Analysis of a pulse-based ToF camera for automotive application

Master Thesis

Simon Theiß

Computer Graphics and Multimedia Systems Group
Institute for Vision and Graphics
University of Siegen

In cooperation with the Delphi Deutschland GmbH

Supervisors: Prof. Dr. Andreas Kolb, M. Sc. Hamed Sarbolandi

27th of March 2015

Analyse einer pulsbasierten ToF Kamera für den Einsatz im Automotive-Bereich

Masterarbeit

Simon Theiß

Lehrstuhl für Computergraphik und Multimediasysteme
Universität Siegen

In Kooperation mit der Delphi Deutschland GmbH

Gutachter: Prof. Dr. Andreas Kolb, M. Sc. Hamed Sarbolandi

27. März 2015

ABSTRACT

This thesis analyzes a pulse based ToF camera for automotive applications. Methods for the calibration of several camera parameters and the distance depending error are described and executed. Also, the charge ratios of the shutters in the camera's pixels are analyzed. Furthermore, the implementation of a program that can operate the camera and a program to visualize and analyze the camera's data is described.

ZUSAMMENFASSUNG

Diese Arbeit analysiert eine pulsbasierte ToF Kamera für den Automotive-Bereich. Es werden Methoden zur Kalibrierung verschiedener Kameraparameter und zur Kalibrierung des distanzabhängigen Fehlers beschrieben und angewendet. Außerdem werden die Ladungsverhältnisse der einzelnen Shutter in den Kamerapixeln betrachtet. Weiterhin wird die Implementierung eines Programms zur Steuerung der Kamera und eines zur Visualisierung und Analyse der gemessenen Daten erläutert.

ACKNOWLEDGEMENTS

This thesis was written in cooperation with the *Delphi Deutschland GmbH* in Wiehl and Wuppertal. A special thank goes to *Diethelm Noll* and *Albrecht Haack* for their great support.

My supervisors were *Prof. Dr. Andreas Kolb* and *M. Sc. Hamed Sarbolandi*. They did a great job discussion problems with me and helped me to figure out the next steps.

I would also like to thank my *wife*, *family* and *friends* who always supported me.

EIDESSTATTLICHE ERKLÄRUNG

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Simon Theiß

CONTENTS

LIST OF FIGURES

ACRONYMS

ToF Time of Flight

API Application Programming Interface

ns nanoseconds

fps Frames per second

FWHM Full width at half maximum

1

INTRODUCTION

The *2015 International CES* in Las Vegas was used by Delphi, Mercedes, BMW and Audi to present innovative concepts of self driving cars. Volkswagen showed a new kind of cockpit that can be operated with hand gestures. [Wit15] [HW15] [Cun15]

For these applications, reliable optical sensors are needed to observe the surrounding of the car by measuring the distance to upcoming objects. For this radar or lidar object detection systems, time of flight cameras (ToF) or conventional CCD cameras can be used. CCD cameras do not deliver any distance information, nevertheless they can be useful if object shapes and colors are detected using image processing algorithms. Radar and lidar systems have only one sender and receiver and must use a mechanical system to extend their visible range. The photo receiver of ToF cameras holds a grid of pixels that are able to measure the time of flight. Hence, they are able to capture several two-dimensional distance images per second. ToF cameras can be used in near and far range. This makes them useful for a wide field of applications in and outside the car. Cameras optimized for the inside could help to detect gestures, track the passengers positions for intelligent safety systems and prevent car theft by observing the interior when the car is parked. Outward cameras can help to increase safety by observing the surrounding of the car. They help to keep track of the lane, the distance to other vehicles and can detect pedestrians and upcoming objects. [Li14]

Different ToF camera methods and implementations exist, but for an application in the automotive environment an accurate, reliable, eye-save and energy efficient system is required.

The goal of this thesis is to evaluate a pulse based ToF camera prototype and find a suitable distance calibration.

In the following the structure of this thesis is explained.

Chapter 2 introduces the mathematical background of the ToF methods and describes the different principals of available ToF cameras.

Chapter 3 describes the general requirements for a ToF system to use in the automotive environment.

Chapter 4 introduces and describes the used ToF camera. Furthermore the calibration of the camera system is explained.

Chapter 5 introduces and explains the software that was designed for the thesis.

Chapter 6 sums up the insights and gives an recommendation for further work.

2

TOF TECHNOLOGY

Using the time of flight of an electromagnetic wave to measure the distance to a distant object was already done in the beginning of the 20th century. Back then, radar waves were used to detect ships which lead to modern versions of *radar* systems. *Lidar* systems were developed shortly after the laser was invented in the 1960s. With smaller silicon the first *time of flight cameras* were developed in the year 2000. [Hol07] [MAI60] [Wei05] [Sys06]

In the following the physical background, which is similar for all kinds of time of flight systems, is explained. Also, some definitions, that are helpful to understand the later sections of this thesis, will be given. Then the different time of flight distance measurement principles are introduced. Finally, problems that can affect the systems accuracy are discussed.

2.1 PHYSICAL AND MATHEMATICAL BACKGROUND

In the following some background information that is used in the later sections is defined. First the speed of light and its relation to distance measurements is explained. Then an introduction to the projection of 3D objects to a 2D image is given.

2.1.1 *Speed of light*

Generally the distance D a light pulse travels in a certain time ΔT depends on the speed of light c_0, which is equal for all kinds of electromagnetic waves in the same medium. [Hec02]

$$D = c_0 \cdot \Delta T \tag{1}$$

$$c_0 :\approx 2.998 \cdot 10^8 m/s \tag{2}$$

For a measurement the pulse travels the distance between camera and object twice. First from the camera to the object and then back from the object to the camera, where it is captured.

Therefore, the length of the distance L from the camera to the object is calculated as:

$$L = \frac{1}{2} \cdot c_0 \cdot \Delta T \tag{3}$$

The amount of photon energy V gathered in a photosensitive unit is proportional to the amount of light P, the time of acquisition T and to a hardware depending slope a.

$$V = a \cdot P \cdot T \tag{4}$$

2.1.2 *Perspective camera model*

A perspective camera consists of an imager that is able to create a 2D image from 3D real world objects. For a time of flight camera the 2D image contains intensity and distance informations, a normal CCD camera delivers just colors. Independent from its output, the lens of the system defines the projection from 3D to 2D with its intrinsic parameters.

This projection can be explained using the pinhole camera model, see figure 1. The real world object is projected through the pinhole to the image plane. The focal length and the size of the image plane define the size of the projected object.

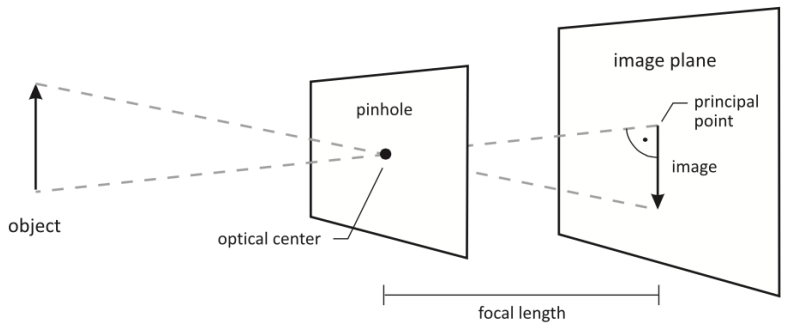

Figure 1: Pinhole camera projection. [Lin10]

The intrinsic camera matrix can be used to compute the exact position of a 3D object point on the 2D image plane.

2.1.2.1 *Intrinsic camera matrix*

The camera projection matrix K is defined as

$$K = \begin{pmatrix} f/s_x & 0 & t_x \\ 0 & f/s_y & t_y \\ 0 & 0 & 1 \end{pmatrix} \tag{5}$$

where f is the focal length of the lens. The size of a single camera pixel is given by s_x and s_y. t_x and t_y are additional translation parameters that map the image center to the $(0,0)$ point, therefore they are related to the resolution of the imager. The intrinsic matrix K is obtained through the process described in section 4.2.2.1.

The position of an object point $\mathbf{x}$ is given by

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{6}$$

were x, y, z are known as the points 3D world coordinates. The origin $(0,0,0)$ is the camera's center, the x and y axes are parallel to the imager and the camera's viewing direction defines the $-z$ axis.

$\mathbf{p} = (u, v)$ is the 2D image of $\mathbf{x}$ on the image plane. It is computed by the multiplication of the 3×3 matrix K and a the 3×1 point $\mathbf{x}$.

$$\mathbf{p} = K\mathbf{x} = \begin{pmatrix} u' \\ v' \\ w \end{pmatrix} \tag{7}$$

As mentioned above, the result of the projection should be a 2D image point. Currently the result appears to be 3 dimensional. Because of the projection to the image plane, $\mathbf{x}$ is a 2D point in its homogenous coordinates form.

The 2D point $\mathbf{p}$ in pixel coordinates, on the projection plane, can be computed by bringing its 3D counterpart in the normalized form. This is done by dividing it by its z component, since in homogenous coordinates the following equation is true:

$$\mathbf{p} \in \mathbb{R}^3 = \begin{pmatrix} u' \\ v' \\ w \end{pmatrix} = \begin{pmatrix} u'/w \\ v'/w \\ w/w \end{pmatrix} = \begin{pmatrix} u'/w \\ v'/w \\ 1 \end{pmatrix} = \begin{pmatrix} u'/w \\ v'/w \end{pmatrix} = \begin{pmatrix} u \\ v \end{pmatrix} = \mathbf{p} \in \mathbb{R}^2 \tag{8}$$

Note that in case of a CCD camera the distance information in the projection is lost when dividing by w. Only time of flight cameras makes it possible to reverse the projection, because the distance information is additionally stored in the pixel. This is used in section 5.3. [Wei15] [Lin10]

2.1.2.2 *Radar and lidar*

Radar and lidar systems emit electromagnetic energy. While radar systems emit radio waves, lidar sends out laser pulses. A

single lidar unit consists of one laser light source and one receiver that is connected to a time to digital converter. It is used to detect the flight time of the emitted laser pulse. To overview a wide field of vision, it is necessary to use a movable mechanical mirror. For each point in the hemisphere the mirror needs to move to the desired angle and one pulse is emitted. The emitter, mirror, and receiver must be synchronized. [ACA15]

2.2 TIME OF FLIGHT CAMERAS

Time of flight cameras illuminate a scene and compute the distance to the objects from the time delay of the returning light. They consist of three main components: An *illumination unit* is used to send out a known light form. The second component is some sort of *computer chip* that connects and triggers transmitter and receiver, and enables the readout. A *receiving unit* consists of a grid of photosensitive pixels that are able to detect the returning light. With thousands of photosensitive pixels in the imager, the camera is capable to acquire all objects in its visible range at once, without the requirement of any mechanical parts.

There are two main principles to perform a time of flight distance measurement with a camera: *pulsed light* and *phase modulated light*.

Pulsed light based ToF cameras emit at least one short light pulse and measure the time of its flight. Systems of the second kind emit light, which is modulated with a specific long wave pattern. They recognize the phase shift in the returning wave to compute the distance. In the following a brief introduction of the two main ToF camera technologies is given. [Erz11]

2.2.1 *Pulse based cameras*

A pulse based ToF camera measures the distance to an object by detecting the flight time of a light pulse that is emitted from the camera and reflected at the object. Since the speed of light is constant, the distance is proportional the traveled time. [Sch11b]

2.2.1.1 *Working principle*

The photo detectors acquisition starts with sending a light pulse and the returning light is collected with a photo detector. It consists of a pixel grid and related to the scene, each pixel can

receive the returning pulse at a different time. The following explanation describes the behavior of a single pixel. A detector pixel is accumulating the light in two intervals, a short and a long one. To measure a distance, a fraction of the returning light pulse must reach the detector before the short accumulation interval ends. The second interval ends after the whole pulse has returned. Figure 2 shows the timing for a single pixel.

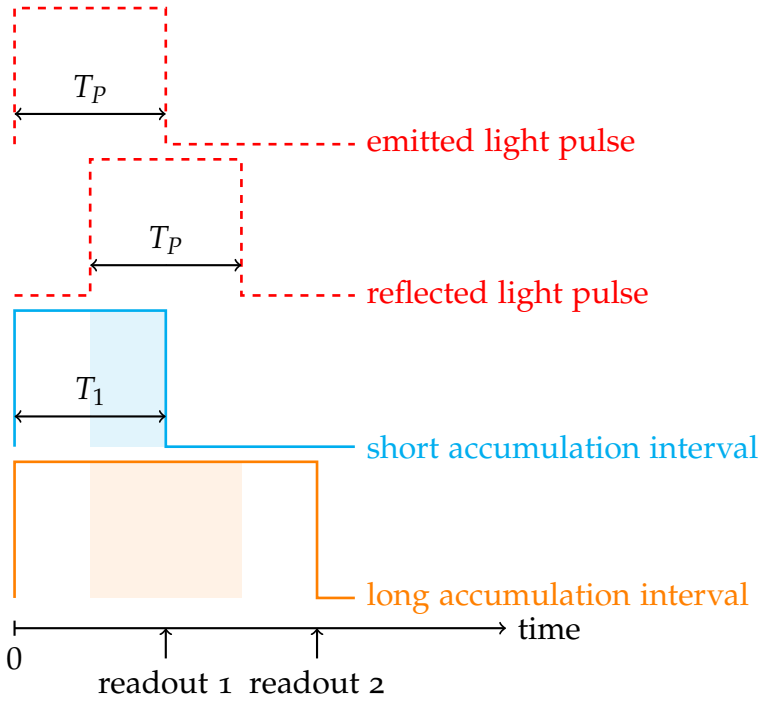

Figure 2: Pulse and acquisition timings of the pulse based camera. The dashed red line is representing the pulse that is reflected by the object and accumulated in the short (cyan) and long (orange) interval. The light colored rectangles in the intervals represent the time where pulse light is received. [Erz11]

From the detectors two values can be obtained that are used for the distance calculation: The intensity of the whole pulse and the intensity caused by the fraction of the pulse in the short interval. In the first acquisition interval the fraction of the light pulse intensity I_F is received. The intensity of the whole light pulse I_P is in the long acquisition interval. During the acquisition not only light pulse photons are collected, but also photons from other light sources. The intensities of a second measurement without active illumination are subtracted from the intensities of the first one, to receive only intensities that are caused by photons of the light pulse. It can be assumed that

I_F and I_P are already cleared from the ambient lights influence and that the light pulse width T_P is equal to the short accumulation intervals width T_1. Then the ratio R of the intensity in the short to the intensity in the long interval is a number between 0 and 1.

$$R = \frac{I_F}{I_P} \qquad \in [0,1] \tag{9}$$

This value is proportional to the distance as

$$d = \frac{1}{2} \cdot c_0 \cdot T_P \cdot (1 - R) \tag{10}$$

where c_0 is the speed of light and the factor $1/2$ comes since the light pulse needs to travel the path from camera to object twice, see equation 3.

Note that it is possible to accomplish that I_F and I_P are read out instantaneous. This is done by not deleting the accumulated potential for the short integration time. [Erz11]

2.2.1.2 *Distributors*

Market ready pulse based camera systems and imagers can be purchased from a few suppliers. **TriDiCam** develops and distributes pulse based time of flight sensors and cooperates with the Fraunhofer IMS in Duisburg, Germany. TriDiCam offers two kinds of imagers. One is row oriented with a resolution of 64×2 pixels, the second is an area sensor with a 128×96 pixel resolution. [Tri15b]

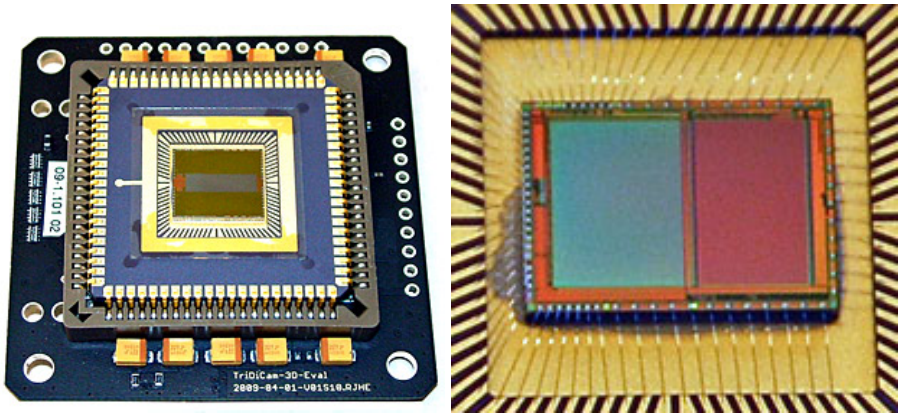

Figure 3: TriDiCams row (left) and area oriented (right) pulse based ToF sensors. [Tri15c] [Tri15a]

Another distributor is **Hamamatsu Photonics**. The japanese company offers different pulse based time of flight imagers.

The S11961-01CR imager is row oriented and offers 256 pixels. The S11962-01CR and S11963-01CR imager are both area sensors with 64×64 and 160×120 pixels. In this thesis the S11963-01CR was used. [Pho14c][Pho14a][Pho14b]

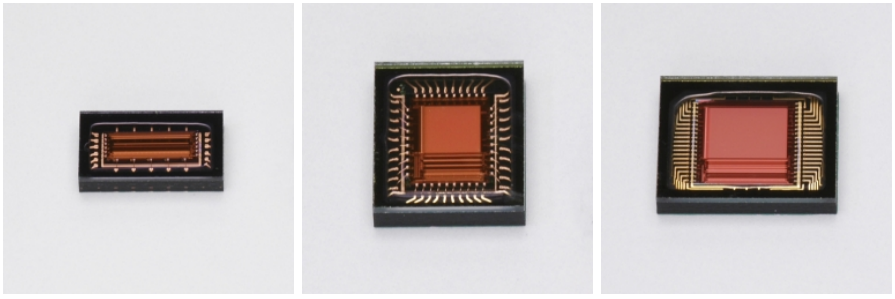

Figure 4: Hamatsu Imagers from left to right: S11961-01CR, S11962-01CR, S11963-01CR [Pho14c][Pho14a][Pho14b]

2.2.2 *Phase modulated cameras*

Other than pulse based, phase modulated cameras constantly emit light during the acquisition of a frame. They also measure the time of flight by detecting the phase shift from the returning light. [Lin10]

2.2.2.1 *Working principle*

The distance is computed from the phase shift of a periodical modulated signal that is emitted by an illumination unit and received by an image sensor that consist of special pixels called smart pixels. The illumination unit emits infrared light ($\lambda \approx 800nm - 1400nm$) which is modulated with a long wavelength ($\lambda \approx 5m - 15m$). The wavelength of the infrared light is indifferent, as long as it is significant shorter than the modulated wavelength, since it merely serves as carrier signal for the long wave. [Lin10]

The distance from the camera to the point of reflection can be recognized by finding the phase shift φ of the reflected $r(t)$ and emitted signal $s(t)$ in the image sensor. To obtain the phase shift, the receiver needs to know the emitted reference signal as reference. Therefore the receiver and the illumination unit are connected to the same signal generator. $r(t)$ the reflected signal and $s(t)$ the reference demodulation signal are sinusoidal

$$r(t) = b + a \cdot sin(\omega_{mod} \cdot t - \varphi) \tag{11}$$

and

$$s(t) = sin(\omega_{mod} \cdot t) \tag{12}$$

Here b stands for the background light, a for the amplitude and $\omega_{mod} = 2 \cdot \pi \cdot f_{mod}$ for the angular modulation frequency.

The smart pixels values depend on the correlation between $s(t)$ and $r(t)$ over the integration time T_{int} like

$$S = c(\psi) = r(t) \otimes s(t) = \lim_{T \to \text{inf}} \frac{1}{T_{int}} \int_0^{T_{int}} r(t) \cdot s(t + \psi) dt \tag{13}$$

where ψ is a phase delay introduced into the reference signal.

This leads to

$$S = c(\psi) = \frac{a}{2} cos(\varphi - \psi) \tag{14}$$

To find the phase shift φ, S is sampled by four sequential images

$$S_i = c(\psi_i), \quad i = 0, 1, 2, 3 \tag{15}$$

where for each sample a different value for ψ is used.

$$\psi_0 = 0^o, \psi_1 = 90^o, \psi_2 = 180^o, \psi_3 = 270^o \tag{16}$$

φ and a can then be computed by

$$\varphi = atan\left(\frac{S_3 - S_1}{S_0 - S_2}\right) \tag{17}$$

$$a = \frac{1}{2}\sqrt{(S_3 - S_1)^2 + (S_0 - S_2)^2} \tag{18}$$

φ leads to the distance

$$d = \frac{1}{4\pi\omega_{mod}} \cdot c \cdot \varphi \tag{19}$$

from the object to the camera. [Lin10]

2.2.2.2 *Distributors*

A distributor for phase modulated time of flight cameras is **PMD technologies** from Siegen, Germany. The company cooperates with **Infineon** and offers a time of flight camera called CamBoard picoS that is based on the Infineon IRS 1010C chip that has a 160×120 pixel resolution. Infineon also offers the IRS 1020C with 352×288 pixels. [PMD15b]

MESA imaging is a swiss company that offers two time of flight cameras. The SR4000 has 176×144 pixels and operates to $5m$, optional a $10m$ version is available. Its accuracy is said to be $\pm 1cm$. A second version, the SR4500 has the same resolution, however operates up to $9m$ with an accuracy of $\pm 2cm(0.5 - 5m)$ and $\pm 4cm(> 5m)$. It also offers a digital signal processor for embedded image processing. [Ima15]

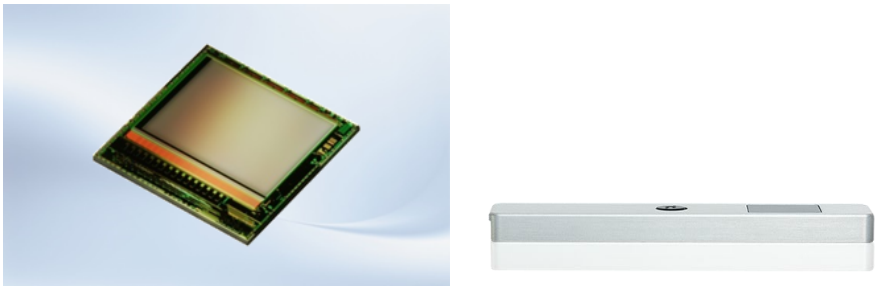

Figure 5: Infineons IRS 1010C (left) and PMDs CamBoard picoS (right). [PMD15a] [Inf15]

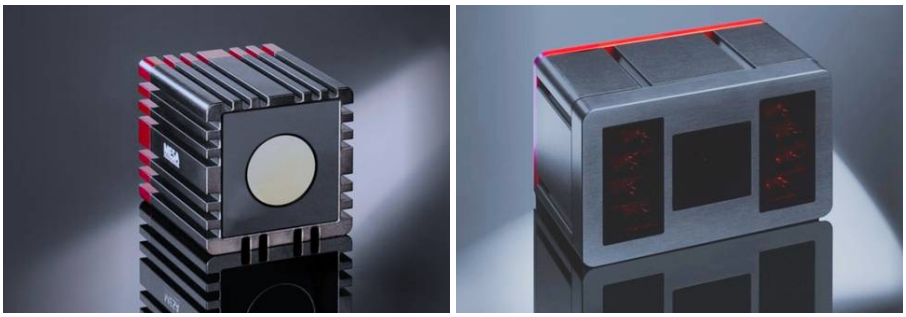

Figure 6: Time of flight cameras from MESA imaging. The SR4000 (left) and the SR4500 (right). [Ima15]

2.3 TOF ERRORS

Time of flight cameras are affected by several errors that can influence the measured distances. In the following these errors are further described.

2.3.1 *Signal Quality*

The outcome of the measurement depends on the reliable detection of the reflected signal. One common effect comes from the IR-reflectivity of the observed surfaces. Objects that turn out as bad reflectors also have a negative impact of the image due to the fact that undersaturated pixels have insufficient signal-to-noise ratios. Heavy mirroring objects reflect almost the whole energy which quickly leads to oversaturation in the pixels. Oversaturated pixels are not even able to deliver any reliable distance information.

As described above, the formulas to compute distances assume perfect physical conditions. For a pulsed camera this means that the pulse form is perfectly rectangular. In the phase modulated camera a perfect sinusoidal shape is assumed. Here

this effect is also known as *demodulation error*, see [Lin10]. In both cases a distance calibration can help to reduce the error of this effect.

2.3.2 *Multipath effects*

Multipath effects occur when the incoming light is a composition of light that has traveled different paths to a single pixel. This means that in one pixel signals from several sources are composed and seen as one. This can be a huge problem when an object is highly reflective. However, for most objects the magnitude of the indirect illumination is low compared to the magnitude of the direct illumination. This is why measured data is still useful in most cases. [Kle14]

2.3.3 *Flying pixels*

An error occurs when the returning light in the pixels' solid angle is from one object in the foreground and another object in the background. This mostly happens at the edges of objects. Because the pixels mix up the intensities, the resulting distance is also a mixture that matches neither the object in the foreground nor the one in the background. The result becomes visible as flying pixels and can be seen in figure 7. [Lin10]

2.3.4 *Motion artifacts*

Motion artifacts occur when the observed object moves during the acquisition of one frame. The error gets higher, the higher the speed or the acquisition time is. If the objects performs a lateral movement the error will be visible in motion artifacts at the object boundaries. Motion artifacts that do not arise on the boundaries can be caused by different IR-reflectivity. Another error occurs due to movement along the viewing axis. [Lin10]

2.3.5 *Intensity related error*

Intensity related error denotes an error that comes when an object has zones of different bright colors. The dark squares of a checkerboard are measured closer than the light ones. The reason for this error is unknown. [Lin10]

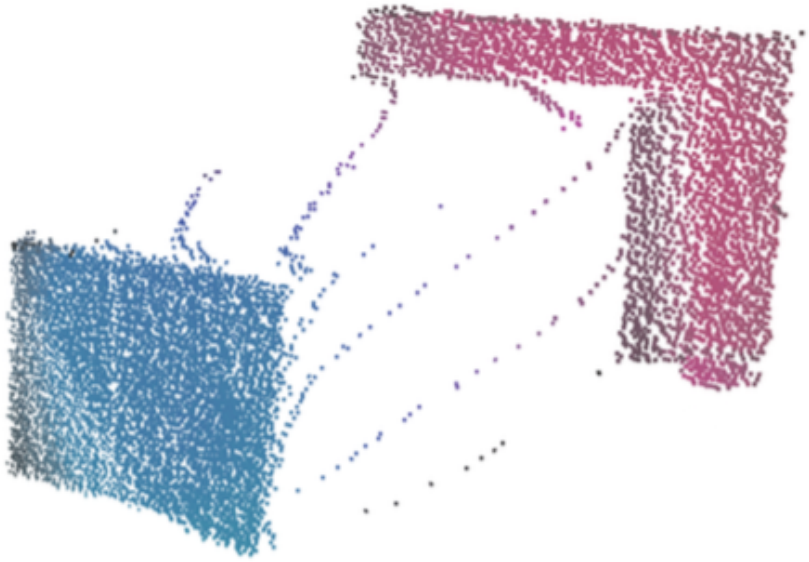

Figure 7: Flying pixels that are caused from different distances in a single pixel. [Lin10]

2.3.6 *Noise*

In general, optical systems are affected from different forms of noise that is classified in the following.

TIME VARIANT NOISE means all types of noise that can change over time. When the optical sensors generate charge from incoming photos, Poisson noise is introduced. [Sch11b] The amount of incoming photons does not reduce this effect, but it can be modeled with a normal distribution. Some forms that are independent from the signal are thermal noise, reset noise and dark current shot noise. These noise forms increase with rising temperature of the system, so active cooling is a proper way to reduce it. [Lin10]

TIME INVARIANT NOISE does not change over time and is individual for each pixel. Inaccuracies in the production process are the reason for this noise. Extreme forms of this are broken pixels or pixels that always measure slightly higher values. It is possible to compensate these due to the constant appearance of their errors.

Some denoising is possible by applying filters to the images, see [Lin10].

3

TOF IN AUTOMOTIVE

Time of flight systems are used in the automotive industry for years in form of radar and lidar. [LLP15] Time of flight cameras could also be good candidates for the use in cars since they are able to acquire a whole scene at once without mechanical parts. In the following, some fields of applications for time of flight cameras are listed. Then, general requirements for electronic components in a car are listed. Finally, a functional goal for this thesis is derived.

3.1 FIELDS OF APPLICATION

For time of flight cameras a lot of applications in the car are conceivable. The camera can be pointed into the car and observe the interior, or it can be pointed outside and observe the surroundings. In the following some possible scenarios are delineated.

INTERIOR SURVEILLANCE: The sitting position of the travelers could be tracked. This could be used to optimize the airbag trigger.

GESTURE RECOGNITION: Hand gestures can be recognized and then used to control the cars entertainment system.

CAR ALARM: The camera could observe the cars' interior and detect suspicious movements when the car is parked.

DISTANCE CONTROL: The camera can measure the distance to the objects in front of the car, to keep the distance or prevent collision with animals and pedestrians.

AUTONOMOUS CAR: Further, a time of flight camera can improve the environment recognition and assist the driver or make a car full autonomous.

3.2 REQUIREMENTS

Measured distances should be as reliable as possible, even if the scene is illuminated with direct sun light. The illumination of the camera needs to be bright enough, but on the other hand the brightness is limited due to human eye safety. Another requirement is that the camera should consume as less energy as needed. This is because the engine is not running and the car's battery serves as the camera's power source for interior surveillance. Another important requirement coming from the automotive manufacturers is that the system needs to work at temperatures from $-40°C$ to $+85°C$, sometimes up to ($+110°C$). Here the temperature of the air that surrounds the system is measured. The system itself could become even hotter due to its own heating. Since car manufacturers buy huge quantities of parts the price to function performance is also very important.

3.3 FUNCTIONAL GOAL

The functional goal of this thesis is to analyze a ToF sensor that is expected to be suitable for the desired fields of application and meets the requirements. To evaluate the system, a distance calibration needs to be performed.

A pulse based ToF camera is expected to meet these better than phase modulated cameras, primarily because of the energy usage. During the acquisition of an image, pulse based cameras emit a small bright pulse every few microseconds, where phase modulated cameras need to emit light constantly. The pulse can have higher energy than sunlight and be eye safe at the same time, as it is very short.

The camera system that was used for this thesis is the pulse based Hamamatsu **S11963-01CR Distance Image Sensor**. Hamamatsu Photonics[1] provided a demo unit that could be used for evaluation. The next chapter contains a detailed description of the camera system.

1 http://www.hamamatsu.com/eu/en/index.html

4

CALIBRATION AND EVALUATION

The calibration and evaluation process can be separated into three parts. The first part contains a brief description of the camera, its working principle and important parameter settings. The second part describes the calibration of the system. In the last section further details of the camera pixels are investigated.

4.1 THE USED CAMERA SYSTEM

The camera is a demo unit that was provided by Hamamatsu Photonics for the evaluation of the pulse based **S11963-01CR Distance Image Sensor**. It is shown in figure 8. In the following, the components, parameters and the working principle of the camera are further explained.

4.1.1 *Description*

The camera consists of an illumination unit (left) and a lens (right in figure 8). The system is powered by two power supplies, one for the controller and one for the camera. It connects to the computer via gigabit Ethernet.

4.1.1.1 *Optical lens*

The system already contains a preadjusted optical lens with an $37,5^\circ \times 27.7^\circ$ angle of view. The F-number is 1.2 the focal length is $f = 8mm$. Additionally, the lens contains a *HOYA IR83N* IR-transmission filter.

4.1.1.2 *Light source*

The light source consists of 64 $5mm$ LEDs that are oriented in a $8x8$ grid. It is covered by a diffuser and the projection angle is $17^\circ \times 17^\circ$. The wavelength is $870nm$ with a FWHM of $45nm$.

Figure 8: The used camera system with illumination unit (left) and the lens (covered, right). [Ham14]

Pulses that are emitted from this light source have a constant width of $40ns$ with a peak power of $10W$.

Instead of using infrared LEDs it would also be possible to use short pulse laser light. Here the luminous power is higher, but laser is more dangerous.

4.1.1.3 *Imager*

As imager the **S11963-01CR Distance Image Sensor** is used. With a size of $4.8mm \times 3.6mm$, a pixel size of $30\mu m \times 30\mu m$ and a $30\mu m$ pixel pitch the imager offers 168×128 pixels whereof 160×120 pixels can be used. Without any filters the imager has a sensitivity that covers the visible spectrum and infrared light to a wavelength of $900nm$, see figure 9.

A block diagram of the imager is visible in figure 10. The incoming light generates charge in the photodiode array. Depending on which circuit in the pixel is closed, the charge is collected in one of the pixels shutters, see figure 11. They are read out at the end of the measurement using the column gain amplifier circuit to $Vout1$ and $Vout2$.

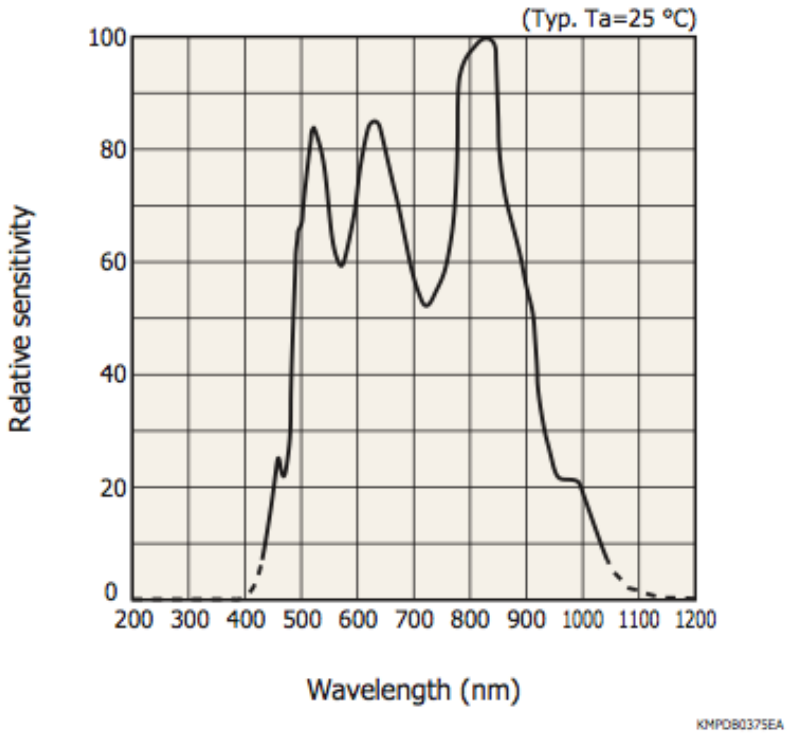

Figure 9: Sensitivity vs. wavelength of the imager without an IR-filter. [Ham14]

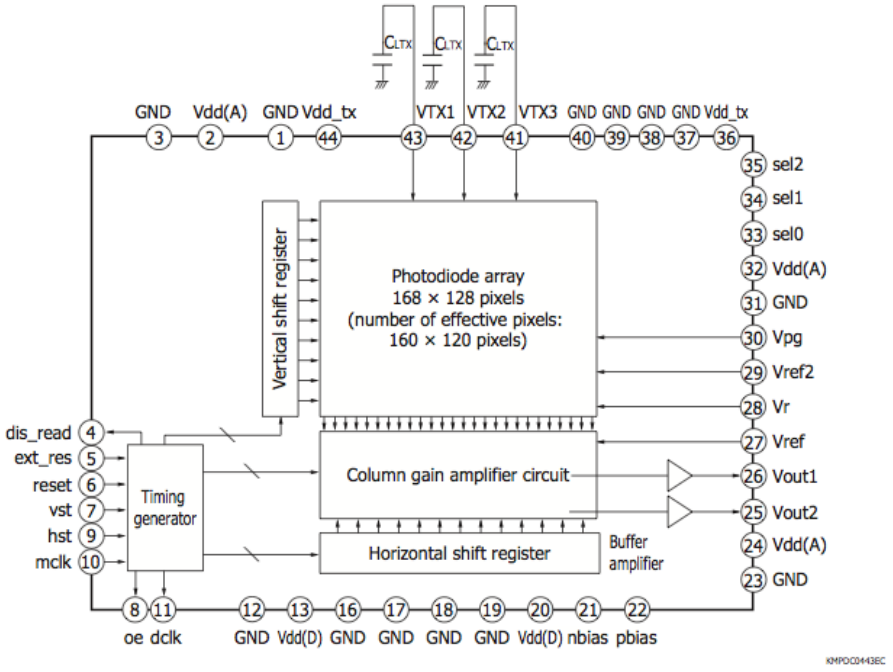

Figure 10: Block diagram of the imager. The photodiode array in the center that is read out by a column gain amplifier circuit to *Vout*1 and *Vout*2. [Ham14]

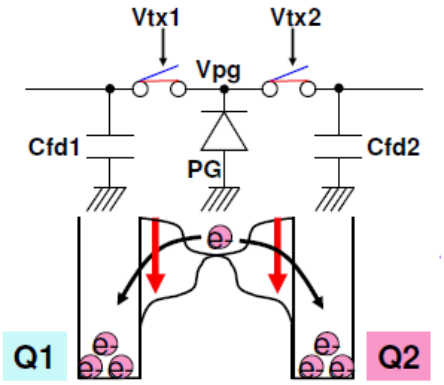

Figure 11: Working principle of a single pixel in the imager. When one of the circuits is closed using the Vtx trigger, the potential barrier is dissolved (red arrow) and the generated charge is collected. Image taken from email correspondence with permission of Hamamatsu.

4.1.2 *Parameters*

In the following the main parameters of the camera and their influence to the system is explained. Table 1 shows the main parameters and their values. Note there were some other parameters that were also changeable in the controller. Most of them could be used to adjust some voltages in the imager, others to set parameters for the data transfer and software decoding. This parameters were not changed as they were already set to recommended values. A full list can be seen in the camera's manual. [Ham14]

name	**unit**	**used value**	**allowed values**
tacc	ms	varied	[200,300,...,1000000]
light pulse width	ns	40	[10,20,...,1000]
light pulse delay	ns	40	[0,5,10,...,10000]
VTX1	ns	40	[10,15,...,1000]
VTX2	ns	40	[10,15,...,1000]
VTX3	ns	3920	[0,5,10,...,50000]

Table 1: Camera parameters and their suggested values.

4.1.2.1 *Light pulse width*

This parameter defines the width of the emitted light pulse and influences the camera's distance range. Increasing the pulse width also increases the maximum possible distance. Unfortu-

nately, attempts to change the light pulse width were unsuccessful. The illumination unit emits pulses with a width of approx. $40ns$, even if a different value is set, see section 4.2.1.

4.1.2.2 *VTX1 and VTX2*

$VTX1$ and $VTX2$ describe the time shutter 1 and shutter 2 are open, respectively. Compare figure 11. They are chosen to be equal, and also equal to the light pulse width. So they are set to $40ns$.

4.1.2.3 *VTX3*

Primary, $VTX3$ defines the length of a third shutter that is opened to prevent discharging of capacitors in the pixels while the $VTX1$ and $VTX2$ shutters are not open. For the acquisition of one light pulse, shutter 1, 2 and 3 are opened in sequence. The complete acquisition time is $VTX = VTX1 + VTX2 + VTX3$. Because $VTX1$ and $VTX2$ are fixed in this setup, VTX can be adjusted by varying $VTX3$. Therefore, $VTX3$ influences the time interval between the emitting of the single pulses.

The following equation may not be violated:

$$light\ pulse\ width < VTX1 + VTX2 + VTX3 \tag{20}$$

On the other hand $VTX3$ can not be set to 0 because the light needs some time between two pulses. For example, a lamp that sends $100ns$ pulses with a duty cycle of 0.1% needs $100\mu s$ between each pulse.

4.1.2.4 *Tacc*

Tacc defines the time of acquisition for a single frame. A longer acquisition time increases the number of pulses per image. This results in a higher amount of photons that are gathered in the shutters. More accumulated photon energy leads to a less noisy image, but can also lead to oversaturation.

The number of pulses for a frame is defined by the ratio of VTX and *Tacc*

$$pulses\ per\ frame = \frac{Tacc}{VTX} \tag{21}$$

4.1.2.5 *Light pulse delay*

In theory, the light pulse must be emitted simultaneously with the opening of the first shutter, see figure 2. Even if triggered

at the same time the pulse is emitted later due to hardware delays. To compensate this behavior the opening of the first shutter happens after a certain delay in time. This time is called *light pulse delay* and is set to $75ns$. A method to find the light pulse delay is described in section 4.2.1.1.

4.1.3 *Working principle*

The working principle of this camera is just insignificantly different to the working principle described in section 2.2.1.1. Instead of accumulating the photons in one shutter that is read out twice, this imager consists of two shutters that are triggered different and read out simultaneously after the accumulation. In the following a detailed derivation of the working principle including the used formulas for distance computation is given. After that, the timings of the shutters are put in the context of a full frame acquisition.

4.1.3.1 *Shutter timings*

A pulse based TOF camera consists of two shutters for each pixel that are opened and closed alternating while gathering the photons of the laser pulse. The distance to an object can be computed using the ratio of charge in the shutters.

When the camera sends out the light pulse with a width of T_0, it instantly opens the first shutter to gather the reflected light for a duration of T_1. Then it switches to open the second shutter for a length of T_2. Figure 12 shows the timings of shutter 1 in cyan, shutter 2 in orange and the returning light pulse in dashed red.

The amount of light that is gathered in the shutters 1 and 2 depends on the traveled distance ΔT and the integration times T_1 and T_2. When the pulse has traveled for the time ΔT and returns to the camera, shutter 1 begins to receive light pulse photons. Shutter 1s integration time is limited by T_1, therefore the time it receives the laser pulse is $T_1 - \Delta T$. Shutter 2 was first opened when shutter 1 was closed at T_1, so it immediately begins to gather the laser pulse photons, until the pulse ends at time $\Delta T + T_0$. Hence the time shutter 2 gathers laser pulse photons is given by $\Delta T + T_0 - T_1$.

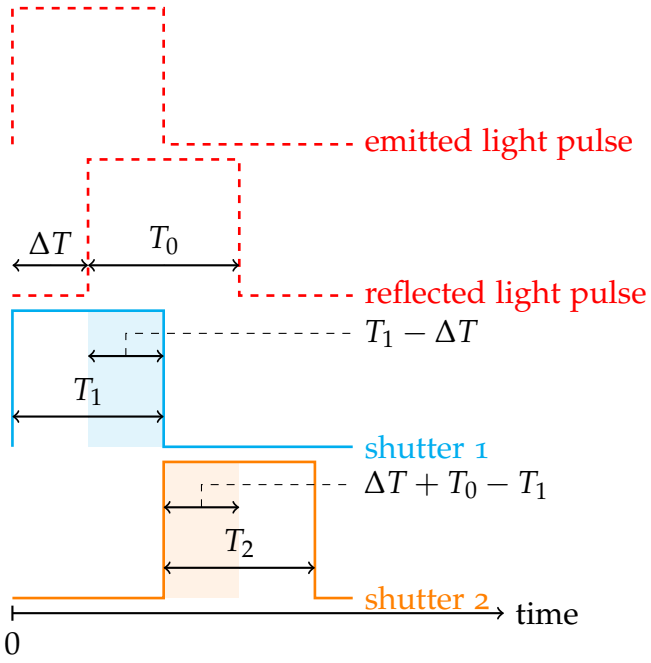

Figure 12: Timings of shutter 1, shutter 2 and the light pulse

4.1.3.2 *Distance computation*

In the following P_{Laser} stands for the amount of light emitted by the laser. The knowledge about the time that each shutter can receive laser pulse photons leads to the amount of gathered charge $VTX1$ and $VTX2$ for shutter 1 and shutter 2, respectively. Equation 4 can be extended to

$$VTX1 = a_1 \cdot P_{laser} \cdot (T_1 - \Delta T) \tag{22}$$

and

$$VTX2 = a_2 \cdot P_{laser} \cdot (\Delta T + T_0 - T_1) \tag{23}$$

It is to be noted that only in theory shutter 1 and 2 receive the same amount of energy from an equal number of photons. In reality, it is possible that shutter 1s and 2s charge is different due to manufacturing varieties. The factors a_1 and a_2 are used to describe this behavior. Since a_1 and a_2 can also be individual for every pixel we assume $a = a_1 = a_2$ here and go deeper into this topic in section 4.3.

Using equation 22 and 23 one can solve for ΔT by eliminating $a \cdot P_{laser}$ which leads to

$$\Delta T = \frac{T_1 \cdot (VTX1 + VTX2) - VTX1 \cdot T_0}{VTX1 + VTX2} \tag{24}$$

Finally, one can compute the distance to the object using equation 3 and 24 as

$$L = \frac{1}{2} \cdot c_0 \cdot \frac{T_1 \cdot (VTX1 + VTX2) - VTX1 \cdot T_0}{VTX1 + VTX2} \tag{25}$$

Considering the relation between T_0 and T_1, one can see $T_0 < T_1$ would mean that the integration time is longer than the pulse. This would lead to the same ΔT when $\Delta T + T_0 < T_1$ and make it impossible to find out the correct propagation time. To choose $T_0 > T_1$ would only be correct if shutter 2 is long enough not to be closed before the signal ends. $T = T_0 = T_1$ is chosen, because it makes it possible to reduce the integration time of shutter 2 to the pulse width. Under these circumstances, the equation 25 is simplified to:

$$L = \frac{1}{2} \cdot c_0 \cdot \frac{T \cdot VTX2}{VTX1 + VTX2} \tag{26}$$

To compute a distance frame the value of $VTX1$ and $VTX2$ must be received. The next section explains how this is done.

4.1.3.3 *Distance frame acquisition*

The timings of the shutters above make just a fraction of what happens to receive a full distance frame. For one full distance frame, eight subframes are transferred from the camera to the computer. Four of the subframes are related to values from the first shutter, the rest belong to the second shutter.

In reality, the photon energy of the pulse is not charging, but discharging an already charged pixel. The difference of the charged potential and the potential after accumulation gives the photon energy. Therefore the pixel is charged and the value of the charge is read out of the pixels and transfered to the computer, before the accumulation of pulse photons begins. These values are denoted with $full$ in the following, since the shutters are fully charged.

After the accumulated photons have discharged the pixels, the charge is read out and another transfer to the computer happens. Since the values are read out after the measurement these values are denoted with $after$ in the following.

Figure 13 shows the timings for charging and readout, the discharging process with photon energy, and the second readout process.

The difference of $full$ and $after$ must be computed once for the $Vout1$ and once for the $Vout2$ values to receive a grid of numbers that give the pulse photons energy in each pixel.

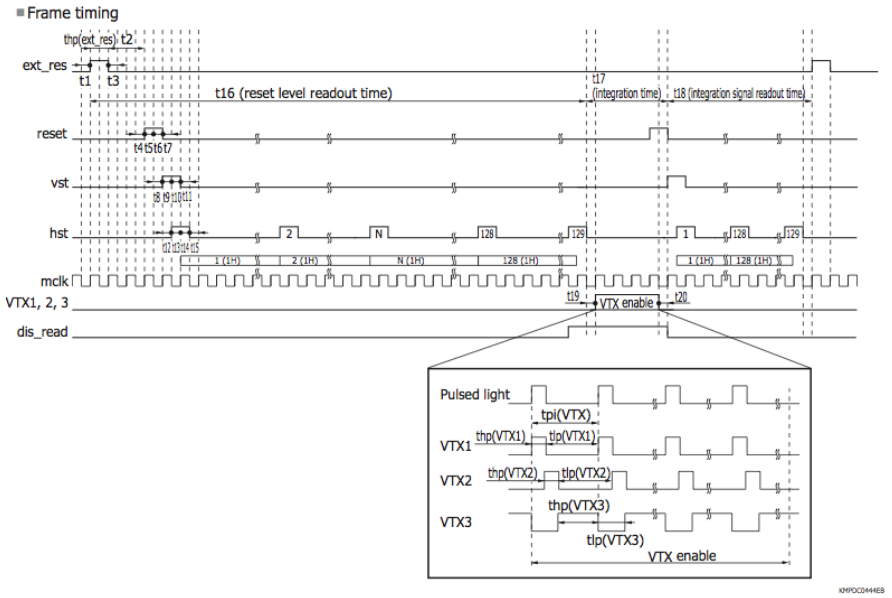

Figure 13: Timing diagram for one frame. First, the pixels are reset and read out, then the shutter are opened (zoomed window) and finally the results are read out. [Ham14]

At this point four of the eight subframes for one full distance frame are received. Where the other four frames come from is explained in the next section.

4.1.3.4 *Obtaining VTX1 and VTX2 from camera data*

$VTX1$ and $VTX2$ define the charge from the returning light pulse that is collected in shutter 1 and 2, respectively. The amount of energy the shutter measures is not only caused by the light pulse. Several other light sources, like the sun or lamps, can increase it. To suppress other light sources, the shutters charge is obtained from two measurements. One measurement is done with ($light\_Vout$), another without active illumination ($dark\_Vout$). The *light* measurement consists of ambient and pulse light. It can be assume that the ambient light remains constant during the short acquisition interval. It is captured in the *dark* measurement. The difference between the two measured values leads to the influence of laser:

$$VTX1 = |light\_Vout1 - dark\_Vout1| \tag{27}$$

$$VTX2 = |light\_Vout2 - dark\_Vout2| \tag{28}$$

Each of the measurement results in a transfer of four subframes, which makes eight subframes that are necessary to compute one distance frame, as mentioned above.

4.1.3.5 *Computing a distance frame*

As explained above, from the two measurements the following data sets are send to the camera.

- light_Vout1 : intensity of shutter 1 with laser illumination
- light_Vout2 : intensity of shutter 2 with laser illumination
- dark_Vout1 : intensity of shutter 2 without laser illumination
- dark_Vout2 : intensity of shutter 2 without laser illumination

Each data set is a 160×120 grid $G = (light|dark)\_Vout(1|2)$, that holds the measured intensity values of a shutter. Where $G_{u,v}$ denotes the measured intensity at pixel position u, v.

Each distance image is computed using two frames, one for the first and one for the second shutter. Each frame is computed using two data sets.

The distance value for each pixel $p_{u,v}$ of the final distance image is computed using equation 25. Where $VTX1_{u,v}$ and $VTX2_{u,v}$ are obtained from the $light\_Vout1_{u,v}$, $dark\_Vout1_{u,v}$, $light\_Vout2_{u,v}$ and $dark\_Vout2_{u,v}$ values using equation 27 and 28 respectively.

4.2 CALIBRATION

In the following the calibration of the camera system is explained. The process of calibration consists of finding suitable parameters to compensate the hardware tolerances. After that, a procedure of a full range distance calibration is explained and realized.

4.2.1 *Parameter calibration*

The hardware does not work like it is expected in theory because of several real life issues, as dark noise, internal delays, and production variations. This is normal for CMOS chips and cannot be avoided, but in some cases compensated when the error is known and measurable. In the following the errors that occur in the different camera parts are explained and if possible, a method or settings to compensate them is discussed.

4.2.1.1 *Light source parameters*

The light source is expected to emit a perfect rectangular pulse within the same moment the first shutter opens. But it does not meet the expectations, since the shape of the pulse is not perfectly rectangular, but rather looks like in figure 14. Also the pulse emitting is delayed and does not happen when the first shutter opens, and also can shift over time due to temperature influences.

LIGHT PULSE DELAY There is an easy setup that makes it possible to find the light pulse delay. In order to understand how it works, we first consider in what way the distance of an object depends on $Vout1$ and $Vout2$ using equation 26:

$$\frac{1}{2} \cdot c_0 \cdot 40ns \cdot \frac{Vout2}{Vout1 + Vout2} = D \tag{29}$$

Equal values for $Vout1$ and $Vout2$ lead to a distance of $2.998m$:

$$\frac{1}{4} \cdot c_0 \cdot 40ns = 2.998m \tag{30}$$

This means that if the system is $2.998m$ away from a wall the shutters should be filled equally. This knowledge can be used to find the light pulse delay. For this, the camera is placed $2.998m$ away from a plain wall. For each possible light pulse delay the ratio of $Vout1$ and $Vout2$ is observed.

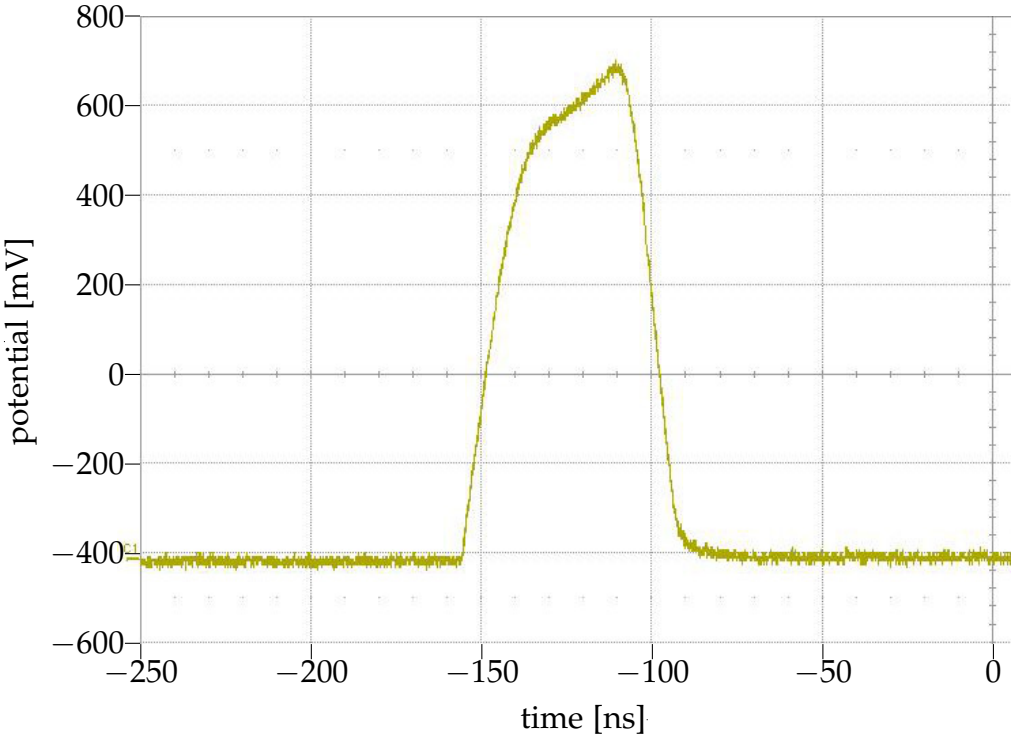

Figure 14: Form of the pulses emitted from the light source in the Hamamatsu Demo Board. Measured with the *1500XP Optical Waveform Analyzer* by *3M Photodyne Inc.* and a *SDA 760Zi-A 6 GHz Osscilloscope* by *Teledyne LeCroy GmbH*.

Figure 15 shows the result. It turns out that 75ns is the best fitting value for the light pulse delay since the camera allows only numbers that can by divided by 5. The measured light pulse delay matches the suggested light pulse delay by Hamamatsu, see section 4.1.2. Note that delays in the system and also the delay of the light pulse can depend on temperature. A continuous correction requires continuous measurement of the delay. This could only be done in hardware with back coupling.

Timeshift To see how the measured distance shifts over time the camera was placed in front of a wall and kept running for three hours while measuring the distance. The result of this test can be seen in figure 16. In the first minutes the time shifts very fast. After one hour the distances become more stable. Over the three hours the distance in average shifts 10cm, from 1.1m to 1.2m.

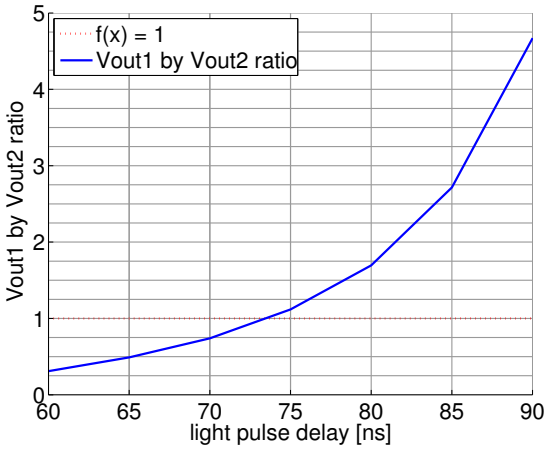

Figure 15: Light pulse delay versus shutter ratio. The ratio is closest to 1 when the delay is set to 75ns.

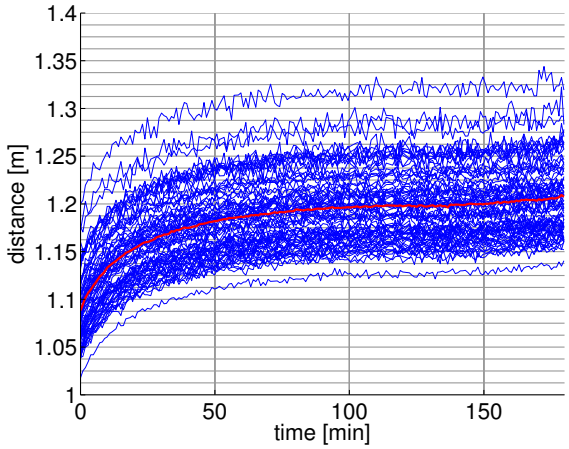

Figure 16: Distance vs. on time of the system. The blue lines show the shift of individual pixels. The red line shows the average distance shift.

4.2.1.2 *Oversaturation*

To measure the range of the numbers that represent the charges in the shutters, a test with a very long acquisition time was performed. It turned out that the highest values for $Vout1$ and $Vout2$ were around 25000. To be sure that the pixels do not oversaturate, the parameters in all tests were chosen that the highest measured charge value is still below 22500 ($= 25000 - 10\%$). The value 22500 is referred as *oversaturation threshold* in the following. Whether the pixels oversaturate depends on the IR-reflectivity, the number of pulses per frame and the distance to the object.

4.2.1.3 *Shutter parameters*

The shutters should open and close instantly and their acquisition shape should also be rectangular. This means that under the same conditions both shutters should measure the same amount of energy. The exact opening and closing times of the shutters are not known, but they are expected to be defective. A further study on this topic is performed in section 4.3.

4.2.2 *Distance calibration*

The goal of the distance calibration is to find a function that corrects the distance error. For this several test based on the distance calibration in [Lin10] are performed. The distance error affects each pixel individually over the camera's range. A function that is able to correct the systematic distance error is the result of the distance calibration. It can be used to improve the accuracy of a measured distance image $D_{measured}$.

$$D_{corrected} = C(D_{measured}) \tag{31}$$

Here C represents such a function. Its result is a corrected distance image $D_{corrected}$. To achieve this, the distance values that the camera measures in each pixel are compared to their ground truth values over the camera's range. A plain wall is the reference object the distance is measured to. The camera is mounted on a movable rail which is placed perpendicular towards the wall. Measurements are performed from $0.5m$ to $5m$ in steps of $2cm$. Each measurement consists of 150 frames that are averaged to reduce noise. Figure 17 shows an outline of the setup.

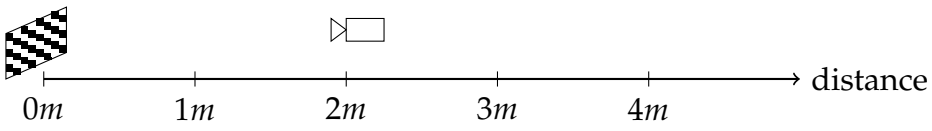

Figure 17: Setup of the distance calibration approach. The wall with the checkerboard at $0m$ and the camera at $2m$.

4.2.2.1 *Camera pose estimation*

To compare the real distances to the measured distances, the camera's pose for each individual measurement needs to be known. The camera's pose consists of the orientation and position of the camera. It can be estimated from an image of an object with known size and position that is taken with the camera. This is done by detecting and analyzing the connection between the reference points and their real world counterparts using the intrinsic parameters. Here the reference points are the corners of a 6×8 checkerboard.

Finding corner positions using the ToF camera's 160×120 images is defective due to the camera's low resolution. To receive

an accurate result, the use of a dual camera approach is necessary, where a high resolution CCD camera [Vis15] is rigidly fixed to the ToF camera.

The intrinsic camera parameters were be found by analyzing several pictures of a checkerboard, taken with the two cameras. For this the *MIP - MultiCameraCalibration* software from [Sch11a] that is based on the approach described in [LSKK10] is used. The dual camera calibrations result were the intrinsic camera parameters including the projection matrices and a matrix to project from the first to the second camera's space and back.

This way the pose of the ToF camera can be obtained from the CCD camera's images. Additionally to each of the ToF measurements one corresponding CCD image must be taken.

In the range of 0m to 1.2m the checkerboard is too big to be entirely visible in the CCD images, so the full set of corners cannot be detected. After a few more meters the checkerboard in the image is too small for accurate results. Therefore, the range of goods results from $1.2m$ to $2.5m$ is used to fit a line of camera centers. This line can be used in a function that finds the camera position and direction for all images taken by the ToF camera.

$$K_{pos,dir}(d) = K_{pos_0} + d * K_{dir} \tag{32}$$

Where d is only valid if chosen as in the range of the rail and only for even $2cm$ steps. With these matrices the distance of each pixel to the wall can be computed leading to the set of ground truth images.

4.2.2.2 *Measurement result*

The data now consist of two distance frame sets. One is the set of measured distance images, the other contains the computed ground truth images. The result of comparing both sets can be seen in the top graph in figure 18.

The error varies over the distance, as seen in the center graph in figure 18. In this and the following graphs, the distance error of a pixel is represented by a semitransparent ($\alpha = 0.01$) line. Depending on how many lines overlay, lighter and darker section appear.

The goal is to find the function C that takes the measured image and corrects it. Since this function depends on the error, the errors are further studied in the following.

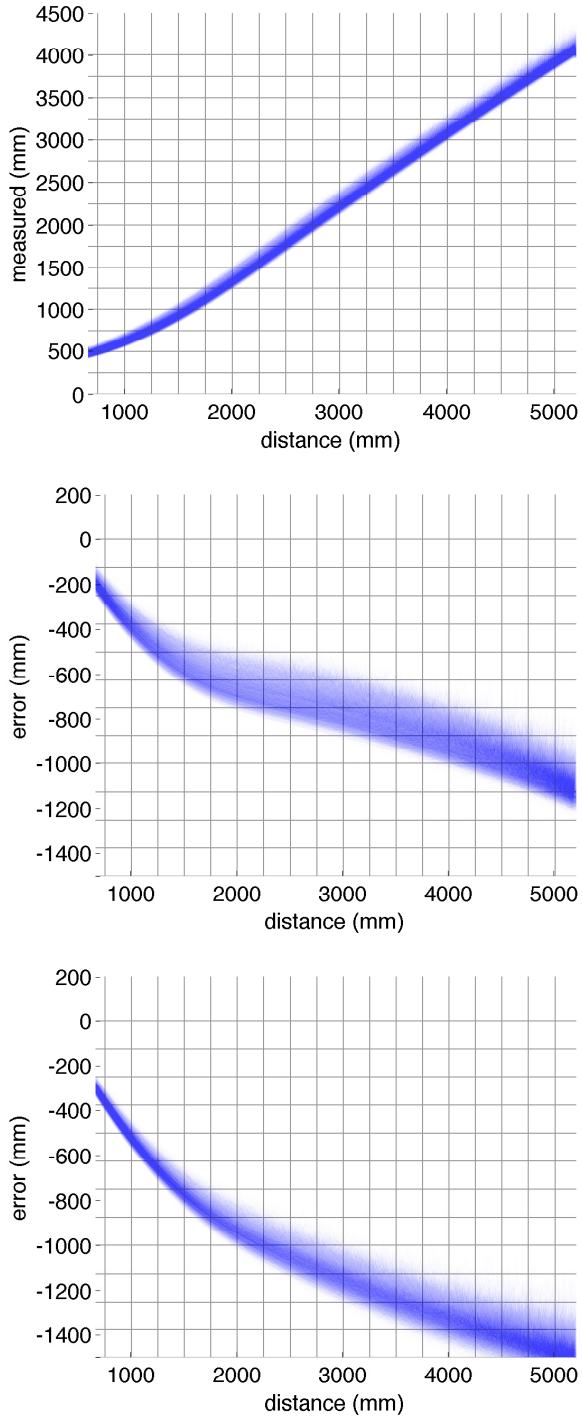

Figure 18: The measured distance (top) and the distance errors of all pixels (center and bottom). The center image shows the distance error that serves as basis for the 5th degree polynomial correction. The bottom image shows the distance error after applying the charge ratio correction that is discussed in section 4.3.1.

4.2.2.3 *Error correction*

The error done by the camera regarding a single reference pixel can be approximated using a 5th degree polynomial like

$$Error_{reference}(d) = a_1x^5 + a_2x^4 + a_3x^3 + a_4x^2 + a_5x + a_6 \quad (33)$$

which is used to correct the distance $d_{u,v}$ for all pixels $(u, v) \in [0, 159]x[0, 119]$ like

$$d_{u,v} = m_{u,v} - Error_{reference}(m_{u,v}) \quad (34)$$

After applying the correction to the measured data the overall error can be reduced as seen in the top left graph in figure 19. The individual errors of the pixels vary in a range from -100mm to 300mm. This is caused by the different errors from the individual pixels.

4.2.2.4 *Pixel related error correction*

Having individual functions for each pixel instead of one to correct these errors would be complex and memory intensive. Instead a pixel wise linear adjustment regarding the reference pixel is done. This produces a set of polynomials of lower degrees

$$Error_{pixel}(u, v, d) = b_1x + b_2 \quad (35)$$

that can be used for an additional pixel dependent correction. Equation 34 is extended to

$$d_{u,v} = m_{u,v} - Error_{reference}(m_{u,v}) - Error_{pixel}(u, v, m_{u,v}) \quad (36)$$

The correction function C (equation 31) can be designed by pixel wise applying equation 36. The impact of the additional pixel dependent correction can be seen in the top center graph in figure 19. The errors now are in a range from -50mm to 50mm in the near range.

In the span where the distance is greater than $3m$ the error tends to become bigger again. This can be explained by the inverse square root law [Heco2]. Just increasing the illumination to improve the higher distance values is counterproductive, since it would lead to oversaturation in the lower distance range. Therefore an advanced calibration is used.

4.2.2.5 *Adaptive intensity calibration*

A scene that is predestined for the application of an adaptive intensity correction contains multiple objects in different distances to the camera. The camera would allow to read out the

intensity of the shutters during accumulation, without a reset of the collected charge. Multiple readouts for the individual pixels would help to find the right intensity setting for each pixel. Stopping the readout shortly before a pixel reaches the maximum valid number would lead to a well illuminated image. This procedure is not designated by the current state of the camera and should be done by the camera's hardware for best performance. Also, the accumulation time and number of readouts is limited by the demand to retain a high frame rate.

In order to see if this idea improves the overall distance quality, another test was performed. The measurement and calibration was repeated using three different intensity settings. The intensities were chosen to be optimal in different ranges. One intensity is optimized for the near $(0.5 + m)$ range, one for the medium $(2.0 + m)$ and another for the far $(4.0 + m)$ distance range. The specific settings were found by moving the rail to each of the three distances, and increasing the integration time of the camera to a value that is close to, but below, the oversaturation threshold. Then for each of the parameter sets an individual distance calibration was performed. Each measurement has an optimal range which is expected to be where the returning intensity is: still far away from noise but not too high for oversaturation. The set of three different calibrations was used for a further improvement of the distance accuracy. Having three different frame sets for each range gives the option to chose the best frame set. Here a frame set is the best if its intensities are the highest below oversaturation. The result of this calibration can be seen in the top right graph in figure 19. Now, also the distances over $3m$ are in a range of -50mm to 50mm.

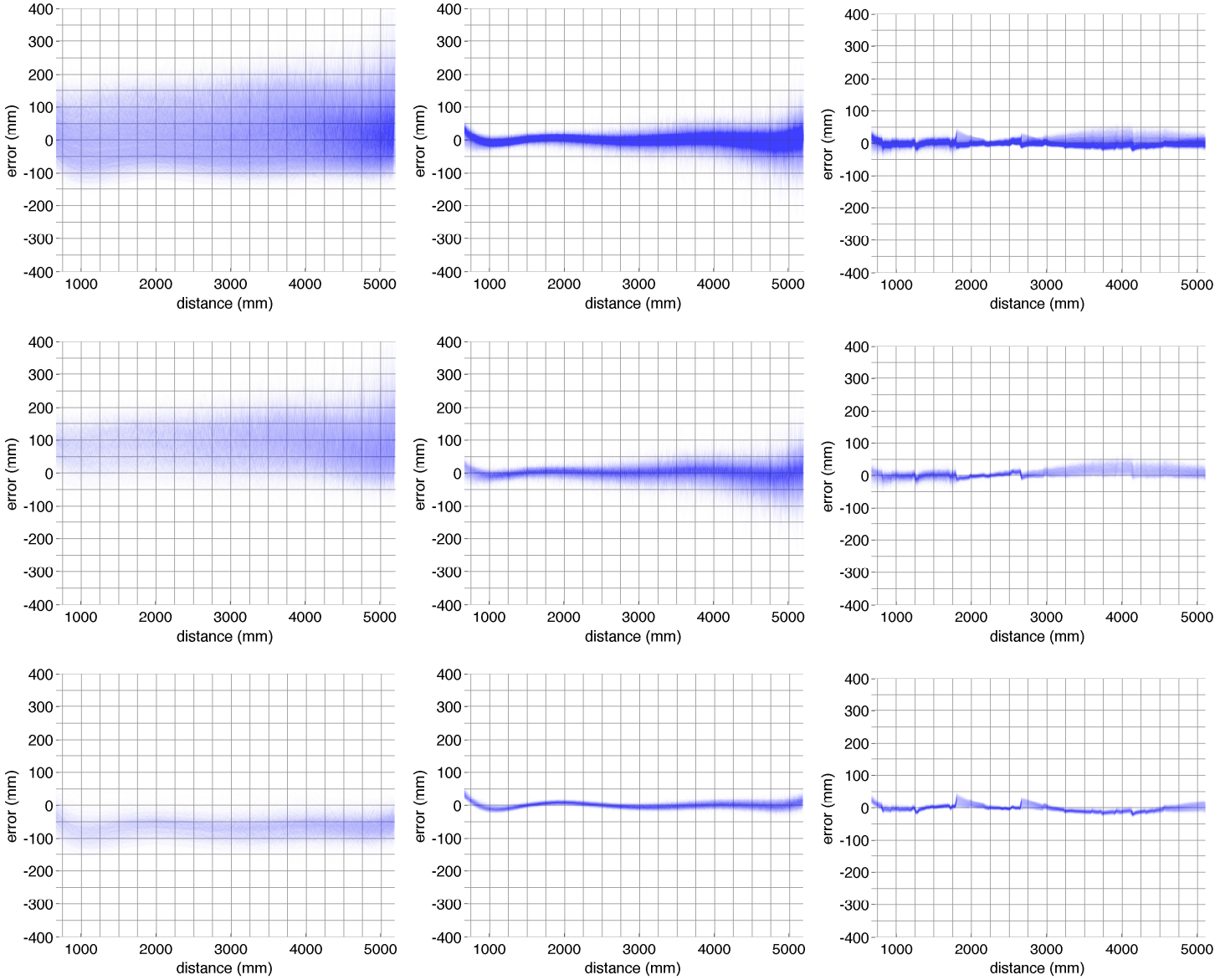

Figure 19: Remaining distance errors after correction with 5th degree polynomial (left column), additional pixel wise correction (center column), and adaptive intensity correction (right column). The graphs in the top row show the errors of all pixels. In the center row only pixels close to the border, in the bottom row only pixels in the center of the image are used.

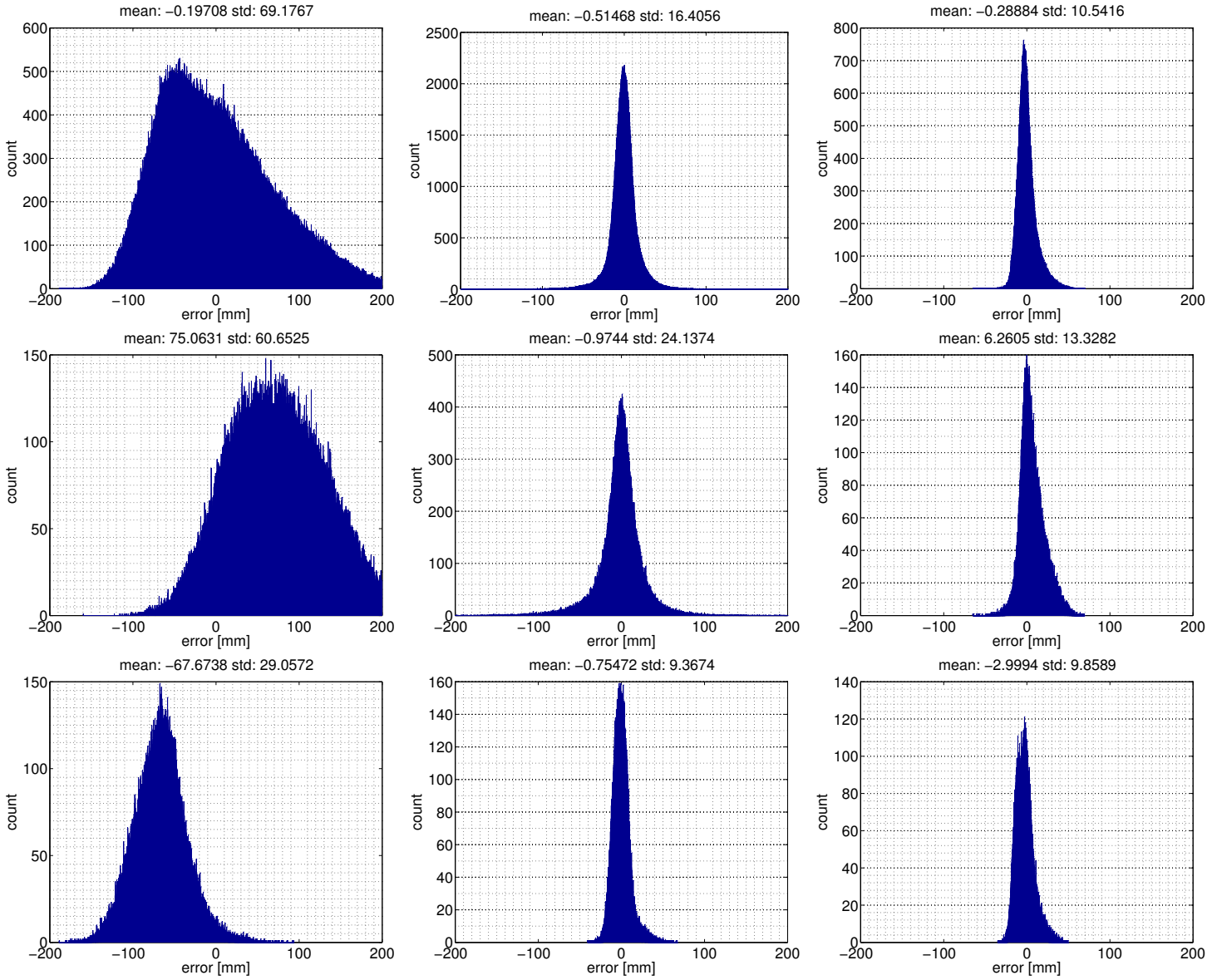

Figure 20: Histogram, mean and standard deviation of the distance errors after correction with 5th degree polynomial (left column), additional pixel wise correction (center column), and adaptive intensity correction (right column). The graphs in the top row show the errors of all pixels. In the center row only pixels close to the border, in the bottom row only pixels in the center of the image are used.

4.2.2.6 *Comparing the calibration results*

The result of the calibration is a function that takes a distance image and corrects the measured distances. By gradually improving the function, the distance errors could be further reduced.

While analyzing the distance images, the distribution of the light spot of the illumination unit attracted attention. When the camera is placed towards a wall the image center is well illuminated, but the pixels in the corners are not, see figure 21.
To see how the different intense regions perform, the graphs

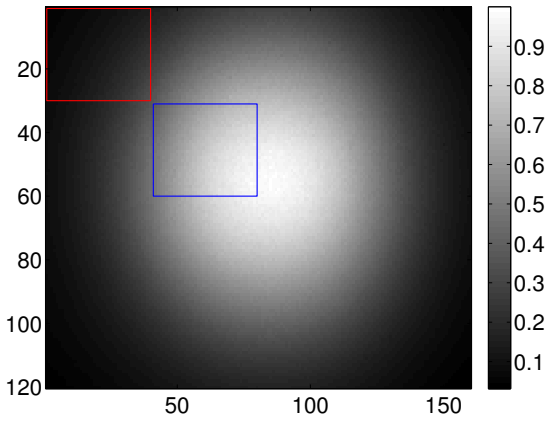

Figure 21: Intensities of each pixel when the camera is placed 1.5m in front of a wall.

were additionally plotted for two subsets of the image pixels. The first subset contains 40×30 dark pixels from the top left corner (red box), the second 40×30 well illuminated pixels near the center (blue box in 21) of the image. The calibration results of the dark pixels can be seen in the center row in figure 19. The bottom row of figure 19 belongs to the center pixels.

It can be seen that the illumination of the pixels has a huge impact on the calibration results. To compare the results, figure 20 shows a histogram, the mean, and the standard deviation of the errors after each correction. It can be seen that the result of the adaptive correction for all pixels (top right histogram) is very similar to the result of the pixel individual correction of only the bright pixels (bottom center histogram).

4.2.2.7 *Calibration speed*

The calibration process is very time consuming. To move the camera from the start to the end of the rail in 2*cm* steps, and

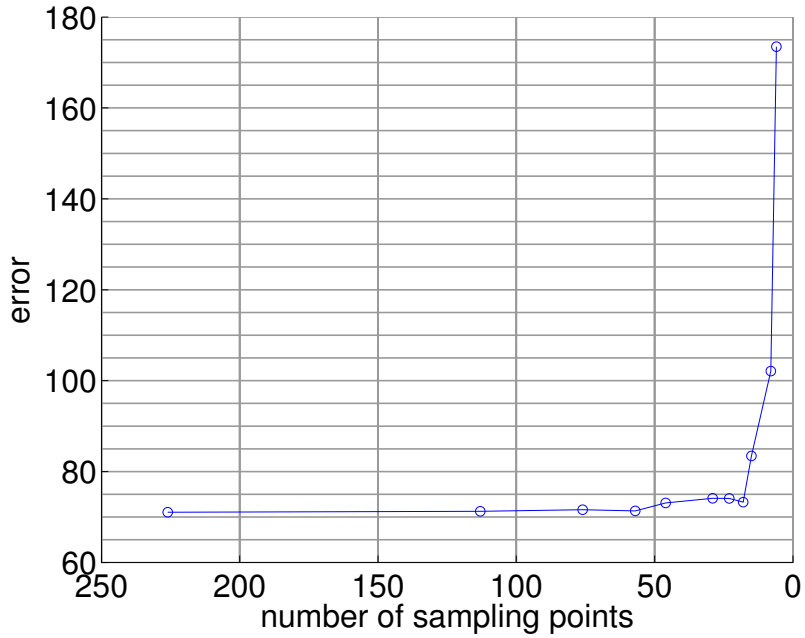

Figure 22: Quality of the 5th degree polynomial vs. the number of sampling points. Note that the interpolation of 226 sampling points already has an error of 76.

capture 226 distance frames, that in turn were averaged from 150 subframes, takes up to 3 hours. This time could be halved if the step size would be doubled, but this would on the other hand lead to a more inaccurate approximation of the correction polynomial. To analyze this, the number of sampling points for the polynomial was gradually decreased. The interpolation error of the fitted polynomial at a sample point s to the real error is described like

$$error_s = ||e - p_s||_2 \tag{37}$$

by the euclidean norm of the difference vector of e and the corrected polynomial p_s. e holds the 226 distance error values of the reference pixel. p_s is a polynomial that approximates e or a subset of e when less sampling points are used. The norm of the difference describes the quality of the approximation. A lower $error_s$ implies a better p_s that fits e. Using just half of sample points to find a calibration, does not affect the quality of the polynomial. The error begins to visibly rise when less than 50 sample points are used. The result of this analysis can be seen in figure 22.

4.2.2.8 *Calculation speed*

For each pixel a polynomial of 5th degree needs to be evaluated, if this correction method is implemented. In context of real time scene analysis, it could become an issue when 160×120 or more pixels need to be corrected at once. On the other hand the problem can be parallelized to a large degree and consequently scales with the number of available CPUs or shader units in the GPU. A method to reduce the number of multiplications in the calculation is to evaluate the polynomial

$$a_0x^5 + a_1x^4 + a_2x^3 + a_3x^2 + a_4x + a_5 \tag{38}$$

like this

$$a_5 + x \cdot (a_4 + x \cdot (a_3 + x \cdot (a_2 + x \cdot (a_1 + x \cdot (a_0))))) \tag{39}$$

The difference is that equation 38 needs $5 + 4 + 3 + 2 + 1 = 15$ multiplications, where equation 39 needs only 5 multiplications due to the precedence of operators in C/C++. [cpl15]

4.3 VOUT1 VS VOUT2

As already mentioned in section 4.1.3.2 the ability to generate electrical charge can be individual for each pixel, due to variations in the production process of the imager and inaccuracies during the assembly of the camera system.

To further analyze this behavior a test was performed. The equipment for the test were the camera, two $400W$ halogen beam lamps and a plain white wall. The idea of the test was to cover the camera's illumination unit and use a continuous light that provides the same illumination for both shutters instead. Shutter 1 and shutter 2 were open for the same time: $40ns$. To suppress the influence of the alternating current to the light, 5000 frames were recorded. The light did not point directly into the camera, but to a wall were the camera was directed to. To preempt that a certain fixed light position could influence the measurement, the two emitters were continual moved by hand in all directions. The emitters were placed left and right from the camera. Brighter and darker intensities were generated by increasing and decreasing the distance between the camera and the illuminated wall.

Each of the recorded frames consist of shutter values that would be declared as *dark* and *light* in a normal measurement. Since the camera's illumination unit is covered, the *dark* and *light* values do not differ. This doubles the effective number of frames to 10000.

For each pixel (u, v) of each frame i the ratio between the $Vout1$ and $Vout2$ value was computed like

$$Ratio_{u,v,i} = \frac{Vout1_{u,v,i}}{Vout2_{u,v,i}} \tag{40}$$

and the frame with all ratios by

$$finalRatio_{u,v} = median_{\forall i}\left(Ratio_{u,v,i}\right) \tag{41}$$

where $i \in 1, ..., 10000$ denotes the number of the frame, $u \in [1, ..., 160]$ and $v \in [1, ..., 120]$ are the pixel indices.

In theory, since the accumulation time for $Vout1$ and $Vout2$ is the same and the emitters can not distinguish between them, the accumulated charge should be the same for $Vout1$ and $Vout2$. This means that equation 40 should always evaluate to 1.0. It turned out that this is not the case.

The left graph in Figure 23 shows the results of this test. The ratios spread a gradient that goes from 0.7 in the top left to 0.9 in the bottom right corner. The right graph in figure 23

shows the results of the same test but without a lens. Here the gradient is visible too, though it is less distinct.

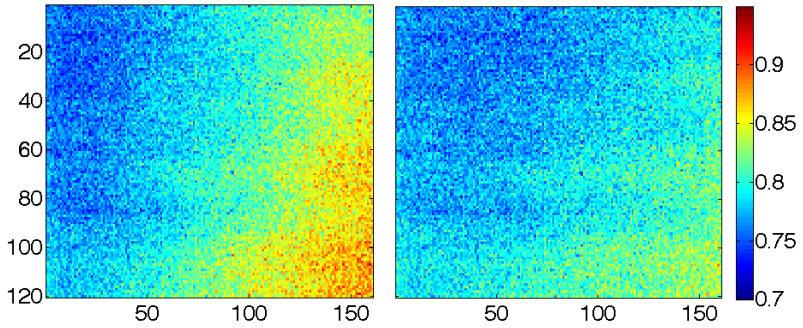

Figure 23: Collected charge ratio $\frac{Vout1}{Vout2}$ for each pixel, with lens (left) and without lens (right).

The regional ratio differences were further studied to research whether the intensity of the illumination has any influence to the ratio. In section 4.1.3.2 the ability to generate charge from photons in the pixel was modeled with a single slope a, see formula 22 and 23. It turns out that this assumption was close to correct, as seen in figure 24. Each graph shows all recorded intensities for a pixel.

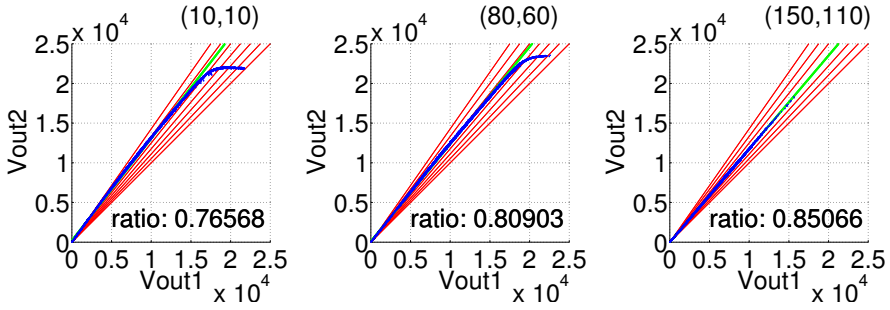

Figure 24: Intensity related charge ratio for the top left, center, and bottom right pixels. The charges of $Vout1$ are on the x- and the corresponding $Vout2$ values on the y-axis. The red lines represent the slopes from 0.7, 0.75, ..., 0.95, 1.0 from left to right. The green line represents the fitted ratio.

This additional analysis of the intensity related charge ratio was done by plotting a point cloud from all measured intensities for a single pixel. The position of each point was

found by using the $Vout1$ value as x coordinate and the $Vout2$ value as y coordinate. To compare the slope of the point cloud the red lines can be used. The ratios the lines represent are $0.7, 0.75, ..., 0.95, 1.0$ from left to right. Also, the exact charge ratio for all pixels that have a charge below 20000 was computed. It is represented by the green line. It can be seen that the $(80, 60)$ pixel follows a charge ratio of 0.80903, before it overturns at intensities that become closer to the oversaturation threshold of 22500.

4.3.1 *Influence on a distance image*

As seen in section 4.1.3.2 at equation 22 the slope of the charge accumulation is expected to be equal in the distance computation. Since this is obviously not the case, a systematic error is accepted. Formula 26 is used to compute the distance from the measured $Vout1$ and $Vout2$ values. To visualize the impact of the real $Vout1$ by $Vout2$ ratio on a distance measurement the following simulation has been performed. For each distance from 0 to 5 meters the theoretical expected $Vout1$ and $Vout2$ values were computed and stored in an array. Then the $Vout1$ values were altered to $Vout1'$ and used with their corresponding $Vout2$ values to recompute the distance. The difference between the real and recomputed distances are shown in the following figures. Figure 25 shows the impact of different slopes to the distance error. The alternation of $Vout1$ to $Vout1'$ was done like

$$Vout1'_d = s * Vout1_d \tag{42}$$

where $d \in [0, 5]m$ represents the distance and $s \in [0.6, 0.062, 0.064, ..., 1.0]$ the slope. Clearly the case where the slope is 1.0 results in an error of 0, which makes the dark red line in the image. The error is proportional to the difference between $Vout1$ and $Vout1'$. When the slope is 0.6 the error goes up to 0.75 meters, as seen by the light red curve.

4.3.2 *Explanations for the rail distance error*

In section 4.2.2.4 the distance error from the rail test did not resemble the curves that were anticipated, compare figure 25. The rail test distance error was negative proportional with higher distance, see center graph in figure 18. That the errors do not match can be explained by the fact that this scenario assumes absolutely perfect conditions of all camera parts, except the al-

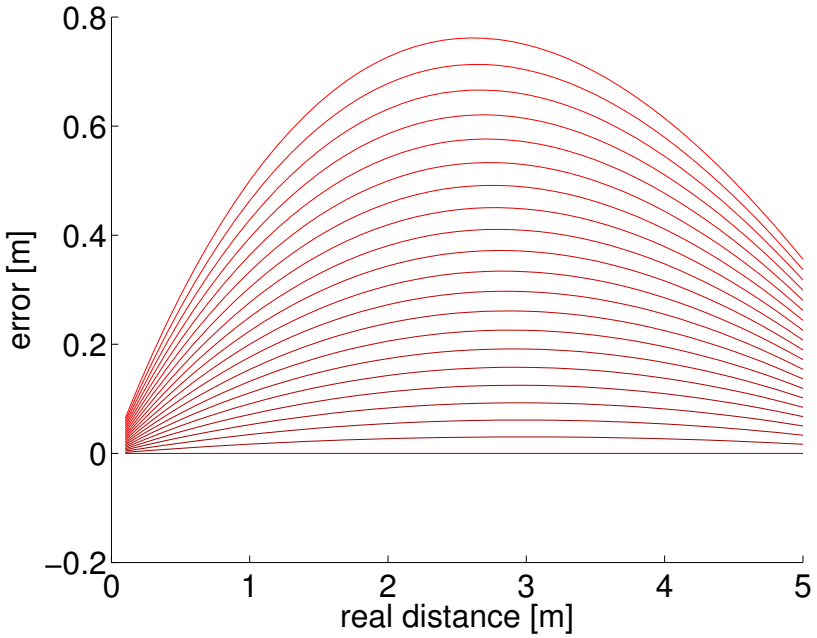

Figure 25: The theoretical distance error caused by a $Vout1/Vout2$-ratio that is 1.0 (dark red line) to 0.6 (light red line) with steps of 0.02.

tered charge ratio. Additionally, the camera system was already calibrated to deliver equal values for $Vout1$ and $Vout2$ by the light pulse delay calibration, see section 4.2.1.1. The camera system unfortunately does not work that way, so this scenario is not able to fully explain the rail distance deviation.

Figure 26 shows the charge ratios for different scenarios. In the optimal case (green) the ratio always is 1, which results in an identity line. The 0.75 scenario (orange) was really measured for the $(10, 10)$ pixel in this test.

Additionally, it is interesting to see the influence of an additional offset to the slope. The 0.7 plus 0.3 offset scenario (red) is one example. The offset would mean that the $Vout1$ shutter would always contain a charge. To simulate this, the alternation of $Vout1$ to $Vout1'$ is done like

$$Vout1'_d = 0.7 * Vout1_d + o \tag{43}$$

where the slope now is set to a fixed value of 0.7 and again $d \in [0, 5]m$, but now the additional offset $o \in [0.0, 0.02, 0.04, ..., 0.3]$ is added. $Vout1$ in the higher distances normally goes to 0, now it is still existing due to the offset. The result of this simulation is visible in figure 27. Here the lowest curve, that represents $Vout1'_d = 0.7 \cdot Vout1_d + 0.3$, comes closest to the rail measured

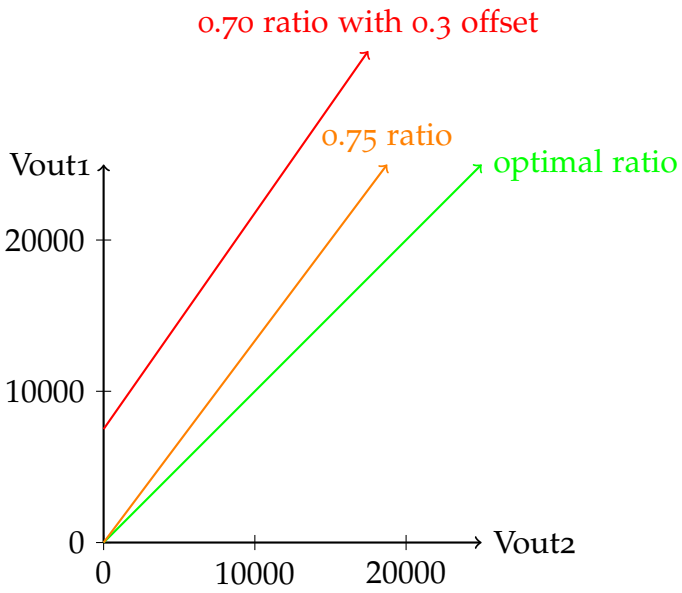

Figure 26: Possible ratio curves for different $Vout1/Vout2$ ratios.

distance error in the center graph of figure 18. The 0.7 slope plus 0.3 offset scenario (red) matches the measured rail test error the most, but it does not work as an explanation because the offset is to huge to be realistic.

The attempt to use the matrix of measured charge ratios, which is visualized in figure 23, to correct the measured charges for $Vout1$ in the distance calibration did not lead to better results. On top of this, the measured distance error became worse, see bottom graph in figure 18. For this another 5th degree correction polynomial was generated. The distance error correction with the new polynomial had comparable results to the previous correction in section 4.2.2.3.

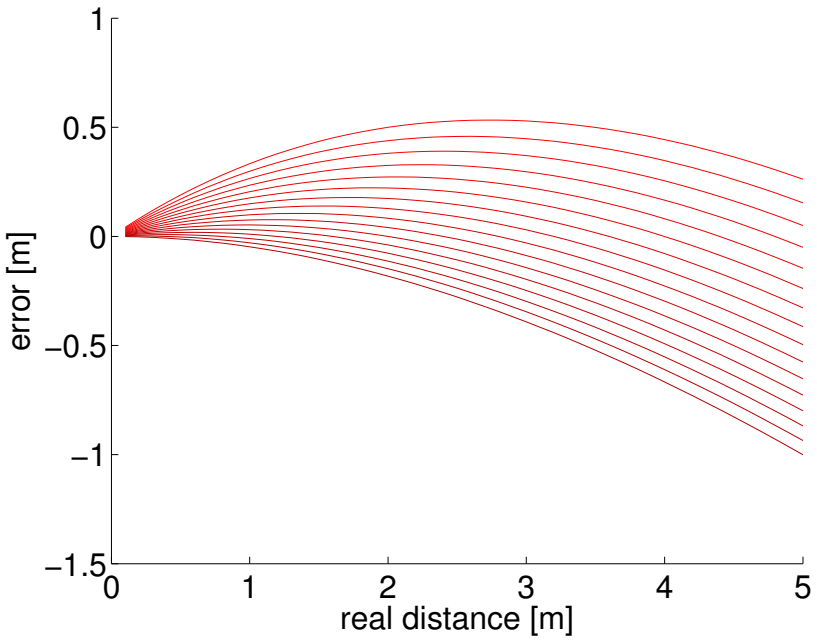

Figure 27: The theoretical distance error caused by a $Vout1/Vout2$-ratio that is 0.7 with additional offsets from 0.0 (light red line) to 0.3 (dark red line) with steps of 0.02.

5

IMPLEMENTATION

In this chapter the implementation of core parts is explained. The process of capturing camera data and the decoding of single camera frames is explained in more detail.

The camera is connected to the computer via gigabit Ethernet and sends the data in UDP packages. It comes with a software that could visualize and save the recorded data. Unfortunately, the software did not provide an API to automate the measurements. Therefore, an API to operate the camera and a command line based camera program was written. In the following the UPD commands that are used by the API and the structure of the API is explained. In the last section of this chapter a method to visualize the recorded data in 3D is shown.

5.1 CAMERA UDP COMMANDS

The imager and the lamp in this camera are controlled by a FPGA that was programmed by Hamamatsu. The software can only be influenced with the commands that are described in the following. To communicate with the camera, several UDP commands can be used. See table 2 for a short overview over those. Then, the structure of the parameters that contain data is explained in more detail.

ENDIANNESS Endianness defines the byte order of the hexadecimal representation of a number. The camera system sends and receives data in little endian order, which means that a number is byte wise ordered that the least significant byte gets the lowest memory address. For example

$$300000003_{10} = 11E1A303_{16} \tag{44}$$

is true, but the hexadecimal value can be stored in two different ways in the computer's memory. So the following equation is also valid:

$$0x11E1A303_{big\ endian} = 0x03A3E111_{little\ endian} \tag{45}$$

Name	**UDP datagram prefix**
reseto	$0x4806040001000000$
reset1	$0x4806040000000000$
response	$0x0006000000000000$
Request to reset the system. The Hamamatsu program sends reseto and reset1 for a reset.	
request parameters	$0x480102000000$
response	$0x0001000058000000$...
Requests the current parameters.	
set parameter	$0x48025800$...
response	$0x0002000000000000$
Sets new parameters.	
start measurement	$0x480302000000$
response	$0x0003$...
Request to start the measurement.	
stop measurement	$0x48070000$
response	$0x0007000000000000$
Request to stop the measurement.	

Table 2: UDP commands to operate the camera.

5.1.1 *System reset*

The reset0 and reset1 commands can be used to reset the camera to its default state. What exactly happens inside the camera controller is not clearly defined. The reset does not reset the parameters to their default values. Also for yet to be known reasons the reset was split in two commands. This behavior was learned from the original Hamamatsu camera program and adapted.

5.1.2 *Parameter request*

A parameter request can be used to read out the current camera parameters. The response of the camera will have the following form:

$0x0001000058000000$			
Tacc	Ext_res width	ldpos	light pulse width
light pulse delay	VTX1 width	VTX2 width	VTX3 width
phipulse	adtiming	Number of pixels(H)	Number of pixels(V)
Vref	Vr	Vsf/Vref2	Vpg
MCLK	Trans Line	mode	Trans Packet
phienable	Vref33		

Where each cell represents the parameter to set, encoded as four byte little endian hexadecimal number. The datagram has a length of 96byte. [Ham14]

5.1.3 *Parameter set request*

To set the parameters one can sent a datagram like this.

$0x480102000000$			
Tacc	Ext_res width	ldpos	light pulse width
light pulse delay	VTX1 width	VTX2 width	VTX3 width
phipulse	adtiming	Number of pixels(H)	Number of pixels(V)
Vref	Vr	Vsf/Vref2	Vpg
MCLK	Trans Line	mode	Trans Packet
phienable	Vref33		

Each cell represents the parameter to set, encoded as four byte little endian hexadecimal number. The datagram has a length of 92byte. As most of the parameters are fixed, it is a possible to receive the current camera parameters, change the desired ones and send them back to the camera. [Ham14]

5.1.4 *Start measurement*

When the camera receives the command to start the measurement it begins capturing and sends the data in several packages. The response UDP datagrams have the following form:

$0x0003$	transmission	size	line
data / type	data	data	data
data	data	data	data
data	data	data	data
data	data	data	...

Where each cell, except the prefix, represents a number in its two byte little endian hexadecimal from. The sending continues until a *stop measurement* command is sent to the camera.

TRANSMISSION describes the number of the distance frame. The values are 0,1,2,3.... An increment occurs after 8 UDP datagrams.

SIZE defines the size of the package in bytes which is always the same: 43072.

LINE defines the number of the subframe. Possible values are 0,8,16,24,32,40,48,56.

TYPE defines whether the light was on when the subframe was recorded. When type = 0 it is a dark subframe without active illumination, when type = 1 it is a light subframe with active illumination. Note that the type is only sent in frames where line is 0,16,32 or 48. Otherwise the position in the datagram contains pixel data.

DATA is the only field that occurs more than once in each UDP datagram. Each data field holds a number that represents the charge of a pixel. The number of data elements after the *line* field is always 168×128, independent whether the first serves as type field. The 160×120 pixel of a $Vout1$ or $Vout2$ subframe are not encoded in one UDP datagram, but in two. The first field after *line* is assigned to $Vout1$, the second to $Vout2$, the third to $Vout1$ again, and so on.

To what kind of subframe a pixel data belongs depends on its position in the payload and the value that is in the *line* field. The table below shows the order the UDP datagrams are received in, and what kind of data is in the payload.

trans.	size	line	type	content of UDP datagram
0	43072	0	0	1th half of *Vout*1 & *Vout*2, dark, full
0	43072	8	data	2nd half of *Vout*1 & *Vout*2, dark, full
0	43072	16	0	1th half of *Vout*1 & *Vout*2, dark, after
0	43072	24	data	2nd half of *Vout*1 & *Vout*2, dark, after
0	43072	32	1	1th half of *Vout*1 & *Vout*2, light, full
0	43072	40	data	2nd half of *Vout*1 & *Vout*2, light, full
0	43072	48	1	1th half of *Vout*1 & *Vout*2, light, after
0	43072	56	data	2nd half of *Vout*1 & *Vout*2, light, after
1	43072	0	0	1th half of *Vout*1 & *Vout*2, dark, full
⋮	⋮	⋮	⋮	⋮

The upper half of the Vout1 and Vout2 data is encoded in the first, the lower half in the second frame. The payload in each of the UDP datagrams that contains camera data can be seen as a 168×128 matrix that contains half of the *Vout*1 and *Vout*2 values in every even and odd cell, respectively.

There are always 2688 cells that contain 1344 *Vout*1 and 1344 *Vout*2 values. Then, there is a 4 cell padding and the next 2688 cells with *Vout* data comes. This repeats 8 times.

The table indices for the *Vout*1 values are:
4 6 ... 2688 2690
2696 2698 ... 5380 5382
5388 5390 ... 8072 8074
8080 8082 ... 10764 10766
10772 10774 ... 13456 13458
13464 13466 ... 16148 16150
16310 16312 ... 18840 18842
18848 18850 ... 21532 21534

The table indices for the *Vout*2 values are:
5 7 ... 2689 2691
2697 2699 ... 5381 5383
5389 5391 ... 8073 8075
8081 8083 ... 10765 10767
10773 10775 ... 13457 13459
13465 13467 ... 16149 16151
16311 16313 ... 18841 18843
18849 18851 ... 21533 21535

When two datagrams of camera data are received, the first and second part of the *Vout* data are put in separate 168×128

matrices. The relevant 160×120 frames are found in the center, see figure 28. After eight UDP datagrams with camera data are received, all necessary data to compute one distance frame is available.

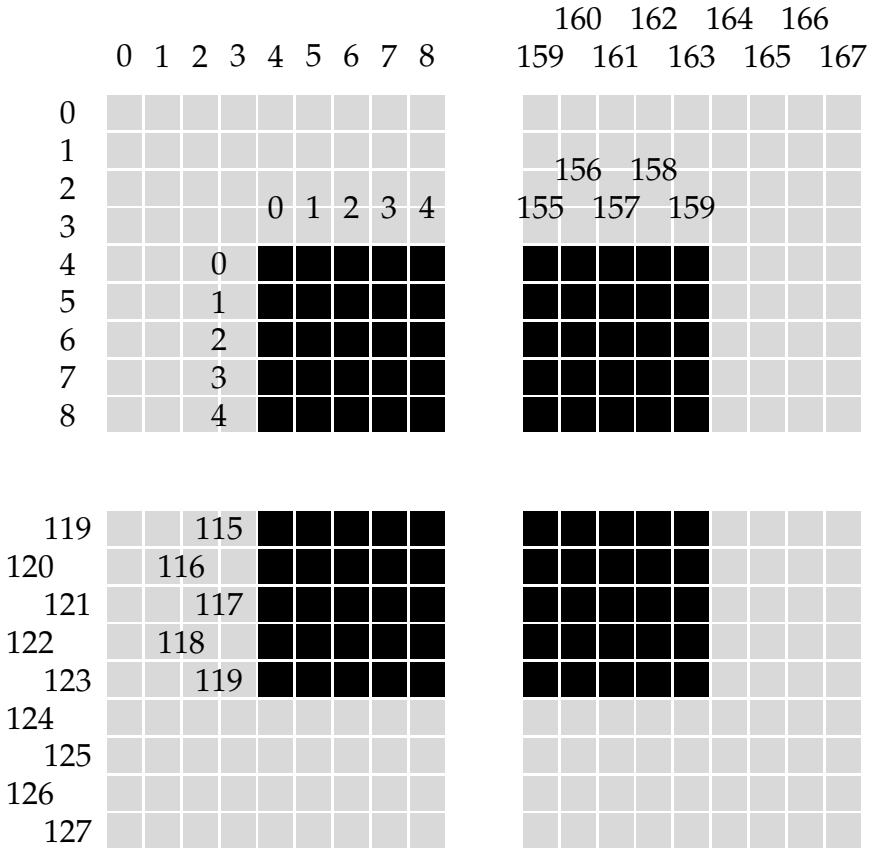

Figure 28: A 160×120 *Vout*1 or a 160×120 *Vout*2 frame that is embedded in the 168×128 matrix which was received from the camera. Each square represents a 2 byte number.

5.2 IMPLEMENTATION

A library that is able to connect to the camera was written in C++ using the Qt5 framework. The library consists of several classes and interfaces that allow to exchange data with the camera. Their interplay is explained by two scenarios. The first scenario it connecting to the camera and sending of a datagram to begin the measurement. The second is receiving a distance frame from the camera as an answer. An overview on the classes in the library and their connections can be found in figure 29.

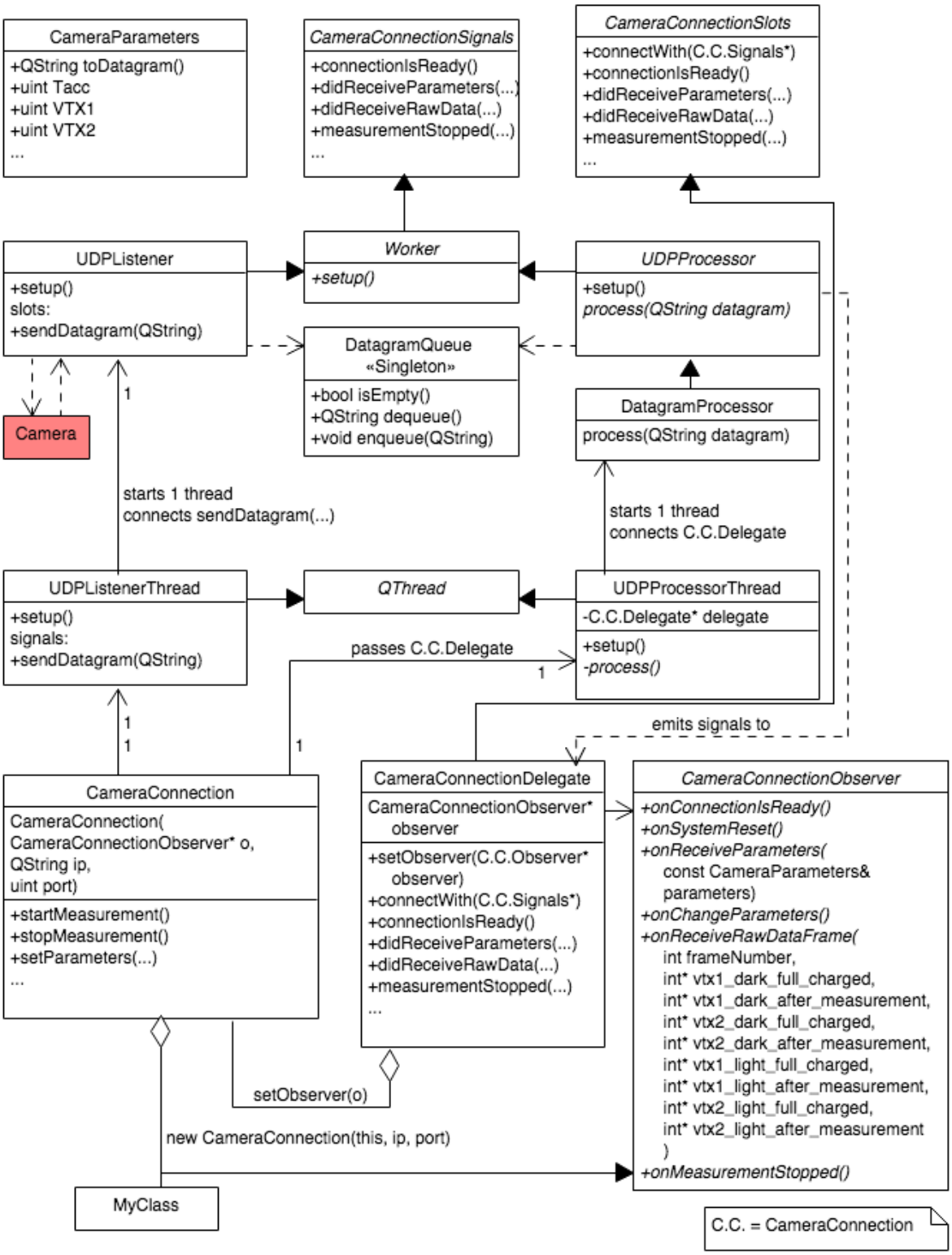

Figure 29: UML class diagram of the camera library.

5.2.1 *Establishing a connection*

In the following a sample class *MyClass* is equipped with the ability to send commands to the camera. The *CameraConnection* class can be used for that. To get feedback on the connection, *MyClass* extends the abstract class *CameraConnectionObserver*. Its methods are called when the connection can be used and frames were received. A short example of *MyClass* is given in listing 5.1.

```
1  #include<Camera.h>
2
3  class MyClass : public CameraConnectionObserver {
4   ...
5   CameraConnection* m_connection;
6   // call to establish a connection.
7   void connect() {
8    m_connection = new CameraConnection(this);
9   }
10  // This method is called when the connection
11  // is ready to use.
12  void onConnectionIsReady() {...}
13
14  void startMeasurement() {
15   // Call after onConnectionIsReady
16   m_connection->startMeasurement();
17  }
18
19  // Is called when data is received
20  void onReceiveRawDataFrame(...) {
21   ...
22  }
23 }
```

Listing 5.1: Class that uses the camera library

When in line 8 a new instance of the *CameraConnection* class is created, the *this* pointer is passed as pointer to the *CameraConnectionObserver* interface implementation.

In the constructor of *CameraConnection* a thread is created. The *UDPListenerThread* class binds a worker object called *UDPListener*. It sends and receives data to and from the camera and enqueues all received UDP datagrams in the thread safe *DatagramQueue*. When the thread was successfully started, *CameraConnectionObserver*s *onConnectionIsReady()* method is invoked.

5.2.2 *Sending commands to the camera*

To send commands to the camera, one of *CameraConnection*s methods is called. This could be *startMeasurement()* for example. This method passes the corresponding byte sequence to *UDPListenerThread* and from there to *UDPListener* using the *Qt Signal and Slot mechanism*, which is required for safe inter-thread communication. *UDPListener* then sends the datagram to the camera.

5.2.3 *Receiving camera data*

The *UDPListener* also receives all UDP datagrams that come from the camera. Because decoding of UDP datagrams takes too long to perform on the receiving thread, in *CameraConnection* another thread, the *UDPProcessorThread* is created. It has a worker object, inherited from *CameraConnectionSignals*, that is called *DatagramProcessor* and dequeues datagrams from the *DatagramQueue*. Depending on the type of the datagram, the content or information is decoded and the corresponding signal of *CameraConnectionSignals* is emitted.

CameraConnection has created an instance of *CameraConnectionDelegate* that is derived from *CameraConnectionSlots* and able to receive the emitted signals. The *connect(sender,signal,receiver,slot)* calls to connect the signals with their slots, are invoked from *UDPProcessorThread*.

CameraConnectionDelegate got a pointer to *MyClass* in form of the *CameraConnectionObserver* interface and passes full decoded distance frames to it. For this, the **onReceiveRawDataFrame(...)** method is called. As seen in listing 5.2, the number of the distance frame and the eight 160×120 *Vout* subframes are passed.

```
1  void onReceiveRawDataFrame(
2   int frameNumber,
3   int* vtx1_dark_full_charged,
4   int* vtx1_dark_after_measurement,
5   int* vtx2_dark_full_charged,
6   int* vtx2_dark_after_measurement,
7   int* vtx1_light_full_charged,
8   int* vtx1_light_after_measurement,
9   int* vtx2_light_full_charged,
10  int* vtx2_light_after_measurement
11  ) {
12    ...
```

```
13  }
```

Listing 5.2: The onReceiveRawDataFrame method in MyClass is called when a full frame was received.

5.2.4 *A data capturing session*

In a typical capturing session several commands are sent to the camera. The sequence diagram in figure 30 shows the interaction between the computer and the camera. First, a system reset is performed. As mentioned above, it consists of two parts. The system reset command is not always needed to establish a connection and receive frames, however it helps to make the camera system more reliable. When the response of the first part is received, the second is sent. After receiving the response for the second reset command, the current camera parameters can be requested. If the camera parameters match the desired ones, it might not be necessary to set them again. Then the start measurement command is sent to the camera. The camera will start to sent the captured subframes. Since UDP is not reliable, package loss can occur. If one subframe is skipped (e.g. the camera receives 0,8,24,32...), the received subframes and all following subframes with equal frame numbers should be dropped. When enough distance frames are received, the stop measurement command can be sent to the camera. It may take some subframes until the response comes and the camera stops sending.

This procedure is implemented in the *CameraTest.exe* program that writes all received frames to disk.

5.2.5 *CameraTest.exe*

A command line call to capture 100 distance frames from the camera and save them with the prefix *filename* looks like

```
CameraTest.exe -cam -numframes 100 -outfile filename
```

As explained in section 4.1.3.4, to compute a distance value for a pixel, the pixels $VTX1$ and $VTX2$ values are needed. The program captures the raw data frames from the camera and writes them to 8 different files. Each file holds 100 frames and one frame can be seen as matrix of 160×120 integer values.

The files are named

$$filename\_[vtx1|vtx2]\_[dark|light]\_[full|after].txt \qquad (46)$$

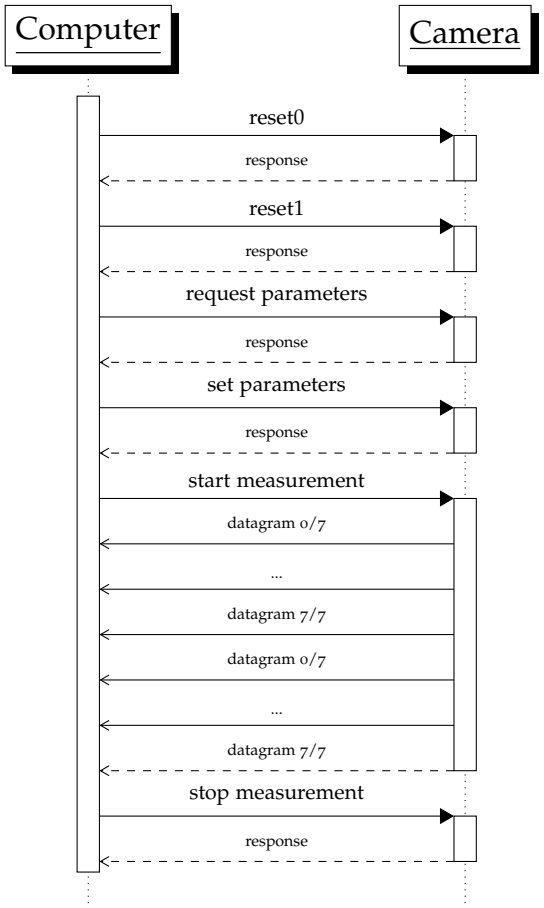

Figure 30: Sequence diagram that shows a typical example of computer and camera interaction.

Where *filename* specifies the name of the file, like chosen above. *vtx1* and *vtx2* specify whether the file belongs to the $VTX1$ or $VTX2$ shutter. *dark* and *light* denote if the file belongs to the dark (ambient light only) or light (ambient light + active illumination) frame. *full* means that the file contains the value of the shutter before photon acquisition. *after* means that the file contains the value of the shutter after photon acquisition.

By subtracting *after* from *full* the value in the shutter for one image is obtained. By subtracting the values of the *dark* from the *light* images, one obtains the charge caused by the active illumination called *vtx*. This is done for both shutters leading to two 160×120 *vtx1* and *vtx2* that can be used to compute the distance in each pixel. See figure 31.

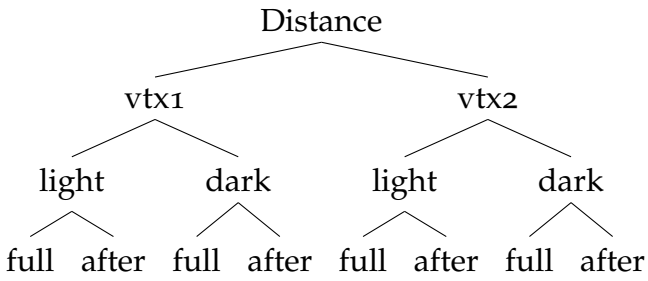

Figure 31: Hierarchical view of the different files.

5.3 VISUALIZATION

In this section the visualization of one distance frame is explained. For the visualization the programmable OpenGL rendering pipeline with vertex and fragment shader is used. In the following, the main parts that are needed for the visualization are explained. The first is the encoding and upload to the graphic memory, then a brief explanation what happens in the shaders is given.

As explained in the previous section the *onReceiveRawDataFrame(...)* method is called when a new frame is received. The first step brings the eight *int* pointers to a GPU friendly form. A 160×120 large array of the struct *VertexData* is created and filled with data. The x,y,z,w components of the *QVector4D* represent the data like: x = light_Vout1, y = light_vout2, z = dark_vout1 and w = dark_vout2. The index component is used to save the pixel index the data belongs to.

```
1 struct VertexData {
2  QVector2D index;
3  QVector4D vout; //
4  QVector4D vout_full;
5 };
```

Listing 5.3: VertexData that holds the data for one pixel.

The *VertexData* frame* is then transferred to GPU memory. For this memory is allocated and its location is written in the variable *buffer*.

```
1 unsigned int buffer;
2 glGenBuffers(1,&buffer);
3 glBindBuffer(GL_ARRAY_BUFFER, buffer);
4 glBufferData(GL_ARRAY_BUFFER, 19200 * sizeof(VertexData),
5     frame, GL_STATIC_DRAW);
```

Listing 5.4: VertexData is transfered to GPU memory.

Next the memory layout is set and the call to draw the points is invoked.

```
1  // which VBOs (frame in memory) to draw
2  glBindBuffer(GL_ARRAY_BUFFER, buffer);
3  glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, m_indexBuffer);
4
5  size_t offset = 0;
6
7  // how to locate index data
8  m_program->enableAttributeArray(m_indexAttr);
9  glVertexAttribPointer(m_indexAttr, 2, GL_FLOAT,
10    GL_FALSE, sizeof(VertexData), (const void *)offset);
11
12 offset += sizeof(QVector2D);
13
14 // how to locate vout data
15 m_program->enableAttributeArray(m_voutAttr);
16 glVertexAttribPointer(m_voutAttr, 4, GL_FLOAT,
17    GL_FALSE, sizeof(VertexData), (const void *)offset);
18
19 offset += sizeof(QVector4D);
20
21 // vout_full data
22 m_program->enableAttributeArray(m_voutFullAttr);
23 glVertexAttribPointer(m_voutFullAttr, 4, GL_FLOAT,
24    GL_FALSE, sizeof(VertexData), (const void *)offset);
25
26 // Draw geometry using indices 0,1,2,...,19199
27 glDrawElements(GL_POINTS, 19200, GL_UNSIGNED_SHORT, 0);
```

Listing 5.5: VertexData is transfered to GPU memory.

Each pixel will be drawn at an individual 3D position. Instead of computing the position once on the CPU, it is more flexible and performant to do this parallel on the GPU. The first stage of the processing that happens on the GPU is the vertex shader. It is a micro program that is called for every vertex of the geometry, here for each 19200 distance points of the distance image. The vertex shader program gets a converted form of *VertexData* as input and sets the 3D position and the intensity of the point as output. In listing 5.5 from line 8 to 24 the *index*, *vout*, and *voutFull* data that is hold in *buffer* is connected with the shader. Earlier the shader arguments locations were bound to the *m_index*, *m_voutAttr*, and *m_voutFullAttr* handles.

For each attribute location the position in memory is set. With this information the shader is able to use the correct values.

A 3D backprojection can be performed from the distance, that is computed using the *Vout* values, and the *index* that holds the pixel coordinates (u, v).

PROJECTION OF A DISTANCE PIXEL INTO 3D SPACE The result of the scheme described in section 4.1.3.5 is a 160×120 image. A single image pixel $\mathbf{p} = (u, v)$ represents a radial distance value d. The goal is to compute the corresponding 3D coordinates $\mathbf{X}$ for the pixel $\mathbf{p}$. To understand how this computation works, the projection of an object point $\mathbf{X}$ to its corresponding image point $\mathbf{p}$ is explained. Then the inverse projection delivers the projection of a distance pixel into 3D space.

In optics a common model to illustrate the projection is a pinhole camera. In the pinhole camera model, all light passes an infinitesimal aperture before it is projected onto the image plane. Each point $\mathbf{x} \in \mathbb{R}^3$ (including $\mathbf{X}$) of an object in the camera's visible range is projected onto a 2D image pixel $\mathbf{p}$ by:

$$\mathbf{p} = K\mathbf{x} \tag{47}$$

Where K is the 3×3 projection matrix that depends on the camera's intrinsic parameters. To handle the 3D to 2D projection, $\mathbf{p}$ is seen as a point in homogeneous coordinates. One can see that the $\mathbf{p}$ on the image plane will be the same for all possible $\mathbf{x}$ on the same visual line. Figure 32 illustrates this.

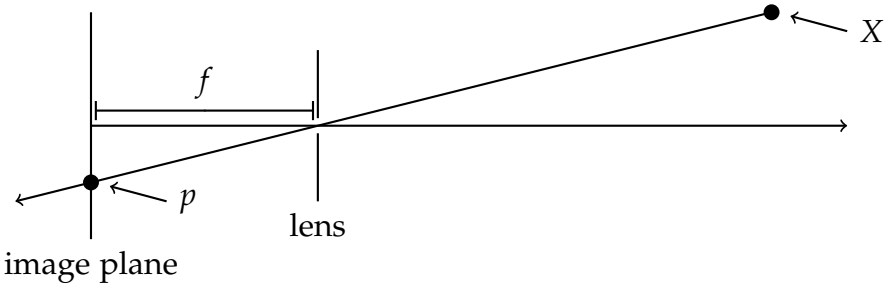

Figure 32: The projection of point X through the lens onto the image plane. All object points along this ray would be projected into the same p.

This implies that the reverse projection

$$\mathbf{x} = K^{-1}p \tag{48}$$

leads to a point $\mathbf{x}$ that must be seen as the direction $\hat{\mathbf{x}} = \frac{\mathbf{x}}{|\mathbf{x}|}$ of the visual line, where the original point $\mathbf{X}$ was on. This can be seen in figure 33.

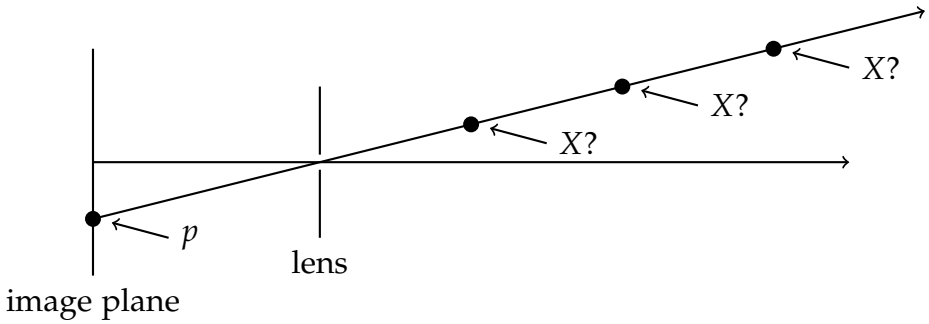

Figure 33: Backprojection of the image point p along the ray $\hat{x}$ of possible points. All points on the ray are qualified to be the original point X.

A normal 2D image would make it impossible to obtain the lines α value. This value describes how far to go from the line's origin to the original point. Since the distances are measured in a distance image, α is known as d. Figure 34 shows this. So the corresponding 3D coordinates $\mathbf{X}$ for a pixel can be computed as:

$$\mathbf{X} = d \cdot \hat{\mathbf{x}} \tag{49}$$

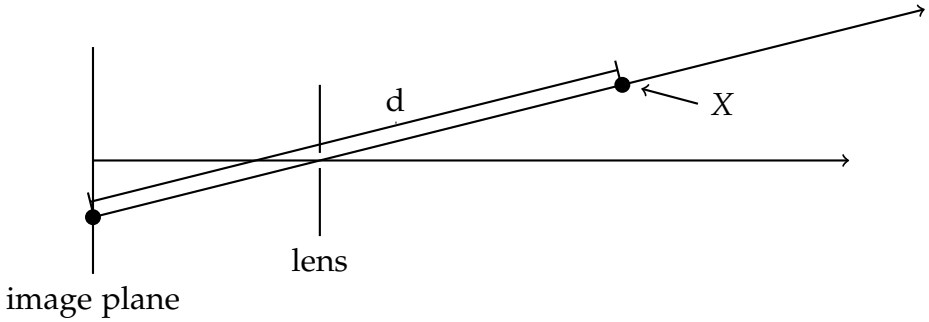

Figure 34: The original point X can be identified from the measured distance d.

When the vertex shaders are finished by setting the positions, the GPU rasterizes the geometry to screen pixels. The colors of the pixels that belong to one 3D point can be assigned in another shader program, the *fragement shader*. It is once called for each pixel on the rasterized image.

The intensity that was computed in the vertex shader is then assigned to the pixel. Depending on user selection, pixel can also be discarded.

This implementation described the core parts of the visualization program *Toffylizer* that was developed in this thesis.

5.3.1 *Toffylizer*

The evaluation of the camera's data on the basis of two dimensional graphs like in section 4.2.2 is rather technical. A 3D view of the data improves the analyzing process as it gives a better imagination of the recorded scene, see figure 35. Since the program is used for the analysis of ToF camera data it is called *Toffylizer*. The Toffylizer is written in C++ using the *Qt-Framework*[1] and the 3D visualization was programmed using *OpenGL*[2]. The program is able to read and visualize the camera data. The data can come directly from the camera, or from files with recorded data.

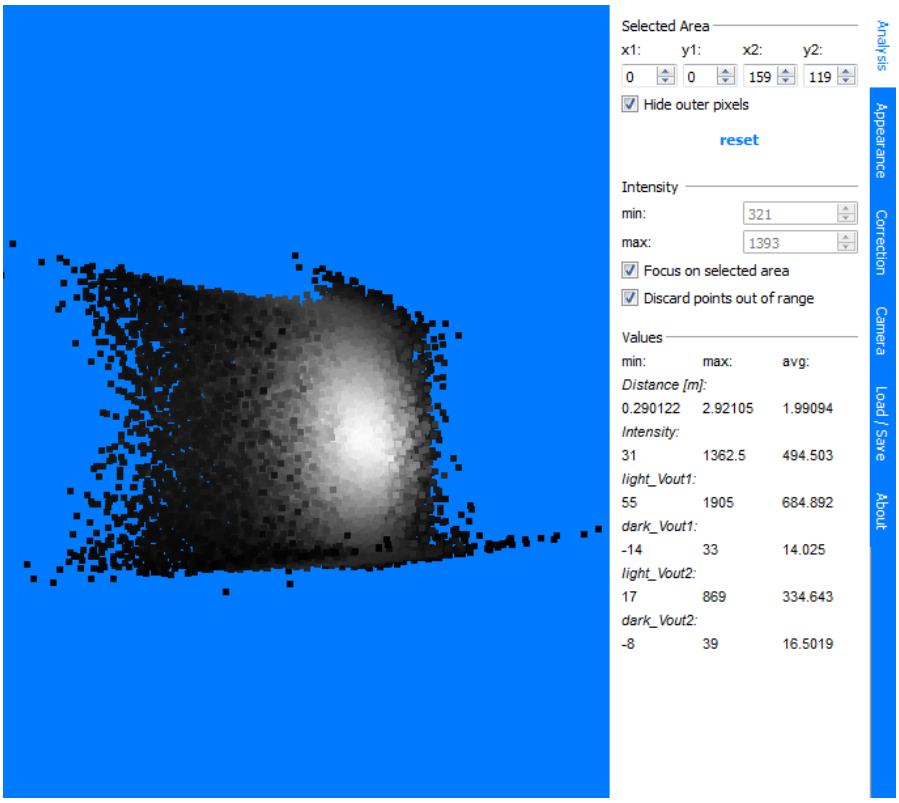

Figure 35: Toffylizer, a program to record and analyze ToF data.

A user manual of the program can be found in the attachment.

1 http://qt-project.org
2 https://www.opengl.org

6

SUMMARY AND FUTURE WORK

One goal of this thesis was to find a distance calibration for the camera system. To find a function to correct the measured distance values, the camera was placed on a movable rail and a lot of reference distance images were recorded. To programmatically record the data, an extendable C++ API was designed and implemented. At some point reverse engineering using the network reader Wireshark[1] was necessary. For a better analysis, a software was also developed, that can capture the distance images from the camera and directly visualize them in a turnable 3D view. From the C++ API a stand alone console program was derived, which made it easy to integrate the camera connection in Matlab programs.

Using a pixel individual calibration approach with multiple measurements for different intensities was successful. The overall error could be reduced to less than $\pm 5cm$ over the camera's range. It could be seen that the well illuminated pixels in the image center outperform the darker ones. For the bright pixels, the error was even lower than $\pm 3cm$.

To explain the uncalibrated camera error further studies were investigated. It was evaluated that the two shutters of one pixel have a different charge sensitivity. Unfortunately, this information could not be used to further improve the calibration result, since the error done by this effect does not correlate to the measured distance error. The result of a polynomial correction of distances that were corrected due to the $Vout1$ by $Vout2$ error lead to similar results as without a $Vout$ correction, because the polynomial did just compensate its influence. Nevertheless, its appearance deserves further studies to find out what other effects let the error increase in larger distances. A more reliable camera in an uncalibrated state could be interesting for mass production of the systems as needed in the automotive industry. A calibration that takes several hours for each camera might not be acceptable because it would be too expensive.

1 https://www.wireshark.org/

The results have shown that the polynomial just barely change when only a fraction of sampling points were used. This could be an important step in the direction of a more efficient calibration.

Since the distance calibration was very time-consuming, it might not be used in this form to calibrate camera's that are produced in mass. It could only be an option if all manufactured camera systems would lead to the same distance correction polynomial and pixel individual functions. This could not be evaluated since only one model of the system was available. Therefore, it might be interesting to find other methods of calibration. One way would be to have a better influence on camera parameters. For this the implementation of an optimized controller with a parallel readout that is faster than Ethernet is suggested. The new controller could also be able to perform some kind of adaptive intensity readout. If the pixels would stop accumulating when the charge reaches a certain value a more reliable measurement could be possible. This would provide a good illumination in each pixel. To perform a correction like in section 4.2.2 the number of pulses that were used to charge the pixel should be additionally provided. Furthermore, an optimized illumination unit could help to improve the system. The possibility to change the light pulse width could also lead to a higher accuracy in the near range. A brighter light spot that is more homogeneous over the image could reduce the intensity differences and lead to even better distance values.

The system could also be tested against further scenarios. Influences of multipath effects, different intensities, IR-reflectivities, and the appearance of moving objects can be studied.

The measurements in this thesis have shown that the errors of the distance images can be reduced using a distance calibration. It could be seen that not only the imager but also the illumination unit influences the quality of the results. The designed software will hopefully help to further understand and analyze the system.

BIBLIOGRAPHY

[ACA15] ACAM: *TDCs - Time-to-Digital Converters*. http://www.acam.de/products/time-to-digital-converters/. Version: Februar 2015

[cpl15] CPLUSPLUS.ORG: *C++ Operators - Precedence of operators*. http://www.cplusplus.com/doc/tutorial/operators/. Version: Januar 2015

[Cun15] CUNNINGHAM, Wayne: *Delphi's computer chauffeur drives me around Las Vegas at CES 2015*. http://www.cnet.com/news/delphi-computer-chauffeur-drives-me-around-las-vegas/. Version: Januar 2015

[Erz11] ERZ, Michael: *Charakterisierung von Laufzeitkamerasystemen für Lumineszenzlebensdauermessungen*, Naturwissenschaftlich - Mathematischen Gesamtfakultät der Ruprecht - Karls - Universität Heidelberg, PhD Thesis, Feb 2011

[Ham14] HAMAMATSU: *S11963-01CR Distance Image Sensor Demo Kit*. K3E-B90894(E)(S11963-01CRdemokit).pdf. Version: Januar 2014

[Hec02] HECHT, E.: *Optics*. Addison-Wesley, 2002 (Pearson education). http://books.google.de/books?id=T3ofAQAAMAAJ. – ISBN 9780321188786

[Hol07] HOLLMANN, Martin: *Radar Development in Germany*. http://www.radarworld.org/germany.html. Version: Januar 2007

[HW15] HOLGER WITTICH, Birgit P.: *VW Golf R Touch auf der CES 2015 - So funktioniert das Golf-Cockpit der Zukunft*. http://www.auto-motor-und-sport.de/news/vw-golf-r-touch-auf-der-ces-2015-so-sieht-das-golf-cockpit-der-zukunft-aus-9143845.html. Version: Januar 2015

[Ima15] IMAGING, MESA: *Standard TOF Cameras*. http://www.mesa-imaging.ch/products/product-overview/. Version: Februar 2015

[Inf15] Infineon: *Infineon 3D Image Sensor*. http://www.infineon.com/cms/de/product/sensor-ics/3d-depth-sensing/3d-image-sensor-for-consumer/infineon-registered-3d-image-sensor/channel.html?channel=5546d4614937379a0149382f21d6007a. Version: Februar 2015

[Kle14] Klein, Jonathan: *Correction of Multipath-Effects in Time-of-Flight Range Data*, Computer Graphics Group, University of Siegen, Master Thesis, Jan 2014

[Li14] Li, Larry: *Time-of-Flight Camera – An Introduction*. http://www.ti.com/lit/wp/sloa190b/sloa190b.pdf. Version: Mai 2014

[Lin10] Lindner, Marvin: *Calibration and Realtime Processing of Time-of-Flight Range Data*, Computer Graphics Group, University of Siegen, PhD Thesis, December 2010

[LLP15] LLP, Delphi A.: *Active Safety - Delphi Electronically Scanning Radar*. http://www.delphi.com/manufacturers/auto/safety/active/electronically-scanning-radar. Version: Februar 2015

[LSKK10] Lindner, Marvin ; Schiller, Ingo ; Kolb, Andreas ; Koch, Reinhard: Time-of-Flight Sensor Calibration for Accurate Range Sensing. In: *Comput. Vis. Image Underst.* 114 (2010), Dezember, Nr. 12, 1318–1328. http://dx.doi.org/10.1016/j.cviu.2009.11.002. – DOI 10.1016/j.cviu.2009.11.002. – ISSN 1077–3142

[MAI60] MAIMAN, T. H.: Stimulated Optical Radiation in Ruby. In: *Nature* 187 (1960), 08, Nr. 4736, 493–494. http://dx.doi.org/10.1038/187493a0

[Pho14a] Photonics, Hamamatsu: *Distance area image sensor S11962-01CR*. http://www.hamamatsu.com/resources/pdf/ssd/s11962-01cr_kmpd1141e.pdf. Version: 2014

[Pho14b] Photonics, Hamamatsu: *Distance area image sensor S11963-01CR*. http://www.hamamatsu.com/

resources/pdf/ssd/s11963-01cr_kmpd1142e.pdf. Version: 2014

[Pho14c] Photonics, Hamamatsu: *Distance linear image sensor S11961-01CR*. http://www.hamamatsu.com/resources/pdf/ssd/s11961-01cr_kmpd1140e.pdf. Version: 2014

[PMD15a] PMD: *Depth matters! Integrating small and ultra-thin 3D imaging*. http://pmdtec.com/products_services/reference_design_pico_pico_s.php. Version: Februar 2015

[PMD15b] PMD: *Reference Design Brief CamBoard picoS 71.19k*. http://pmdtec.com/html/pdf/PMD_RD_Brief_CB_pico_71.19k_V0103.pdf. Version: Februar 2015

[Sch11a] Schiller: *MIP-MCC: MIP-MULTICAMERACALIBRATION*. http://www.mip.informatik.uni-kiel.de/tiki-index.php?page=Calibration. Version: September 2011

[Sch11b] Schmidt, Mirko: *Analysis, Modeling and Dynamic Optimization of 3D Time-of-Flight Imaging Systems*, IWR, Fakultät für Physik und Astronomie, Univ. Heidelberg, Dissertation, 7 2011. http://www.ub.uni-heidelberg.de/archiv/12297. – started 01.07.2008

[Sys06] Systems, 3DV: *3D video cameras by 3DV*. http://web.archive.org/web/20090228203547/http://www.3dvsystems.com/technology/product.html. Version: Januar 2006

[Tri15a] Tridicam: *Flächensensor*. http://www.tridicam.de/de/produkte/flaechensensor.html. Version: Februar 2015

[Tri15b] Tridicam: *Time-of-Flight*. http://www.tridicam.de/de/technologie.html. Version: Februar 2015

[Tri15c] Tridicam: *Zeilensensor*. http://www.tridicam.de/de/produkte/zeilensensor.html. Version: Februar 2015

[Vis15] Vision, Allied: *Marlin - Acclaimed best-seller digital IEEE 1394 machine vision camera*. http://www.alliedvisiontec.com/us/products/cameras/firewire/marlin.html. Version: Januar 2015

[Wei05] Weitkamp, C.: *Lidar: Range-Resolved Optical Remote Sensing of the Atmosphere*. Springer, 2005 (Springer Series in Optical Sciences). https://books.google.de/books?id=8AT2smoj4MkC. – ISBN 9780387400754

[Wei15] Weisstein, Eric W.: *Homogeneous Coordinates*. http://mathworld.wolfram.com/HomogeneousCoordinates.html. Version: Februar 2015

[Wit15] Wittich, Holger: *Consumer Electronics Show in Las Vegas - Die Stars der CES 2015*. http://www.auto-motor-und-sport.de/news/consumer-electronics-show-in-las-vegas-die-stars-der-ces-2015-9143738.html. Version: Januar 2015

ATTACHMENT

The attached DVD contains the following data:

- **Hamamatsu CD Juni 2014** - the content of a CD that came with the camera.
- **Hamamatsu source code** - the source code of the original Hamamatsu program.
- **Literatur** - A collection of literature that was cited in the thesis.
- **matlab** - The folder that contains the code of the test and calibration.
- **Toffylizer** - The source code of the Toffylizer and the CameraTest.exe program.
- **Toffylizer Build** - A deployed version of the Toffylizer and CameraTest.exe program including sample data.
- **Thesis Simon Theiss.pdf** - The pdf version of this thesis.
- **contents.txt** - An index of the DVD with information about the folders