Xu Z. Research and Realization of an Anti-noise Auto-focusing Algorithm. In: Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2013 5th International Conference on. vol. 2; 2013. p. 255–258.

[27] Florea C, Florea L. A parametric non-linear algorithm for contrast based auto-focus. In: Intelligent Computer Communication and Processing (ICCP), 2011 IEEE International Conference on; 2011. p. 267–271.

[28] Mo C, Liu B. An auto-focus algorithm based on maximum gradient and threshold. In: Image and Signal Processing (CISP), 2012 5th International Congress on; 2012. p. 1191–1194.

[29] He J, Zhou R, Hong Z. Modified fast climbing search auto-focus algorithm with adaptive step size searching technique for digital camera. Consumer Electronics, IEEE Transactions on. 2003 May;49(2):257–262.

[30] Śliwiński P, Wachel P. A Simple Model for On-Sensor Phase-Detection Auto-focusing Algorithm. Journal of Computer and Communications. 2013;1:11–17.

[31] Canon. Dual Pixel CMOS AF Technology;. [WWW] Cited: April 4, 2014. Available from: `www.usa.canon.com/cusa/consumer/products/cameras/standard_display/daf_technology`.

[32] Nokia. Detailed specifications for the Nokia Lumia 1520;. [WWW] Cited: February 6, 2014. Available from: `www.nokia.com/gb-en/phones/phone/lumia1520/specifications/`.

[33] Hennessy JL, Patterson DA. Computer Architecture, Fifth Edition: A Quantitative Approach. 5th ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.; 2011.

[34] ARM. ARM Processor Architecture;. [WWW] Cited: February 20, 2014. Available from: `www.arm.com/products/processors/instruction-set-architectures/index.php`.

[35] ARM. Introducing NEON;. [WWW] Accessed on: March 3, 2014. Available from: `infocenter.arm.com/help/topic/com.arm.doc.dht0002a/DHT0002A_introducing_neon.pdf`.

[36] Photonis. Electro-Optics Handbook;. [WWW] Cited: April 10, 2014. Available from: `www.photonis.com/attachment.php?id_attachment=95`.

[37] MathWorks. MATLAB - The Language of Technical Computing;. [WWW] Cited: April 11, 2014. Available from: `http://www.mathworks.com/products/matlab/`.

[38] Jang M, Kim K, Kim K. The Performance Analysis of ARM NEON Technology for Mobile Platforms. In: Proceedings of the 2011 ACM Symposium on Research in Applied Computation. RACS '11. New York, NY, USA: ACM; 2011. p. 104–106. Available from: http://doi.acm.org/10.1145/2103380.2103401.

[39] Welch E, Patru D, Saber E, Bengtson K. A study of the use of SIMD instructions for two image processing algorithms. In: Image Processing Workshop (WNYIPW), 2012 Western New York; 2012. p. 21–24.

[40] Ha SW, Park HC, Han TD. Mobile digital image stabilisation using SIMD data path. Electronics Letters. 2012 July;48(15):922–924.

[41] Rintaluoma T, Silven O. SIMD performance in software based mobile video coding. In: Embedded Computer Systems (SAMOS), 2010 International Conference on; 2010. p. 79–85.

[42] Lin Z, lihua Cai, Yu Y, Chen T. An auto-focus algorithm based on image processing. In: Automatic Control and Artificial Intelligence (ACAI 2012), International Conference on; 2012. p. 394–397.

[43] Choi J, Kang H, Lee CM, Kang MG. Noise insensitive focus value operator for digital imaging systems. Consumer Electronics, IEEE Transactions on. 2010 May;56(2):312–316.

[44] Gamadia M, Kehtarnavaz N, Roberts-Hoffman K. Low-Light Auto-Focus Enhancement for Digital and Cell-Phone Camera Image Pipelines. IEEE Trans on Consum Electron. 2007 May;53(2):249–257. Available from: http://dx.doi.org/10.1109/TCE.2007.381682.

[45] Gamadia M, Kehtarnavaz N. Enhanced low-light auto-focus system model in digital still and cell-phone cameras. In: Image Processing (ICIP), 2009 16th IEEE International Conference on; 2009. p. 2677–2680.

[46] Weiss B. Fast Median and Bilateral Filtering. ACM Trans Graph. 2006 Jul;25(3):519–526. Available from: http://doi.acm.org/10.1145/1141911.1141918.

[47] Perreault S, Hebert P. Median Filtering in Constant Time. Image Processing, IEEE Transactions on. 2007 Sept;16(9):2389–2394.