Correction of Multipath-Effects in Time-of-Flight Range Data

Master Thesis

Jonathan Klein

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

Supervisors: Prof. Dr. Andreas Kolb, Damien Lefloch

6th of January 2014

Korrektur von Multipath-Effekten in Time-of-Flight Tiefendaten

Masterarbeit

Jonathan Klein

Lehrstuhl für Computergraphik und Multimediasysteme
Universität Siegen

Betreuer: Prof. Dr. Andreas Kolb, Damien Lefloch

6. Januar 2014

ABSTRACT

In this thesis, we work on correcting the multipath effect that occurs in time-of-flight camera measurements.

A time-of-flight depth camera illuminates the scene with a modulated light source and uses the phase shift of the incident light to compute the distance for each pixel. However, the assumption, that the light is only reflected once in the scene is not met in reality; many objects will also reflect light that comes from indirect paths. This causes wave interference and can lead to significant errors in the depth estimation.

The thesis focuses on the multipath correction method proposed by Dorrington et al. It models the multipath effect as two interfering light components and uses multiple modulation frequencies to obtain additional information which are used to demodulate the components and correct the multipath error.

We present a successful implementation and extension of this method. We motivate many implementation choices and demonstrate how the results of Dorrington et al. can be reproduced. However, our thorough analysis of the method also shows some limitations of the approach, some of which are of a principal nature.

ZUSAMMENFASSUNG

Diese Masterarbeit beschäftigt sich mit der Korrektur des Multipath-Effekts der bei Time-of-Flight-Kameramessungen auftritt.

Eine Time-of-Flight-Tiefenbildkamera beleuchtet die Szene mit einer modulierten Lichtquelle und verwendet the Phasenunterschied des zurückgestrahlten Lichts um die Entfernung für jeden Pixel zu berechnen. Die Annahme, dass das Licht nur einmal reflektiert wird ist in der Realität jedoch nicht erfüllt, da viele Objekte auch Licht über einen indirekten Weg reflektieren. Dies verursacht Welleninterferenz und kann zu signifikanten Fehlern in der Tiefenbestimmung führen.

Diese Arbeit ist fokussiert auf die Multipath-Korrekturmethode die von Dorrington et al. veröffentlicht wurde. Sie modelliert den Multipath-Effekt als zwei interferierende Lichtkomponenten und benutzt mehrere Modulationsfrequenzen um zusätzliche Informationen zu erhalten mit deren Hilfe die Komponenten demoduliert und der Multipath-Fehler korrigiert werden kann.

Wir legen eine erfolgreiche Implementeirung und Erweiterung dieser Methode vor. Dabei motivieren wir viele Implementationsentscheidungen und demonstrieren wie die Ergebnisse von Dorrington et

al. reproduziert werden können. Allerdings zeigt unsere eingehende Analyse auch einige Einschränkungen des Verfahrens, von denen einige prinzipbedingt sind.

CONTENTS

1 FOREWORD ix

2 TIME-OF-FLIGHT PRINCIPLES 1
- 2.1 Depth Imaging Overview 2
- 2.2 Photonic-Mixer-Device Cameras 3
- 2.3 Wave reconstruction 5
- 2.4 Error Sources 10

3 THE PMD DIGICAM 15
- 3.1 Polar Coordinates 15
- 3.2 Calibration 17
- 3.3 Noise measurement 24

4 THE MULTIPATH EFFECT 29
- 4.1 Sources of Multipath Effects 29
- 4.2 Mathematical Description 31
- 4.3 Multipath Resolving 32
- 4.4 Wave Demodulation Utilizing Different Frequencies 34

5 METHOD ANALYSIS ON SIMULATED DATA 37
- 5.1 Simulation of the Multipath Effect 37
- 5.2 General Demodulation Limitations 38
- 5.3 Result Metric and Post Processing 39
- 5.4 Numerical Minimization 41

6 IMPROVEMENT OF THE DEMODULATION 49
- 6.1 A New Approach to the Minimization 49
- 6.2 Visualization 51
- 6.3 Evaluation of Algorithms and Parameters 52
- 6.4 Run Time 53
- 6.5 Noise Analysis 54

7 RESULTS ON REAL DATA 57
- 7.1 Demodulating Camera Images 57
- 7.2 Test Setup 59
- 7.3 Basic Demodulation Results 60
- 7.4 Special Purpose Test Scenes 64
- 7.5 Comparison Between the Old and New Method 68

8 CONCLUSION FUTURE WORK 71
- 8.1 Conclusion 71
- 8.2 Future work 71

A APPENDIX 73

List of Figures 80

List of Tables 81

BIBLIOGRAPHY 82

1 FOREWORD

OVERVIEW

Time-of-flight depth cameras acquire full field depth maps that are useful for many different tasks like robotics or gesture recognition. They use a modulated light source to illuminate the scene and measure the phase shift of the light induced by traveling the distance from the camera to the object and back again. This assumes, that only a single light wave is measured which is not the case in reality.

If an object reflects not only the light coming directly from the camera but also light reflected by other objects in the scene, this leads to wave interference and the measured phase does not correspond to the distance of the object anymore. This is known as multipath effect and can significantly reduce the quality of resulting depth maps. The topic of this thesis is the implementation and extension of the multipath correction method proposed by Dorrington et al. [DGC+11].

Chapter 2 describes the principals of time-of-flight cameras. Different depth imaging techniques are presented and photonic-mixer-device time-of-flight cameras are explained in detail. This includes the formulas needed to reconstruct the wave of the incident light and to extract its phase and amplitude. Furthermore, common error sources of time-of-flight cameras are explained.

Chapter 3 explains the calibration process that we performed on our camera and also contains an analysis of its noise behavior.

Chapter 4 describes the multipath effect in detail. Its different sources are explained and a state-of-the-art section covers previous correction attempts. Especially the method proposed by Dorrington et al. is covered in detail as it is the foundation of this thesis.

Chapter 5 describes, how the multipath correction can be applied. It contains evaluations of different approaches to the present optimization problem including their optimal parameters.

Chapter 6 improves this approach by introducing a new minimization function which halves the dimensionality of the search space. The new method is compared to the old one and it is analyzed, how noise affects the demodulation.

Chapter 7 applies the new method to camera data. The results of Dorrington et al. are reproduced and an analysis of other scenarios shows some limitations of the method.

ACKNOWLEDGMENTS

I would like to thank some people that helped me during my work on this thesis and without whom I would not have been able to finish it.

Damien Lefloch was my main supervisor for this thesis. In many discussions he helped me to understand the topic and to determine the next steps.

Christoph Peters was a great help for some math related problems. Most importantly he had the idea for the modification of the minimization function that is shown in 6. He also was a great proofreader with many helpful comments.

Prof. Dr. *Andreas Kolb* was my section supervisor and helped me with some of the harder problems.

Oliver Schwaneberg helped me to evaluate some ideas that concerned a quick decision tree. They did not make it into the thesis, but I am still thankful for the time that he spent to help me.

pmdTec are the manufacturers of the camera that I used. They also offered great technical support and provided me with a prototype of their latest camera model.

I would also like to thank the *authors* of the open source software that I used. The most important projects are: Python with SciPy, NumPy and matplotlib, LaTeX and Lyx, SumatraPDF, Inkscape, Notepad++, Subversion and CloudCompare.

A special thanks goes to my *family* and *friends* who always supported me and prayed for me when I had trouble with one of the many problems I encountered during this thesis. This was a great emotional help!

And I also want to thank the various *composers* of the music that I heard during my work. This also helps very much sometimes.

NANOS GIGANTUM HUMERIS INSIDENTES

2 TIME-OF-FLIGHT PRINCIPLES

depth cameras

A time-of-flight camera is a special kind of depth camera. Contrary to common cameras which measure a color value per pixel, depth cameras measure a distance for each pixel which results in a so called depth image or depth map. Where color cameras are capable of capturing material and lighting information of a scene, depth cameras capture geometrical information. The depth image can easily be converted into a point cloud which resembles the scene, but it will only contain points that are in the field of view of the camera and not occluded by other objects. Figure 1 shows an example of a depth image and its point cloud.

depth vs. distance

It should be noted, that the terms *depth* and *distance* will most of the time be used synonymously throughout this thesis. *Depth* comes from the idea, that a third dimension is added to the width and height of an image and *distance* is what this third dimension represents. So the choice of terms for a specific explanation depends only on the current point of view.

areas of application

Depth images are useful for many different applications, which include robotics, automotive, gesture recognition and home security [Sch11b, p. 1]. They can be used to find unoccupied areas or to determine the position and orientation of objects in a scene (possibly in combination with a color image). This makes depth imaging a useful and widely used technique. One of the most widely known examples is the Microsoft Kinect camera, which utilizes a depth camera for gesture control of video games.

Figure 1: Example of a depth image. Left: The depth map, where darker means nearer. Middle: A color picture taken with a normal camera from the same position. Right: A Point cloud computed from the depth map, shown from a different perspective.

2.1 DEPTH IMAGING OVERVIEW

There are various techniques for depth measurement with different advantages and disadvantages. Beside contact based approaches (which are despite of their good accuracy not suitable for many applications [Lin10, p. 5]), the majority of these techniques can be categorized into triangulation and time-of-flight [Lin10, p. 5-7].

2.1.1 *Triangulation*

stereoscopy

A natural approach (as we humans use it for our three dimensional sight) is stereoscopy, where two cameras at different positions take a picture of the scene. Due to the perspective projection, the observed position of near objects differs more between the pictures than the position of farther objects. To compute the depth, features of the first picture must be brought into correspondence with features of the second picture and the offset between them must me computed. With this offset, the distance of the features to the camera can be determined. This process is very complex and demands fast computers or dedicated hardware [Sch11b, p. 5].

structured light

The structured light approach uses a light pattern that is projected onto the scene. A camera at a different position then measures the deformation of the pattern and computes depth values from it [Sch11a, p. 10]. This approach is very similar to the stereoscopy because the light pattern can be thought of as features that have to be brought into correspondence There is no need for a second camera here as the light pattern itself acts as an image taken from the viewpoint of the projector from which the light pattern will always be undisturbed. Additionally the registration of features is simplified by the pattern because its structure is known [Sch11a, p. 10].

2.1.2 *Time-of-Flight*

The time-of-flight approach uses the fact, that light has a finite speed. The scene is illuminated by a light source and the sensor measures the time, that it took the light to fly to an object in the scene and back to the camera again. In a more general case, other signals like electromagnetic or acoustical waves can also be used [Lin10, p. 7]. There are two main categories for time-of-flight sensing, *pulse-based* systems and *continuous-wave* systems.

pulse based

Pulse-based systems emit discrete pulses of light and the sensor measures the time difference between sending and receiving the light pulse [Sch11b, p. 14]. This is done by using an extremely fast shutter in front of the image sensor, to integrate the intensity of the incident light over short windows [Sch11b, p. 15]. The time difference is then

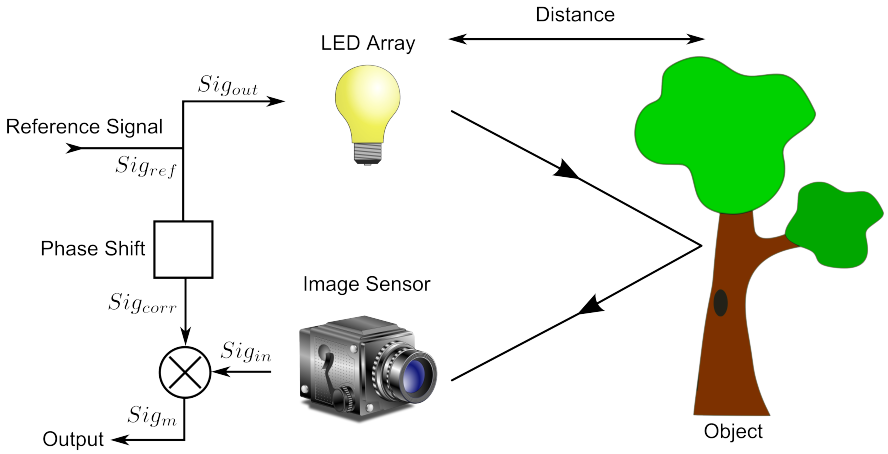

Figure 2: Basic principle of a time-of-flight camera. The phase shift of the incoming signal due to the traveled distance is measured by correlating it with a reference signal. Note that the LED array and the image sensor are roughly at the same position in reality.
(The light bulb, the camera and the tree are public domain images from http://openclipart.org/.)

calculated from the mismatch between the integration window and the incident light pulse.

Continuous-wave systems use a modulated light source and measure the phase difference between the outgoing and incoming light wave [Sch11b, p. 15]. They are explained in more detail in the next section, by means of the Photonic-Mixer-Devices.

continuous wave

2.2 PHOTONIC-MIXER-DEVICE CAMERAS

A Photonic-Mixer-Device camera (or PMD camera) is a special continuous-wave time-of-flight camera and consists of two main parts: An active illumination unit and an image sensor.

The active illumination unit emits near infrared light (with a wavelength of 780nm - 1400nm) modulated with a much longer wavelength (in the magnitude of 10 meters). In practice, only the modulation frequency is important, the wavelength of the carrier signal is mostly arbitrary, as long as it is significantly shorter than the modulation wavelength [Sch11a, p. 16]. Therefore the term wavelength or frequency will in further explanations always relate to the modulation wavelength or modulation frequency, and not to the carrier wavelength or carrier frequency.

light source

Figure 2 shows the basic principle of the distance measurement. The reference signal Sig_{ref} (with the frequency f_{mod}) is used to modulate the LED array, which illuminates the scene. The light Sig_{out} (which is equal to Sig_{ref}) travels to the object and back to the image sensor. This means that it has to travel twice the distance d (it is assumed, that the light source and the image sensor are at the same lo-

explanation of the different signals

cation) which leads to a phase shift φ. The amplitude is also changed, due to the distance attenuation and imperfect reflection on the object. This means, that the incoming Sig_{in} is a phase shifted and attenuated version of Sig_{out}.

the PMD sensor

The image sensor cannot measure the phase shift of Sig_{in} directly, it is limited to integration over time. However, the sensitivity of the special image sensor of a PMD camera can be changed at a very high frequency [Lin10, p. 10]. By using Sig_{ref} as the sensitivity, the sensor effectively measures integrals of the correlation function $\text{Sig}_{corr} = \text{Sig}_{in} \cdot \text{Sig}_{ref}$, which can be used to compute the phase as shown in section 2.3.

From the phase shift φ, the speed of light c and the angular frequency $\omega = 2 \cdot \pi \cdot f_{\text{mod}}$, the distance d can be computed:

$$d = \frac{\varphi \cdot c}{2 \cdot \omega} \tag{1}$$

It should be noted, that d is the distance between the camera and the object, and not the distance, that the light traveled, which is twice as high (this is were the $1/2$ comes from) as the light has to go from the camera to the object and back again.

As every pixel corresponds to a different part of the scene, the phase shift and thus the distance can also be different for each pixel. Therefore, it is necessary to do the whole process described above for each pixel in parallel, which makes the PMD sensor more complex compared to a normal image sensor.

hardware implementation

On the hardware side, each PMD pixel consists of two tabs (also called quantum wells), which measure the incoming light. Between these two tabs an electrical switch in form of an electrical field controls which tab collects the electrons generated by the incident photons. If the modulation frequency is used to control this switch, the number of electrons collected in one tab is approximately the product of the incoming light and the reference signal, that is a sample of the correlation function As all other electrons will be collected by the other tab, which equals a sample with a phase shift of 180°. In this way, two samples of the correlation function are measured at the same time [Sch11b, p. 18].

A major advantage of PMD sensors is, that the cross-correlation is directly done in one semiconductor based circuit, which removes the major sources of errors of conventional systems doing the detection of the optical signal and its cross-correlaction separately [Sch08, p. 13].

Compared to traditional image sensors, PMD sensors have rather big pixels for two reasons: Firstly, bigger pixels can detect more light thus increasing the depth precision and secondly, the pixel electronics are far more complex due to the cross-correlation functionality [Sch11b, p. 21]. This results in a much lower resolution compared to conventional cameras. For example, the PMD Digicam used in this thesis has a resolution of 288×352 which is already unusually high.

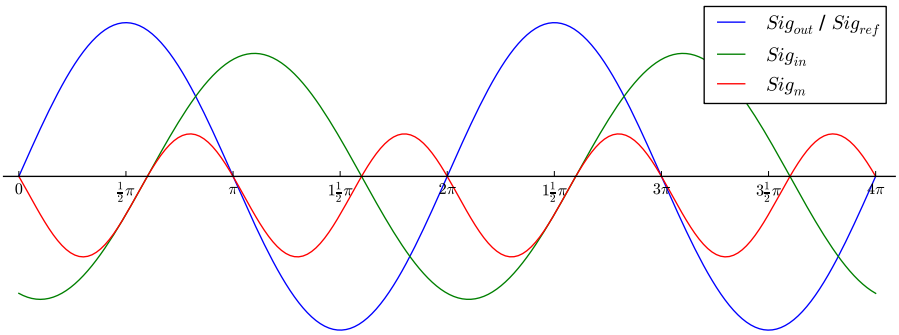

Figure 3: Two whole periods of the 4 signals. Here $\tau = 0$ which means $\text{Sig}_{out} = \text{Sig}_{ref}$.

2.3 WAVE RECONSTRUCTION

This section describes, how the phase shift and amplitude of the incident light can be computed from the sample values of the correlation function.

2.3.1 *Correlation Function Sampling*

Before deriving the computation of the phase shift and amplitude, it is necessary to formalize the involved signals, which all depend on the reference signal $\text{Sig}_{ref}(t)$:

$$\text{Sig}_{out}(t) = \text{Sig}_{ref}(t) \tag{2}$$

$$\text{Sig}_{in}(t) = a \cdot \text{Sig}_{out}(t - \varphi) \tag{3}$$

$$\text{Sig}_{corr}(t) = \text{Sig}_{ref}(t + \tau) \tag{4}$$

$$\text{Sig}_{m}(t) = \text{Sig}_{corr}(t) \cdot \text{Sig}_{in}(t) \tag{5}$$

This formulas model all the signals at a specific time t. It is important to note, that $\text{Sig}_{out}(t)$ is the light, that is leaving the camera at a specific time t (thus it equals the reference signal Sig_{ref}), and is different from the incoming light $\text{Sig}_{in}(t)$ at the same time t. $\text{Sig}_{in}(t)$ is based on an attenuated Sig_{out} from the past, with a time difference of $\frac{\varphi}{\omega}$, which results in formula 3. Sig_{corr} is built from Sig_{ref} by applying a phase shift of τ (formula 4), and the measured signal Sig_m is simply the product of Sig_{corr} and Sig_{in} (formula 5).

sinusoidal vs. rectangular

From now on, we assume the simplification that Sig_{ref} is sinusoidal ($\text{Sig}_{ref} = \sin(\omega \cdot t)$). Hence, all other signals are sinusoidal as well. In reality, this assumption is not met. We discuss this issue in section 2.4. It is also possible to work with a rectangular signal but this is not covered in this thesis. Instead, the resulting formulas can be found in [Scho8, p. 19].

Figure 3 shows a plot of the resulting signals. It should be noted, that Sig_m has a doubled frequency compared to the other signals and

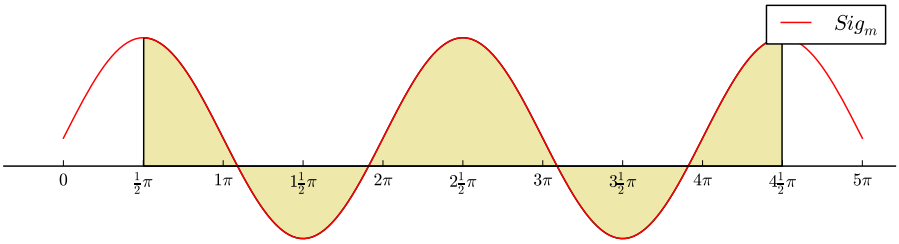

Figure 4: Integral over Sig_m. Note that in reality, the integration bounds span several million periods and not just a few, as in this picture.

an additional offset. This can be explained by the following calculation:

$$\begin{aligned}\text{Sig}_m(t) &= \text{Sig}_{corr}(t) \cdot \text{Sig}_{in}(t) \\ &= \sin(t+\tau) \cdot \sin(t-\varphi) \\ &= \frac{1}{2}\left(\cos(t+\tau-t-\varphi) - \cos(t+\tau+t+\varphi)\right) \\ &= -\frac{1}{2}\cos(2t+\tau+\varphi) + \frac{1}{2}\cos(\tau-\varphi)\end{aligned}$$

In the third line, the identity

$$\sin x \cdot \sin y = 1/2\left(\cos(x-y) - \cos(x+y)\right)$$

is used.

As the second summand does not depend on t, we have a cosine function with a doubled frequency $(2 \cdot t)$ and a constant offset (the second summand).

2.3.1.1 *Basic Sampling*

We proceed to reconstruct the resulting wave by sampling it at different points. However, the sensor can only integrate over a nonzero exposure time, which means that no specific value of Sig_m can be measured directly. This exposure time is also several orders of magnitudes longer than a single period of the signal (typically the factor is around 10^6) which is shown in figure 4. In general, the integration bounds are not bound to the period of the signal in any way, so the measurement just starts at some point and ends at another point a bit later.

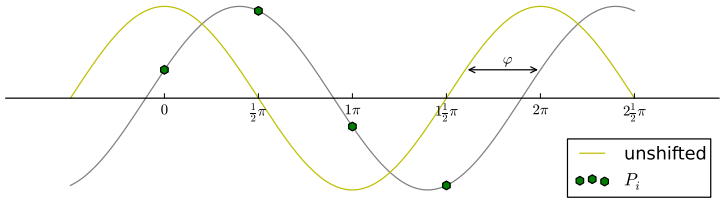

Figure 5: The correlation function of Sig_{ref} and Sig_{in} sampled at four points. From the reconstruction of this wave, the phase difference to the original wave can be computed.

When we normalize over the integration time, we get the following formula:

$$\begin{aligned}\frac{1}{t_1 - t_0}\int_{t_o}^{t_1} \text{Sig}_m(t)dt &= \frac{1}{t_1 - t_0}\int_{t_o}^{t_1} -\frac{1}{2}\cos(2t + \tau + \varphi) + \frac{1}{2}\cos(\tau - \varphi)\,dt \\ &= \frac{1}{2}\cos(\tau - \varphi) + \frac{1}{t_1 - t_0}\int_{t_o}^{t_1} -\frac{1}{2}\cos(2 \cdot t + \tau + \varphi)\,dt \\ &= \frac{1}{2}\cos(\tau - \varphi) \text{ for } t_1 \to \infty\end{aligned}$$

sampling the correlation function

As the integral over a sinus function is periodic, the error converges through the normalization to 0. And as the integration time is so much bigger than the period duration, this means that the actual start and end time are irrelevant, as long as the result is normalized. If we do multiple measurements with different τ, we can sample the function $\frac{1}{2}\cos(\tau - \varphi)$, which is the correlation function between Sig_{ref} and Sig_{in}. This way we can measure sample points which we will call D_i by integrating over a longer period. In practice, we use four sample points with a distance of $\pi/2$, hence for $i \in \{0, \ldots, 3\}$:

$$\tau_i = i \cdot \frac{\pi}{2} \tag{6}$$

$$D_i = \frac{1}{2}\cos(\tau_i - \varphi) \tag{7}$$

Figure 5 shows an example of such sample points. Once the correlation function is known, the phase shift φ and the amplitude α can computed from it as described in 2.3.2 and 2.3.3.

2.3.1.2 *Considering the Amplitudes*

The calculation in the previous section assumed an amplitude of one for both sine functions, which means, that they are completely ignored. If we add them, we get

$$\text{Sig}_m = \alpha_1 \sin(t + \tau) \cdot \alpha_2 \sin(t - \varphi)$$

Which results just in a minor change of the integral:

$$\begin{aligned}\frac{1}{t_1 - t_0}\int_{t_o}^{t_1} \mathrm{Sig}_m(t)dt &= \frac{1}{t_1 - t_0}\int_{t_o}^{t_1}\left(-\frac{1}{2}\alpha_1\alpha_2\cos\left(2t+\tau+\varphi\right)\right.\\ &\quad\left.+\frac{1}{2}\alpha_1\alpha_2\cos\left(\tau-\varphi\right)\right)dt\\ &= \frac{1}{2}\alpha_1\alpha_2\cos\left(\tau-\varphi\right) \text{ for } t_1 \to \infty\end{aligned}$$

This means, that all the sample points D_i will also be scaled by the two amplitudes.

2.3.1.3 *Background Illumination*

In equation 3 we assumed that only the emitted light from the active illumination unit will be measured by the image sensor. This is only correct for an otherwise perfectly dark scene. In a normal scene, we will always have background illumination of some kind.

We can assume, that this background illumination is constant over the time of one acquisition, which means that all electrons generated from background light will be evenly distributed among both tabs. This happens because the integral over Sig_{corr} equals 0, so no tab gets more electrons than the other, as long as their frequency is not the same as the modulation frequency. In this case, peaks in the incoming light match up with intensity peaks for one tab, which means that the other tab will receive less electrons or even none at all.

If we call the background illumination b in our formal notation, this means, that $\mathrm{Sig}_{in}(t) = \mathrm{Sig}_{out}(t-\varphi) + b$, but $\mathrm{Sig}_m \neq \sin(t+\tau)\cdot(\sin(t-\varphi)+b)$, because b will not be multiplied by Sig_{corr}. We rather get $\mathrm{Sig}_m(t) = \sin(t+\tau)\cdot\sin(t-\varphi)+b$ and thus by the same integral calculation $\frac{1}{2}\alpha_1\alpha_2\cos(\tau-\varphi)+b$.

To get correct sample points anyway we can compute them with

$$D_i = A_i - B_i \tag{8}$$

where A_i and B_i are the values from the two tabs. This will eliminate the background illumination, as shown in the following calculation.

Let $A_i' = A_i + b$ be the sample value without background illumination. Since B_i is the same as A_i with an additional phase shift of 180° and our signal is sinusoidal, we get that $A_i' = -B_i'$. Now we obtain

$$\begin{aligned}D_i &= A_i - B_i\\ &= \left(A_i' + b\right) - \left(B_i' + b\right)\\ &= \left(A_i' + b\right) + \left(A_i' - b\right)\\ &= 2\cdot A_i'\end{aligned}$$

which means that we get the desired value scaled by a constant factor (which can be ignored).

This means, that we can remove all background illumination with this computation. However, there remains a problem because too high background illumination can lead to saturation of the tabs, in which case the removal no longer works. This can especially happen in outdoor scenes with bright sunlight, but to overcome this, some PMD cameras have special suppression of background illumination circuits, that adjust the charge level in the tabs and can instantaneously reduce background illumination [Lin10, p. 10].

2.3.2 *Phase Reconstruction*

To compute the correlation function from the sample points, the arctan2 function is used. It takes the x and y component of a vector and returns the angle (between $-\pi$ and π) that this vector has in the polar coordinate system. As the correlation function is a cosine, the sample point D_0 is directly the x coordinate of the vector. Likewise the sample point D_1 is directly the y component, because the phase shift between them is $\pi/2$, which is the same as the phase shift between sine and cosine. This leads to $\varphi = \operatorname{arctan2}(D_1, D_0)$. The sample points are also scaled by the amplitudes, but as the length of the vector that we put in the arctan2 function does not matter, the whole calculation stays the same.

increased accuracy

To increase accuracy we use not only D_0 and D_1, but use the fact that D_2 should be the same as $-D_0$ (and likewise for D_1, D_3) as they are phase shifted by 180°. So $D_0 - D_2$ is in the ideal case the same as $2 \cdot D_0$ but if the values are slightly different, their difference will interpolate between the values, which makes it more robust to noise. So we get the final formula:

$$\varphi = \operatorname{arctan2}(D_1 - D_3, D_0 - D_2) \tag{9}$$

2.3.3 *Amplitude Reconstruction*

To reconstruct the amplitude from the sample points D_i, the following formula is used:

$$a = \sqrt{(D_0 - D_2)^2 + (D_1 - D_3)^2} \tag{10}$$

Figure 6 explains, why this formula is correct. As described in 2.3.2, $(D_0 - D_2)$ and $(D_1 - D_3)$ are just two sample values shifted by 90°. As the wave is sinusoidal, they can also be thought of as points on a circle, as in the figure. To compute the length of the vectors (and therefore the radius of the circle which equals the amplitude of the wave), we wood normally need an x and y coordinate, but the sample value gives us just the y (or x, depending on the point of view). However, as the two vectors are rotated by 90°, the x coordinate of the first one equals the y coordinate of the second one. So we can just use Pythagoras to get the amplitude, as is done in formula 10.

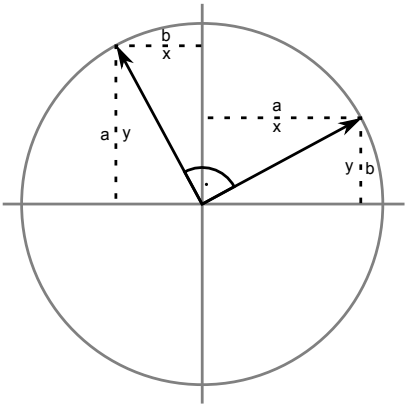

Figure 6: Derivation of the amplitude reconstruction formula. The radius of the circle can be computed with Pythagoras by using the two x or y values.

2.4 ERROR SOURCES

There is a variety of error sources affecting the measurement of PMD cameras. Some are based on model assumptions that are not met and can be overcome by calibration while others like noise are more random.

2.4.1 *Incorrect Phase Shifts*

The time-of-flight principle relies on the idea, that the distance to an object can be determined by the phase shift of a wave traveling from the camera to the object and back again. However, there are a number of situations, where this phase shift does not correspond to the distance, which leads to wrong results, even if the phase shift could be measured perfectly.

light source offset

One of these problem is, that the the light source is neither at the exact same position as the camera, nor is it punctiform. Figure 7 for instance shows a PMD camera with two big LED-Arrays which have an offset of multiple centimeters from the image sensor. Furthermore, as all LEDs have a slightly different distance to any given point in the scene, it will not only reflect one wave with a wrong phase shift but actually a mixture of different waves which all have wrong phase shifts.

multiple reflection inside the scene

Another problem is, that the light is not necessarily reflected only once in the scene. If a point not only reflects light directly from the light source but also from other objects in the scene (which are illuminated by the same source), the waves will mix and result in a new wave with wrong phase shift [DGC+11]. This is called the multipath effect and its correction is the main topic of this thesis.

multiple cameras

A similar problem arises, when multiple time-of-flight cameras are used to measure a scene at the same time. A possible solution is

to use different parameters such as different modulation frequencies [Sch11b, p. 25].

flying pixels

Another problem occurs at the edges of objects. Because the pixels have a non-zero spatial extent it can happen that light from the object and its background falls on the same pixel. This can be seen in figure 42. Although these rays do not interfere in the scene itself, they are still measured together and result in the same problems as inter scene interference.

lens errors

The lens system of the camera can also be a source of errors. It will not only cause a geometric distortion of the picture but also alter the optical path lengths of light rays through the different refractive indexes of the lens material [Sch11b, p. 23, 25].

2.4.2 *Wiggling Error*

The above wave reconstruction assumes, that all signals are sinusoidal. In reality, this assumption is often not met, as on the one hand sometimes rectangular signals are used and on the other hand the precision of the signal generator is limited.

The result is a periodic depth error, which can also be seen in figure 12 [Sch11b, p. 22]. This error can be quite big if, for instance, a sinusoidal signal is assumed (and the corresponding formulas are used) when it is rectangular in reality, the error can be up to 29% of the measured phase [Sch08, p. 23].

2.4.3 *Intensity-Related Errors*

For yet unknown reasons, the measured distance of a point depends also on its reflectivity [Lin10, p. 16, 44]. Black surfaces result in a wrong depth measurement which means that for example the black circles in the calibration pattern shown in figure 9 become clearly visible in the depth map, although they do not stick out from the wall in anyway.

The effect causes dark surfaces to be shifted towards the camera and decreases with larger distances, which indicates that it is related to the absolute amount of incoming light [Lin10, p. 46]. As it is additionally overlayed by the wiggling error, the separate correction is difficult. Nonetheless, Lindner proposes two approaches to address it.

2.4.4 *Saturation*

Time-of-flight cameras need to handle high dynamic ranges because on the one hand the light intensity decreases quadratically with the distance to the object (which means that the intensity is very high for near objects but decreases rapidly when the distance is increased)

and on the other hand interfering light from other sources is also detected [Sch11b, p. 22]. Additionally, the reflectivity of objects in the scene can be very different, which has a great impact on the measured results [Lin10, p. 11].

While underexposed pixels have a bad signal-to-noise ratio, overexposed pixels might not be able to provide any useful information at all. Both can however at least be detected, especially if the raw values and not only the derived phase and amplitude are examined [Lin10, p. 11]. If only some pixels are affected, they can also be corrected with techniques like *inpainting* or simple interpolation; additionally they can also be ignored for further processing [Sch08, p. 28].

2.4.5 *Noise*

time dependent noise

A basic problem of all image sensors is, that the charge generation by incident photons always introduces Poisson noise, which makes statistical errors in the depth measurement inevitable [Sch11b, p. 20]. This cannot be suppressed and increases with the amount of incoming photons. For large enough exposures times, it can be modeled by a normal distribution [Lin10, p. 12].

Other sources of noise are thermal noise, reset noise and dark current shot noise which are all independent from the incoming light but increase with the temperature, so cooling is an effective way to reduce this kind of noise [Lin10, p. 12].

fixed pattern noise

Another kind of noise is the constant pixel error pattern. It does not change over time but is different for each pixel on the sensor. Causes are minimal inaccuracies in the production process, such as variations in oxide thickness, size of gate area or doping concentrations [Lin10, p. 12]. Although the two common problems of conventional CCD or CMOS sensors, the *dark-response nonuniformity* and *photo-response nonuniformity* are canceled out in the phase and amplitude calculation, PMD sensors still suffer from these problems, caused by slightly varying capacities and other hardware-design and -processing related reasons [Sch08, p. 25]. But by measuring all pixels, these errors can be detected and due to their constant nature also compensated.

2.4.6 *Motion Artifacts*

If objects in the scene or the camera itself are moved during the capturing of a phase image, motion artifacts occur. They result either in a mixture of foreground and background if the object is moved perpendicular to the view direction of the camera, or in an additional phase shift, if the object is moved along the view direction. Additionally, if the reflectivity of the object texture changes this can also change the measured distance, although the true distance might stay constant [Lin10, p. 16].

Many scenes are static but especially if the camera should be used to recognize gestures, the prevention and correction of motion artifacts becomes an important problem.

2.4.7 *Unambiguity Range*

As the distance calculation is based on the phase shift, only distances less than half the wavelength (as the light also has to travel back to the camera) are unambiguous. If the distance to an object gets slightly higher, the measured phase will just jump back to the beginning of this unambiguity range. This means that it is not possible to determine the true distance purely based on the measured phase.

A solution for this problem is to also take the amplitude into account and analyze the depth map for sudden changes as proposed in Droeschel et al. [DHB10].

3 THE PMD DIGICAM

PMD DigiCam

For this thesis a prototype of the new PMD DigiCam is used. Its main features are the (compared to other depth cameras) very high resolution of 352×288 pixel, the two bright LED-arrays (which are useful for depth measurements in big rooms) and the low camera noise. The camera is shown in figure 7.

Figure 7: The PMD Digicam prototype. Above and under the lens are two powerful LED-arrays which illuminate the scene. The backside consists mainly of the cooler.

3.1 POLAR COORDINATES

polar coordinates

The pictures taken by this camera are in a polar coordinate system rather than in a Cartesian. This means, that a straight wall will appear as a curve in the depth image, because pixels measure the distance between the camera position and the point in the scene. A Cartesian coordinate system on the other hand would mean, that each pixel contains the shortest distance between the image sensor plane and the point in the scene.

Figure 8 shows the two different coordinate systems and how the shape of a flat wall in a polar camera picture appears. If d is the distance to the wall and x is the position on the wall, the length of

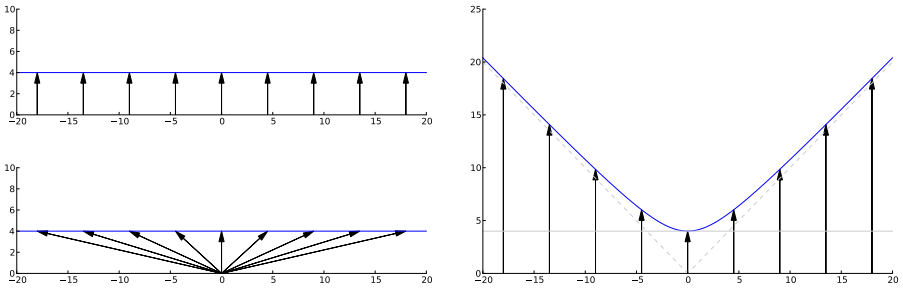

Figure 8: Cartesian (upper left), polar (lower left) and polar values on a Cartesian scale. The right image shows also the absolute value function (gray dotted) and the Cartesian distance values (gray line).

the arrows is $\sqrt{d^2 + x^2}$. Because d is constant in this case, this function converges to the absolute value function for large values of x (shown as a gray dotted line in the right picture), but for small values, the wall looks similar to a sphere.

coordinate transformation

If the focal length of the lens is known, the depth values can be converted into the Cartesian coordinate system, where the wall looks flat again. This can be done by performing the following steps:

1. Change the origin of the coordinate system. The pixel position must be moved so that the center pixel of the sensor has the coordinates $(0, 0)$. Note that the center pixel does not necessarily have the coordinates $(X_{res}/2, Y_{res}/2)$, as the image sensor might not be exactly aligned with the camera lens due to production inaccuracies. Instead, the true center pixel has to be measured in the calibration process.

2. Calculate the view direction of each pixel. The center pixel will always have a view direction parallel to the camera alignment, the view directions of the other pixels depend on their position and the pixel size relative to the focal length of the lens. If the z component is set to one, the x and y components can be computed by simply multiplying the pixel position by appropriate constants (which could be different for the x and y axis).

3. Multiply by the measured depth. If the view direction of each pixel is multiplied by the depth measured by this pixel, the resulting vector will end exactly at the point that was measured by this pixel and the three components of this vector will be the position of the point in a cartesian coordinate system.

The values in the polar coordinate system are not wrong, but their intuitive interpretation is misleading. It depends on the current use case, whether the values should be transformed, or not.

3.2 CALIBRATION

As we use a prototype of the PMD DigiCam, no calibration data is provided. This means, that we have to do everything by ourselves and to make sure, that at least the assumptions described in section 2.3 are met. Although there are some quite advanced ways for calibrating cameras ([Sch11a], [Lin10], [LNL+13]), we choose to use a fairly simple approach due to time constraints and to ensure a focus on multipath-effects.

overview

The basic idea behind calibration is to find a function mapping measured values to accurate distance values This can be done by generating a huge amount of sample points for which ground truth values are known. Than a best fitting function to approximate these samples can be searched. The critical point of this method is, that the sample points have to be representative. If sample points for a specific case do not exist, the calibration has no chance to be correct in this case, so the very first step is to decide, for which cases the camera should be calibrated.

what to calibrate

We do not want to change the lens of the camera and decide that distances between 1 and 4 meters will be sufficient for our needs. Furthermore, the modulation frequency and integration time of the camera have to be chosen which results in our case in the three different frequencies 10Mhz, 20Mhz and 30Mhz and a fixed integration time of 500ms. We find, that the lens distortion in the edges is too high to get reliable information, and thus only pixels nearer to the center are used. While the other pixels are still calibrated, one should not rely on their correctness.

calibration input and output

Another important decision is, on what level the camera should be calibrated. Choices are the raw channel values, the phase and amplitude or the final distance. For the calculations in 2.3, we need the raw channel values, so the intuitive idea is to map raw values to calibrated raw values. However it is nearly impossible to obtain ground truth raw values because this would require some kind of ground truth raw channel measurement device. Instead, our approach is to compute a calibrated phase and amplitude, build a wave based on these values and sample it. This ensures, that our raw channels match a perfect sinusoid function, which is one of the requirements in later calculations.

Normally, the input values should be the 4 raw channels, but as functions with four dimensional arguments are far more complex, we decided to use only the phase and amplitude, computed from these values as input values, which turns out to lead to sufficient results. It is however possible, that some information, that could lead to better calibration results, is lost through this decision. Furthermore, it would also be possible to use the pixel position as an input value. As described in 2.4 it is likely that the sensor contains a fixed pixel er-

Figure 9: Example pictures of the lens calibration. In total, 41 such pictures are used for the calibration.

ror pattern and the lens distortion also depends on the pixel position, which leads to a high optimization potential, but as described above, we intend a fairly simple calibration model, so in the end we only use the phase and amplitude derived from the raw channels to calculate calibrated raw channels.

For this calibration, a set of scripts based on [Zhaoo] are used.

3.2.1 *Lens Intrinsic Parameters*

The first step in calibrating the camera is to measure the so called lens intrinsic parameters. They include properties like the focal length, the pixel size and distortion parameters of the lens. To get these parameters, a script is used to automatically extract them from a series of calibration pictures.

The calibration pictures show a RIG calibration pattern from different distances and angels, three of them are shown in figure 9. The script has parameters for the size and distance of the dots and the knowledge that the should resemble a regular quad, and searches for them in the pictures. Based on the position of the points, the lens intrinsic parameters can be determined.

The script works fully automated. It was only necessary to enhance the contrast of some pictures so that the position of the dots could be found. Results are better if more pictures are used. In total, 41 pictures are used for this calibration. It should be noted, that as in this process only the lens parameters should be determined, the used frequency and integration time do not matter at all.

3.2.2 *Phase calibration*

Once the lens intrinsic parameters are available, they can be used to create ground truth information for the phase calculation.

Another script can use a picture of the same calibration pattern along with the lens intrinsic parameters, to determine the relative position and orientation of the plane, in which said pattern lies. If it is pinned as tightly as possible to a wall, one can consider the plane

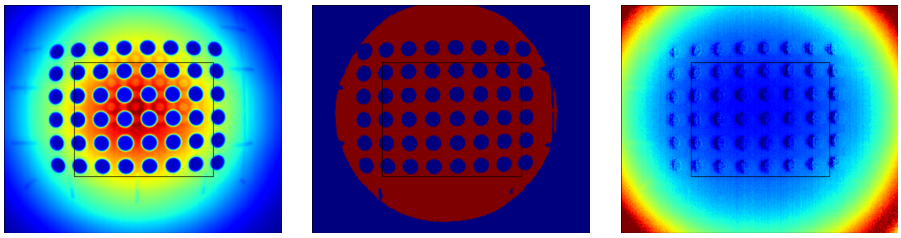

Figure 10: Amplitude, selection mask and phase values (from left to right) of a calibration picture.

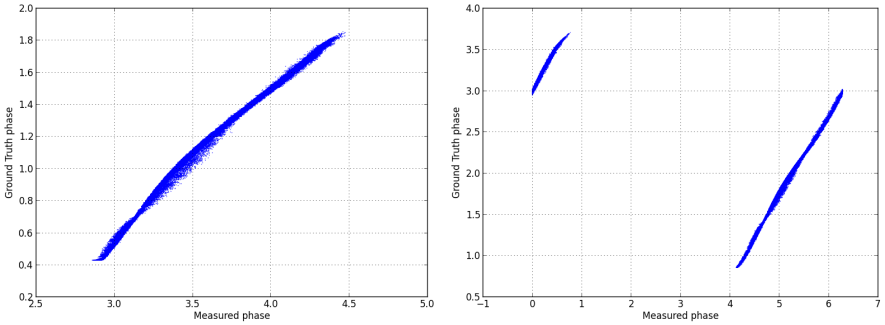

Figure 11: The phase calibration samples for 10Mhz (left) and 20Mhz (right).

of the wall the same as the calibration pattern plane and thus the whole wall becomes a source of ground truth phase information.

calibration pixel selection

Now another series of pictures is taken, using all the frequencies that we want to calibrate. In theory, we now already have a measured / ground truth pair for each pixel of each picture, that we could use to find a calibration function. In reality however, not all of these values are good enough to be used for calibration. A known problem of the PMD Digicam is, that the depth information becomes unreliable for pixels with low amplitude. As the calibration pattern uses black circles, all points inside a circle have a low amplitude and should not be used for calibration. Furthermore, as the lens has a high distortion in the corner, only center pixels are used for calibration.

pixel masking

Figure 10 shows the mask (center image) that is used to select pixels for the calibration data. Only pixels whose amplitudes are higher than the average amplitude (marked as red) and who are in the center of the picture (marked by the black rectangle, which is $1/4$ of the total image size) are used as calibration samples. In the phase image (on the right), it is clearly visible, that the black circles of the checkerboard influence the phase measurement, although their distance should be exactly the same, as in their surroundings. In a perfect measurement, the calibration pattern would not be visible at all in the depth image (see also section 2.4 for further details).

calibration sample description

Figure 11 shows the phase calibration samples for 10Mhz and 20Mhz. They are generated from the same set of pictures (for each distance the pictures for the 3 frequencies are taken in parallel), but there are

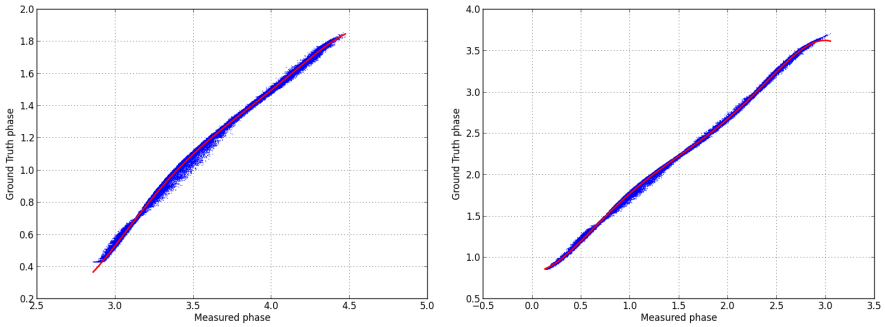

Figure 12: The phase calibration polynomials (red) for 10Mhz (left) and 20Mhz (right).

two major differences. First of all the ground truth phases for the 20Mhz case are twice as high as for the 10Mhz case, which is exactly what has to be assumed (note that in formula 1 on page 4 the phase is divided by the frequency to get the distance). Secondly, there is a major gap in the 20Mhz data set which is due to a constant phase offset inside the camera for this frequency. As the phase values can only be in the interval $[0 \dots 2\pi]$, higher values are projected back to valid ones (which leads to the left part of the sample points). Although it would of course be possible to find a function, that matches these sample points, it is far easier, to first move all the values by a correction offset φ_{off} so that they form a consecutive line like in the 10Mhz case.

After applying this calibration phase offset, we fitted in a fifth degree polynomial function using the least square fitting (tests show, that a higher degree does not lead to a better approximation of the function and would only lead to a harder computation).

This results in the calibration functions shown in table 1. Example plots can be seen in figure 12. The general formula is:

$$\text{calib}(\varphi) = \sum_{i=0}^{5} a_i \cdot (\varphi + \varphi_{off})^{5-i}$$

limitation of the calibration

Figure 12 also shows a major problem with the calibration: As the band that the sample points resemble is pretty broad at some parts, it is clear that it is impossible to find a function which is close to all sample points. So even if the same points that were used to build the calibration function are calibrated with that function, some of them will still have wrong values. It is important to remember this inaccuracy if pictures are later considered to be calibrated.

3.2.3 *Amplitude calibration*

calibration difficulties

The calibration of the amplitude is far more difficult than the phase calibration, as there is no direct way to get ground truth information

	10Mhz	20Mhz	30Mhz
a_0	−0.46861088	−0.11776117	−0.00451956
a_1	8.63853605	0.88826474	0.05610486
a_2	−63.20817976	−2.37231896	−0.26067655
a_3	229.21818499	2.64900201	0.55224721
a_4	−410.61616649	−0.11113942	0.48622625
a_5	290.18364156	0.83142054	0.98368383
φ_{off}	0	−4	−5

Table 1: Coefficients of phase calibration polynomials for the 3 different frequencies.

for the amplitude. While the distance, and therefore the phase shift, can also be measured by a simple ruler, it is much harder to find a device that can measure the amplitude of a modulated light wave with a given frequency. It is also not clear, what exactly the calibrated amplitude should be. The "phase shift corresponding to the actual distance between an object and the camera" is a simplified model of the phase measurement that can be used to generate ground truth information for the phase, but a similar model is harder to define for the amplitude. Should the amplitude be only dependent of the light hitting the sensor, or should inaccuracies of the light emitting unit also be considered? How should the sensitivity of the sensor be handled? There is no single answer to such questions, so it is hard to find a definite calibration model.

workaround approach

However, it turns out that we do not actually require a full amplitude calibration for our application. The demodulation model in Chapter 4 only assumes, that the amplitude does not change, if the modulation frequency changes. Furthermore, the amplitude is only linear in all the formulas, which means, that a amplitude scaled by a value s before the demodulation will result in demodulated values scaled by the same value s after the demodulation. And as we are only interested in correcting the distance measurement, the demodulated amplitudes are far lower.

An easy solution would be to simply set the amplitude to one for each measurement. This would satisfy the constraint, that the amplitude does not change with the modulation frequency and as we are only considering one pixel at a time in the demodulation process and do not use the amplitude afterwards, we would not lose any important information either.

amplitude changes

There is however one important issue with the approach: The idea of a constant amplitude for all frequencies is only valid in a non multipath case. As discussed in section 4.4, the amplitude as well as the calculated distance will change for different frequencies, if the light is reflected multiple times. Setting the amplitude to a constant

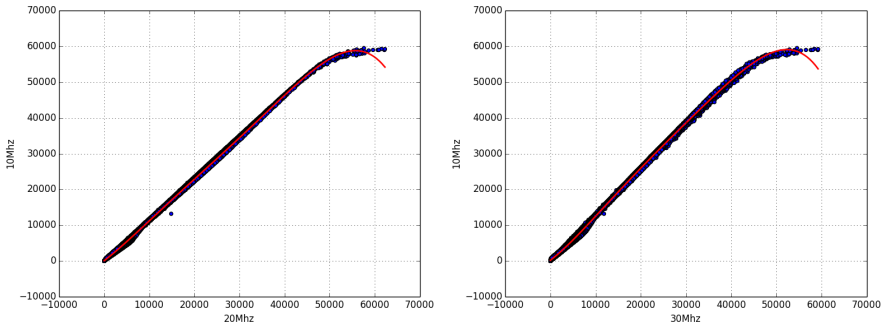

Figure 13: The amplitude calibration functions for 20Mhz (left) and 30Mhz (right). Both try to map the amplitude values to the values that would have been measured with 10Mhz (the reference value).

value would lead to a loss of an important part of the information gained from the second measurement.

amplitude calibration model

This means, that we need at least some simple amplitude calibration. As we only have to deal with the constraint, that the amplitude normally does not change, if the frequency changes, we can just define the amplitude of one frequency as ground truth and compute a function, that will adjust amplitudes measured with other frequencies to this value. This means, that the same value of one frequency will always be mapped to the same ground truth value and if the amplitude changes differently than expected, it will also be mapped to a different ground truth value. This way, we can can preserve the information of the amplitude change and still do a calibration that matches our constraint.

calibration process

The process is similar to the phase calibration: A large number of samples is taken from a series of measurements and then a best fitting polynomial can be found for these samples. The generation of sample points is even easier because we do not need to find good points like in the phase measurement. Instead we select random points from each image.

Figure 13 shows the resulting functions which are again a polynomial of the fifth degree. This was mainly done to be consistent with the phase calibration, a linear function would not have resulted in a much worse approximation. This time, the breadth of the sample values is also lower which should result in a better calibration. The polynomial coefficients are shown in table 2

3.2.4 *Results*

phase calibration

Figure 14 shows the results of the camera phase calibration. The ground truth picture contains the synthetic values of the wall plane as extracted from the calibration pattern, which means, that the floor in the lower right corner of the image (visible in the *measured* picture) is not taken into account.

	20Mhz	30Mhz
a_0	−0.11776117	−0.00451956
a_1	0.88826474	0.05610486
a_2	−2.37231896	−0.26067655
a_3	2.64900201	0.55224721
a_4	−0.11113942	0.48622625
a_5	0.83142054	0.98368383

Table 2: Coefficients of phase calibration polynomials for the 2 different frequencies. As 10Mhz is the reference frequency, it needs no calibration function.

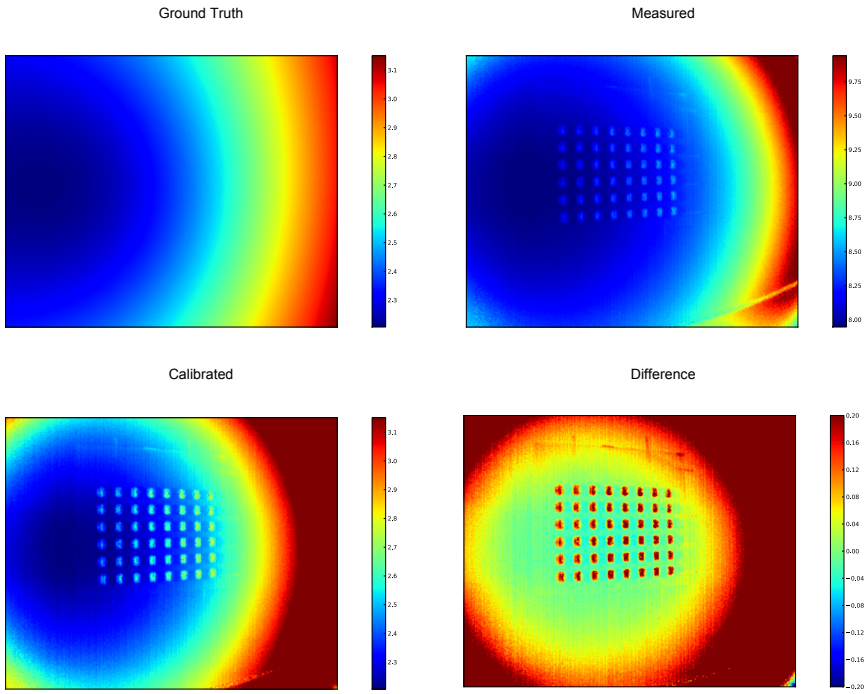

Figure 14: Results of the camera phase calibration. All values are given in meters.

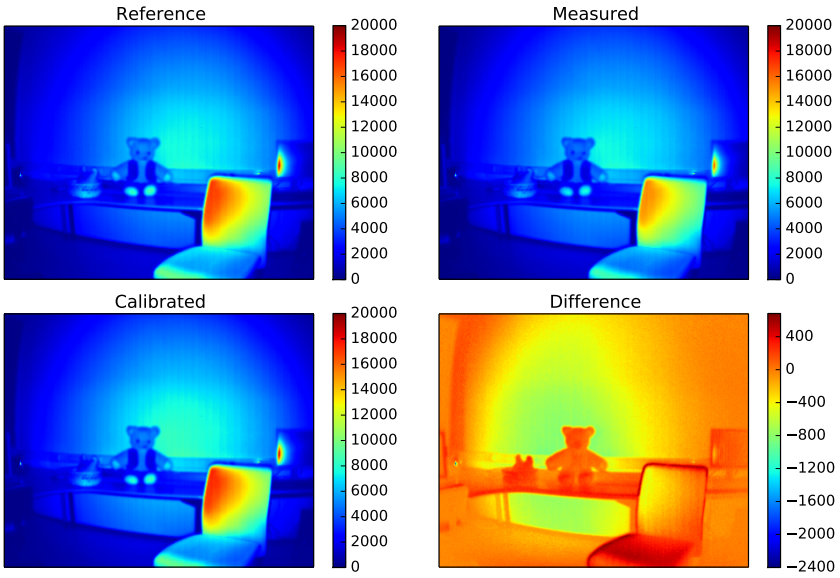

Figure 15: Results of the amplitude calibration.

The *difference* picture shows, that the calibration works reasonably well in the bright part of the picture, while darker areas show a higher error. Especially the error for the black circles is quite high, but as the phase values from the camera are already far too big, there is no chance in calibrating this with a function that just takes one phase and returns the other. Altogether the calibration leads to plausible but not exact values.

amplitude calibration

The results for the amplitude calibration are shown in 15. After applying the calibration function, the amplitude images for two different frequencies are very similar. Apart from some outliners, the calibrated image is quite close to the original, most of the time the error is less than 2%. On the one hand this means, that the amplitude calibration works better than the phase calibration but on the other hand, a somewhat simpler method was used, which can explain the better results.

partial calibration

It should again be noted, that the calibration is only partial. We calibrated the camera only for the frequencies 10Mhz, 20Mhz and 30Mhz at a fixed integration time of 500ms. Furthermore only the center pixels are calibrated and only for a distance between one and four meters.

3.3 NOISE MEASUREMENT

As described in section 2.4, a common error in all sensor data is noise. Even though the exact same sample point is measured multiple times, the results cannot be expected to be exactly the same each time. Instead they usually vary around the true value with a Gaussian distribution. This noise can have different causes and cannot be calibrated

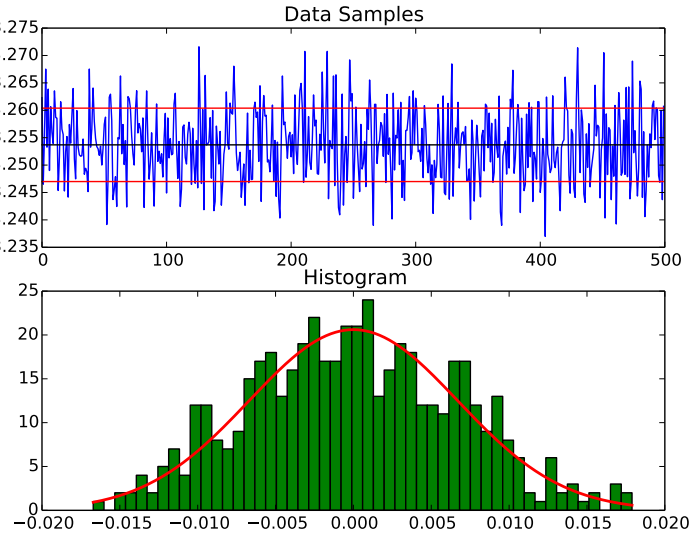

Figure 16: Noise analysis for the phase computed from the raw values. The upper part shows the individual samples (but only 500 for the sake of visual clarity), with the computed expected value (red line). The lower part shows the histogram of these samples with the corresponding normal distribution (red line).

directly. Instead the usual way to deal with this problem, is to perform a series of measurements and average the results.

If the noise behavior of a specific device is known, it is easier to compare it to other devices and it can also be possible to estimate, how many measurements should be averaged to get a desired accuracy.

measurement

To measure the noise behavior of the PMD DigiCam, a series of 1000 images of the same blank white wall were taken. Although the values for the different pixels are expected to vary (as the wall might not be perfectly flat or white), a perfect camera should return the same value for the same pixel position in each measurement.

3.3.1 *Results*

single pixel noise

Figure 16 shows the resulting plot for the phase values calculated directly from the raw data of the center pixel. It can be seen, that the noise distribution matches approximately a normal distribution, as expected. Plots for other values (namely the amplitude and the raw channels) are not shown, as they look all very similar to the one of the phase.

The exact noise parameters are shown in table 3. As described in section 2.3.2, the value for D_0 should be the negative value of D_2 and likewise with D_1 and D_3, which is especially in the case of D_1 and D_3 only partially fulfilled, but at least the deviations are comparable.

	Expected Value (μ)	Deviation (σ)
D_0	-9689	93.102
D_1	-1290.2	91.264
D_2	9203.4	91.583
D_3	834.08	87.882
Phase	3.2536	0.0066336
Amplitude	19012	133.27

Table 3: Noise values for the raw channels, the amplitude and the noise of the center pixel. All values are rounded to five significant digits.

The relative error of phase noise, i.e. the quotient of its standard deviation and its expected value, is 0.2% which is quite good.

full image noise

To see, how other pixels behave, we perform a full image noise analysis. The same computations as for the center pixel are done for all pixels, but instead of showing a histogram per pixel, all the deviation for one value is shown in a single map in figure 17.

The main observation is, that the noise is the highest in the center of the raw channel and amplitude plot, but the phase noise looks inverse to this. This might be explained by the fact, that the center is also the brightest area in the image and higher values usually mean also higher noise. To analyze this, the images are normalized (again by dividing the standard deviation by the expected value), the results are shown in figure 18. D_1 and D_3 look very noisy but apart from that the noise tends to get lower towards the center of the image. This can be explained by a higher brightness in the center which is generally good for the measurement (as long as there is no oversaturation).

Altogether we end up with a phase noise of $0.2\% - 0.3\%$ for the interesting part of the image, which is generally good.

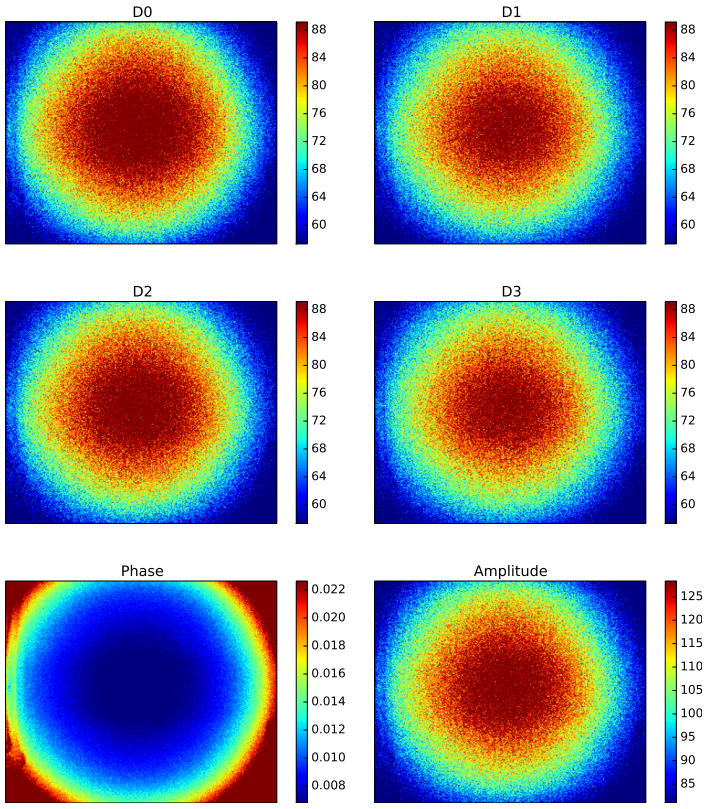

Figure 17: Noise of the camera raw channels, the phase and the amplitude.

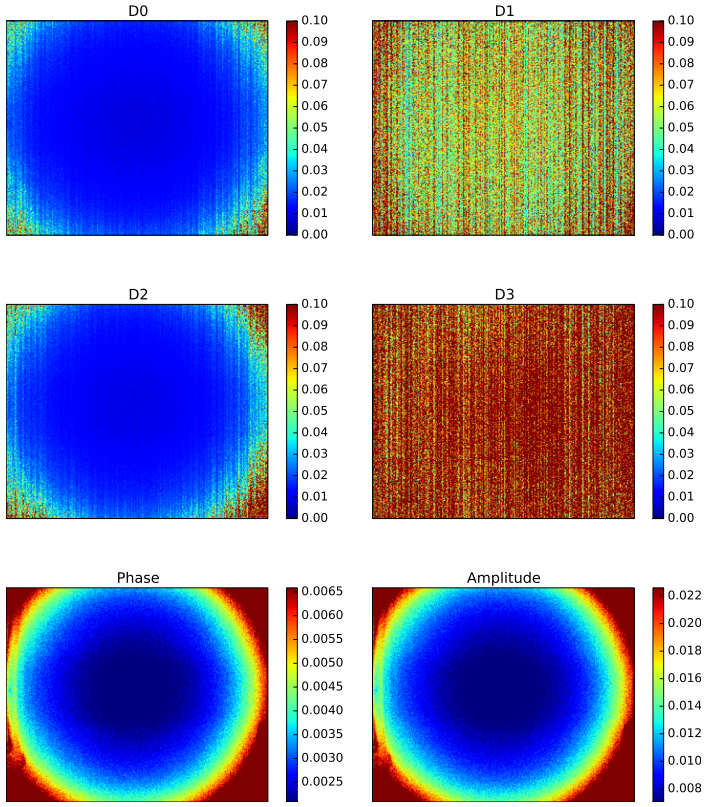

Figure 18: Normalized noise of the camera raw channels, the phase and the amplitude.

4 THE MULTIPATH EFFECT

The multipath effect occurs, whenever the image sensor measures a superposition of interfering waves instead of a single wave. This can cause significant errors in the distance calculation as the phase shift of the incident light wave will be used, which will no longer correlate to the correct distance in a multipath case.

As already sketched in 2.4 there are multiple causes for the multipath effect and the solutions proposed to correct it do also vary with the different causes as not all can be addressed by all methods.

4.1 SOURCES OF MULTIPATH EFFECTS

This section characterizes the different sources of the multipath effect in more detail. Various publications cover different subsets of them mostly depending on the capabilities of the presented correction algorithms or the general topic of the publication. The main sources of this section are [DGC+11, KWB+13, KBC13, Fuc10].

background interference

As all diffuse materials reflect incoming light in all directions, a normal scene will contain indirect reflections everywhere and the camera will not only measure the combination of two but actually of countless waves. The reason why measured data is still useful is, that most of the time the indirect reflections are order of magnitudes weaker than the direct reflections. Only in special situations, where the indirect reflections are unusually high (compared to the direct reflections) this effect becomes a problem.

highly reflective objects

A typical example is, that a highly reflective object in the scene reflects not only the light, that comes directly from the camera but also light from other objects in the scene. This is illustrated in figure 19: The camera sends out light to the wall and the cup. The wall reflects this light (red ray) in all directions, a small portion of it reaches the cup, which is also illuminated directly (green ray). Both of the incoming rays are reflected towards the camera (yellow ray), which can only measure a mixture between the two rays. Although most scenes do not contain objects like mirrors, many surfaces can become highly reflective under a certain angle due to the Fresnel effect. This is especially problematic as their reflectivity might not be obvious and may lead to surprising errors in the measurement.

transparent objects

Transparent (or semi transparent) objects have a similar behavior although the involved materials have different properties. The light from other objects is not necessarily reflected by transparent objects but can rather pass through it with a little attenuation. This light in-

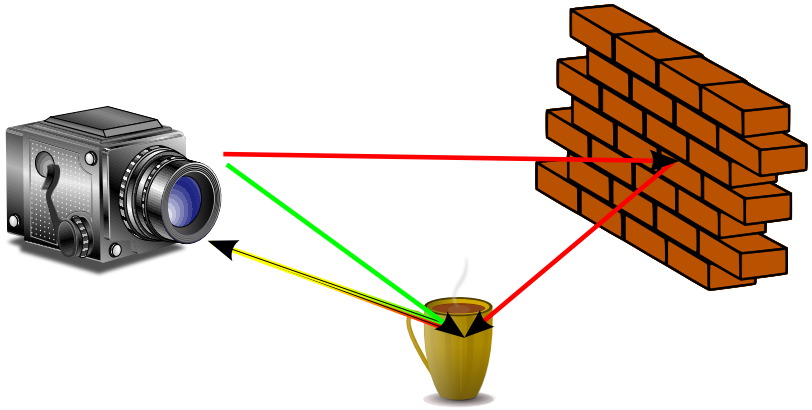

Figure 19: Sketch of the multipath effect. The light returned by the cup consists not only of direct illumination, but also of indirect illumination from a nearby wall.
(The camera, the cup and the wall are public domain images from http://openclipart.org/.)

terferes with the light reflected by the transparent surface and causes multipath errors.

An expansion of this scenario are diffuse objects like fog. Each point can potentially scatter light then, which results in a superposition of an infinite amount of waves.

corners

Another source for multipath effects are concave corners, which often appear rounded in time-of-flight images. This happens because each point on the one wall will receive light reflected by any point of the other wall and reflect parts of it towards the camera. Normally indirect reflections a very weak and would not be significant, but in corners there are enough reflecting points that are close enough to result in a significant amount of light that is reflected additionally to the directly reflected light. Like in the diffuse object scenario, there will be countless interfering waves, but in this case, the phase shifts are a lot more similar as only close points can have a significant influence.

camera intern

The interference of different waves must not necessarily happen inside the scene. Like all cameras, time-of-flight cameras contain a lens system to form the image on the sensor. Although camera lenses are usually of a high quality, they can never refract all the incoming light but will also scatter and reflect small parts of it. Additionally, the other components inside the camera will also reflect small amounts of light.

Under some circumstances this can be significant enough to produce multipath errors. A possible explanation of the intensity related errors is that as the direct illumination is pretty weak, reflections inside the camera have more influence and disturb the measurement this way [Sch11b, p. 25]. Another example are scenes, where a big part of the camera frustum is occluded by a nearby object. As the near object is very bright due to its short distance to the illumination

unit, it causes strong reflections inside the camera resulting in much smaller depth values for the unoccluded part of the scene compared to the same scene without the occluding object.

flying pixels

An effect very similar to the multipath effect is the so called *mixed pixel* or *flying pixel* effect. It happens not because of interfering waves inside the scene or the camera, but because the pixels are too big to measure only a single wave. Especially at the border of objects, a pixel will often receive light from the background and the object itself. Although they do not interfere anywhere, the result is the same as if they would. As the symptoms are so similar, the mixed pixel effect is often considered together with the multipath effect, although their causes are different.

4.2 MATHEMATICAL DESCRIPTION

In a multipath situation with n components, the camera measures the sum of n different waves. If we assume a sinusoid for each wave (where *wave* means not the actual light wave, but its modulation; the light wave itself is always a sinusoidal, the modulation can be arbitrary), this leads to:

$$\begin{aligned} w_m(t) &= \sum_{i=0}^{n-1} \alpha_i \cdot \sin(\omega \cdot t + \varphi_i) \\ &= \alpha_m \cdot \sin(\omega \cdot t + \varphi_m) \end{aligned}$$

Each component of the measured wave w_m can have a unique amplitude α_i (resembling the intensity of the component) and a unique phase shift φ_i (resembling the distance of the component). As explained in A, the sum of sinusoidal functions with the same frequency is again a sinusoidal function, which means that the formulas from chapter 2.3 can be applied, which is exactly what happens, when the multipath effect is completely ignored. But in general, none of the distances in any component relates to the new phase shift φ_m, which means that using it for distance calculation will lead to wrong values.

depth correction

To get the correct depth information from the measurement, it is necessary to determine the values $\varphi_0, \ldots, \varphi_{n-1}$. Normally, one of these components will be chosen to be the correct one and the other ones are discarded, but in some situations it can be useful to have multiple depths for each pixel (e.g. for transparent or diffuse objects).

4.2.1 *Two Component Multipath*

In many cases, it is sufficient, to use only two components to model the multipath effect. In this case, we have a primary component, which represents the true distance and a secondary component which represents the error due to interference. In figure 19 the green wave

would be the primary component while the red wave would be the secondary (both times including the yellow wave which is the superposition of both).

model limitations

Although all multipath affected measurements can in theory be split into a primary and a secondary component (by subtracting the error) it can in practice be hard to find this error, if more than two waves interfere. The main reason to limit the model to two components is, that some correction methods get tremendously more difficult with the number of components they have to decompose. Additionally, many cases can still be modeled accurately with this approach [DGC+11]. Other cases are however very likely to fail in a correction constrained in such a way.

We will call the primary component w_0 and the secondary component w_1. This results in the following formula:

$$\begin{aligned} w_m(t) &= w_0(t) + w_1(t) \\ &= \alpha_0 \cdot \sin(\omega \cdot t + \varphi_0) + \alpha_1 \cdot \sin(\omega \cdot t + \varphi_1) \end{aligned}$$

4.3 MULTIPATH RESOLVING

There have been many different attempts to solve the multipath problem, which are described in this section.

corner correction

Fuchs proposed a method that addresses multipath errors in corners. He models all points in the scene as Lambertian emitters which illuminate all the other points and calculates the effect, that each pixel has on the scene. A few simplifications are needed to make this approach applicable such as using the measured (and slightly wrong) distances and considering only points that are visible from the camera. The algorithm cannot work in real time, it takes roughly ten minutes (on not closer specified hardware) to correct an image, but the results show a significant correction of the round corner problem. [Fuc10]

advanced corner correction

Jimenez et al. extends the approach of Fuchs by using an iterative method. This was already suggested by Fuchs as it helps to overcome the issue, that the computations are based on the distorted depth values of the measurement. This method also expands the model and considers different albedo factor of surface areas. Furthermore, it takes the normal vectors of the sampling points into account, which need to be updated after each iteration step. Due to the complex iteration steps, this method is also very costly (taking several minutes even for a small scene on not closer specified hardware), but the results are better than the previous approach. [JPMP12]

two component model

Dorrington et al. proposed a new method that uses multiple measurements with different modulation frequencies to gain additional

information about the multipath interference. Using a numerical optimization they can directly determine the components of the measured wave. Although the method works with an arbitrary number of components, only two measurements are used to separate two components because the optimization is already difficult and time demanding in this case. As the improvement of this method is the main topic of this thesis, it is described in more detail in the next chapter. [DGC+11]

closed form solution

Godbaz et al. proposed a new solution for the same approach, which does not rely on a numerical optimization by presenting two different closed form solutions for the overdetermined case. The first models the reflectivity as a Cauchy distribution over range, while the second one uses attenuation ratios. Instead of two measurements, four measurements are required for both methods to separate the two components. Both methods are able to improve the range measurement in most cases, however there is a threshold depending on the relative amplitudes of the components and the signal-to-noise ratio, below which the components can no longer be separated. Additionally the results are not as good as the ones from the numerical optimization. The computation times were not compared but in general closed form solutions are much faster than iterative optimizations. [GCD12]

SPUMIC framework

Kirmani et al. proposed a framework called SPUMIC (simultaneous phase unwrapping and multipath interference cancellation) that addresses phase unwrapping and multipath correction at the same time. At least five different modulation frequencies are used to measure the scene. As the maximum unambiguous distance depends on the modulation frequencies, measuring with different frequencies can increase it to the greatest common divisor of these frequencies. Furthermore, the measurements are used to build a Hankel matrix whose rank is used to test, whether the measurement is affected by multipath errors. If this is the case, the total least square Prony's method is used to separate two multipath components. The biggest drawback of this method is, that it requires a high number of measurements but on the other hand it is implementable in real time due to its low computational complexity. [KBC13]

peak modulation

Kadambi et al. use a custom built camera with a special modulation. Instead of using a sinusoidal, a sharp peak is used. The resulting correlation function is then also no longer a sinusoidal but contains peaks for every interfering multipath component. This does not just enable to correct the multipath effect, measure near-transparent objects correctly or see through diffuse objects, it is furthermore possible to compute a full time-profile movie that shows the propagation of the light in the scene. The main limitation of this method is the time resolution which is to low to distinguish components with similar phases that occur for instance in multipath affected corners. Also

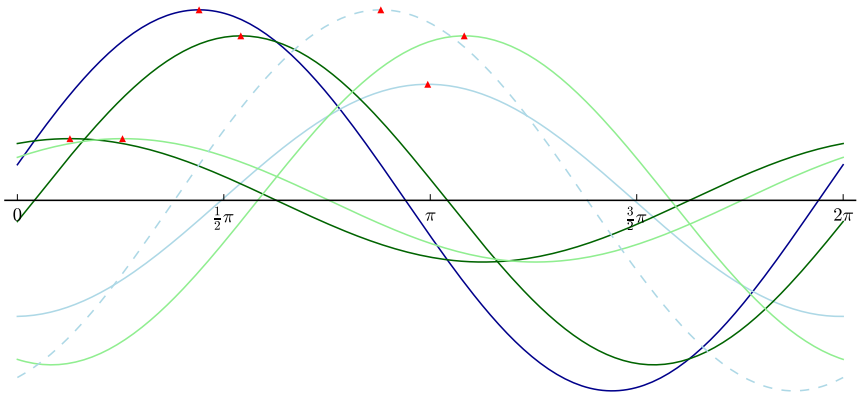

Figure 20: The measured waves (blue) of two pairs of interfering waves (green) for two different frequencies (light and dark colors). The red triangles mark the phase shift of the waves. The dotted wave has the double phase shift of the dark blue one.

off-the-shelve cameras are not capable of this measurement, as long as they do not have an FPGA that can be reprogrammed to use the new modulation. [KWB+13]

4.4 WAVE DEMODULATION UTILIZING DIFFERENT FREQUENCIES

We now describe the multipath correction approach proposed by Dorrington et al. whose main idea is to measure the scene with two different modulation frequencies to gain more information about the multipath interference. We also point out where this additional information comes from and explain the process of function fitting in a bit more detail.

two components

Most off-the-shelve cameras today have the ability to change the modulation frequency. This is done between the measurements and will provide enough information to find w_0 and w_1. The method could be extended to separate more waves but the additional effort for the measurement process (more frequencies would be needed) and computation time must be weighted against the additional improvement (which might be small in many cases).

source of additional information

Figure 20 demonstrates, where this additional information comes form: If the modulation frequency of the camera is doubled, it is expected, that the amplitude of the measured wave will not change but that its phase shift will be doubled. This is however only true for the non multipath case. The image shows the resulting waves, if two waves interfere: The dark blue wave is the sum of the two dark green waves. If the frequency is doubled, the phase shift of the green waves is also doubled, but the amplitude stays the same (displayed as light green waves). However, the phase shift of the resulting sum (the light blue wave) is more than doubled and its amplitude is smaller. This

wave is significantly different from the expected wave (which is displayed as a dotted line) and this difference is basically the additional information from the second measurement.

The difference in the second wave can also be used as an indicator of the multipath effect: If there is no additional information from the second measurement, it means that there are no interfering waves.

formula simplification

We now have two measurements with two components each, which means that we have a total of four different waves. To simplify the model, it can be assumed, that the amplitude of a measurement stays constant if the modulation frequency is changed. This can be assured by the camera calibration that is described in 3.2.3. Furthermore, the second modulation frequency can be described as a multiple of the first one, which means that the phase shift for the second measurement will change by the same factor (because the distance d stays constant, the phase shift φ in equation 1 must change at the same rate as the angular frequency ω). Together this means, that although we have four distinct waves, we only need two amplitudes and two phase shifts to describe them (instead of four amplitudes and four phase shifts).

In practice, the second frequency is twice as high as the first. This leads to the following formulas for the measured correlation functions (where the high indices denote the measurement and the low indices denote the component):

$$m^0 = \frac{1}{2}a^0 \cdot \cos\left(\varphi^0\right) = \frac{1}{2}a_0 \cdot \cos\left(\varphi_0\right) + \frac{1}{2}a_1 \cdot \cos\left(\varphi_1\right) \tag{11}$$
$$m^1 = \frac{1}{2}a^1 \cdot \cos\left(\varphi^1\right) = \frac{1}{2}a_0 \cdot \cos\left(2 \cdot \varphi_0\right) + \frac{1}{2}a_1 \cdot \cos\left(2 \cdot \varphi_1\right)$$

Where the following substitutions were used:

$$\begin{aligned} a_0^0 &= a_0 & a_0^1 &= a_0 \\ a_1^0 &= a_1 & a_1^1 &= a_1 \\ \varphi_0^0 &= \varphi_0 & \varphi_0^1 &= 2 \cdot \varphi_0 \\ \varphi_1^0 &= \varphi_1 & \varphi_1^1 &= 2 \cdot \varphi_1 \end{aligned}$$

The notation that is used here differs a bit from the one used in [DGC+11] where some indexes are different and all sinusoids are in the exponential notation. This makes however no difference in the following calculations.

The four variables $a^0, a^1, \varphi^0, \varphi^1$ can directly be measured as described in chapter 2. Based on them, the four variables for the individual components $a_0, a_1, \varphi_0, \varphi_1$ must be computed. So far, closed form analytical solutions where only presented for the overdetermined case of four different measurements [DGC+11, GCD12], so a numerical solver is used. It tries to find values for $a_0, a_1, \varphi_0, \varphi_1$, so that a correlation function built from them matches the measured one as closely as possible.

To distinguish the measured correlation function from the correlation function that results from inserting the estimated values in formula 11, we name the measured correlation functions $\hat{m}^0$ and $\hat{m}^1$. This leads to the following minimization function:

$$\underset{(\alpha_0,\alpha_1,\varphi_0,\varphi_1)}{\arg\min} \left|m^0 - \hat{m}^0\right|^2 + \left|m^1 - \hat{m}^1\right|^2 \tag{12}$$

function comparison

This function tries to find matching arguments so that the estimated correlation function equals the measured one. By definition, one function equals another function if their values equal each other over the whole function domain. However, instead of matching an infinite amount of samples (which would be impossible), only the four sample values from the measurement process are used to match the functions. As the functions that should be matched are sinusoids with a known frequency and a zero offset, two points would already be sufficient to define the functions and therefore also to match them, but the four values are given anyway by the measurement and additional points might increase the stability of the optimization.

This results in the following error function

$$\begin{aligned} &f(\alpha_0,\alpha_1,\varphi_0,\varphi_1) = \\ &\sum_{\tau\in T}\left(w_0(\tau) - \left(\frac{1}{2}\alpha_0\cdot\cos(\varphi_0-\tau) + \frac{1}{2}\alpha_1\cdot\cos(\varphi_1-\tau)\right)\right)^2 \\ &+\left(w_1(\tau) - \left(\frac{1}{2}\alpha_0\cdot\cos(2\cdot\varphi_0-\tau) + \frac{1}{2}\alpha_1\cdot\cos(2\cdot\varphi_1-\tau)\right)\right)^2 \end{aligned}$$

where $T = \left\{i\cdot\frac{\pi}{2} \mid i\in[0\ldots3]\right\}$ denotes the set of sample positions.

The publication of Dorrington et al. lacks information how this minimization should be applied to multipath errors in practice. The different choices that can be made are evaluated in the next chapter.

5 METHOD ANALYSIS ON SIMULATED DATA

general difficulties

Solving an equation system like the one described in 4.4 with a numerical optimization approach leads to a number of problems. Contrary to an analytical solution, it is not directly clear, if a solution exists (although this is given in our case [GCD12]) and how hard it will be to find it. Furthermore, there are different algorithms available, all with different strengths and weaknesses, and not all of them are suitable for finding a solution to every problem.

This means that it is not sufficient to implement a simple formula, but rather a complete evaluation of algorithms and parameters is needed. The results of this evaluation are described in this chapter.

5.1 SIMULATION OF THE MULTIPATH EFFECT

In order to evaluate the results of the optimization, it is important to have a ground truth. A common way is to use a simulator to generate test data whose optimal solution is then already known.

state of the art

Simulating ToF images with effects like multiple reflections (which are the source of the multipath effect) for a given scene is a complex task, which can be compared to generating photo realistic images of a scene. Meister et al. describes a method based on pathtracing, which is capable of simulating complex scenes correctly, including flying pixels, transparent or strongly reflecting materials and the multipath effect [MNK]. However this method is quite complex and provides more than required for our evaluation of the pure demodulation. This is the reason why we are using a custom, simplified approach for the simulation.

basic principle

The model described in chapter 4 is quite simple, it basically contains just two waves with different properties. Based on this, a very simple yet functional algorithm was developed. Instead of simulating a whole scene, individual pixels are simulated and for each pixel a situation similar to figure 19 is assumed: One wave that travels a long distance and is reflected twice (the red one), and another wave that travels a shorter distance and is reflected only once (the green one). With given values for the albedo of the materials (which can in practice be chosen quite freely) and the two distances, it is simple to compute for both waves the resulting phase shift (based on the traveled distance) and amplitude (attenuated by the reflections and the distance) and also the sum of both waves, which is, what the image sensor will measure.

stochastic sampling

A very handy way of using this simulation algorithm is the stochas-

tic sampling. For a common test scenario, the user would decide, which distances and reflectivities he wants to sample and would write nested loops to go through the domains of each parameter. If the number of tests should be changed, the step size in every loop must also be adjusted. It should also be taken care, that the parameters are sampled equally and that none is oversampled (which might increase the calculation time needed for the same quality of the results). Furthermore, a fixed step size might skip interesting values in the domain completely.

Contrary to this, the stochastic sampling will choose random parameters each time the method is called. If a pseudo random number generator with a fixed seed is used, this will produce the same results for each run of a test (similar to the common sampling), but it is no longer needed to specify a step size and at the same time all parameters are guaranteed to be sampled equally.

the simulator

Therefore, the whole simulator consists only of a single function that computes a random multipath point. The user first sets the random number generator seed and then calls the function in every loop cycle. The function will return the sample values for both of the frequencies (which is what the real camera would provide) and additionally the ground truth values of the two waves that can be used to evaluate the demodulation results.

The simulator uses the two modulation frequencies 10Mhz and 20Mhz which are also used later on for the acquisition of real camera data.

5.2 GENERAL DEMODULATION LIMITATIONS

Apart from finding a good numerical optimization algorithm and determining its optimal parameters, there exists also a more fundamental problem for the demodulation: There are quite different pairs of waves that will result in nearly the same wave, if they are added. If this wave is measured with only a limited accuracy, it becomes impossible to say, which of these wave pairs is the correct result.

Figure 21 shows an example of this, which was found by a failed demodulation. The values for the lower frequency are:

	φ_0	φ_1
w_0	≈ 0.618	≈ 0.852
w_1	≈ 0.560	≈ 0.739

accuracy limit

If the lower phase is both times considered to be the correct value, this lead to distances of $d_0 \approx 1.474m$ and $d_1 \approx 1.336m$, and a distance difference of $13cm$ which is a significant error for a near field depth measurement. The difference between the correlation functions (as measured by formula 12 on page 36) is approximately $7 \cdot 10^{-11}$, which

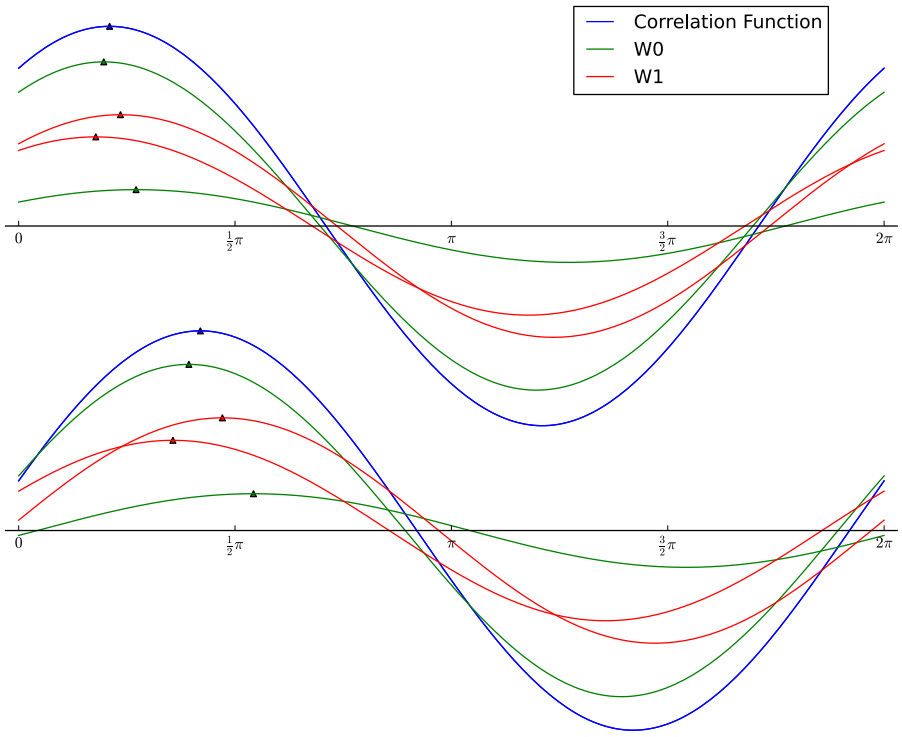

Figure 21: Two pairs of waves resulting in the same correlation function for two different frequencies. The markers are placed at the maximum of each wave and correspond to the respective phase shift.

is extremely small (in figure 21 both correlation functions are actually drawn, they are just far too similar to be distinguishable).

real world limitation

The main problem is not, that the numerical accuracy would not be sufficient, but rather that the noise ratio of the camera is order of magnitudes higher (the noise for values of this size is around $3 \cdot 10^{-3}$, see 3.3.1) than the difference between these two correlation functions. This means, that even if the optimization works perfectly (or a closed analytical solution is found), it can be impossible to obtain the correct result from the measurement, as there might be multiple candidates within the noise ratio of the camera without a clear favorite.

5.3 RESULT METRIC AND POST PROCESSING

As seen in 5.2, the minimization error returned by the minimization algorithm alone is not sufficient to evaluate the result, because even if the error is very small, it cannot be guaranteed, that the result is near the correct solution. Instead, a better metric is needed which takes the actual phases and amplitudes into account.

result vector

The results of a demodulation are four values - two phases and two amplitudes. These values can be regarded a a four dimensional vector on which the usual vector norms can be applied. As some demodulation results show, this is however a too strict criteria because

two vectors can look very different but still mean the same solution. Examples of such ambiguities are:

- Phases can lie outside their usual range. A negative phase shift or a phase shift greater than 2π would be mathematically correct but not a desired solution.
- The waves could be in a different order. In formula 11, the order of w_0 and w_1 does not matter, which means that the optimization algorithm could return them in any order.
- The sign of the amplitude can be exchanged by an additional phase shift. If the phase of a cosine is shifted by π it effectively changes only the sign of the function. If both is done, the wave does not change at all.
- The amplitudes are arbitrary for equal phases. If there is no multipath interference, the correlation function can be demodulated into two waves with the same phase but different amplitudes which add up to the amplitude of the correlation function. How the amplitude is divided is completely arbitrary.
- The phase can be less important than the amplitude. If one of the waves has a very low amplitude (relative to the other wave), it means, that its contribution to the correlation function is small and even big differences in its phase don't change the correlation function strongly.

result normalization

These points make clear, that a traditional vector norm should not be used to compare demodulation results. Instead they motivate a normalization of the optimization results: The amplitude should always be positive, the phase shift should be in $[0 \dots 2\pi]$ and the wave with the smaller phase shift should be the primary one (because the true distance of an object is always the minimal one, thus interfering waves can make it only longer).

result comparison

This alone does not resolve all issues mentioned above so a vector norm can still not be used. Instead a comparison function is used, which performs an analysis of the normalized results. To make further analysis easier, this function only returns a Boolean value depending on whether two pairs of waves match or not. The following series of tests are performed, as soon as one fails, the result is negative, if all pass, it is positive (again, low indices denote components of a wave pair, while high indices denote different pairs):

1. Main phase match: $\left|\varphi_0^0 - \varphi_0^1\right| < \epsilon$
2. Primary waves match: $\left|\alpha_0^1 - \alpha_0^1\right| + \left|\varphi_0^0 - \varphi_0^1\right| < 2 \cdot \epsilon$
3. Secondary amplitude match: $\left|\alpha_1^0 - \alpha_1^1\right| < \epsilon$
4. Secondary phase match: $\frac{\alpha_1^0}{\alpha_0^0} > \frac{1}{1000} \quad \Rightarrow \quad \left|\varphi_1^0 - \varphi_1^1\right| < \epsilon$

finer tests

The threshold ϵ can be chosen freely, but is typically chosen as $\frac{1}{10000}$. The second test is partially redundant, which comes from the idea, that the tests are in a sense independent from each other and the result is better, the more test are passed (however in our case we use only a binary classification, so all tests must always be passed for a positive result). As the extra costs are negligible the redundancy is therefore kept.

The implication in the fourth test means, that the second phase shift should only be considered if the amplitude is significant which matches the point discussed above.

5.4 NUMERICAL MINIMIZATION

After these basic considerations, we can finally move on to the demodulation. In this section the evaluation of different algorithms and their parameters is described.

5.4.1 *Minimization Algorithm*

This thesis uses the `scipy.optimize` package which offers a wide variety of different minimization algorithms in different categories (general purpose, constrained, global, scalar function) [com09]. It also comes with a good guide on these algorithms which can be used to choose the preferred one for a given problem [Var]. In our case, besides the classical Levenberg–Marquardt algorithm (*LevMar*), both the Broyden, Fletcher, Goldfarb and Shanno Algorithm (*BFGS*) and Powell's Method (*Powell*) are expected to work well, so these three are further investigated. Additionally, a simple test for the other algorithms is performed which executes them with their standard parameters on a fixed set of demodulation problems. The results were significantly worse than those obtained with the main candidates, so these algorithms are not considered further.

gradient of the error function

As BFGS and LevMar are Newton based optimizers, they rely on the gradient of the error function. Although it can also be estimated numerically, this increases the computation time significantly and decreases the quality of the results, so it is worth the effort to compute it analytically. As this is very straight forward, only the resulting formulas are given in A. Powell on the other hand is a gradient free algorithm, which means that the gradient is not needed at all.

algorithm parameters

To get the best optimization results, the algorithm parameters must be chosen carefully. Apparently, the values of the error function can get extremely small, even for wrong solutions. Because numerical algorithms cannot be expected to find the exact solution, they need a termination criteria. For the BFGS implementation, this is the gradient norm, which must be below a threshold, for Powell and LevMar it is the relative error in the error function. As the standard values

are far too high for our case, they needed to be changed to avoid early termination. The starting point is another critical choice, which is discussed in the next section.

5.4.2 *Starting Point*

The starting point is one of the most important parameters in almost every method. If it lies near the global minimum, it is easier to reach [Var]. The problem is, that the position of the minimum is unknown so the starting point has to be guessed.

We evaluate five different methods for guessing the starting point. Each one has to return two amplitudes and two phases in a four dimensional vector. By convention, the amplitudes come first, so the vector has the format $(\alpha_0, \alpha_1, \varphi_0, \varphi_1)$ where the indices denote the primary and secondary wave. The variables α_m and φ_m in the following description stand for the amplitude and phase shift of the measured values.

1. $(0, 0, 0, 0)$: The null vector is an intuitive choice if little is known about the parameters.

2. $(\alpha_m, \alpha_m, \varphi_m, \varphi_m)$: If the multipath effect is not too strong, both of the waves could be similar to the measured one.

3. $(\alpha_m/2, \alpha_m/2, \varphi_m, 2 \cdot \varphi_m)$: Two waves are added, so the amplitudes of the components are probably smaller than the amplitude of the sum. And if the multipath effect is caused by a reflective surface, one phase shift might be similar to the measured one, where the other is significantly higher.

4. $(0, \alpha, 0, \varphi)$: If there is no multipath effect, this starting point should be the minimum and no computation time is wasted to demodulate a wave that was never modulated. This might be a good choice for pictures where only small areas are expected to have interfering waves, but although this is more of a performance optimization, it still has to be checked if it yields good results.

5. (a, b, c, d) where a, b, c, d are determined by a brute force search. Finding the minimum with a brute force optimization is very time-consuming but using a low resolution search to determine a good starting point for a further optimization might be worth the search time.

5.4.3 *Constraints vs. Post Processing*

Some algorithms support constraints for the solution, which can be helpful to find the correct minimum. However, constraints like “the

phase shift should be in the interval $[0 \dots 2\pi]$" can also be achieved by post processing the optimization result. Furthermore, a solution violating such constraints would not be a wrong solution, it would rather be an unwanted representation of the desired solution. Hence, optimization constraints are not used at all for the demodulation and only the post processing is applied.

5.4.4 *Algorithm Comparison*

The three selected algorithms are tested with all five starting point methods. The simulator is used to generate a set of 1000 demodulation problems, which are then used in each test to ensure equal conditions for each test.

plot explanation

The results are show in the figures 22-24. They consist of a histogram that show the distribution of the minimization errors color coded in three categories: Green means a full ground truth fit as defined in section 5.3. Yellow means that only the primary phase shift was correct. It is not directly apparent, why this is a truly weaker requirement and although not so many results fall in this category, the differentiation becomes more interesting if the values are affected by noise, which is analyzed in section 6.5. The third category contains all the results that failed completely and is shown in red.

The comparison plot shows the size of all three categories in percentages and visualizes which method performed best.

It is interesting that they are usually very small even if the correct minimum was not found. Furthermore, a small minimization error correlates with a ground truth fit, there are however also cases where a minimum with a relatively big error is correct or where one with a small error is incorrect.

results

The most important observation is, that the starting point has a highly significant influence on the demodulation result. The default initialization with zeros seems to work best for Powell, more sophisticated guesses make the results only worse.

For BFGS however we get a completely different picture: While it performs poorly for the default starting point, the results get tremendously better with a more sophisticated one.

LevMar is similar to BFGS, but it tends to be a little better for most of the starting point methods (with the fourth method as a notable exception).

Altogether BFGS with starting point method 3 has the best results (which works in over 90% of the cases), followed by LevMar with method 3 and Powell with method 1, so these staring point methods will be used for further analysis.

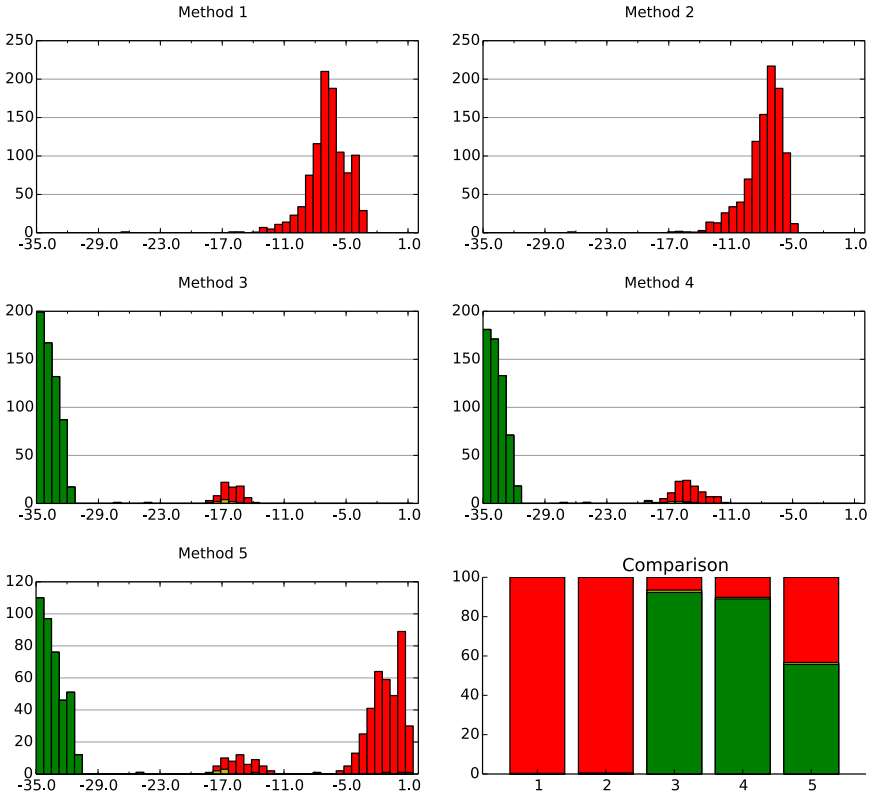

Figure 22: Starting points histograms and comparison for BFGS.

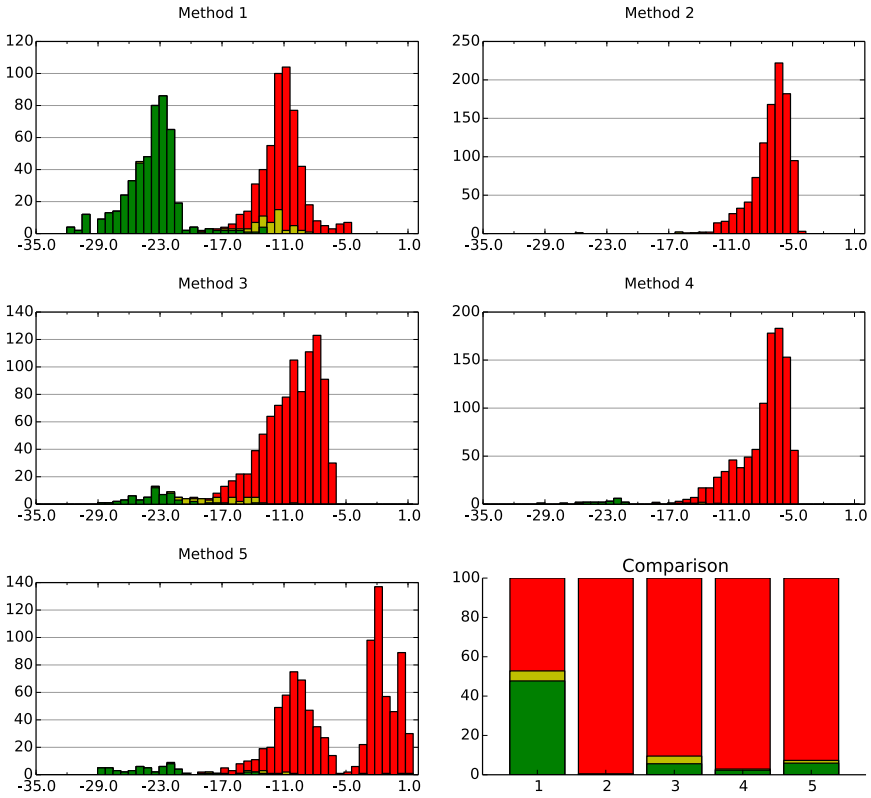

Figure 23: Starting points histograms and comparison for Powell.

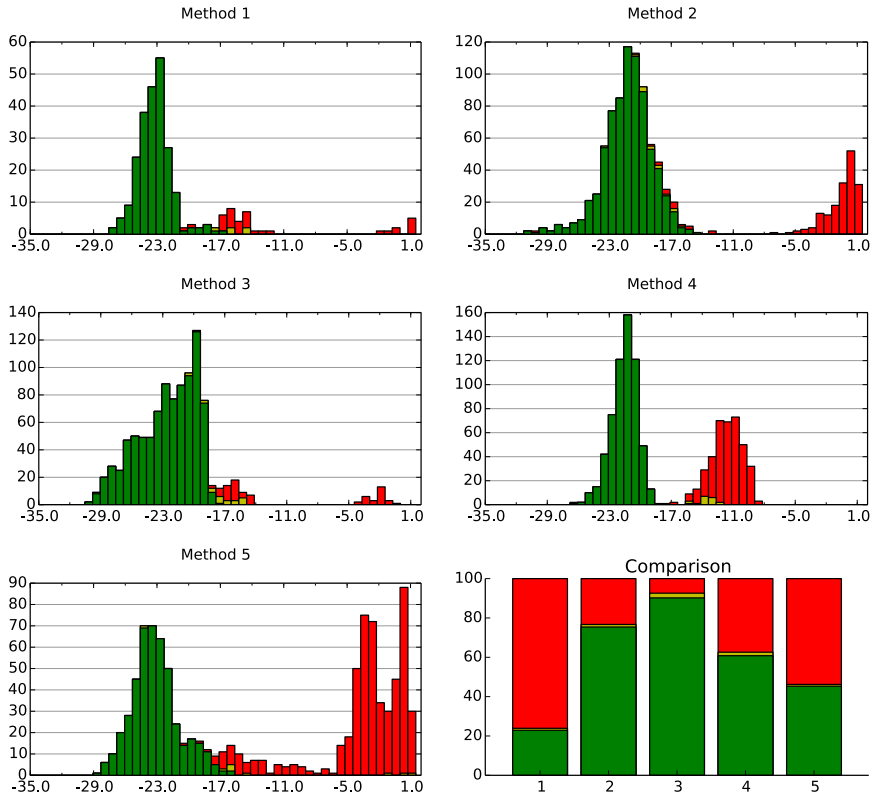

Figure 24: Starting points histograms and comparison for LevMar.

5.4.5 *Brute-Force Starting Point Analysis*

As the starting point has such a tremendous effect on the minimization result, it is interesting to search for even better starting points.

To investigate the behavior of the optimization further, we perform a comprehensive analysis of starting points. The demodulation is performed on a single problem with a huge number of different starting points using BFGS.

A key problem of the minimization function is its four dimensionality. While a three dimensional data set is already difficult to plot in a understandable manner, a four dimensional plot is nearly always infeasible. One approach could be a volume visualization that changes with the time as a fourth dimension, but this is still very hard to understand. To simplify the visualization, only the two phases of the starting point are varied whereas the amplitudes stay constant. This approach simplifies the problem to two dimensions which makes it easier to visualize.

One should notice, that the purpose of this test was not to find the optimal solution by doing a brute force test of different starting points. It is rather about showing how sensitive the optimization is to the parameters that can be chosen. If chosen correctly, the results are

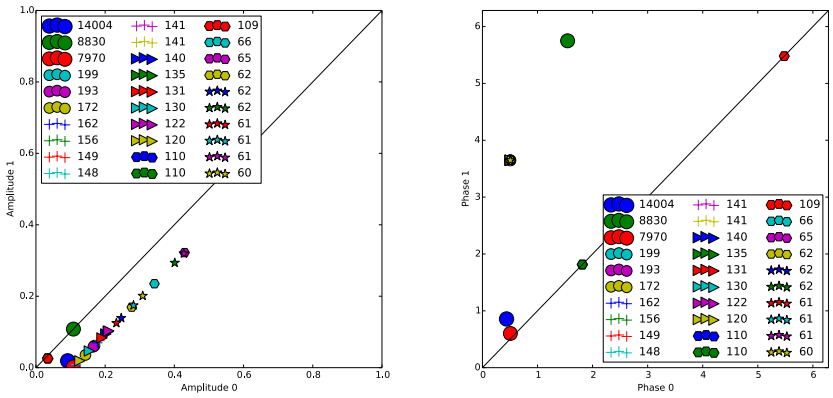

Figure 25: The amplitudes and phases of the found clusters.

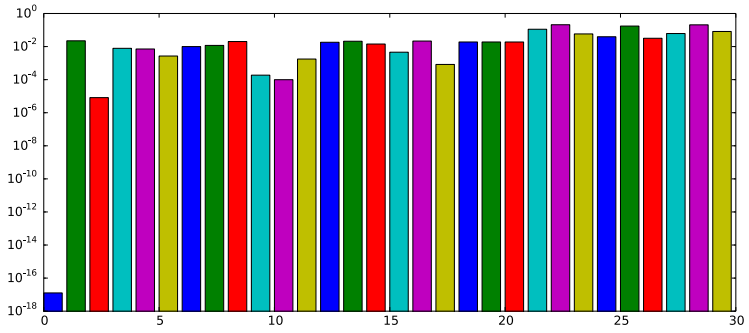

Figure 26: The errors of the clusters. They are sorted descending in cluster size and the colors are the same as in figure 25.

way better than in this test, but it is interesting to see, how important this choice actually is.

In total, 200 different values are used for each phase which results in 4000 different starting points in total. These 4000 minimums are clustered and the 30 largest clusters are visualized. The other ones are rejected because their size is insignificant and they are considered to be outliers.

Figure 25, 26 and 27 show the result of this analysis. The biggest cluster has the smallest error and is very close to the ground truth value. The other clusters however have far higher errors and are distributed all over the parameter range.

Figure 27 is particularly interesting as it shows, where the values for the individual clusters come from. The image is symmetric, which was expect as the minimization function and its gradient are also symmetric. Apart from that, the pattern is very chaotic and it is hard to say anything more useful about it.

results

It does not seem possible to find a generally good starting point with this analysis approach. As a result, we stick to the starting point methods from 5.4.4, which offers good results, at least in most cases.

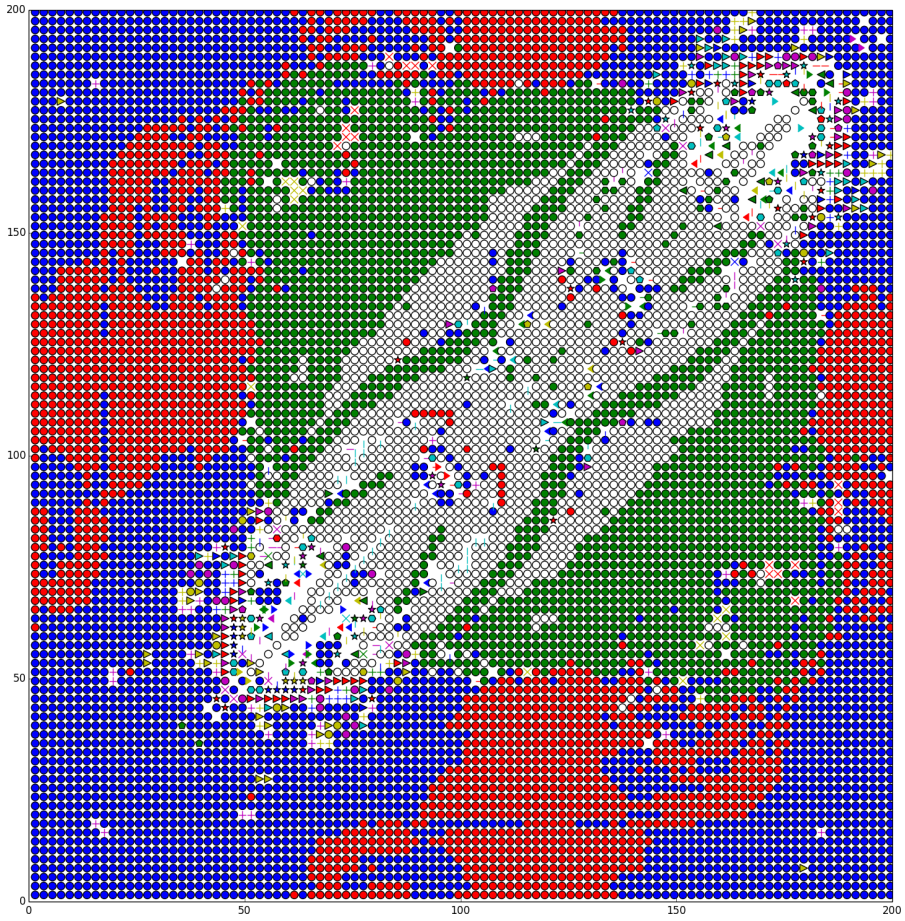

Figure 27: Each starting point is marked with the cluster to which it will converge. White points mean, that none of the 30 big clusters belong to this starting point.

The next chapter will describe the enhanced method for numerical demodulation and contains also an analysis, how noise affects the process and how fast the demodulation can be computed.

6 IMPROVEMENT OF THE DEMODULATION

This chapter explains how the original minimization problem can be reduced from four dimensions to two dimensions, which makes the solution more robust and faster. It also compares the new method with the old method in terms of runtime and sensitivity to noise.

6.1 A NEW APPROACH TO THE MINIMIZATION

We want to minimize the following function:

$$\begin{aligned} & f(\alpha_0, \alpha_1, \varphi_0, \varphi_1) \\ &= \sum_{\tau \in T} \left(w_0(\tau) - \left(\frac{1}{2}\alpha_0 \cdot \cos(\varphi_0 - \tau) + \frac{1}{2}\alpha_1 \cdot \cos(\varphi_1 - \tau) \right) \right)^2 \\ &\quad + \left(w_1(\tau) - \left(\frac{1}{2}\alpha_0 \cdot \cos(2 \cdot \varphi_0 - \tau) + \frac{1}{2}\alpha_1 \cdot \cos(2 \cdot \varphi_1 - \tau) \right) \right)^2 \end{aligned}$$

One thing that can be noticed is, that the amplitudes are just scalar values in this equation. This means, that they can be pulled out of the parenthesis and that the whole equation can be written as the norm of a sum of vectors. In order to do so, we first define some helper variables to make the equation shorter and more readable:

$$\begin{aligned} A_i &= -\frac{1}{2}\alpha_0 \cdot \cos(\varphi_0 - \tau_i) \\ A'_i &= -\frac{1}{2}\alpha_0 \cdot \cos(2 \cdot \varphi_0 - \tau_i) \\ B_i &= -\frac{1}{2}\alpha_1 \cdot \cos(\varphi_1 - \tau_i) \\ B'_i &= -\frac{1}{2}\alpha_1 \cdot \cos(2 \cdot \varphi_1 - \tau) \\ C_i &= -w(\tau_i) \\ C'_i &= -w'(\tau_i) \end{aligned}$$

With these variables, the function can be rewritten as

$$f(\alpha_0, \alpha_1, \varphi_0, \varphi_1) = \sum_i^{|T|} (-C_i + \alpha_0 \cdot A_i + \alpha_1 \cdot B_i)^2 + \left(-C'_i + \alpha_0 \cdot A'_i + \alpha_1 \cdot B'_i\right)^2$$

which can be rewritten as a 2-norm (from now on we use $|T| = 4$, but the method would work with other sizes likewise):

$$f = \left\| \alpha_0 \cdot \begin{pmatrix} A_0 \\ A_1 \\ A_2 \\ A_3 \\ A'_0 \\ A'_1 \\ A'_2 \\ A'_3 \end{pmatrix} + \alpha_1 \cdot \begin{pmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \\ B'_0 \\ B'_1 \\ B'_2 \\ B'_3 \end{pmatrix} - \begin{pmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \\ C'_0 \\ C'_1 \\ C'_2 \\ C'_3 \end{pmatrix} \right\|_2^2$$

This in turn can be thought of as the error term of a least squares problem $\|A \cdot x - b\|_2 \to \min$:

$$\left\| \begin{pmatrix} A_0 & B_0 \\ A_1 & B_1 \\ A_2 & B_2 \\ A_3 & B_3 \\ A'_0 & B'_0 \\ A'_1 & B'_1 \\ A'_2 & B'_2 \\ A'_3 & B'_3 \end{pmatrix} \cdot \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} - \begin{pmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \\ C'_0 \\ C'_1 \\ C'_2 \\ C'_3 \end{pmatrix} \right\|_2^2$$

Hence, we have shown that minimization with respect to α_0, α_1 is equivalent to a least squares problem. We can solve it using the normal equation$A^T \cdot A \cdot x = A^T \cdot b$. In our case, this leads to:

$$\begin{aligned} & A^T \cdot A \cdot x = A^T \cdot b \\ \Leftrightarrow \quad & \begin{pmatrix} A_0 A_1 A_2 A_3 A'_0 A'_1 A'_2 A'_3 \\ B_0 B_1 B_2 B_3 B'_0 B'_1 B'_2 B'_3 \end{pmatrix} \cdot \begin{pmatrix} A_0 & B_0 \\ A_1 & B_1 \\ A_2 & B_2 \\ A_3 & B_3 \\ A'_0 & B'_0 \\ A'_1 & B'_1 \\ A'_2 & B'_2 \\ A'_3 & B'_3 \end{pmatrix} \cdot \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} = \begin{pmatrix} A_0 A_1 A_2 A_3 A'_0 A'_1 A'_2 A'_3 \\ B_0 B_1 B_2 B_3 B'_0 B'_1 B'_2 B'_3 \end{pmatrix} \cdot \begin{pmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \\ C'_0 \\ C'_1 \\ C'_2 \\ C'_3 \end{pmatrix} \\ \Leftrightarrow \quad & \left(A^T \cdot A\right)^{-1} \cdot \left(A^T \cdot A\right) \cdot x = \left(A^T \cdot A\right)^{-1} \cdot A^T \cdot b \\ \Leftrightarrow \quad & x = \left(A^T \cdot A\right)^{-1} \cdot A^T \cdot b \end{aligned} \tag{13}$$

This way we can compute the solution vector x which equals $(\alpha_0, \alpha_1)^T$. However, most of the helper variables still depend on the two phases, which means, that we can only calculate the optimal amplitudes if the phases are known. However, from an alternative point of view, we can also formulate two functions $\alpha_0(\varphi_0, \varphi_1)$ and $\alpha_1(\varphi_0, \varphi_1)$ taking two phases and returning the optimal amplitudes for them. These functions can be derived from equation 13 but the resulting formulas are rather long. They are given in appendix A.

With these functions, our original minimization problem can be rewritten as

$$
\begin{aligned}
&f_2(\varphi_0, \varphi_1) = \\
&\sum_{\tau \in T} \left(w_0(\tau) - \left(\frac{1}{2} \alpha_0(\varphi_0, \varphi_1) \cdot \cos(\varphi_0 - \tau) + \frac{1}{2} \alpha_1(\varphi_0, \varphi_1) \cdot \cos(\varphi_1 - \tau) \right) \right)^2 \\
&+ \left(w_1(\tau) - \left(\frac{1}{2} \alpha_0(\varphi_0, \varphi_1) \cdot \cos(2 \cdot \varphi_0 - \tau) + \frac{1}{2} \alpha_1(\varphi_0, \varphi_1) \cdot \cos(2 \cdot \varphi_1 - \tau) \right) \right)^2
\end{aligned}
$$

where the amplitudes have been replaced by the optimal amplitude functions and only the two phases remain as parameters. It should be noted that although the second row uses the doubled phase shifts (due to the doubled frequency), the phase shifts are not doubled for the amplitude functions (which would be $\alpha_0(2 \cdot \varphi_0, 2 \cdot \varphi_1)$). We only replace the amplitudes as a given parameter by the optimal amplitudes according to the other parameters, regardless where they occur in the formula.

6.1.1 *Demodulation with the New Function*

The demodulation with the new function works analogous to the demodulation with the old function. With a given starting point, the optimization of f_2 is performed. Then the resulting phases are used to compute the associated optimal amplitudes with the functions α_0 and α_1. As a last step, the same post processing steps as before is applied (see 5.3).

As many optimization algorithms require the gradient of f_2 it must also be computed. On the one hand this is now easier, as the function domain has only two dimensions, but on the other hand f_2 is far complexer due to the functions computing the optimal amplitudes. This makes the gradient formulas enormously long, they are given in appendix A.

6.2 VISUALIZATION

As the optimization function is now only two-dimensional, it becomes easy to plot it. Figure 28 shows the function for two different problem instances.

Due to the high complexity of f_2, it is hard to say general things about the function. As a compromise, we compare plots similar to figure 28 for a large number of problems. They all look similar to the two examples in 28. Hence, we can conclude:

- The plots can be tiled perfectly. This was necessary as all phase values outside the interval $[0 \ldots 2\pi]$ correspond to values inside this interval.
- If both phases are equal, the function is undefined (which causes the thin line in the images). In such a situation, any combina-

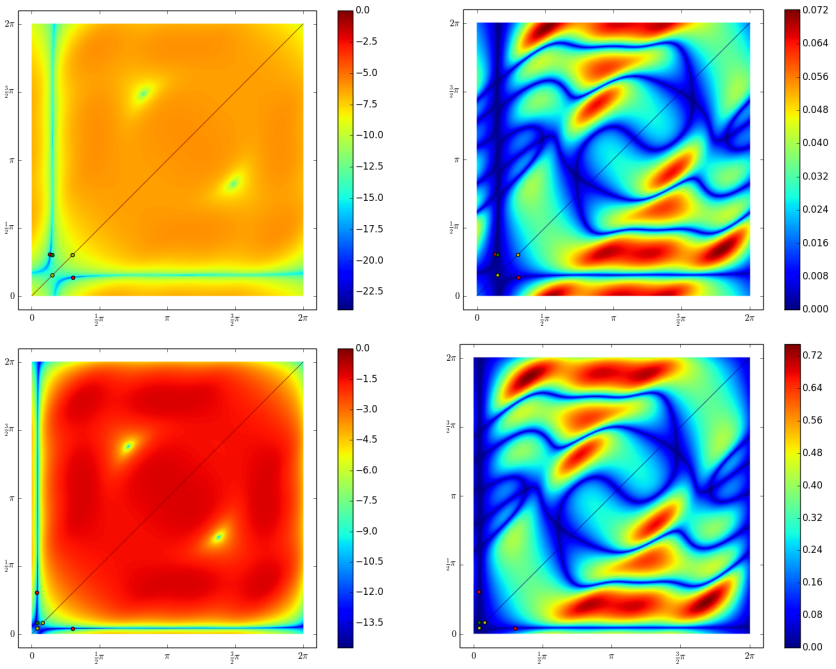

Figure 28: The function f_2 (left side) and its derivative for φ_0 (right side) for two different optimization problems. The red dot shows the ground truth optimum, the yellow dots the two phase shifts extracted from the two measurements and the green dot is a combination of both phases which is often near the optimum. The plots show the logarithm of the respective value.

tion of amplitudes would result in an optimal value, so there is no single optimal value. See 5.3 for details.

- There is a thin valley with a curve in which the minimum lies. All values in this valley are very small but just one is minimal.
- The green dot derived from the two measured phases (by taking one phase for each axis) is often very close to the global minimum shown as red dot.

6.3 EVALUATION OF ALGORITHMS AND PARAMETERS

To find the best minimization algorithm for the new approach, we repeat the evaluation performed in section 5.4. Due to our observations in section 6.2, we choose the starting point to be (φ_0, φ_1).

Figure 29 shows the results. For all three algorithms the results got much better, especially for Powell. However, it still falls behind BFGS and LevMar which now succeed in nearly all cases. LevMar now actually outruns BFGS with a failure rate of only 0.2% instead of 0.9% of BFGS, making it the new favorite for demodulation.

In conclusion, the new approach is a major improvement compared to the original one. All tested methods work better with the new

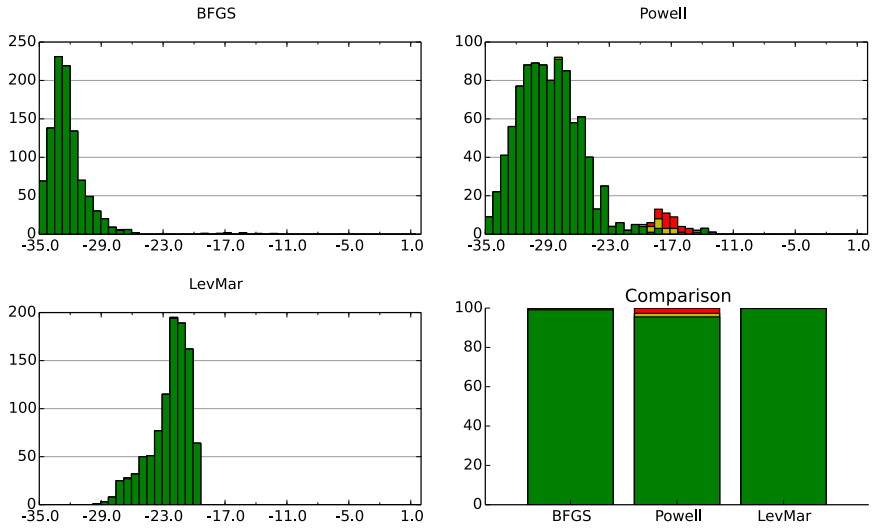

Figure 29: Results for the new minimization function.

formula, although it is quite complex. Obviously, the reduction to two parameters is more beneficial than a simpler functional.

6.4 RUN TIME

Besides the accuracy, the run time is also very important. If the multi-path correction should be part of a real time post processing pipeline, the computation of a single image must be possible in a fraction of a second.

speed improvement

The current implementation is unsuitable for this. However, it is still interesting to see, that the new method provides also a major improvements in terms of speed, which is shown in Figure 30. The BGFS demodulation works twice as fast as before; the Powell demodulation is even three times faster now (however still slower than BFGS). LevMar became only around 40% faster but is by far the fastest solution. All computations were done on an Intel Core i7-2600 system with 8GB RAM and a 32bit Python environment on Windows 7 64bit.

optimization options

It should be noted, that this implementation is not really optimized for maximum speed. Although the `scipy` framework is quite efficient, as it uses Fortran and C for many built-in functions, the error function and its gradient are still only implemented in Python and need to be called many times during the optimization. Furthermore, no parallelization is used and the computation runs purely on the CPU. It would however not be trivial to utilize a GPU, as the optimization algorithms are quite complex and would probably behave badly in terms of coherent branches.

Altogether it would certainly be possible to improve the performance and the new method is already a great help for this, but it

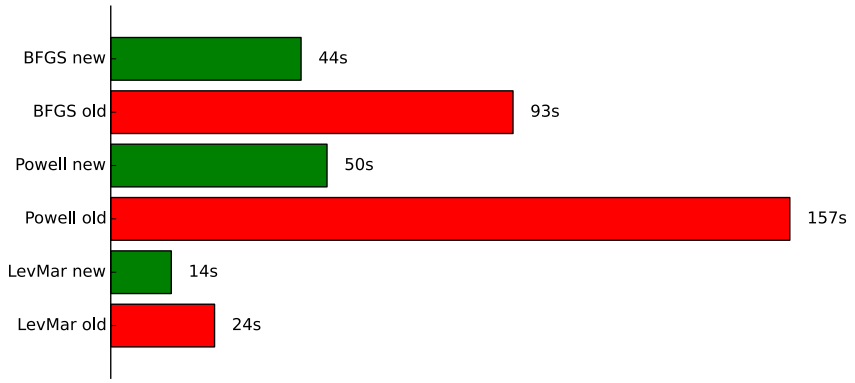

Figure 30: Computation time for 1000 demodulations for the old (red) and new (green) methods.

seams unrealistic, that a real time demodulation can be reached without a notable improvement of the method itself.

6.5 NOISE ANALYSIS

Before the demodulation is applied to real camera data, it is important to consider another point: While the simulator will produce perfect data (in a sense that it matches all our assumptions), camera data is imperfect in many different ways (see 2.4). Most importantly, it is affected by noise, which means that the measured values will vary around the true values. We already analyzed the noise behavior of the PMD DigiCam used in this thesis in section 3.3, so we know which kind of noise we have to deal with. This section will analyze, how this noise will affect the demodulation process.

noise simulation

The simulation of camera noise is straight forward. As expected and confirmed in chapter 3, the noise is approximately normal distributed. This means, that after we have computed sample points for a multipath situation, we can simply add normal distributed values (provided by the `numpy` pseudo random number generator) to them. By changing the variance of the generator, we can control our virtual camera noise.

test series

This noised, simulated data is then used for another test series. The demodulation is again applied to a set of 1000 problems which are now noised with variances ranging from 10^{-2} to 10^{-11}. Additionally, one data set is completely unnoised to show the difference between very small noise and no noise at all. The results are shown in figure 31.

As this is a long test series, the individual minimization error distributions are not shown. Instead, only the values from the former pie charts are shown as bars, which resemble the function of the noise af-

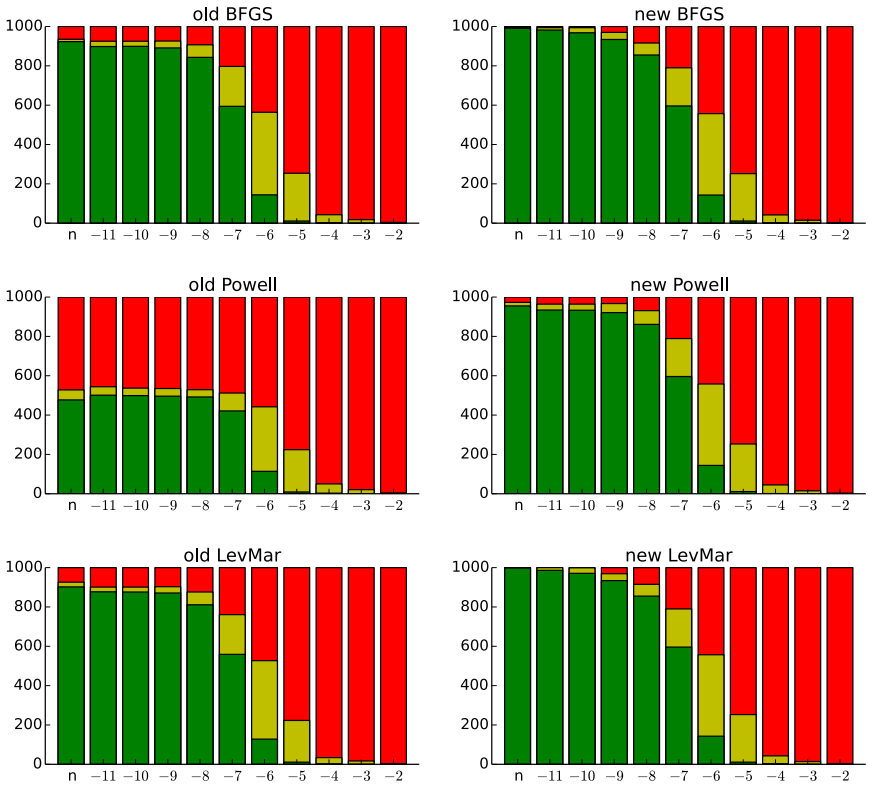

Figure 31: Noise affection on the demodulation results. The x axis is labeled with the exponent of the noise amount, n labels the test without noise.

fection. The latter is similar for all algorithms: A very small variance does not affect the results much, but at at some point, the results get drastically worse until the demodulation fails nearly always.

interpretation

This means, that there is not much of a choice, how much noise is tolerable. Until a variance of 10^{-8} the noise affection should most of the time be negligible, variances between 10^{-7} and 10^{-6} might still be practical in some cases but anything worse will not be worth the demodulation effort (especially because there is no practical way to tell, if a result is correct or not).

7 RESULTS ON REAL DATA

7.1 DEMODULATING CAMERA IMAGES

image acquisition

We now describe the basic workflow for the demodulation of the camera images. The PMD DigiCam is connected to the PC via a USB connection, a SDK provides the necessary drivers and a C interface, which can control the camera and acquire images. The interface is used by a C++ application called `CameraServer`[1] which can access the different streams. These are images that contain either phase and amplitudes values or the unprocessed raw values. We use the latter one, as we want full control over the calibration process. `CameraServer` also provides a moving average filter, that can be combined with any stream to reduce camera noise.

As all further computations are implemented in Python, `CameraServer` must somehow forward the recorded data. We choose to write the streams as raw binary files, which can easily be read with the Python `struct` package. This method has the additional advantage, that both programs can run independently from each other which means that the measurement process and the data processing is separated (which is useful as the processing takes some time).

`CameraServer` was also extended with some convenience features to simplify the capturing process. As we need to measure the scene with different frequencies from the same position, a feature was implemented to perform the measurements for different frequencies directly after each other. Another mechanism ensures, that the moving average contains only valid values, before an image is saved. Both is bundled in a single command that collects and saves all data needed for the further analysis.

capturing parameters

We use 10Mhz and 20Mhz as the two frequencies for the demodulation. Furthermore, the integration time of the camera is fixed to 500ms which provides enough light for most scenes while avoiding oversaturation of objects in a reasonable distance at the same time. Additionally, we averaged 50 frames for each image. Experiments have shown that this reduces noise significantly but that averaging more frames reduces it only slightly further while increasing the overall capturing time. As there are different sources for noise and not all can be reduced by averaging (see 2.4), there is still some noise left in our measurements, but there is no easy way to avoid this.

calibration and processing

After capturing, the binary files are loaded by a Python script for all further processing. As a first step, it applies the calibration described

1 written by the author as part of another project

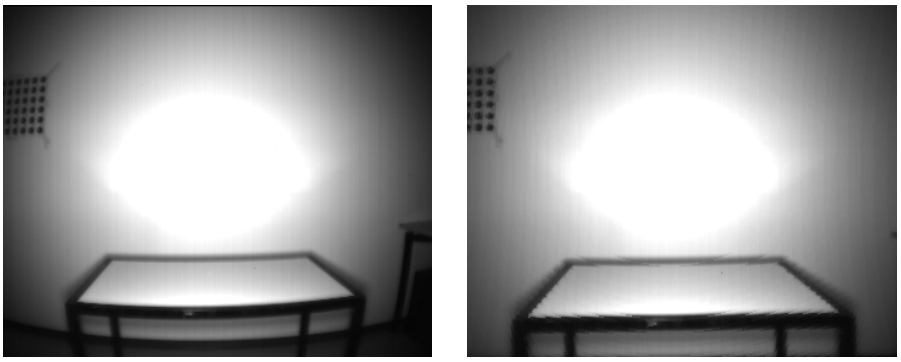

Figure 32: Undistortion of camera images. The left side shows the original image, the right side the corrected one.

in 3.2. Then it iterates over the image pixel by pixel and invokes the demodulation function.

point cloud generation

To visualize the results, it is helpful to generate a three dimensional point cloud. Although the changes by the multipath correction can also be seen in a two dimensional image, it is difficult to get an intuition for the depth values solely from color mapped images. In contrast, a point cloud is a very intuitive representation and allows even layman to judge the quality of the results. For the sake of visualization in this thesis we fix a single perspective to present point clouds.

undistortion

The PMD DigiCam has a quite high lens distortion that must be corrected before the point cloud can be computed. The difference between a distorted and an undistorded image is shown in figure 32. In the distorted image, the table and the bar at the bottom appear curved. After the undistortion, the table is straight, but the image is cropped slightly. The undistortion is performed with a script based on [Zhaoo], which uses the lens intrinsic parameters from the calibration process. It is not further discussed here but can be found in the appendix.

After the undistortion, the original image is no longer rectangular, which means that it will either contain invalid pixels (that do not have any counterpart in the original image) or that the image has to be cropped. The latter one was the default setting of the used undistortion function and is also more useful in our case, as the camera images tend to get blurry towards the edges and the values are only calibrated for the center of the image.

coordinate mapping

Now the depth values must be mapped from polar to Cartesian coordinates, which is done using the method described in 3.1. Once the depth values are transformed, the pixel position can be used as x, y coordinates and the depth value as z coordinate. Theses coordinates are then written to a `PLY` file, which can be read by many different tools. In this thesis we use `CloudCompare`[Mon], a tool that is not

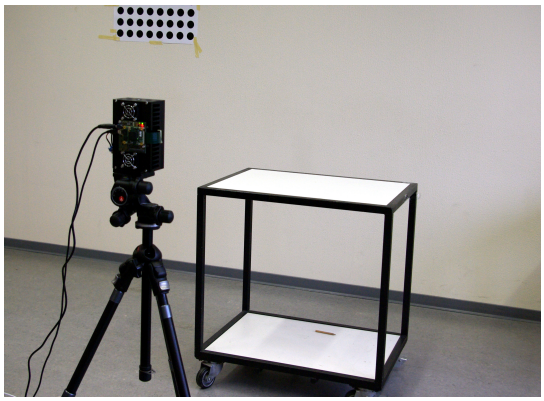

Figure 33: Setup for the table test scene.

only a versatile point cloud viewer, but that also has some interesting analysis functionality.

7.2 TEST SETUP

We used a test scene similar to the one used in [DGC+11] as our main test and evaluation scene. Results on some additional scenes, which are prone to specific problems, are discussed in section 7.4.

A small table was put in front of a white wall as can be seen in figure 33. The surface of the table is slightly specular, especially at low angles, which makes it a very good candidate for multipath errors (as the table will also reflect light coming from the wall behind it in addition to its direct reflection). A tripod is used to mount the camera at a position slightly above the table surface to provide an optimal angle for the multipath effect to occur.

7.2.1 *Ground Truth Data*

A major drawback of real data in contrast to simulated data is, that ground truth is hard to come by. This makes it harder to judge the errors caused by the demodulation.

Although it is possible to create two point clouds of the original and the corrected measurement and judge them by eye, this attempt is very subjective and more error-prone than a real ground truth comparison.

To create at least near ground truth values, it is possible to measure the scene very precisely and create a virtual model based on the hand measured values. Then the distance from each point in the scene to the camera can be computed and used as ground truth depth information. However, this is a very elaborate task and it still might suffer from inaccuracies, as measuring all objects in the scene by hand is also error-prone.

ground truth acquisition

Instead, we came up with a simple, yet efficient way to reduce the

multipath errors to a minimum: A big black sheet was held in front of the wall so that it will reflect only a minimum of light. This way, the reflections on the table are eliminated and only the primary reflection remains, which is than used to compute a near ground truth distance. These distances are not as accurate as the ground truth data from a simulated scene, but they can still be used for a comparison and can be acquired very easily.

It is important to notice, that this will only produce ground truth data for the table surface. As low intensity areas are known to cause wrong depth measurements, the rest of the image should not be used.

7.2.2 *Absolute and Relative Pixel Improvement*

To visualize the improvements achieved by our technique, we use the following formula (where O is the original measurement, C is the demodulated value and G the ground truth value):

$$v_{rel} = 1 - \left| \frac{C - G}{O - G} \right|$$

relative improvement

v_{rel} can be interpreted in the following way: One means, that the optimization is perfect and the ground truth value is directly matched. Zero means that there is no correction at all, negative values mean, that the error is increased. The advantage of taking the absolute value is, that a too strong correction that is still closer to the original value is still counted as a positive result. Therefore, if the value is directly flipped to the other side of the ground truth value, it is still considered as unchanged in terms of a correction relative to the ground truth.

absolute improvement

In some cases however, not only the relative but also the absolute correction can be interesting. This is particularly true when the measured value is already close to the ground truth but the process of correction introduces some noise. The formula to describe the absolute correction is:

$$v_{abs} = |O - G| - |C - G|$$

v_{abs} tells us how many centimeters the new value is nearer to the old one. Positive values mean an improvement, whereas negative values mean, that the new value is worse.

7.3 BASIC DEMODULATION RESULTS

Figure 34 shows the components returned by the demodulation of our test scene. Inspired by the results of chapter 6, LevMar with the new error function was used for the demodulation process.

interpretation of the results

Although the demodulation results of the wall look strange and seem to be wrong (when φ_0 is considered to be the corrected wave

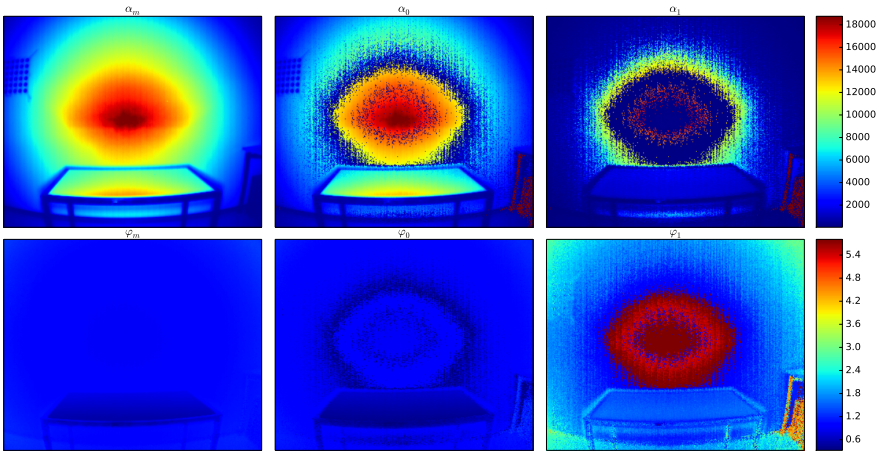

Figure 34: Demodulation results of the table scene. The demodulated phases and amplitude compared to the original measurement are shown.

and φ_1 is the phase shift of the interfering wave), they can be explained by the assumptions made in 5.3. As the lower phase resulting from the demodulation is considered as the primary component, φ_0 will not be the corrected value if the interfering wave has a smaller phase shift than the corrected wave. In the ideal case, this is impossible (as the ground truth distance is per definition the minimal distance), but inaccuracies in the values can lead to such errors (the interfering wave could also be outside the unambiguous distance, but the modulation frequencies and dimensions of the test scene where chosen to avoid this).

Especially in the visualization of the amplitude it becomes visible, that the expected values are just oddly distributed on the two components, depending on whether the interfering component has a smaller or higher phase than the main component. To get better values, the main component can be determined by the amplitude rather than the phase shift, so that the component with the higher amplitude is always the main component. However, this still resulted in a distortion of the points on the wall where no multipath is expected because the demodulation wrongly assumes an interfering wave in the still slightly noisy values.

7.3.1 *Multipath Detection*

If a specific point is not affected by multipath errors, there is no need to pick one of two wrong components as correction. Instead, the original measured value can be used, which is always free of any errors introduced by a failed correction. This motivates the idea to find a detection method for multipath errors, that can be used to decide, whether a point should be changed or not.

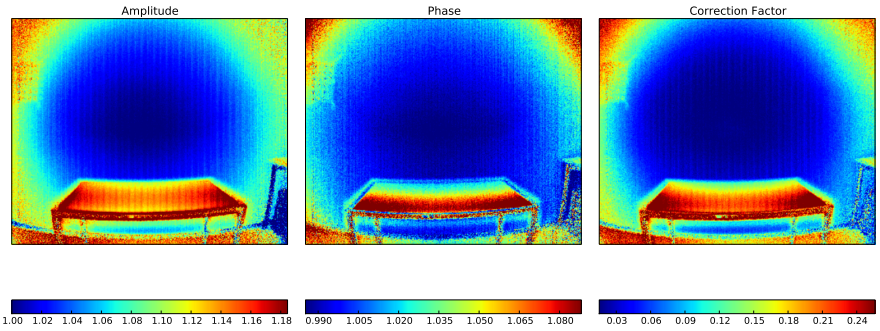

Figure 35: Multipath detection by analyzing the amplitude and phase difference between different modulation frequencies.

indicator candidates

The error value returned by the optimization algorithm is a candidate for this purpose, but it performs poorly because the optimization errors (seen in figure 45) are also small in areas without multipath effects (which means that the demodulation found a wrong pair of components which still add up to the measured values pretty well). The smaller amplitude found by the demodulation is another candidate for a detection method (in figure 34 it has only a significant value on the surface of the table), but it could also be misleading and there is also a third way which should be more robust and can additionally be used even before the demodulation.

multipath indicator

As described in 4.4, the multipath effect changes the measured phase and amplitude based on the used modulation frequency. If the (correctly calibrated) amplitude or the phase divided by the frequency changes between different frequencies this is a good indicator for multipath errors.

Figure 35 shows the relative difference in amplitude and phase between the measurements (showing $\frac{\alpha_0}{\alpha_1}$ and $\frac{2\varphi_0}{\varphi_1}$). Each highlights slightly different parts of the image and combining them by calculating

$$c_v = \left|1 - \frac{\alpha_0}{\alpha_1}\right| + \left|1 - \frac{2\varphi_0}{\varphi_1}\right|$$

leads to a good masking of multipath affected regions (pixels near the border are ignored because they are considered to be uncalibrated, see 3.2.4).

7.3.2 *Multipath Correction*

We now correct the multipath errors by applying a threshold to the multipath masking to decide whether the original phase or the demodulated phase should be used. Figure 36 shows the measurement compared to our ground truth values and the correction. The black

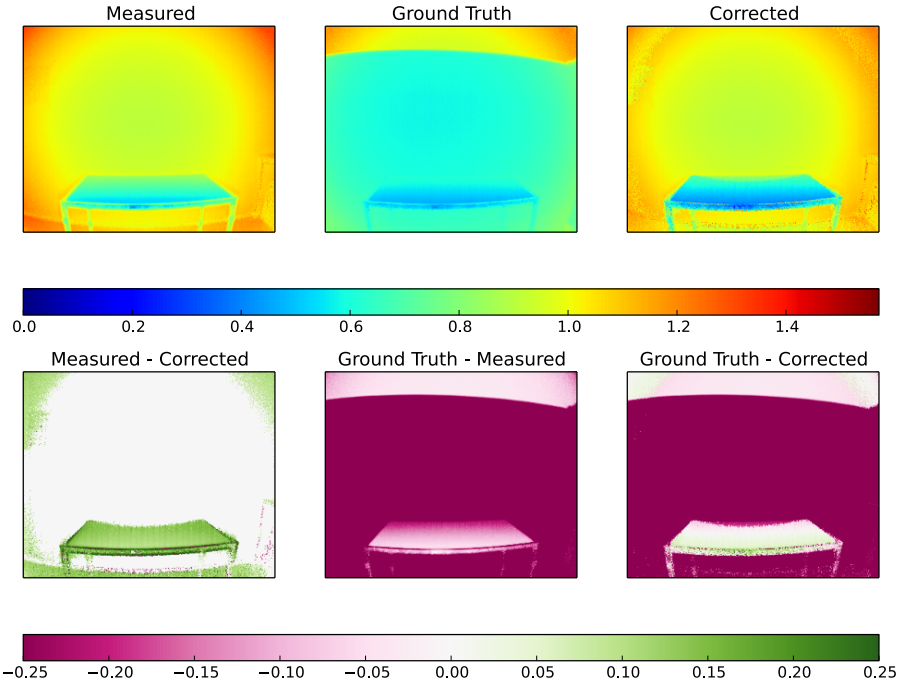

Figure 36: Multipath correction compared the the original measurement and the ground truth values. The absolute scale of the top row and the relative scale of the bottom row are both in radians.

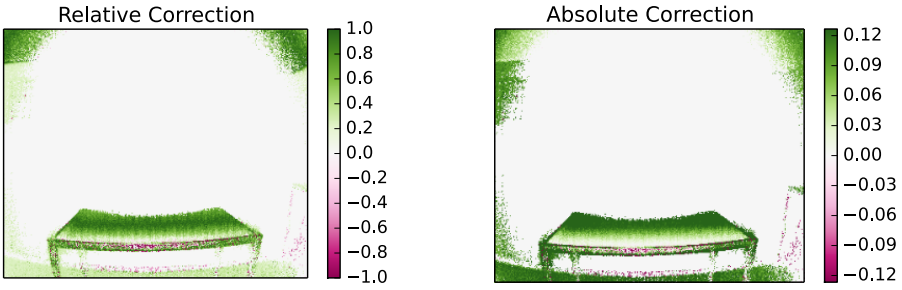

Figure 37: Relative and absolute correction computed with the formulas from section 7.2.2.

sheet used to measure ground truth values on the table is clearly visible, the ground truth values should only be used for the table surface.

correction results

The difference images show how much the table is affected by multipath errors, how much the values changed and how close it is to the ground truth values after the correction. Only small part of the back of the table was not corrected.

Figure 37 shows some more details about the correction. Some points on the front side of the table are worse than before but the majority of points is now very close to the ground truth value. The front part of the table surface suffers from less multipath errors (due to the different viewing angle) is also only slightly changed by the correction.

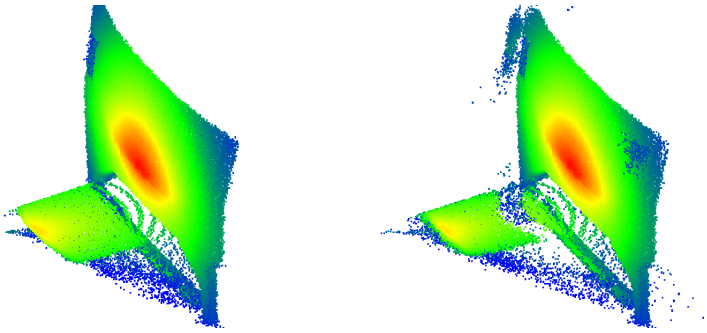

Figure 38: Point cloud of the original and corrected table scene. The color of the points represents the measured amplitude.

Figure 38 shows the results visualized as point clouds which gives a better insight in some details. The correction moved the table closer to the camera (visible by the higher distance to the wall), however some points at the end of the table failed to correct.

The results presented so far are very similar to the ones presented in [DGC+11]. In the rest of this chapter, we address some more multipath problems that were not documented in the original paper.

7.4 SPECIAL PURPOSE TEST SCENES

We also applied the multipath correction to other types of scenes, to see how they are handled.

7.4.1 *Partially Occluded View Frustums*

As explained in 4.1, the rest of the image changes, if a part of the view frustum is occluded. To test, if this can be corrected, the camera was moved closer to the table and a cardboard piece was put on top of it, around 20 centimeters in front of the camera, occluding half of its frustum. At this distance it was fully illuminated by the LED arrays which caused significant distortion in the rest of the scene.

Figure 39 shows the results of applying the correction. Although the correction suffers from many artifacts, the wall is represented far more accurate.

7.4.2 *Rounded Corners*

To test how the correction performs on the rounded corner problem, the table was put directly in front of a wooden wall closet.

Figure 40 shows the result of the correction. In the original measurement the corner clearly is not a sharp edge and the surface of the table seems to bend down. The latter artifact is however not due to light scattered in the corner, it rather results from directly reflected

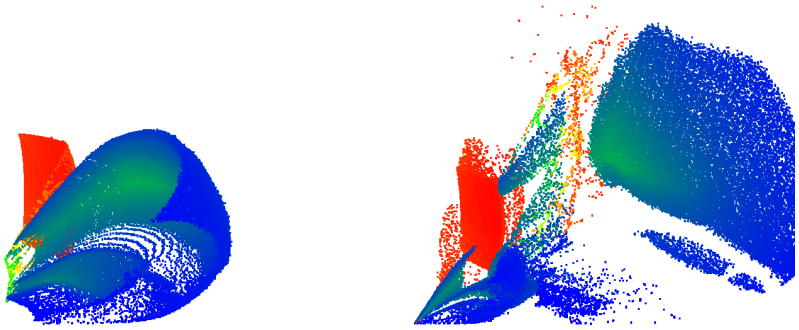

Figure 39: Results of correcting a partially occluded scene. As the colors represents the measured amplitude, the brightly illuminated cardboard appears red while the rest of the scene is mostly blue.

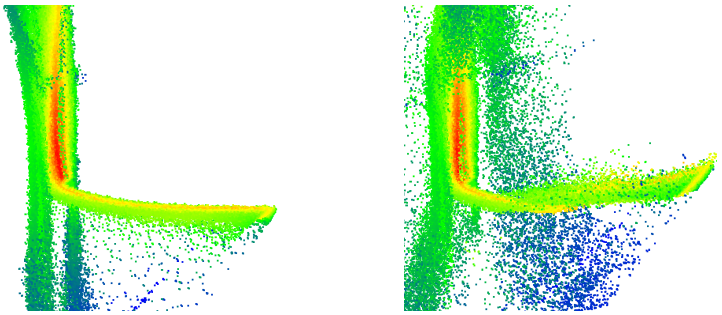

Figure 40: Results of the corner scene viewed from the side.

light on the surface of the table and has therefore the same cause as the errors in the first scene.

The bending was mostly corrected by the multipath correction, although it introduced some noise. The corner however remains round. Figure 41 shows, that the multipath detection does not mask the corner but only a part of the table surface. Furthermore the calculated phases for the corner are not an improvement, which means that the correction would not have been better, if the failed multipath detection was ignored.

The failure for this scene is not surprising. Our multipath model assumes that we have only two components that interfere, but in the corners there is an infinite number of components. Furthermore, their phases lie in a small but continuous interval which makes it practically impossible to extract individual components.

The round corner problem is therefore much better modeled by the approach presented in [JPMP12] which can also successfully correct it.

7.4.3 *Low Intensity Areas and Flying Pixel*

Distance errors due to low intensity areas and flying pixels on object edges occurred in a combined test scene. A board with a checker-

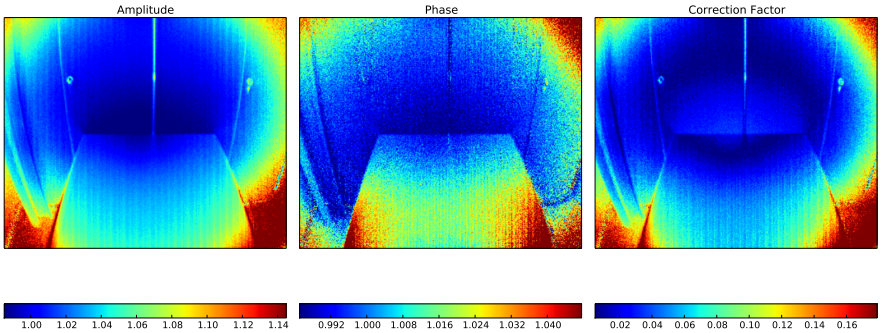

Figure 41: Multipath detection in the corner scene.

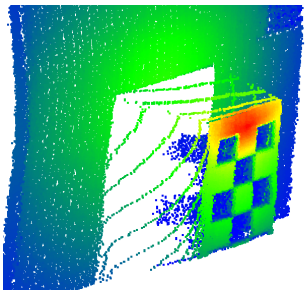

Figure 42: Flying pixels at the edge of the board. The black areas on the board have far to high depth values due to the low intensity of the reflected light.

board pattern was positioned around half a meter away from the wall. Figure 42 shows, that the sharp edges of the board produced flying pixels and the low intensity areas of the checkerboard have a significantly higher distance compared to the rest of the board.

The multipath detection shown in figure 43 works well for the low intensity areas, but for the flying pixels, no clear line around the board is visible.

The multipath correction shown in figure 44 fails however. The algorithm is unable to find correct phases for the low intensity areas and the demodulated phases contain so much noise that even if some flying pixels were corrected, they could not be distinguished from the rest of the noise which makes a correction impossible.

The results presented for this scene are representative for other tested flying pixel and low intensity scenes. While the algorithm performed well for correcting reflective surfaces it fails in these two task.

While the exact cause for the errors in low intensity areas is still not fully understood, the correction should at least work for flying pixels, as they match the model of exactly two different components perfectly.

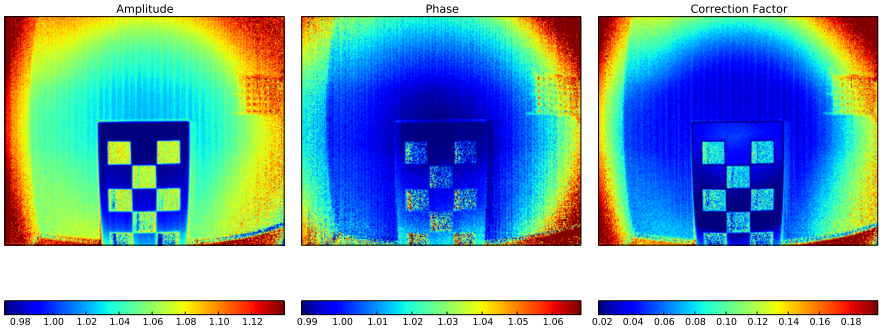

Figure 43: Multipath detection for the flying pixels scene. The dark areas are masked correctly but there is no detection of the flying pixels visible.

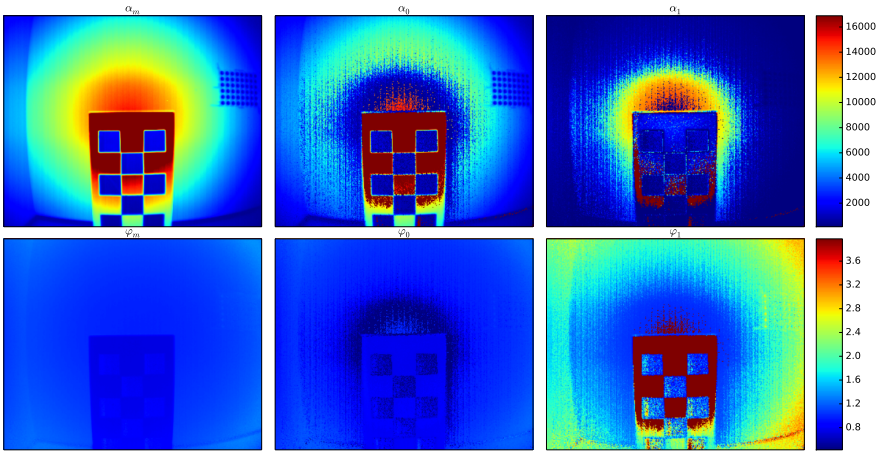

Figure 44: Demodulation results of the flying pixel scene. It contains very much noise and neither the dark areas nor the flying pixels could be corrected.

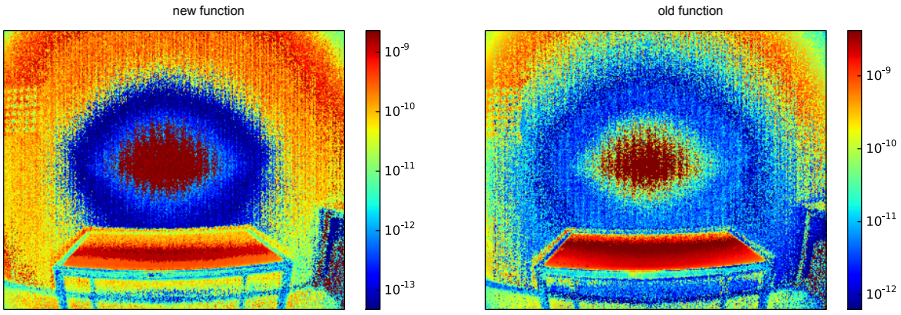

Figure 45: Minimization errors of the new and old error function in a logarithmic scale.

7.5 COMPARISON BETWEEN THE OLD AND NEW METHOD

To evaluate how the new minimization function developed in chapter 6 performs on real data, the table scene was demodulated using the new and the old function with LevMar and BFGS. Powell was not further considered, as it already had too bad results on simulated and data without noise.

Apart from minimal differences, the demodulation results of all four methods are equal and no method is superior. The minimization errors are also comparable, they are completely equal for LevMar and BFGS and have only small differences between the new and the old methods that are shown in 45. The new method has slightly smaller errors, but in practice this is negligible.

The following table shows the run times of the algorithms:

	BFGS	LevMar
new method	302,50s	303,38s
old method	270,25s	271,13s

The table shows the results for the table scene but is representative for other scenes. The times are similar, especially it does not matter, whether BFGS or LevMar is used. The old minimization function is however faster than the new one.

7.5.1 *Interpretation*

The result of the comparison is somewhat surprising as it differs from the values calculated in 6. Normally, the new minimization function should be clearly faster and the choice of the optimization algorithm should also have a greater impact on the computation speed.

early termination

A possible explanation for this is, that the optimization ends earlier, if no good demodulation can be found. If the data are affected by noise too much, there might not even be a good minimum that can be found and the optimization returns just a local minimum. This was

also the case in some previous experiments with other optimization algorithms, which failed to find a solution even in the simulated data but had a far lower computation time compared to the algorithms that succeeded.

poor camera data

As shown in section 3.2 the calibration of the camera is far from being perfect. Additionally, the optimization is very noise sensitive as shown in 6.5. Together this makes the minimization very hard and error-prone. As it was not possible to get better camera data, it could not be tested, how the multipath correction would have worked with them.

The computation time is also not only affected by the big parts of the scene, that do not suffer from the multipath effect and thus make the demodulation there senseless. If only the pixels that contain multipath errors are demodulated, the computation time behaves the same.

This means, that the quality of the measured data is simply not high enough, to benefit from theoretical advantages in applications with real data.

8

CONCLUSION FUTURE WORK

8.1 CONCLUSION

Our implementation of the multipath correction approach described in Dorrington et al. can reproduce the results presented in the original paper. The depth information of surfaces reflecting light of other parts of the scene can be significantly enhanced and the corrected values are very close to the ground truth. We also tested the approach in a number of other scenarios but the results where mixed. While stray light errors inside the camera caused by close and brightly illuminated objects could be corrected, flying pixels and low intensity areas could not be enhanced. Multipath errors occurring in corners could also not be corrected but as their cause does not match our model, this cannot be considered to be a failure.

Problems like these are not addressed in the original publication directly, but it states that the correction fails in some scenes and only makes the measurement worse. It is therefore likely that they suffer from the same problems as we do.

The result of our thorough analysis of the demodulation shows, that it can produce wrong results even for precisely measured values which gives us a theoretical limit for the correction potential that cannot be overcome by more advanced optimization approaches or even closed form solutions. The method is able to correct multipath errors, but it has to be used with care as it is likely to introduce more errors than it corrects. It can be used for a manual enhancement of images, but it is too unstable to be a default post processing step in the capturing process.

The new minimization function that we introduced to enhance the numerical optimization has proven its potential when it is applied on simulated data. All evaluated minimization algorithms work significantly faster with the new function and the probability of finding the correct minimum increases likewise. However this cannot be carried on to the real data, as their accuracy is not high enough.

8.2 FUTURE WORK

There are various approaches that could help to improve the multipath correction. As the demodulation is very sensitive to inaccurate values, a better camera calibration would most likely improve the results, but as there is a theoretical limit, there is no chance to improve the captured data so far that a correct demodulation can be guaran-

teed in all cases. The only way to improve the robustness would be, to measure the scene with a higher number of frequencies. This would make the optimization more difficult, but would certainly increase its stability. The newly proposed minimization function could also be extended to work with more components.

However, using more frequencies makes the image acquisition more complex and time demanding. Furthermore, there are also other multi-frequency methods that use four or five measurements and are either faster to compute or lead to better results. This raises the question, whether the method implemented in this thesis would benefit enough from more frequencies to surpass the other methods.

A

APPENDIX

SUM OF TWO SINUS FUNCTIONS

$$
\begin{aligned}
w_0(t) &= \alpha_0 \sin(t - \varphi_0) \\
w_1(t) &= \alpha_1 \sin(t - \varphi_1) \\
w_m(t) &= \alpha_0 \sin(t - \varphi_0) + \alpha_1 \sin(t - \varphi_1) \\
&= \sqrt{\alpha_0^2 + \alpha_1^2 + 2\alpha_0\alpha_1 \cos(\varphi_0 - \varphi_1)} \cdot \sin(t + \delta) \\
&\text{with } \delta = \operatorname{atant2}(\alpha_0 \sin\varphi_0 + \alpha_1 \sin\varphi_1,\ \alpha_0 \cos\varphi_0 + \alpha_1 \cos\varphi_1)
\end{aligned}
$$

DEVIATION OF THE MINIMIZATION FUNCTION

The error function for numerical minimization is:

$$
\begin{aligned}
f(\alpha_0, \alpha_1, \tau_0, \tau_1) = \sum_{\varphi \in \mathbb{P}} &\left(w(\varphi) - \left(\frac{1}{2}\alpha_0 \cdot \cos(\tau_0 - \varphi) + \frac{1}{2}\alpha_1 \cdot \cos(\tau_1 - \varphi) \right) \right)^2 \\
&+ \left(w'(\varphi) - \left(\frac{1}{2}\alpha_0 \cdot \cos(2 \cdot \tau_0 - \varphi) + \frac{1}{2}\alpha_1 \cdot \cos(2 \cdot \tau_1 - \varphi) \right) \right)^2
\end{aligned}
$$

with the 8 constants $(w(\varphi_0), \ldots, w(\varphi_3), w'(\varphi_0), \ldots, w'(\varphi_3)$.
The derivation of this function is:

$$
\begin{aligned}
f(\alpha_0, \alpha_1, \tau_0, \tau_1) &= \sum_{\varphi \in \mathbb{P}} T_0^2 + T_1^2 \\
T_0 &= w(\varphi) - \left(\frac{1}{2}\alpha_0 \cdot \cos(\tau_0 - \varphi) + \frac{1}{2}\alpha_1 \cdot \cos(\tau_1 - \varphi) \right) \\
T_1 &= w'(\varphi) - \left(\frac{1}{2}\alpha_0 \cdot \cos(2 \cdot \tau_0 - \varphi) + \frac{1}{2}\alpha_1 \cdot \cos(2 \cdot \tau_1 - \varphi) \right)
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial f}{\alpha_0} &= \sum_{\varphi \in \mathbb{P}} 2\left(-\frac{1}{2}\cos(\tau_0 - \varphi) \cdot T_0 \right) + 2\left(-\frac{1}{2}\cos(2\tau_0 - \varphi) \cdot T_1 \right) \\
&= \sum_{\varphi \in \mathbb{P}} -\cos(\tau_0 - \varphi) \cdot T_0 - \cos(2\tau_0 - \varphi) \cdot T_1
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial f}{\alpha_1} &= \sum_{\varphi \in \mathbb{P}} 2\left(-\frac{1}{2}\cos(\tau_1 - \varphi) \cdot T_0 \right) + 2\left(-\frac{1}{2}\cos(2\tau_1 - \varphi) \cdot T_1 \right) \\
&= \sum_{\varphi \in \mathbb{P}} -\cos(\tau_1 - \varphi) \cdot T_0 - \cos(2\tau_1 - \varphi) \cdot T_1
\end{aligned}
$$

$$\frac{\partial f}{\tau_0} = \sum_{\varphi \in \mathbb{P}} 2\left(-\frac{1}{2}\alpha_0 \cdot -\sin\left(\tau_0 - \varphi\right) \cdot T_0\right) + 2\left(-\frac{1}{2}\alpha_0 \cdot -\sin\left(2\tau_0 - \varphi\right) \cdot 2 \cdot T_1\right)$$
$$= \sum_{\varphi \in \mathbb{P}} \alpha_0 \cdot \sin\left(\tau_0 - \varphi\right) \cdot T_0 + 2\alpha_0 \cdot \sin\left(2\tau_0 - \varphi\right) \cdot T_1$$

$$\frac{\partial f}{\tau_1} = \sum_{\varphi \in \mathbb{P}} 2\left(-\frac{1}{2}\alpha_1 \cdot -\sin\left(\tau_1 - \varphi\right) \cdot T_0\right) + 2\left(-\frac{1}{2}\alpha_1 \cdot -\sin\left(2\tau_1 - \varphi\right) \cdot 2 \cdot T_1\right)$$
$$= \sum_{\varphi \in \mathbb{P}} \alpha_1 \cdot \sin\left(\tau_1 - \varphi\right) \cdot T_0 + 2\alpha_1 \cdot \sin\left(2\tau_1 - \varphi\right) \cdot T_1$$

OPTIMAL AMPLITUDE FUNCTIONS

Unfortunately the optimal amplitude functions are far to long as that they could be printed in the normal mathematical notation; instead we show the Python code for the functions.

$\alpha_0\left(\tau_0, \tau_1\right)$

```
def a0Fun(p0, p1, w0, w1):
    tau0, tau1 = p0, p1
    w00, w01, w02, w03 = w0
    w10, w11, w12, w13 = w1

    a=sin(tau0)
    b=sin(tau1)
    c=sin(2*tau0)
    d=sin(2*tau1)

    e=cos(tau0)
    f=cos(tau1)
    g=cos(2*tau0)
    h=cos(2*tau1)

    return -(((g*h+a*b+e*f)*d-c*h**2-c)*w13+
        (c*h*d+g*h**2+(a*b+e*f)*h-2*g)*w12+
        ((-g*h-a*b-e*f)*d+c*h**2+c)*w11+
        (-c*h*d-g*h**2+(-a*b-e*f)*h+2*g)*w10+
        (c*b*d+g*b*h+a*b**2+e*f*b-2*a)*w03+
        (c*f*d+g*f*h-e*b**2+a*f*b-e)*w02+
        (-c*b*d-g*b*h-a*b**2-e*f*b+2*a)*w01+
        (-c*f*d-g*f*h+e*b**2-a*f*b+e)*w00)/(
        (2*g*c*h+2*a*c*b+2*e*c*f)*d+(2*g**2-1)*h**2+
        (2*a*g*b+2*e*g*f)*h+(2*a**2-1)*b**2+2*e*a*f*b-g**2-a**2-2)
```

$\alpha_1\left(\tau_0, \tau_1\right)$

```
def a1Fun(p0, p1, w0, w1):
    tau0, tau1 = p0, p1
    w00, w01, w02, w03 = w0
    w10, w11, w12, w13 = w1

```

```
    a=sin(tau0)
    b=sin(tau1)
    c=sin(2*tau0)
    d=sin(2*tau1)

    e=cos(tau0)
    f=cos(tau1)
    g=cos(2*tau0)
    h=cos(2*tau1)

    return (((g**2+1)*d-g*c*h-a*c*b-e*c*f)*w13+
        (-g*c*d+(2-g**2)*h-a*g*b-e*g*f)*w12+
        ((-g**2-1)*d+g*c*h+a*c*b+e*c*f)*w11+
        (g*c*d+(g**2-2)*h+a*g*b+e*g*f)*w10+
        (-a*c*d-a*g*h+(2-a**2)*b-e*a*f)*w03+
        (-e*c*d-e*g*h-e*a*b+(a**2+1)*f)*w02+
        (a*c*d+a*g*h+(a**2-2)*b+e*a*f)*w01+
        (e*c*d+e*g*h+e*a*b+(-a**2-1)*f)*w00)/(
        (2*g*c*h+2*a*c*b+2*e*c*f)*d+(2*g**2-1)*h**2+
        (2*a*g*b+2*e*g*f)*h+(2*a**2-1)*b**2+2*e*a*f*b-g**2-a**2-2)
```

DERIVATION OF THE ERROR FUNCTION

Calculating the derivations of f_2 is in one way easier, as we now have only two dimensions, but on the other hand we need the derivations of α_0 and α_1, which are again only given as Python code.

$\frac{\partial \alpha_o}{\tau_0}$

```
def a0tau0Fun(p0, p1, w0, w1):
    tau0, tau1 = p0, p1
    w00, w01, w02, w03 = w0
    w10, w11, w12, w13 = w1

    a=sin(tau0)
    b=sin(tau1)
    c=sin(2*tau0)
    d=sin(2*tau1)

    e=cos(tau0)
    f=cos(tau1)
    g=cos(2*tau0)
    h=cos(2*tau1)

    return (((-4*c**2*h+4*g**2*h+2*e*c*b+4*a*g*b-2*a*c*f+4*e*g*f)*
            d-8*g*c*h**2+(-4*a*c*b+2*e*g*b-4*e*c*f-2*a*g*f)*h+4*
            e*a*b**2-2*a**2*f*b+2*e**2*f*b+4*g*c-2*e*a)*(
            ((g*h+a*b+e*f)*d-c*h**2-c)*w13+
            (c*h*d+g*h**2+(a*b+e*f)*h-2*g)*w12+
            ((-g*h-a*b-e*f)*d+c*h**2+c)*w11+
            (-c*h*d-g*h**2+(-a*b-e*f)*h+2*g)*w10+
            (c*b*d+g*b*h+a*b**2+e*f*b-2*a)*w03+
            (c*f*d+g*f*h-e*b**2+a*f*b-e)*w02+
```

```
            (-c*b*d-g*b*h-a*b**2-e*f*b+2*a)*w01+
            (-c*f*d-g*f*h+e*b**2-a*f*b+e)*w00))/((
            (2*g*c*h+2*a*c*b+2*e*c*f)*d+(2*g**2-1)*h**2+
            (2*a*g*b+2*e*g*f)*h+(2*a**2-1)*b**2+2*e*a*f*b-g**2-a**2-2)
                **(2))-(
            ((-2*c*h+e*b-a*f)*d-2*g*h**2-2*g)*w13+
            (2*g*h*d-2*c*h**2+(e*b-a*f)*h+4*c)*w12+
            ((2*c*h-e*b+a*f)*d+2*g*h**2+2*g)*w11+
            (-2*g*h*d+2*c*h**2+(a*f-e*b)*h-4*c)*w10+
            (2*g*b*d-2*c*b*h+e*b**2-a*f*b-2*e)*w03+
            (2*g*f*d-2*c*f*h+a*b**2+e*f*b+a)*w02+
            (-2*g*b*d+2*c*b*h-e*b**2+a*f*b+2*e)*w01+
            (-2*g*f*d+2*c*f*h-a*b**2-e*f*b-a)*w00)/(
            (2*g*c*h+2*a*c*b+2*e*c*f)*d+(2*g**2-1)*h**2+
            (2*a*g*b+2*e*g*f)*h+(2*a**2-1)*b**2+2*e*a*f*b-g**2-a**2-2)
```

$\frac{\partial \alpha_0}{\tau_1}$

```
def a0tau1Fun(p0, p1, w0, w1):
    tau0, tau1 = p0, p1
    w00, w01, w02, w03 = w0
    w10, w11, w12, w13 = w1

    a=sin(tau0)
    b=sin(tau1)
    c=sin(2*tau0)
    d=sin(2*tau1)

    e=cos(tau0)
    f=cos(tau1)
    g=cos(2*tau0)
    h=cos(2*tau1)

    return ((d*(-4*g*c*d-2*e*c*b+2*a*c*f)-4*(2*g**2-1)*h*d-2*
            (2*a*g*b+2*e*g*f)*d+2*h*(2*g*c*h+2*a*c*b+2*e*c*f)+
            (2*a*g*f-2*e*g*b)*h-2*e*a*b**2+2*(2*a**2-1)*f*b+2*e*a*f**2)*(
            ((g*h+a*b+e*f)*d-c*h**2-c)*w13+
            (c*h*d+g*h**2+(a*b+e*f)*h-2*g)*w12+
            ((-g*h-a*b-e*f)*d+c*h**2+c)*w11+
            (-c*h*d-g*h**2+(-a*b-e*f)*h+2*g)*w10+
            (c*b*d+g*b*h+a*b**2+e*f*b-2*a)*w03+
            (c*f*d+g*f*h-e*b**2+a*f*b-e)*w02+
            (-c*b*d-g*b*h-a*b**2-e*f*b+2*a)*w01+
            (-c*f*d-g*f*h+e*b**2-a*f*b+e)*w00))/((
            (2*g*c*h+2*a*c*b+2*e*c*f)*d+(2*g**2-1)*h**2+
            (2*a*g*b+2*e*g*f)*h+(2*a**2-1)*b**2+2*e*a*f*b-g**2-a**2-2)
                **(2))-(
            (d*(-2*g*d-e*b+a*f)+4*c*h*d+2*h*(g*h+a*b+e*f))*w13+
            (-2*c*d**2-4*g*h*d-2*(a*b+e*f)*d+2*c*h**2+(a*f-e*b)*h)*w12+
            (d*(2*g*d+e*b-a*f)-4*c*h*d+2*h*(-g*h-a*b-e*f))*w11+
            (2*c*d**2+4*g*h*d-2*(-a*b-e*f)*d-2*c*h**2+(e*b-a*f)*h)*w10+
            (-2*g*b*d+c*f*d+2*c*b*h+g*f*h-e*b**2+2*a*f*b+e*
            f**2)*w03+(-c*b*d-2*g*f*d-g*b*h+2*c*f*h-a*b**2-2*e*f*
            b+a*f**2)*w02+(2*g*b*d-c*f*d-2*c*b*h-g*f*h+e*b**2-2*
            a*f*b-e*f**2)*w01+(c*b*d+2*g*f*d+g*b*h-2*c*f*h+a*
            b**2+2*e*f*b-a*f**2)*w00)/((2*g*c*h+2*a*c*b+2*e*c*f)*d+
            (2*g**2-1)*h**2+(2*a*g*b+2*e*g*f)*h+(2*a**2-1)*b**2+2*e*a*f*b
                -g**2-a**2
```

```
            -2)
```

$\frac{\partial \alpha_1}{\tau_0}$

```
def a1tau0Fun(p0, p1, w0, w1):
    tau0, tau1 = p0, p1
    w00, w01, w02, w03 = w0
    w10, w11, w12, w13 = w1

    a=sin(tau0)
    b=sin(tau1)
    c=sin(2*tau0)
    d=sin(2*tau1)

    e=cos(tau0)
    f=cos(tau1)
    g=cos(2*tau0)
    h=cos(2*tau1)

    return ((-4*g*c*d+2*c**2*h-2*g**2*h-e*c*b-2*a*g*b+a*c*f-2*
            e*g*f)*w13+(2*c**2*d-2*g**2*d+4*g*c*h+2*a*c*b-e*g*b+2*e*
            c*f+a*g*f)*w12+(4*g*c*d-2*c**2*h+2*g**2*h+e*c*b+2*a*g*
            b-a*c*f+2*e*g*f)*w11+(-2*c**2*d+2*g**2*d-4*g*c*h-2*a*c*
            b+e*g*b-2*e*c*f-a*g*f)*w10+(-e*c*d-2*a*g*d+2*a*
            c*h-e*g*h-2*e*a*b+a**2*f-e**2*f)*w03+(a*c*d-2*e*g*
            d+2*e*c*h+a*g*h+a**2*b-e**2*b+2*e*a*f)*w02+(e*c*d+2*
            a*g*d-2*a*c*h+e*g*h+2*e*a*b-a**2*f+e**2*f)*w01+(-a*
            c*d+2*e*g*d-2*e*c*h-a*g*h-a**2*b+e**2*b-2*e*a*f)*
            w00)/((2*g*c*h+2*a*c*b+2*e*c*f)*d+(2*g**2-1)*h**2+
            (2*a*g*b+2*e*g*f)*h+(2*a**2-1)*b**2+2*e*a*f*b-g**2-a**2-2)-((
            (-4*c**2*h+4*g**2*h+2*e*c*b+4*a*g*b-2*a*c*f+4*e*g*f)*d-8*
            g*c*h**2+(-4*a*c*b+2*e*g*b-4*e*c*f-2*a*g*f)*h+4*e*a
            *b**2-2*a**2*f*b+2*e**2*f*b+4*g*c-2*e*a)*(
            ((g**2+1)*d-g*c*h-a*c*b-e*c*f)*w13+
            (-g*c*d+(2-g**2)*h-a*g*b-e*g*f)*w12+
            ((-g**2-1)*d+g*c*h+a*c*b+e*c*f)*w11+
            (g*c*d+(g**2-2)*h+a*g*b+e*g*f)*w10+
            (-a*c*d-a*g*h+(2-a**2)*b-e*a*f)*w03+
            (-e*c*d-e*g*h-e*a*b+(a**2+1)*f)*w02+
            (a*c*d+a*g*h+(a**2-2)*b+e*a*f)*w01+
            (e*c*d+e*g*h+e*a*b+(-a**2-1)*f)*w00))/((
            (2*g*c*h+2*a*c*b+2*e*c*f)*d+(2*g**2-1)*h**2+
            (2*a*g*b+2*e*g*f)*h+(2*a**2-1)*b**2+2*e*a*f*b-g**2-a**2-2)
                **2)
```

$\frac{\partial \alpha_1}{\tau_1}$

```
def a1tau1Fun(p0, p1, w0, w1):
    tau0, tau1 = p0, p1
    w00, w01, w02, w03 = w0
    w10, w11, w12, w13 = w1

    a=sin(tau0)
    b=sin(tau1)
    c=sin(2*tau0)
    d=sin(2*tau1)

    e=cos(tau0)
```

```
    f=cos(tau1)
    g=cos(2*tau0)
    h=cos(2*tau1)


    return ((2*g*c*d+2*(g**2+1)*h+e*c*b-a*c*f)*w13+
            (-2*(2-g**2)*d-2*g*c*h+e*g*b-a*g*f)*w12+
            (-2*g*c*d+2*(-g**2-1)*h-e*c*b+a*c*f)*w11+
            (-2*(g**2-2)*d+2*g*c*h-e*g*b+a*g*f)*w10+
            (2*a*g*d-2*a*c*h+e*a*b+(2-a**2)*f)*w03+
            (2*e*g*d-2*e*c*h-(a**2+1)*b-e*a*f)*w02+
            (-2*a*g*d+2*a*c*h-e*a*b+(a**2-2)*f)*w01+
            (-2*e*g*d+2*e*c*h-(-a**2-1)*b+e*a*f)*w00)/(
            (2*g*c*h+2*a*c*b+2*e*c*f)*d+(2*g**2-1)*h**2+
            (2*a*g*b+2*e*g*f)*h+(2*a**2-1)*b**2+2*e*a*f*b-g**2-a**2-2)-((
                d*
            (-4*g*c*d-2*e*c*b+2*a*c*f)-4*(2*g**2-1)*h*d-2*
            (2*a*g*b+2*e*g*f)*d+2*h*(2*g*c*h+2*a*c*b+2*e*c*f)+
            (2*a*g*f-2*e*g*b)*h-2*e*a*b**2+2*(2*a**2-1)*f*b+2*e*a*f**2)*(
            ((g**2+1)*d-g*c*h-a*c*b-e*c*f)*w13+
            (-g*c*d+(2-g**2)*h-a*g*b-e*g*f)*w12+
            ((-g**2-1)*d+g*c*h+a*c*b+e*c*f)*w11+
            (g*c*d+(g**2-2)*h+a*g*b+e*g*f)*w10+
            (-a*c*d-a*g*h+(2-a**2)*b-e*a*f)*w03+
            (-e*c*d-e*g*h-e*a*b+(a**2+1)*f)*w02+
            (a*c*d+a*g*h+(a**2-2)*b+e*a*f)*w01+
            (e*c*d+e*g*h+e*a*b+(-a**2-1)*f)*w00))/((
            (2*g*c*h+2*a*c*b+2*e*c*f)*d+(2*g**2-1)*h**2+
            (2*a*g*b+2*e*g*f)*h+(2*a**2-1)*b**2+2*e*a*f*b-g**2-a**2-2)
                **2)
```

With that, we get the Jacobi matrix for our error function with:

$$
\begin{aligned}
f_2(\tau_0,\tau_1) &= \sum_{\varphi\in\mathbb{P}} T_0^2 + T_1^2 \\
T_0 &= w(\varphi) - \left(\frac{1}{2}\alpha_0(\tau_0,\tau_1)\cdot\cos(\tau_0-\varphi) + \frac{1}{2}\alpha_1(\tau_0,\tau_1)\cdot\cos(\tau_1-\varphi)\right) \\
T_1 &= w'(\varphi) - \left(\frac{1}{2}\alpha_0(\tau_0,\tau_1)\cdot\cos(2\cdot\tau_0-\varphi) + \frac{1}{2}\alpha_1(\tau_0,\tau_1)\cdot\cos(2\cdot\tau_1-\varphi)\right)
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial f_2(\tau_0,\tau_1)}{\tau_0} = \sum_{\varphi\in\mathbb{P}} & T_0\cdot\left(-\left(\alpha_1(\tau_0,\tau_1)\frac{d}{d\tau_0}\right)\cdot\cos(\tau_1-\varphi) - \left(\alpha_0(\tau_0,\tau_1)\frac{d}{d\tau_0}\right)\cdot\cos(\tau_0-\varphi)\right) \\
& + T_0\cdot(\alpha_0(\tau_0,\tau_1)\cdot\sin(\tau_o-\varphi)) \\
& + T_1\cdot\left(-\left(\alpha_1(\tau_0,\tau_1)\frac{d}{d\tau_0}\right)\cdot\cos(2\cdot\tau_1-\varphi) - \left(\alpha_0(\tau_0,\tau_1)\frac{d}{d\tau_0}\right)\cdot\cos(2\cdot\tau_0-\varphi)\right) \\
& + T_1\cdot(2\cdot\alpha_0(\tau_0,\tau_1)\cdot\sin(2\cdot\tau_o-\varphi))
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial f_2(\tau_0,\tau_1)}{\tau_1} = \sum_{\varphi\in\mathbb{P}} & T_0\cdot\left(-\left(\alpha_1(\tau_0,\tau_1)\frac{d}{d\tau_1}\right)\cdot\cos(\tau_1-\varphi) - \left(\alpha_0(\tau_0,\tau_1)\frac{d}{d\tau_1}\right)\cdot\cos(\tau_0-\varphi)\right) \\
& + T_0\cdot(\alpha_1(\tau_0,\tau_1)\cdot\sin(\tau_1-\varphi)) \\
& + T_1\cdot\left(-\left(\alpha_1(\tau_0,\tau_1)\frac{d}{d\tau_1}\right)\cdot\cos(2\cdot\tau_1-\varphi) - \left(\alpha_0(\tau_0,\tau_1)\frac{d}{d\tau_1}\right)\cdot\cos(2\cdot\tau_0-\varphi)\right) \\
& + T_1\cdot(2\cdot\alpha_1(\tau_0,\tau_1)\cdot\sin(2\cdot\tau_1-\varphi))
\end{aligned}
$$

UNDISTORTION FUNCTION

```
#compute distorted coordinate
def DistordCoordinates(u, v):
    # lens intrinsics measured by us
    CenterPixel=[1.6582831268049244e+002, 1.3244300024849093e+002]
    Focal=[3.4774794698371841e+002, 3.4774794698371841e+002]
    Distortion=[ -4.6859880979773544e-001, 2.0101237961910157e-001, 0.,
        0., 0. ]

    x=(u-CenterPixel[0])/Focal[0]
    y=(v-CenterPixel[1])/Focal[1]

    r2=(x**2+y**2)
    rad_final=1 + r2*Distortion[0] + r2**2*Distortion[1]
    xUnd=x*rad_final + Distortion[2]*(2*x*y) + Distortion[3]*(r2 + 2*x
        **2)
    yUnd=y*rad_final + Distortion[3]*(2*x*y) + Distortion[2]*(r2 + 2*y
        **2)

    uUnd=xUnd * Focal[0]+CenterPixel[0]
    vUnd=yUnd * Focal[1]+CenterPixel[1]

    return uUnd,vUnd
```

SCRIPTS AND MATERIALS

The demodulation and analysis scripts that were written for this thesis are available on the homepage of the Institute for Vision and Graphics. The package also includes the camera data that can be used to reproduce the result that are shown in this thesis.

http://www.cg.informatik.uni-siegen.de/en/publications

LIST OF FIGURES

Figure 1 Depth Map, Color Image and Point Cloud of a Scene 1
Figure 2 Time-of-Flight Principle Overview 3
Figure 3 Different Signals for the Time-of-Flight Measurement 5
Figure 4 Integration over the Correlation Function 6
Figure 5 Sample Points of the Correlation Function 7
Figure 6 Derivation of the Amplitude Reconstruction Formula 10
Figure 7 PMD Digicam Prototype 15
Figure 8 Cartesian vs. Polar Coordinates 16
Figure 9 Camera Calibration Pictures 18
Figure 10 Calibration Picture Masking 19
Figure 11 Samples for Camera Phase Calibration 19
Figure 12 Phase Calibration Functions 20
Figure 13 Amplitude Calibration Functions 22
Figure 14 Phase Calibration Results 23
Figure 15 Amplitude Calibration Results 24
Figure 16 Camera Noise Distribution 25
Figure 17 Camera Channel Noise 27
Figure 18 Normalized Camera Channel Noise 27
Figure 19 Illustration of the Multipath Effect 30
Figure 20 Multiple Frequencies Measurement 34
Figure 21 Ambiguous Correlation Function Demodulation 39
Figure 22 Starting Point Analysis for BFGS 44
Figure 23 Starting Point Analysis for Powell 44
Figure 24 Starting Point Analysis for LevMar 45
Figure 25 Mimimum Cluster Positions 46
Figure 26 Minimum Cluster Errors 46
Figure 27 Minimum Cluster Origins 47
Figure 28 Twodimensional Minimization Function and Derivative 52
Figure 29 New Minimization Function Algorithm Comparison 53
Figure 30 Run Time Comparison 54
Figure 31 Noise Affection of the Demodulation 55
Figure 32 Distorted and Undistorted Camera Image 58
Figure 33 Table Test Scene Setup 59
Figure 34 Demodulation Results of the Table Scene 61
Figure 35 Multipath Detection in Table Scene 62

Figure 36 Ground Truth Comparison 63
Figure 37 Relative and Absolute Correction 63
Figure 38 Table Scene Point Clouds 64
Figure 39 Occluded Scene Point Clouds 65
Figure 40 Corner Scene Point Clouds 65
Figure 41 Corner Scene Multipath Detection 66
Figure 42 Flying Pixels Scene 66
Figure 43 Flying Pixel Scene Multipath Detection 67
Figure 44 Flying Pixel Scene Demodulation Results 67
Figure 45 Minimization Error Comparison 68

LIST OF TABLES

Table 1 Calibration Polynomials for the Phase 21
Table 2 Calibration Polynomials for the Amplitude 23
Table 3 Camera Noise Values 26

BIBLIOGRAPHY

[com09] The Scipy community. Optimization and root finding (scipy.optimize), 2008-2009. visited on 28.11.2013. URL: http://docs.scipy.org/doc/scipy/reference/optimize.html. (Cited on page 41.)

[DGC+11] A. A. Dorrington, J. P. Godbaz, M. J. Cree, A. D. Payne, and L. V. Streeter. Separating true range measurements from multi-path and scattering interference in commercial range cameras, 2011. URL: http://dx.doi.org/10.1117/12.876586, doi:10.1117/12.876586. (Cited on pages ix, 10, 29, 32, 33, 35, 59, and 64.)

[DHB10] D. Droeschel, D. Holz, and S. Behnke. Multi-frequency phase unwrapping for time-of-flight cameras. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1463–1469, 2010. doi:10.1109/IROS.2010.5649488. (Cited on page 13.)

[Fuc10] Stefan Fuchs. Multipath interference compensation in tof-camera images. In *Proceedings of the ICRA 2010, Istanbul, Turkey*, August 2010. (Cited on pages 29 and 32.)

[GCD12] John P. Godbaz, Michael J. Cree, and Adrian A. Dorrington. Closed-form inverses for the mixed pixel/-multipath interference problem in amcw lidar, 2012. URL: http://dx.doi.org/10.1117/12.909778, doi:10.1117/12.909778. (Cited on pages 33, 35, and 37.)

[JPMP12] D. Jimenez, D. Pizarro, M. Mazo, and S. Palazuelos. Modelling and correction of multipath interference in time of flight cameras. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 0:893–900, 2012. doi:http://doi.ieeecomputersociety.org/10.1109/CVPR.2012.6247763. (Cited on pages 32 and 65.)

[KBC13] A. Kirmani, A. Benedetti, and P.A. Chou. Spumic: Simultaneous phase unwrapping and multipath interference cancellation in time-of-flight cameras using spectral methods. In *Multimedia and Expo (ICME), 2013 IEEE International Conference on*, pages 1–6, 2013. doi:10.1109/ICME.2013.6607553. (Cited on pages 29 and 33.)

[KWB+13] A. Kadambi, R. Whyte, A. Bhandari, L. Streeter, C. Barsi, A. Dorrington, and R. Raskar. Coded time of flight cam-

eras: sparse deconvolution to address multipath interference and recover time profiles. *ACM Transactions on Graphics (TOG)*, 32(6):167, 2013. (Cited on pages 29 and 34.)

[Lin10] Marvin Lindner. *Calibration and real-time processing of time-of-flight range data*. PhD thesis, 2010. (Cited on pages 2, 4, 9, 11, 12, and 17.)

[LNL+13] Damien Lefloch, Rahul Nair, Frank Lenzen, Henrik Schäfer, Lee Streeter, MichaelJ. Cree, Reinhard Koch, and Andreas Kolb. Technical foundation and calibration methods for time-of-flight cameras. In *Time-of-Flight and Depth Imaging. Sensors, Algorithms, and Applications*, pages 3–24. Springer Berlin Heidelberg, 2013. URL: http://dx.doi.org/10.1007/978-3-642-44964-2_1, doi:10.1007/978-3-642-44964-2_1. (Cited on page 17.)

[MNK] Stephan Meister, Rahul Nair, and Daniel Kondermann. Simulation of Time-of-Flight Sensors using Global Illumination. pages 33–40. URL: http://diglib.eg.org/EG/DL/PE/VMV/VMV13/033-040.pdf, doi:10.2312/PE.VMV.VMV13.033-040. (Cited on page 37.)

[Mon] Daniel Girardeau Montaut. Cloudcompare: 3d point cloud and mesh processing software. visited on 12.11.2013. URL: http://danielgm.net/cc/. (Cited on page 58.)

[Sch08] Martin Otmar Schmidt. Dissertation, IWR, Fakultät für Physik und Astronomie, Univ. Heidelberg, 2008. (Cited on pages 4, 5, 11, and 12.)

[Sch11a] Ingo Schiller. *Dynamic 3D scene analysis and modeling with a time-of-flight camera.* PhD thesis, University of Kiel, 2011. (Cited on pages 2, 3, and 17.)

[Sch11b] Mirko Schmidt. Dissertation, IWR, Fakultät für Physik und Astronomie, Univ. Heidelberg, 2011. (Cited on pages 1, 2, 3, 4, 11, 12, and 30.)

[Var] Gaël Varoquaux. Mathematical optimization: finding minima of functions. visited on 08.12.2013. URL: http://scipy-lectures.github.io/advanced/mathematical_optimization/. (Cited on pages 41 and 42.)

[Zha00] Zhengyou Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1330–1334, 2000. doi:10.1109/34.888718. (Cited on pages 18 and 58.)

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 angenom- men wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Siegen, den 6. Januar 2014

Jonathan Klein