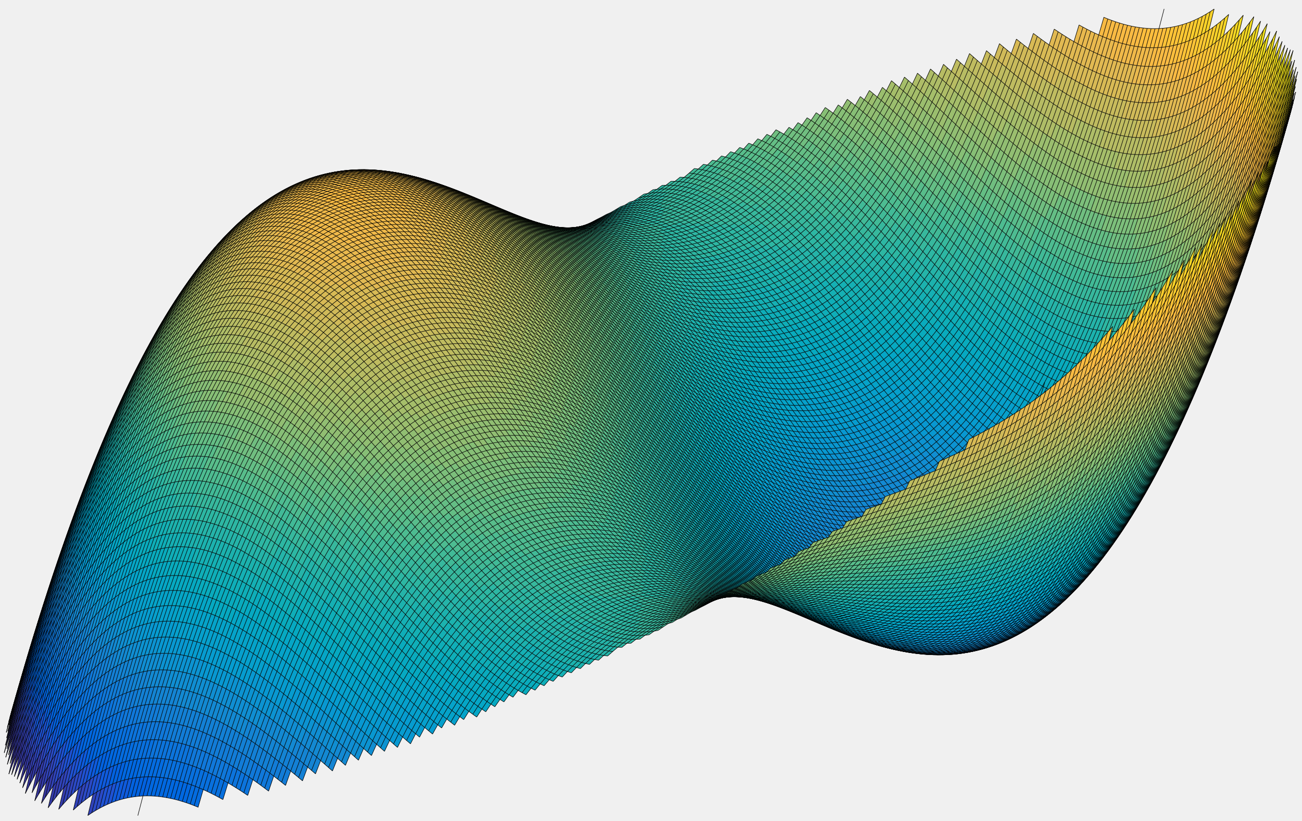
Comparison of different wavefront reconstruction methods

With Zernike polynomials

J. H. Salverda

Delft University of Technology



Comparison of different wavefront reconstruction methods

With Zernike polynomials

by

J. H. Salverda

in partial fulfillment of the requirements for the degree of

Bachelor of Science
in Applied Physics

at the Delft University of Technology,
to be defended publicly on Friday July 1, 2016 at 2:00 PM.

Student number:	4226046	
Project duration:	February, 2016 – June, 2016	
Supervisor:	Dr. S. F. Pereira	
Thesis committee:	Dr. S. F. Pereira,	TU Delft, Optics (AS)
	Dr. J. Kalkman,	TU Delft, Quantitative Imaging (AS)
	Ir. L. Cisotto,	TU Delft, Optics (AS)

TU Delft Delft University of Technology

Abstract

In this thesis, different wavefronts have been generated with a phase-only spatial light modulator and reconstructed with different methods. Both single optical aberrations and combinations of optical aberrations are addressed to the spatial light modulator. First, the wavefront is measured with a Shack-Hartmann sensor. This device measures the spot displacement of a lens array with respect to the local origin. From these, the local derivatives of the wavefront can be calculated. We compare the obtained Zernike coefficients of three wavefront sensor reconstruction methods. First, we use the recently developed analytical algorithm of Janssen [1]. Second, an iterative integration algorithm is used. Last, they are compared with the Zernike coefficients given by the software of the Shack-Hartmann sensor. The wavefronts will also be measured with a Michelson interferometer as an independent measurement. All four methods appeared to be well capable of measuring the wavefront. After normalization, the retrieved Zernike coefficients agree with the input of the aberrations addressed to the spatial light modulator. The coefficients given by the software of the Shack-Hartmann sensor have the highest RMS-error with respect to the input. The runtime of the algorithm of the analytical method is the shortest.

Preface

This research is done under the supervision of the Optics Research Group of the Imaging Physics Department of the Faculty of Applied Sciences at the Delft University of Technology. It is part of the bachelor's programme Applied Physics. This bachelor's thesis is represented by the course code TN2983, which corresponds to 12 ECTS. All the experiments were done in the Van Leeuwenhoek Laboratory. It is a collaboration between the Delft University of Technology and a national research organization TNO. This research is a continuation of the research done by Hassan Al Mahmoedi at the Optics Group during his internship.
I would like to thank:

- Silvania: you have been a great supervisor. You were so helpful with answering my questions and giving me good instructions, especially regarding the experimental part of this research. Thank you very much!
- Luca: thank you for your effort and patience while explaining me all the ins and outs in the Van Leeuwenhoek Laboratory. It was very nice to work with you.
- All the other members of the Optics Research Group: it was a great working environment. Even though I could not always attend the group meetings, they were an addition to my time at the group. Above all, the readings of the speakers were quite informative.

J. H. Salverda
Delft, June 2016

Contents

List of Figures

List of Tables

List of Abbreviations

For convenience, some abbreviations are used throughout the thesis. In table 1 below, these are listed in alphabetical order.

Table 1: List of abbreviations used in this thesis

Abbreviation	*Meaning*	*Short description*
AS	Faculty of Applied Sciences	This faculty offers the bachelor Applied Physics. Optics is a research group within this faculty.
Ast. 0°	Astigmatism 0°	This is an optical aberration, which corresponds to the fifth Zernike aberration according to the ISO-convention.
Ast. 45°	Astigmatism 45°	This is an optical aberration, which corresponds to the sixth Zernike aberration according to the ISO-convention.
CCD	Charged-coupled device	This chip converts electromagnetic radiation into electrical charge.
Interf.	Interferometer	This set-up can be used to reconstruct the phase of light.
ISO	International Organization for Standardization	This organization standardizes various norms and conventions.
LCoS	Liquid crystal on silicon	It is a kind of a microdisplay, which is able to modulate the phase of incoming light.
RMS	Root-mean-square	This means the square root of the mean of the squares. This particular type of mean is often used in error analysis.
SHS	Shack-Hartmann Sensor	This is a type of wavefront sensor.
SLM	Spatial Light Modulator	This is a device, which is able to modulate the phase of the incoming light beam.
WFS	Wavefront sensor	This is a device, which can measure a wavefront.

1 Introduction

Nowadays, we cannot imagine a world without lenses. They are used worldwide in many different fields. Think for example of the camera of your smartphone, the contact lenses or glasses you may wear and the telescope or other advanced optical set-ups that are used to discover new life in the universe. As you probably know, lenses could be either concave or convex. Theoretically, they have an ideal shape and size. However, due to manufacturing processes the actual shape and size could differ from the ideal design, which influences the refraction of light. This will result in deviations in the wavefront. In optics, these deviations are called optical aberrations. Due to the wide variety of usage of lenses it is quite important to quantify in detail what kind of aberrations a certain lens or optical system causes.

In this research, a so-called spatial light modulator will be used. This device could construct any wavefront, to a certain extent. A Shack-Hartmann wavefront sensor will be used to measure this wavefront. It consists of a lens array which focus the incoming light on a CCD. An aberrated wavefront does not focus in the ideal focal point. Hence, from the focus points, the displacements with respect to the ones corresponding to a perfectly flat wavefront could be determined. These spot displacements could be converted to derivatives of the local wavefront. Recently, Janssen found a new way to retrieve the wavefront from these derivatives [1]. He used an analytical approach to relate the derivatives of the wavefront with the so called Zernike polynomials [2]. These polynomials are a complete orthogonal set on the unit circle which therefore can describe any circular wavefront. The advantage of the use of Zernike polynomials, apart from their mathematical properties, is that they each have a physical meaning. We are going to compare Janssen's method with the coefficients given by the Shack-Hartmann sensor, which uses an iterative integration method to calculate the Zernike coefficients. This iterative integration method is also implemented in a Matlab script. These retrieved coefficients will as well be compared with the ones obtained by Janssen's method. Moreover, as an independent measurement, we compare the wavefronts obtained through these three methods with the wavefront retrieved from a Michelson interferometer.

In short, the goal of this research is to find out whether it is possible to measure a wavefront qualitatively and quantitatively with a wavefront sensor by using Janssen's recently found analytical algorithm. The results will be compared with the results obtained with the commercially used and conventional wavefront reconstruction methods.

For this research the algorithms of the students Ruud Bokdam, Leendert van Veen, Ike Mulder, Joost Wooning and Michel van der Kaay are used. Besides, some scripts of Sven van Haver, Hassan Al Mahmoedi and Luca Cisotto are used. Some of them have been modified by us and some scripts have been rewritten.

The thesis will be structured as follows. First, some theory necessary to understand the experiment will be provided. We will briefly introduce the theory behind Janssen's method, the iterative method, the Shack-Hartmann wavefront sensor, the Michelson interferometer and the spatial light modulator. Thereafter, two chapters are devoted to the experimental set-up and methods in order to let you understand how the experiment is done and why we made some choices regarding the analysis of the data. After that, the results will be presented for the various measurements and some discussion on these will be given. Finally, conclusions are made and recommendations for future research are given.

2
Theory

In this chapter of the thesis, some theory needed to understand the research will be presented. Light can be considered both as a particle and as a wave. A wave has an amplitude, a phase and a direction in which it propagates. In this research the wavefront will be measured and thus we are interested in the wave-like characteristics of the light.

2.1. Zernike theory

A wavefront is a surface of points in $\mathbb{R}^3$ corresponding to waves with the same phase. They can be represented by a wavefront function $W(x, y)$.

Frits Zernike, winner of the Nobel Prize in 1953 [3], published in 1934 an article about the so-called Zernike polynomials [2]. They form an orthogonal set of functions on the unit circle [4] (i.e. $\{(x, y) \in \mathbb{R}^2 : x^2 + y^2 \leq 1\}$), which means that the integral of the product of two different Zernike polynomials over the unit circle equals zero. Because these functions form a complete set, every surface on the unit circle ($W : \{(x, y) \in \mathbb{R}^2 : x^2 + y^2 \leq 1\} \to \mathbb{R}$) can be written as an infinite linear combination of Zernike polynomials[1], just like every function $f : \mathbb{R} \to \mathbb{R}$ can be written as a linear combination of sines and cosines[1], which is known as Fourier series representation. A Zernike polynomial consists of a radial and an angular component. When n and $n - |m|$ are both any non-negative integer including 0, then the following holds:

$$Z_n^m(\rho, \theta) = \begin{cases} R_n^{|m|}(\rho)\cos(m\theta) & \text{if } m \geq 0 \\ R_n^{|m|}(\rho)\sin(|m|\,\theta) & \text{if } m < 0 \end{cases} \tag{2.1}$$

$R_n^{|m|}(\rho)$ is defined as:

$$R_n^{|m|}(\rho) = \sum_{k=0}^{\frac{n-|m|}{2}} \frac{(-1)^k (n-k)!}{k!\,(\frac{n+m}{2} - k)!\,(\frac{n-m}{2} - k)!}\rho^{n-2k} \tag{2.2}$$

Since light beams often pass circular apertures, wavefronts are therefore often functions on a disk. These could be scaled to the unit circle and hence they could be written as a Zernike polynomial expansion. Let $Z_n^m(x, y)$ be the same function as (2.1), but with substitutions $\rho = \sqrt{x^2 + y^2}$ and $\theta = \arctan\frac{y}{x}$. We should use the proper definition of the $\arctan$ such that θ will correspond to the azimuthal angle [5]. This means that the range will be $[-\pi, \pi]$. So, if we scale our wavefront to the unit circle, the wavefront $W(x, y)$ could be written in the following form, with α_n^m yet undetermined coefficients:

$$W(x, y) = \sum_{n,m} \alpha_n^m Z_n^m(x, y) \tag{2.3}$$

[1]This is valid if certain convergence criteria are satisfied. The surface $W(x, y)$ should at least be piecewise continuous.

Table 2.1: The first eleven Zernike polynomials according to the ISO-convention (from [6])

n	m	Z_n^m	*Name*
0	0	1	Piston
1	1	$\rho \cos \theta$	Tilt X
1	-1	$\rho \sin \theta$	Tilt Y
2	0	$2\rho^2 - 1$	Defocus
2	2	$\rho^2 \cos 2\theta$	Astigmatism $0°$
2	-2	$\rho^2 \sin 2\theta$	Astigmatism $45°$
3	1	$(3\rho^2 - 2\rho) \cos \theta$	Coma X
3	-1	$(3\rho^2 - 2\rho) \sin \theta$	Coma Y
4	0	$6\rho^4 - 6\rho^2 + 1$	Spherical aberration
3	3	$\rho^3 \cos \theta$	Trefoil $0°$
3	-3	$\rho^3 \sin \theta$	Trefoil $30°$

A list of the first eleven Zernike polynomials is given in table 2.1. Note that there are many different conventions on how to order the Zernike polynomials. Because of consistency, this table follows the so called ISO-convention, since this is the convention used by the software of the wavefront sensor.

2.2. Wavefront sensor

The wavefront sensor used in this research is a Shack-Hartmann sensor. It consists of an array of lenses and a CCD camera. The focal plane of these lenses coincides with the CCD. The principle behind this kind of wavefront sensor is that they measure the local, i.e. at the location of such a lens, slope of the wavefront. From basic geometrical optics, it is known that a plane wave perpendicular to the optical axis, after refraction due to a lens, will converge to the point where the focal plane and the optical axis intersect. This can be seen in the left part of figure 2.1. In the right part, it becomes clear that a local slope will cause this point to move away from the optical axis. This deviation from the optical axis will be called from now on a spot displacement. This spot displacement has x- and y-components.

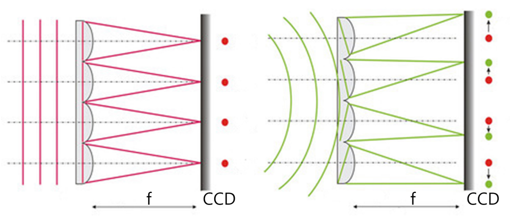


Figure 2.1: Left: a plane wave hits the wavefront sensor. Right: a spherical wave hits the wavefront sensor.

In the right side of figure 2.1, it seems that the wavefront at each lens is locally a plane wave. Actually, even if we are considering a simplified model, it is locally a smooth curved wavefront instead of a plane wave. This means that there won't be just one single point where the light focuses. The way the wavefront sensor determines the spot displacement is actually by calculating the 'center of mass'. Since there are many more pixels on the CCD than there are lenses, the wavefront sensor could identify the precise spot displacement. It is possible to calculate the local average wavefront from these determined spot displacements.

Using the Eikonal equations of optics, Pythagoras' theorem and some basic mathe-

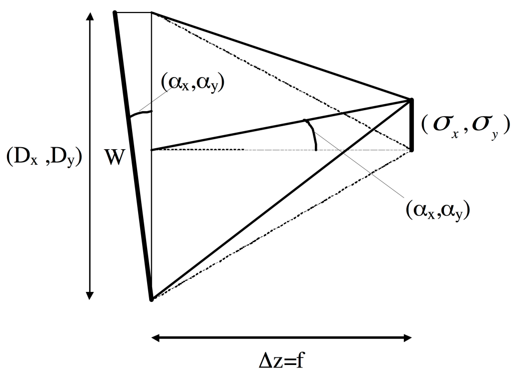


Figure 2.2: Dependency of the spot displacements on the local slope of the wavefront (from [7])

matics, the following equation could be obtained [7].

$$\frac{\partial W}{\partial x} = \frac{\sigma_x}{\sqrt{f^2 + {\sigma_x}^2 + {\sigma_y}^2}} \tag{2.4}$$

$$\frac{\partial W}{\partial y} = \frac{\sigma_y}{\sqrt{f^2 + {\sigma_x}^2 + {\sigma_y}^2}} \tag{2.5}$$

For the interested reader, the precise derivation can be found in the Technical Manual of the SHS [7]. When the spot displacements σ_x and σ_x are very small in comparison with the focal length f, then is it valid to approximate (2.4) and (2.5) with the following expressions:

$$\frac{\partial W}{\partial x} = \frac{\sigma_x}{f} \tag{2.6}$$

$$\frac{\partial W}{\partial y} = \frac{\sigma_y}{f} \tag{2.7}$$

The following two paragraphs will explain two different algorithms, which will be used to reconstruct the wavefront from the partial derivatives with respect to x and y.

2.2.1. Janssen's analytical method

A detailed description of the whole method can be found [1], but we will emphasize what is relevant for our purposes.

It turns out that is convenient, in accordance with formulas for derivatives presented in [8], to combine the partial derivatives with respect to x and y as follows: $\frac{\partial W}{\partial x} + \mathrm{i}\frac{\partial W}{\partial y}$ and $\frac{\partial W}{\partial x} - \mathrm{i}\frac{\partial W}{\partial y}$. These expressions can be considered to be known, since the partial derivatives with respect to x and y can be obtained from the measurements done with the SHS. Both expression can be written as a Zernike expansion:

$$\frac{\partial W}{\partial x} + \mathrm{i}\frac{\partial W}{\partial y} = \sum_{n,m} (\beta_+)_n^m Z_n^m(x,y) \tag{2.8}$$

$$\frac{\partial W}{\partial x} - \mathrm{i}\frac{\partial W}{\partial y} = \sum_{n,m} (\beta_-)_n^m Z_n^m(x,y) \tag{2.9}$$

These Zernike coefficients, $(\beta_+)_n^m$ and $(\beta_-)_n^m$, can be calculated by a least squares Zernike polynomial fit of the left-hand sides of (2.8) and (2.9). According to Janssen, the following estimator for the coefficients α_n^m of (2.3) is valid, using the coefficients $(\beta_\pm)_n^m$:

$$\hat{\alpha}_n^m = \frac{1}{n\epsilon_{\frac{n-|m|}{2}}}\left(\frac{1}{2}(\beta_+)_{n-1}^{m+1} + \frac{1}{2}(\beta_-)_{n-1}^{m-1}\right) - \frac{1}{(n+2)\epsilon_{\frac{n-|m|}{2}+1}}\left(\frac{1}{2}(\beta_+)_{n+1}^{m+1} + \frac{1}{2}(\beta_-)_{n+1}^{m-1}\right) \tag{2.10}$$

where

$$n = \begin{cases} 2, 4, \ldots & \text{if } |m| = 0 \\ |m|, |m|+2, \ldots & \text{if } |m| \neq 0 \end{cases} \tag{2.11}$$

and

$$\epsilon_k = \begin{cases} 1 & \text{if } k = 0 \\ 2 & \text{if } k > 0 \end{cases} \tag{2.12}$$

2.2.2. Iterative method

There is also a second reconstruction method we will use. A more detailed description of the derivation of this method can be found in the Technical Manual of the wavefront sensor [7].

By using the definition of a derivative, we can estimate the wavefront at the position $(x+h, y)$. If h is small enough, the following expression holds:

$$W(x+h,y) \approx W(x,y) + h\frac{\partial W}{\partial x}(x,y) \tag{2.13}$$

From measurements, the derivatives at all points are known. Therefore, a better estimation could be made:

$$W(x+h,y) \approx W(x,y) + \frac{h}{2}\left(\frac{\partial W}{\partial x}(x,y) + \frac{\partial W}{\partial x}(x+h,y)\right) =: W(x,y) + \Delta_x(x,y) \tag{2.14}$$

$$W(x-h,y) \approx W(x,y) - \frac{h}{2}\left(\frac{\partial W}{\partial x}(x-h,y) + \frac{\partial W}{\partial x}(x,y)\right) =: W(x,y) - \Delta_x(x-h,y) \tag{2.15}$$

This leads to:

$$W(x,y) \approx W(x+h,y) - \Delta_x(x,y) \tag{2.16}$$

$$W(x,y) \approx W(x-h,y) + \Delta_x(x-h,y) \tag{2.17}$$

The same reasoning can also be applied to the y-direction. By doing so, we have n expressions for $W(x,y)$, where n represents the number of adjacent sampling points. Normally, $n = 4$. See figure 2.3. Hence, we could take an average of the four expressions of $W(x,y)$. This will lead to the following estimation of $W(x,y)$, depending on the values of the wavefront of adjacent points and the values of the partial derivatives with respect to x and y at the sampling points:

$$W(x,y) \approx \frac{1}{n}\big(W(x+h,y) + W(x-h,y) + W(x,y+h) + W(x,y-h) \\ -\Delta_x(x,y) + \Delta_x(x-h,y) - \Delta_y(x,y) + \Delta_y(x,y-h)\big) \tag{2.18}$$

When the relevant point is on the boundary of the domain, this point won't have four adjacent sampling points. For example, when there is no sampling point at left, the calculation of $W(x,y)$ by (2.18) will be done without these missing values of the wavefront and the partial derivatives (i.e. $W(x-h,y)$ and $\Delta_x(x-h,y)$). Needless to say, n equals three in that case.

In (2.18) we see that the wavefront at (x,y) depends on the wavefront at adjacent points. Since the wavefront is unknown, we use an iterative method, with iteration number $i \in \{1, 2, \ldots, I\}$, to obtain $W(x,y)$. Here I represents the number of the last iteration.

$$W^{(i+1)}(x,y) \approx \frac{1}{n}\big(W^{(i)}(x+h,y) + W^{(i)}(x-h,y) + W^{(i)}(x,y+h) + W^{(i)}(x,y-h) \\ -\Delta_x(x,y) + \Delta_x(x-h,y) - \Delta_y(x,y) + \Delta_y(x,y-h)\big) \tag{2.19}$$

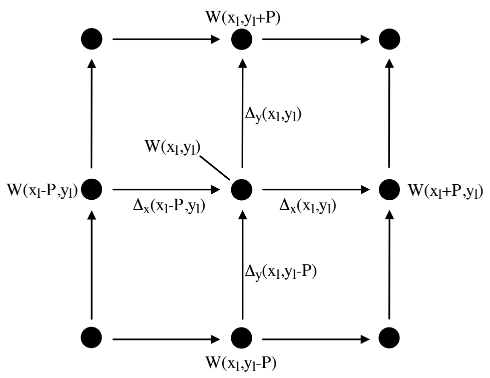


Figure 2.3: Relation between adjacent sampling points (from [7])

In order to perform the first iteration, $i = 1$, we set all values of $W^{(1)}(x, y)$ equal to zero: $\forall(x, y) \in \mathbb{R}^2 : W^{(1)}(x, y) = 0$.

2.3. Interferometry

When you drop two stones in a lake, waves will arise in the water from the points where the stones hit the water, going outward. Those two waves will interfere with each other. Constructive interference will occur when both waves have a maximum at the same time and at the same place. The phase difference between those two waves will be an even multiple of π. Destructive interference will occur when one wave has a maximum and the other a minimum at the same time and at the same place. In that case the phase difference is an odd multiple of π. This concept holds as well for light waves.

In this thesis a Michelson interferometer (figure 2.4) will be used to measure the phase of the wave and thus determine the wavefront using interferometry. It is based upon the principle of interference: differences in optical path length will result in constructive and destructive interference.

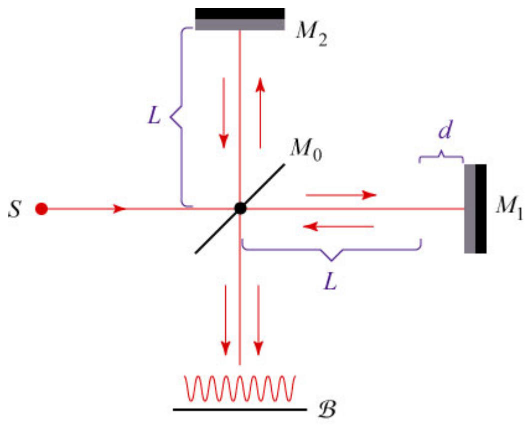


Figure 2.4: Michelson Interferometer (from [9])

The light source S emits a coherent light beam. A coherent light beam consists of waves of the same frequency (monochromatic) which propagate synchronized through the medium. This means that there is no phase difference between them. A laser is a well-known example of a light source which can emit a coherent light beam. M_0 is a beam splitter: the light from S is partially reflected in the direction of M_2 and partially transmitted in the direction of M_1. M_1 and M_2 are mirrors, which reflect the beams propagating in their direction. The reflected light from M_1 and M_2 propagates back in the direction of the beam splitter and will there be respectively reflected and transmitted to the screen or the detector. An interference pattern of the two light beams is formed at the detector. By moving M_1, the optical path length of the ray coming from M_1 changes. This will result in a different phase difference between the two rays hitting the screen or the detector, which will cause a different interference pattern. From all these different interference patterns corresponding to different displacements of M_1, a wavefront can be reconstructed.

2.4. Spatial light modulator

To construct a certain wavefront, we use a spatial light modulator. In this section the principle upon which a spatial light modulator is based will be explained briefly.

The SLM is being illuminated by a coherent collimated light beam. It will reflect this beam with a modified phase profile, which is caused by the image that is addressed to the SLM.

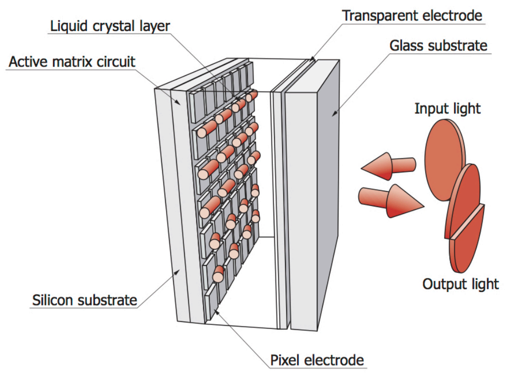


Figure 2.5: Schematic representation of a spatial light modulator (from [10])

The spatial light modulator used in this research is a LCoS-SLM. It means that a liquid crystal layer is placed on a silicon substrate. On top of it there is a layer containing many pixels made by aluminium. Each of these pixels can be controlled separately. Adjusting the voltage will cause the electric field to change. This change of electric field will result in a different tilt angle of the molecules of the liquid crystal layer. The tilt of the molecules effects the refractive index of the liquid crystal layer. Light going through the liquid crystal layer containing molecules with different alignments will experience different refractive indices and will thus experience an optical path difference. This optical path difference results in a phase difference after reflection at the silicon substrate and subsequently leaving the SLM. The polarization of light must be in the same direction as the alignment of the molecules. A schematic representation is shown in figure 2.5.

3 Experimental set-up

In this chapter of the thesis the experimental set-up will be explained in detail. In this research we used two set-ups, one for measuring the spot displacements with the wavefront sensor and the other for the interferometric measurements. These two set-ups don't differ a lot from each other. They, including all different used optical elements, will be discussed in this chapter. Besides, the choices we made with respect to the set-up will be justified. Moreover, a picture and a sketch of the set-up will be presented.

3.1. Measuring using a Shack-Hartmann wavefront sensor

Figure 3.1 shows a picture of the set-up where a wavefront is being measured by a wavefront sensor. From 1, light from a laser travelling through an optical fiber is colli-

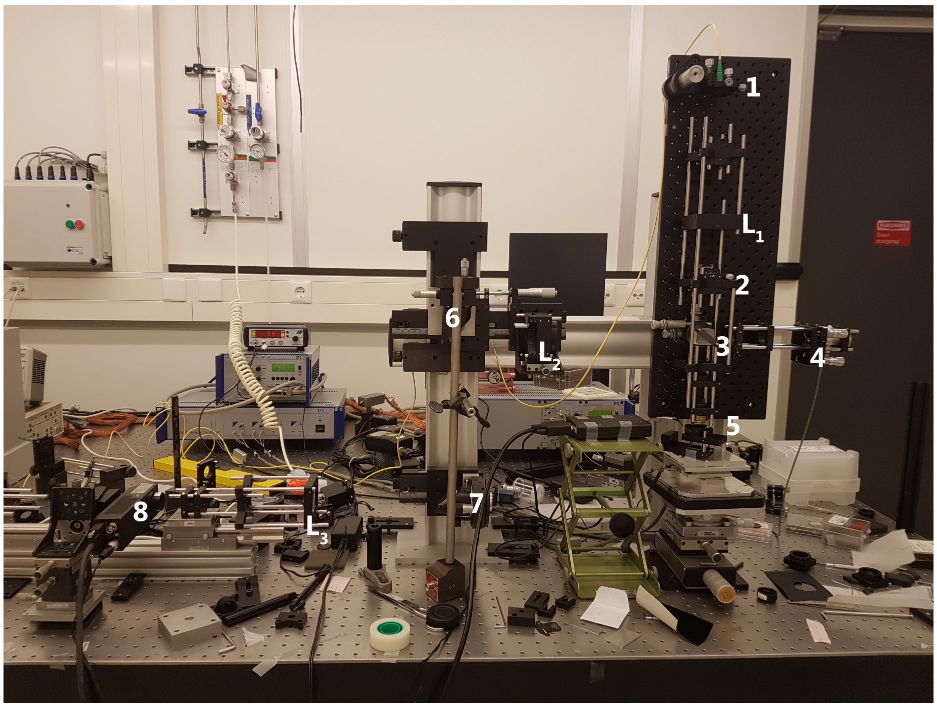


Figure 3.1: Experimental set-up with the spatial light modulator and the wavefront sensor

mated ($f_{L_1} = 20\,\mathrm{cm}$) and sent downward. It is a coherent light beam with $\lambda = 633\,\mathrm{nm}$ (HeNe laser). After the collimator, an approximately flat wavefront is formed. Next, there is a linear polarizer (2). As explained in paragraph 2.4, the polarization axis should be in the same direction as the alignment of the molecules of the liquid crystal of the SLM, which is 5 in the picture. After being polarized, the beam gets split by the beam splitter at 3. One beam goes to the SLM. The other beam, which goes to the right in figure 3.1, gets blocked because no reference beam is needed in this part of the experiment. The reference mirror will be used for the interferometric measurements. A certain wavefront is formed after reflection at the SLM, which is 5 in the figure. At the beam splitter the beam gets split again. One part of it will propagate to the left direction and passes a telescopic system with lenses L_2 and L_3 ($f_2 = 40\,\mathrm{cm}$ and $f_3 = 25\,\mathrm{cm}$). Due to practical reasons, there are two mirrors placed at 6 and 7. Finally, the wavefront of the beam will be measured with the wavefront sensor at 8.

The distances between the lenses, the SLM, and the wavefront sensor are important. In order to be able to measure the wavefront created by the SLM as well as possible, the wavefront at 5 needs to be (almost) the same as the wavefront at 8. From Fourier optics it is known that a lens performs a Fourier transform of the object plane one focal length behind the lens. In order to avoid aberrations due to propagation, we use a 4f-lens system, which is shown schematically in figure 3.2. The two mirrors at 6 and 7 are omitted in 3.2.

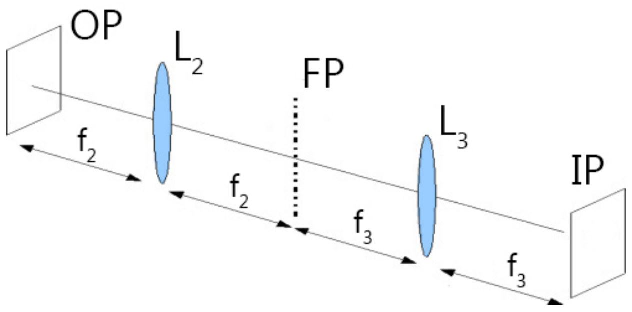


Figure 3.2: Schematic sketch of a 4f-lens system with an object plane (OP), a Fourier plane (FP), an image plane (IP) and two lenses (L_2 and L_3) with corresponding focal lengths f_2 and f_3 (from [11], modified)

Thus, the SLM is positioned in the object plane, one focal length away from L_2. The wavefront sensor is positioned in the image plane, one focal length away from L_3. The distance between the two lenses equals the two focal lengths summed. Between the two lenses, the Fourier transform is performed by L_2. The second lens reconstructs the wavefront from the Fourier transform.

When positioning the SLM, it is important to adjust the tilt in such a way that the impinging beam is perpendicular to the SLM. In this way we try to avoid extra aberrations due to misalignment.

3.2. Measuring using an interferometer

When the wavefront is being measured using interferometry, almost exactly the same set-up is used as just discussed in paragraph 3.1. However, there are two differences. First, the reference beam, at the right hand side of the beam splitter, doesn't get blocked any more, because it is needed for interferometry. At 4 a flat mirror is positioned, which can be moved by a piezoelectric translation stage. Second, at 8 a CCD camera is placed instead of a wavefrontsensor. In order to obtain the phase of the wavefront we record the intensity of the interference signal at the CCD camera for various positions of the piezoelectric translation stage within one wavelength. Afterwards, a five-step phase shifting algorithm is applied [12].

4 Experimental methods

In this chapter there will be a detailed explanation of the experimental methods and the way the data and their corresponding results have been analyzed. In paragraph 4.1, some description is given on how the input images of the spatial light modulator are generated. Thereafter, a short paragraph is devoted to how the orientation of the set-up should be verified. In paragraphs 4.3 and 4.4, some parts of the analysis, which apply to both the wavefront sensor data and the interferometric data, are explained. Finally, in paragraphs 4.5 and 4.6 the method specific analysis of the wavefront sensor and respectively the interferometer is described.

4.1. Addressing an aberration to the spatial light modulator

The liquid crystal display of the SLM has a rectangular shape. The resolution is 1920x1080 pixels. Each pixel is a square with sides of $8\ \mu\mathrm{m}$. The SLM is an 8-bit device, so it could maximally address 256 different grey levels, which correspond to 256 different phase levels. In this experiment, a so-called 5-5 configuration is used. This means that there are only 192 different grey values, but the liquid crystal is more stable and therefore there is less flickering in the measurements. The input of the spatial light modulator are BMP-images.

A brief description of the steps in the Matlab script follows. First, a certain mesh grid is made with x from -960 to 959 and y from -540 to 539, both with steps equal to one. These Cartesian coordinates are being transformed to polar coordinates:

$$\rho = \sqrt{x^2 + y^2} \tag{4.1}$$
$$\theta = \operatorname{atan2}(y, x) \tag{4.2}$$

In (4.2), $\operatorname{atan2}$ is a modified version of the $\arctan$ function. The angle will be 0 at the positive x-axis. In the first and second quadrant θ changes counterclockwise from 0 to π. In the third and fourth quadrant, θ goes clockwise from 0 to $-\pi$. A pupil with a certain radius will be cut out and the ρ-matrix will be normalized such that $\rho = 1$ holds at the edge of the pupil. This should be done, because the Zernike polynomials are a complete set of orthogonal functions on the unit circle only, as stated before.

For example, when a wavefront consisting of Zernike aberrations coma X and coma Y should be addressed, we add these particular aberrations and normalize them with the Euclidean norm, as done below.

$$W = \frac{1}{\sqrt{2}}\left((3\rho^3 - 2\rho)\cos\theta + (3\rho^3 - 2\rho)\sin\theta\right) \tag{4.3}$$

In figure 4.1 there is an example of an image containing an aberration consisting of coma X and coma Y which could be used as input for the SLM.

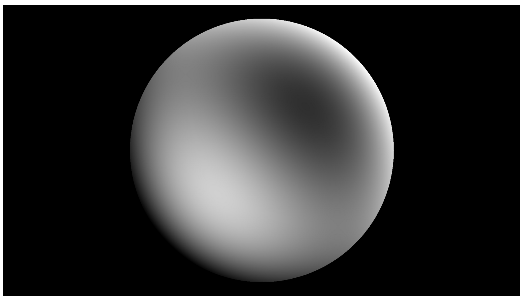


Figure 4.1: Example of an image of a combination of the aberrations coma X and coma Y

4.2. Orientation

Because lenses have the property to invert images, it is quite important to check the orientation of the image addressed to the SLM, the SLM itself, the wavefront sensor and the camera. This could be done by addressing an asymmetric image to the SLM and notice how it is displayed upon the wavefront sensor and the camera. To check the orientation of the camera, the character L is used for this purpose, see figure 4.2. For the wavefront sensor, we used a modified version of 4.2. In fact, we cannot allow this image to have jump discontinuities at its boundaries, because the wavefront sensor is not able to measure them, since the device is only able to measure spot displacements, which are approximately proportional to the partial derivatives. As a jump discontinuity corresponds to an infinitely large partial derivative, then it should be evident that the SLM will not see these discontinuities. Therefore, a smooth edged character L is addressed to the SLM to check the orientation of the wavefront sensor.

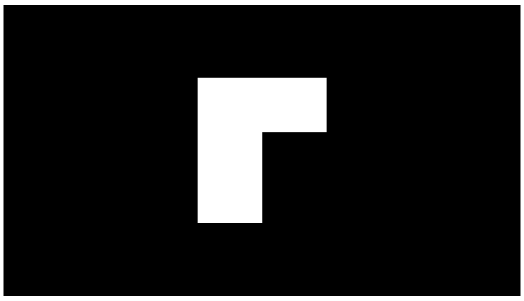


Figure 4.2: This image was addressed to the SLM to check the orientation of the camera. In order to check the orientation of the wavefront sensor, a smooth edged L is used.

If we don't carefully check the orientation of all the components we are dealing with, it is possible that the wrong aberration is being measured. For example, if there is a rotation of $90°$ between the image addressed to the SLM and the wavefront sensor, then coma Y will be measured when coma X is addressed to the SLM and vice versa.

4.3. Omitting piston, tilt X and tilt Y

As stated earlier, the wavefront is measured with a wavefront sensor and with a camera using interferometry. It is very difficult to place the camera or the wavefront sensor perfectly perpendicular to the light beam. So the camera or wavefront sensor will always be under a little angle with respect to the horizontal or vertical direction. Moreover, it is practically impossible to place the camera and the wavefront sensor in exactly the same position. There will always be a little angle between them. Therefore, it is convenient to leave out the first three Zernike aberrations, i.e. piston, tilt X (horizontal direction) and tilt Y (vertical direction), in our analysis. They are irrelevant for the purpose of this research and will therefore not be taken into consideration.

4.4. Normalization

The manual of the wavefront sensor does not clearly state what the units of the spot displacements and the Zernike coefficients are. In the next chapter, it will be clear that, indeed, the order of magnitude of the coefficients is different for the various wavefront reconstruction methods. Nevertheless, in order to be able to compare the results of the different methods with each other, the obtained Zernike coefficients will be normalized with the corresponding Euclidean norm, which is defined as follows.

$$||\alpha|| = \sqrt{\sum_{n,m} (\alpha_n^m)^2} \tag{4.4}$$

Here, α_n^m are the coefficients corresponding to the aberration Z_n^m. See again table 2.1. The summation is over all evaluated aberrations. This amount must be the same for both methods. Note that we don't consider the Zernike aberrations piston, tilt X and tilt Y, as just explained in paragraph 4.3.

4.5. Wavefront sensor

4.5.1. Centering the SLM

When we want to measure an aberration addressed to the spatial light modulator using the wavefront sensor, the alignment of the set-up is quite important. The wavefront sensor is connected to a computer and is controlled by the software program SHSWorks. Within this application there is a real time display of the camera, as can be seen in figure 4.3.

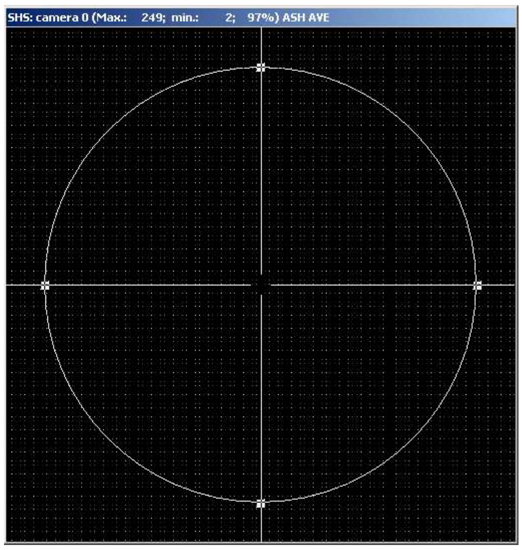


Figure 4.3: Window of the real-time spot displacements of the wavefront sensor

The software program SHSWorks does a Zernike polynomial fit of the data inside the circle with the white border. Moreover, when we are only interested in the spot displacement, we can export the measured spot displacements inside the circle. The radius and the center of this circle could be adjusted. It is very important that this circle perfectly coincide with the circular aperture made in the input file of the SLM (e.g. figure 4.1). After all, the area of the data which will be analysed and will be fit with Zernike polynomials, should coincide with exactly that area in which the Zernike aberrations are made. Only then, it is justifiable to compare the outcome with the input. In order to center the SLM, it is helpful to address a cross to the SLM. This cross cannot have a jump discontinuity at its boundaries, as explained earlier. Therefore a cross with a cosine-like intensity profile is made, see figure 4.4.

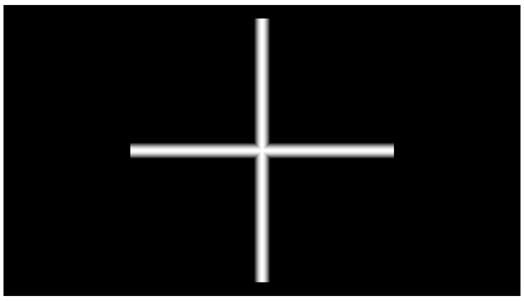


Figure 4.4: Image addressed to the SLM, in order to be able to recognize the center of the SLM in the display of the wavefront sensor

4.5.2. Reference measurement

Aberrations in the wavefront which will be measured at the location of the wavefront sensor are not only caused by the image addressed to the SLM. After all, it is likely that the optical elements used in the experiment, like the lenses and the mirrors, could also cause some aberrations. Moreover, it is also possible that the SLM itself, without any aberration addressed, is responsible for aberrations in the measured wavefront. However, we are interested whether we could measure the aberrations, which are actually addressed to the SLM and which are not caused by any undesirable properties of the used optical elements or the SLM itself. To achieve this, a reference measurement was done by addressing a black screen to the SLM. The wavefront of this measurement, i.e. the Zernike coefficients, should be subtracted from the wavefront of the measurement with the desired aberration addressed to the SLM. In this way, they will be corrected for unwanted irrelevant aberrations. Note that it is important that the normalization with the Euclidean norm will be done after the correction with the reference measurement.

4.5.3. Zernike coefficients given by the wavefront sensor

The software of the Shack-Hartmann wavefront sensor itself also executes an iterative integration algorithm to calculate the Zernike coefficients from the spot displacements. These Zernike coefficients will also be subtracted by the coefficients corresponding to the reference measurement with a black screen addressed to the SLM, calculated and given by the software of the Shack-Hartmann sensor. After this correction, they will be normalized as well.

4.6. Interferometer

4.6.1. Alignment

The pupil of the image addressed to the SLM should be entirely visible on the display of the CCD camera. This could be done by addressing a cross (figure 4.5) to the SLM (not necessarily a smooth edged one). In this case it is even better to address a thin and sharp edged cross, in order to be able to see the cross. For this purpose it is more convenient to block the reference beam.

When the cross is centered and after verifying that it is not shown at an angle on the screen, this image should be saved. We will need it later on to determine the center of the cut out pupil, i.e. the center of the cross. A certain aberration is addressed to the SLM and for interferometric measurements the reference beam needs to be unblocked. An interference pattern with bright and dark fringes could be seen at the screen. In theory, a very bright spot, where maximum constructive interference occurs, should get less bright when the mirror moves in any direction. However, it is possible that the display of the CCD is saturated. Therefore the saturation should be checked. If this is the case, then the exposure time needs to be decreased until the image is not saturated anymore.

The tilt of the reference mirror should be adjusted in order to try to center the fringes around the center of the camera. Moreover, as regards the algorithm used later on, it is necessary that there are not too many fringes formed on the display for more accurate

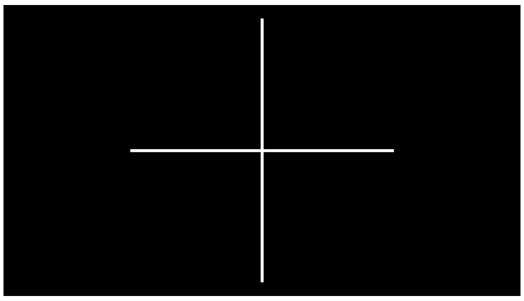


Figure 4.5: Image addressed to the SLM, in order to be able to recognize the center of the SLM in the display of the camera for the interferometric measurements

phase retrieval.

4.6.2. Measuring the aberration

As stated earlier in this thesis, the voltage applied to the piezoelectric translation stage, which drives the reference mirror, is approximately proportional to the displacement of the mirror. For this experiment it is sufficient to take images with the camera each $0.02\,\mathrm{V}$ starting at $45.50\,\mathrm{V}$ until $49.50\,\mathrm{V}$. This range corresponds to a displacement of approximately a few wavelengths. The time between each measurement should be around $800\,\mathrm{ms}$.

4.6.3. Reference measurement

As stated earlier in this chapter, aberrations measured by the interferometer are not only caused by the image addressed to the SLM. The optical elements, the SLM itself and last but not least, the reference arm including its mirror could cause aberrations as well. These can be corrected by taking a reference measurement with a black image addressed to the SLM. The range of the voltage applied to the piezoelectric translation stage remains the same as with the measurement of the aberrations. The Zernike coefficients obtained by the analysis of the reference measurement should be subtracted from the measurement of the aberration itself. In order to be able to compare these results with the results of the measurements with the wavefront sensor, the Zernike coefficients will be normalized with the Euclidean norm after the correction with the reference measurement. See paragraph 4.4.

4.6.4. Algorithm

The analysis of the interferometric data is done with Matlab and a C++ compiler of Microsoft Visual Studio. In this paragraph the steps taken in the algorithm will be explained.

Cut out pupil

The pupil, which can be seen for example in figure 4.1, needs to be cut out as well in the images taken by the camera. The center of the pupil can be determined with the image of the cross, which was taken earlier. The radius of this pupil can be calculated using the ratio between the focal lengths of the two lenses of the interferometer. Looking at figure 3.1, the magnification of the optical system is:

$$M = \frac{f_{L_3}}{f_{L_4}} \tag{4.5}$$

To calculate the radius of the pupil in pixels of the CCD of the camera (R_{CCD}), we need the pixel size of the SLM (d_{SLM}), the pixel size of the CCD (d_{CCD}) and the radius of the pupil in pixels of the SLM (R_{SLM}), which is known.

$$R_{\mathrm{CCD}} = \frac{d_{\mathrm{SLM}} \cdot R_{\mathrm{SLM}} \cdot M}{d_{\mathrm{CCD}}} \tag{4.6}$$

An example of an image taken by the CCD camera, where the pupil has been cut, can be seen in figure 4.6.

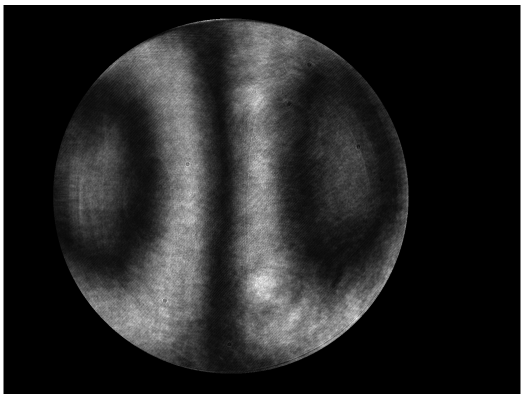


Figure 4.6: Image shot by the camera on which an interference pattern can be seen, when coma X was addressed to the SLM

Correlation

In order to retrieve the wavefront, the correlation between the first image I_1, corresponding to a voltage of $45.50\ \mathrm{V}$ applied to the piezoelectric translation stage, and the other images I_j should be calculated. The following correlation coefficient could be calculated with Matlab:

$$r_j = \frac{\sum_m \sum_n \left(I_j(m,n) - \bar{I}_j\right)\left(I_1(m,n) - \bar{I}_1\right)}{\sqrt{\sum_m \sum_n \left(I_j(m,n) - \bar{I}_j\right)^2 \left(I_1(m,n) - \bar{I}_1\right)^2}} \tag{4.7}$$

$I_j(m,n)$ is the intensity matrix of the j-th image with pixel positions (m,n). The mean of I_j is defined as $\bar{I}_j$. In figure 4.7 the correlation coefficient, obtained by the interferometric data, is plotted against the voltage applied to the piezoelectric translation stage.

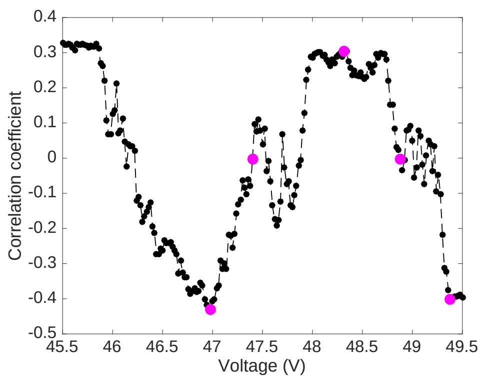


Figure 4.7: Correlation coefficient against the voltage

Phase reconstruction

For the phase reconstruction, only five images are necessary. First, a whole period of the cosine-shaped correlation graph needs to be analyzed. It is convenient to choose a period starting at a maximum or a minimum. As an example, consider now the part of the graph of figure 4.7 which starts with a minimum. We denote the five images with I_{j_1}, I_{j_2}, I_{j_3}, I_{j_4} and I_{j_5}. They correspond respectively to the locations within the period with relative phase $\Phi_r \in \{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}, 2\pi\}$, which means respectively the first minimum, the first intersection with the x-axis, the maximum, the second intersection with the x-axis and the second minimum. These locations are highlighted in figure 4.7. The two minima, $\Phi_r = 0$ and $\Phi_r = 2\pi$, and the maximum $\Phi_r = \pi$ are determined by choosing the lowest or respectively the highest correlation within the specific part of the period of the cosine-shaped graph. Thereafter, the first intersection with the x-axis, $\Phi_r = \frac{\pi}{2}$, is determined by choosing from the data points between $\Phi_r = 0$ and $\Phi_r = \pi$ that point which corresponds to a correlation closest to zero. The second intersection with the x-axis, $\Phi_r = \frac{3\pi}{2}$, is determined in the same way: the data point between $\Phi_r = \pi$ and $\Phi_r = 2\pi$, which corresponds to a correlation closest to zero.

The phase ϕ can be reconstructed with the following five-step algorithm [13]:

$$\tan\phi = \frac{2(I_2 - I_4)}{I_1 - 2I_3 + I_5} \tag{4.8}$$

The result of this algorithm is a phase profile, as can be seen in figure 4.8. Due to the range of the atan2 function, there are jump discontinuities in the reconstructed phase profile. This can be seen in the transition from blue to yellow and vice versa.

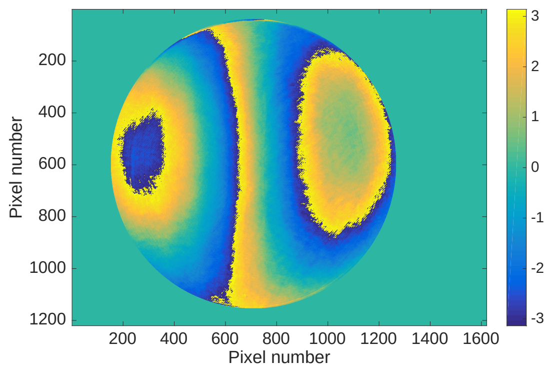


Figure 4.8: Reconstructed phase profile in radians, where the aberration addressed to the SLM was coma X

Filtering and unwrapping the phase profile

In order to get rid of the jump discontinuities in the phase, the phase profile needs to be unwrapped, as described in reference [14]. Before that, the data gets filtered, which eases the unwrapping. In short, this filtering does a Fourier transform and filters out the high frequencies. In figure 4.9 and 4.10 the filtered phase profile and respectively the unwrapped phase profile are shown. Eventually, a least squares Zernike polynomial fit is done.

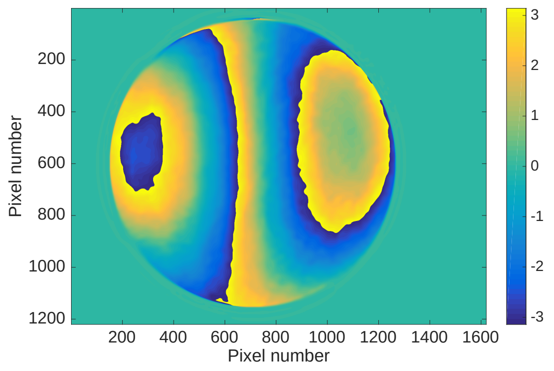


Figure 4.9: Filtered phase profile in radians, where the aeration addressed to the SLM was coma X

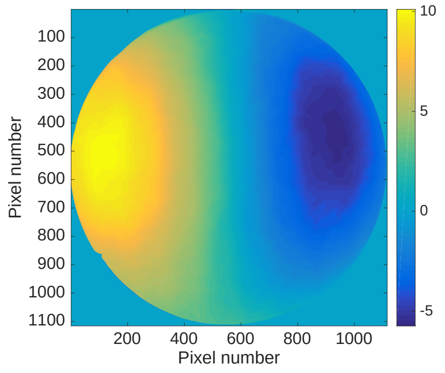


Figure 4.10: Unwrapped phase profile in radians, where the aberration addressed to the SLM was coma X

5 Results

In this chapter the results of the experiments will be presented. First, some examples will be shown to illustrate how the Zernike coefficients are being corrected and being normalized. After that, the runtime of the algorithms will be considered.

In paragraph 5.3, only the measurements done with the wavefront sensor will be considered. Thus, the following Zernike coefficients will be compared:

- The Zernike coefficients obtained from the spot displacements by following Janssen's analytical method, from now on abbreviated as *analytical*.
- The Zernike coefficients obtained from the spot displacements by following the Iterative method, from now on abbreviated as *iterative*.
- The Zernike coefficients given by the software of the SHS, from now on abbreviated as *SHS*.
- The Zernike coefficients which where the input of the images addressed to the SLM, from now on abbreviated as *input*.

In the second part, paragraph 5.4, the measurements done with both the wavefront sensor and the interferometer will be considered. This means that the following Zernike coefficients will be compared:

- The Zernike coefficients obtained through the analysis of the images from interferometry, from now on abbreviated as *interferometer*.
- *Analytical*
- *Iterative*
- *SHS*
- *Input*

In table 5.1, all the measurements are listed with corresponding reconstruction methods. The second column show which aberrations where addressed to the SLM. In the third column is indicated whether there is measured only with the wavefront sensor (WFS) or also with the interferometer (Interf.).

Table 5.1: This table shows the different measurements with their corresponding input aberrations. The last columns show which method is used to evaluate the measurement.

Measurement	*Input*	*Measured with*		*Methods*			
		WFS	*Interf.*	*Analytical*	*Iterative*	*SHS*	*Interf.*
1	Defocus	•		•	•	•	
2	Ast. $0°$	•		•	•	•	
3	Ast. $45°$	•		•	•	•	
4	Coma X	•		•	•	•	
5	Coma Y	•		•	•	•	
6	Combination	•		•	•	•	
7	Combination	•		•	•	•	
8	Combination	•		•	•	•	
9	Combination	•		•	•	•	
10	Combination	•		•	•	•	
11	Coma X	•	•	•	•	•	•
12	Combination	•	•	•	•	•	•

5.1. Example of calculation of Zernike coefficients

Table 5.2 on page 28, corresponding to measurement 11, serves as an example of how the corrected and normalized Zernike coefficients are calculated for the various reconstruction methods. The Zernike aberration of the image addressed to the SLM was coma X, see table 5.2a. Regarding table 5.2b to 5.2e: the reference measurement, i.e. a black screen addressed to the SLM, is analyzed by every method separately. The obtained raw Zernike coefficients, in units of λ, are listed in the second column. The third column shows the raw Zernike coefficients, again in units of λ, of the measurement when *input* was addressed to the SLM. The fourth column equals the third subtracted by the second column. As can be seen in the table, the input of coma X was 0.5λ. The analytical method, the iterative method, the SHS and the interferometer measured a coefficient of respectively 0.254λ, 0.239λ, 0.578λ and 0.554λ for coma X, see figure 5.1.

The coefficients of the SHS and the interferometer have approximately the same order of magnitude as the input. However, the coefficients obtained through the analytical and iterative method don't have the same order of magnitude as the input. It seems like there is a factor two difference between them. Maybe there is some ambiguity regarding the units used by the software of the wavefront sensor. Nevertheless, in order to be able to compare the coefficients either way, we normalized them by their Euclidean norm. The results can be seen in the last columns of table 5.2a to 5.2e. After normalization, the input coefficient of coma X equals 1 and the normalized coefficients of the four reconstruction methods agree quite well with the input (0.998, 0.998, 0.998 and 0.999). In the next paragraphs, the normalized coefficients will be compared with each other. The calculation of the results of measurements 1 to 10 and 12 are shown in appendix A.

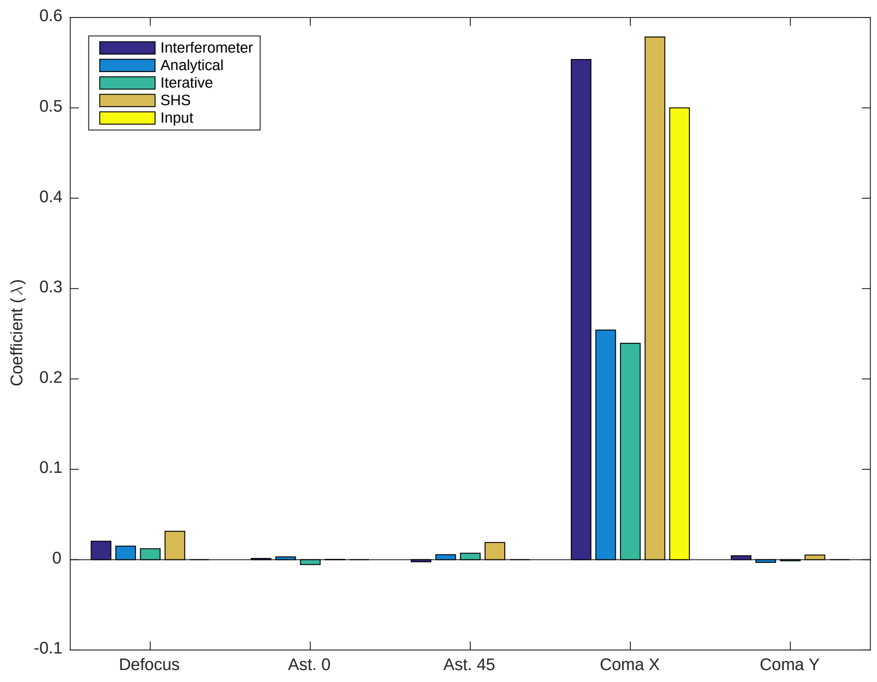


Figure 5.1: Bar plot of the corrected Zernike coefficients, in units of λ, corresponding to measurement 11. Note that they are not normalized.

Table 5.2: Calculation of corrected and normalized Zernike coefficients of measurement 11

(a) Input

Aberration	*Raw* (λ)	*Normalized*
Defocus	0	0
Ast. 0°	0	0
Ast. 45°	0	0
Coma X	0.5	**1**
Coma Y	0	0

(b) Analytical

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	−0.016	−0.002	−0.015	0.059
Ast. 0°	0.084	0.087	0.003	0.012
Ast. 45°	0.043	0.048	0.006	0.022
Coma X	0.014	0.268	0.254	**0.998**
Coma Y	−0.006	−0.009	−0.003	−0.012

(c) Iterative

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	−0.017	−0.005	0.012	0.050
Ast. 0°	0.084	0.079	−0.005	−0.023
Ast. 45°	0.043	0.050	0.007	0.030
Coma X	0.013	0.252	0.239	**0.998**
Coma Y	−0.007	−0.008	−0.001	−0.005

(d) SHS

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	−0.041	−0.010	0.031	0.054
Ast. 0°	0.201	0.201	0.000	0.001
Ast. 45°	0.103	0.122	0.019	0.033
Coma X	0.034	0.612	0.578	**0.998**
Coma Y	−0.019	−0.014	0.005	0.009

(e) Interferometer

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	0.036	0.056	0.020	0.037
Ast. 0°	0.162	0.163	0.001	0.002
Ast. 45°	0.183	0.181	−0.002	−0.004
Coma X	0.000	0.554	0.554	**0.999**
Coma Y	0.012	0.016	0.004	0.008

Table 5.3: The runtime of the algorithms corresponding to the different wavefront reconstruction methods

Method	*Runtime* (s)
Interferometer	53.6481
Analytical	0.0876
Iterative	0.4293

5.2. Runtime

Apart from the RMS-error of the Zernike coefficients with respect to the input, the runtime of the different algorithms is also important to be considered. In this paragraph some attention will be given to this subject. We can subdivide this issue into two parts. The first one is the amount of time a measurement takes. The second part is the amount of time the analysis of the retrieved data takes, i.e. executing the different algorithms.

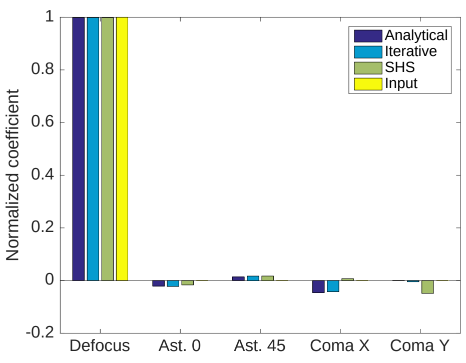
First, we did not measure the precise amount of time of measuring the Zernike aberrations. However, a simple insight could be given. When measuring with the interferometer, an image should be taken with multiple different voltages applied to the piezoelectric translation stage. The voltage goes from $45.5\,\mathrm{V}$ to $49.5\,\mathrm{V}$ with steps of $0.02\,\mathrm{V}$, each taking a time of $800\,\mathrm{ms}$. It should be evident that this will take a few minutes. As regards the wavefront sensor, measuring the spot displacements will definitely not take that long.

Second and more important, we will consider the runtime of the different algorithms of the wavefront reconstruction methods. Because the software given by the SHS gives the Zernike coefficients directly on the screen, we will only consider the runtime of the algorithms of the interferometer, the analytical method and the iterative method, which is implemented by us. Regarding the interferometer, we will only consider the time it takes to obtain the raw, i.e. not yet corrected and normalized, Zernike coefficients out of the images taken with the CCD camera. Regarding the analytical and iterative method, we consider the time it takes to retrieve the raw Zernike coefficients out of the spot displacements. In table 5.3, we evaluated these times for measurement 11 (figure 5.4a), where only coma X was addressed as input to the SLM.

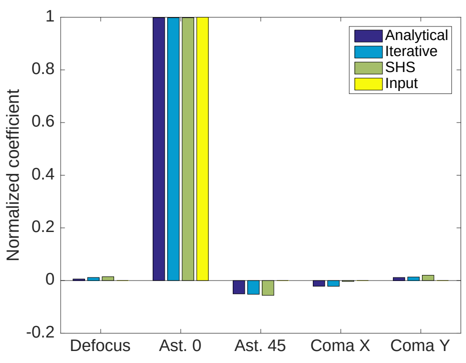
5.3. Comparison of wavefrontsensor methods

This paragraph will present the results of the comparison of the different wavefront sensor reconstruction methods. The figures 5.2a up to and including 5.2e show the outcome when only single aberrations were addressed to the SLM. Remember that they have already been corrected for any undesired aberrations caused by optical elements in the set-up. Afterwards they have been normalized. We see that the SHS itself and both the analytical and the iterative method are quite well capable of measuring all the five different Zernike aberrations, which were addressed to the SLM. However, sometimes a small coefficient of a certain aberration is measured while the input was zero for that aberration. For example, a little bit defocus is measured, as can be seen in figures 5.2c up to and including 5.2e, while there was actually no defocus addressed to the SLM. Further error analysis will be discussed in paragraph 5.5.

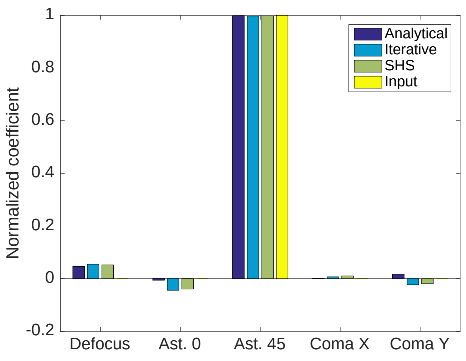
Figures 5.3a to 5.3e show the bar plots of the Zernike coefficients obtained through the different wavefront reconstruction methods, when a combination of Zernike aberrations were addressed as input to the SLM. Note that only the wavefront sensor was used to measure. All the three methods agree quite well with the input, although the Zernike coefficients given by the software of the SHS does show a greater error with respect to the input for some measurements, for example the coefficient of astigmatism 0° of measurements 6 and 10 in figures 5.3a and 5.3e.



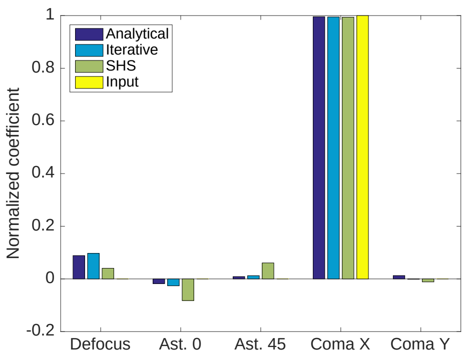
(a) Measurement 1



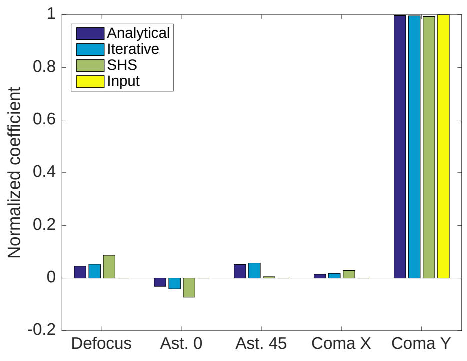
(b) Measurement 2



(c) Measurement 3

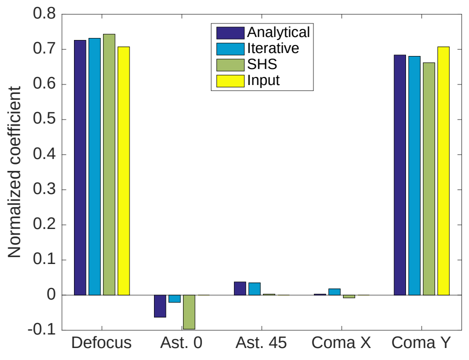


(d) Measurement 4

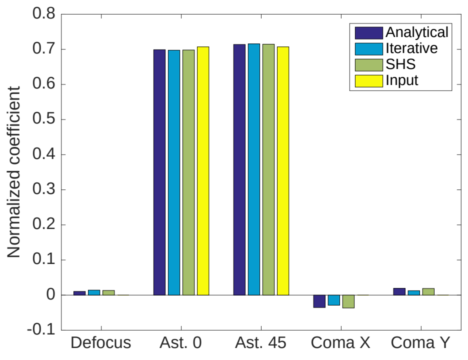


(e) Measurement 5

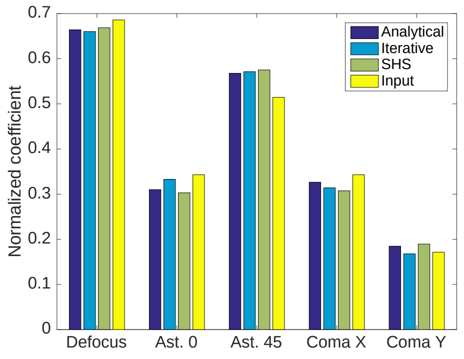
Figure 5.2: Bar plots of the normalized Zernike coefficients of different wavefront reconstruction methods, i.e. *analytical*, *iterative* and *SHS*. Note that they have already been corrected with the reference measurement. The measurements were done with the wavefront sensor. The images addressed to the SLM contained a single Zernike aberration, which is presented as *Input*.



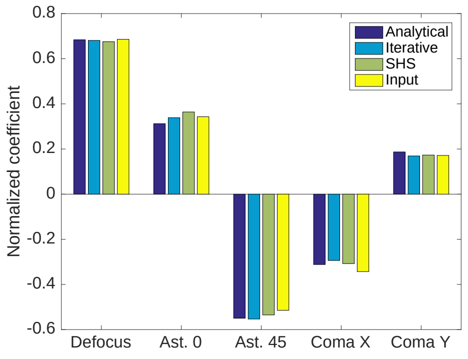
(a) Measurement 6



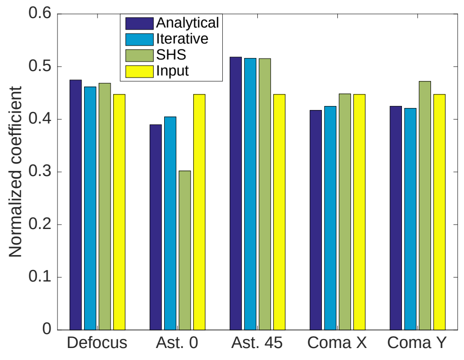
(b) Measurement 7



(c) Measurement 8



(d) Measurement 9

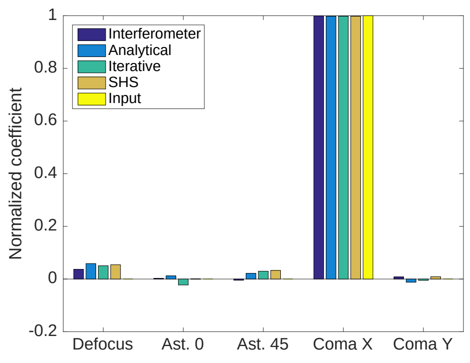


(e) Measurement 10

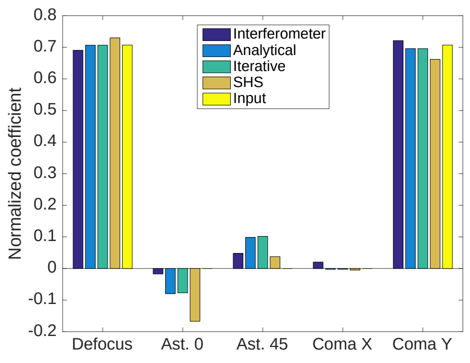
Figure 5.3: Bar plots of the normalized Zernike coefficients of different wavefront reconstruction methods, i.e. *analytical*, *iterative* and *SHS*. Note that they have already been corrected with the reference measurement. The measurements were done with the wavefront sensor. The images addressed to the SLM contained a combination of Zernike aberrations, which is presented as *input*.

5.4. Comparison between interferometer and wavefront sensor methods

In this paragraph the results of the comparison between interferometry and wavefront sensor reconstruction methods will be presented. The results are shown in figure 5.4a and 5.4b. We see that all the methods are able to measure the Zernike aberrations, which were addressed to the SLM. However, also here, the Zernike coefficients given by the software of the SHS have greater relative errors with respect to the input than the other methods. This will become visible in the following paragraph.



(a) Measurement 11



(b) Measurement 12

Figure 5.4: Barplots of the normalized Zernike coefficients of different wavefront reconstruction methods, i.e. *interferometer*, *analytical*, *iterative* and *SHS*. Note that they have already been corrected with the reference measurement. The measurements were done with the wavefront sensor and with the camera using interferometry. The images addressed to the SLM contained (a) a single Zernike aberration and (b) a combination of two Zernike aberrations, which is presented as *input*.

5.5. Error analysis

In this paragraph the error of the retrieved Zernike coefficients will be analysed. The root-mean-square-error of a given method of a given measurement with respect to the input is defined as follows:

$$\mathrm{RMS} = \sqrt{\frac{1}{5}\sum_{i}\left(\alpha_i^{\text{method}} - \alpha_i^{\text{input}}\right)^2} \tag{5.1}$$

We are only evaluating the first five Zernike aberrations, without including piston, tilt X and tilt Y. That explains the factor $\frac{1}{5}$ within the square root. It is possible to calculate the RMS-error for all measurements for every method. Figure 5.5 shows the outcome. In this figure it can be seen that the RMS-errors of the Zernike coefficients given by the software of the SHS is clearly higher than the errors of other wavefront reconstruction methods. This means that *SHS* is less accurate than the other methods. The analytical method is as accurate as the iterative method is. Eventually, the interferometer is clearly the most accurate method. However, it should be noted that only two measurements have been made using interferometry, which is in contrast with the other methods.

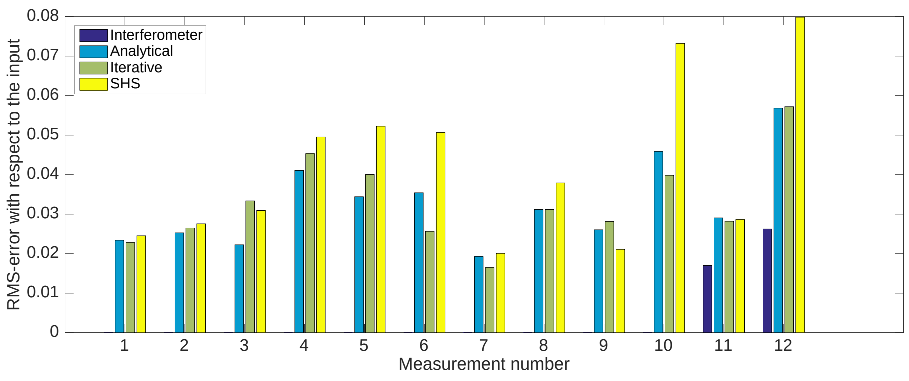


Figure 5.5: The RMS-error with respect to the corresponding input calculated for all measurements and every wavefront reconstruction method

6 Conclusions and future work

6.1. Conclusions

In this thesis we measured a wavefront through various methods. The goal was to find out whether it is possible to measure a wavefront qualitatively and quantitatively with Janssen's recently developed analytical algorithm [1]. We used the spot displacements given by the wavefront sensor to relate the derivatives of the wavefront with respect to x and y with the Zernike coefficients of the wavefront itself. We compared the results with some conventional and commercially used methods: an iterative integration method of the SHS, which we also implemented ourselves in a Matlab script. Moreover, we compared some of them with the reconstructed wavefront obtained through interferometry.

We constructed the wavefront to be measured by addressing certain aberrations to a spatial light modulator. First, we addressed single Zernike aberrations to the spatial light modulator. These were measured with the wavefront sensor only. Both the analytical and iterative algorithm were capable of measuring these aberrations. That is also valid for the Zernike coefficients given by the SHS, but they showed a higher error with respect to the input of the Zernike coefficients addressed to the spatial light modulator in comparison with the analytical and iterative method. Apart from single aberrations, combinations of Zernike aberrations were as well addressed to the spatial light modulator and all results of all different methods showed a high degree of agreement with the input. The results obtained through interferometry were actually most precise, i.e. they had the lowest RMS-error with respect to the input addressed to the spatial light modulator. However, it should be noted that the runtime of the algorithm corresponding to the interferometer is much higher in comparison to the runtime of the algorithms of the analytical and iterative method and the set-up is more complicated since it requires several exposures and a reference mirror. In short, Janssen's analytical algorithm is a good, reliable and fast wavefront reconstruction method to qualify of which aberrations a certain wavefront consists.

6.2. Recommendations for future work

First of all, only a few measurements were done with the interferometer. In future work more measurements could be done to investigate whether the error of the obtained Zernike coefficients with respect to the input remains low when varying the Zernike aberrations addressed to the spatial light modulator. Moreover, the correlation curve of the interferometric data (figure 4.7) should have a smoother cosine-like shape without irregular fluctuations, like in [12]. They may be caused by the so-called 5-5 configuration of the spatial light modulator. Another configuration may lead to less flickering of the display of the SLM. Besides, these irregular fluctuations of the correlation curve may also be caused by the fact that the SLM might have been unstably positioned. This could be explored in future research.

It needs to be mentioned that the Zernike coefficients, before normalization, didn't have the same order of magnitude for all different methods. The analytical and iterative method did have the same order of magnitude with respect to each other, but they didn't agree with the coefficients given by the software of the SHS and the ones obtained through interferometry before normalization. Thus, all the methods are capable of measuring the aberrations qualitatively, but not quantitatively. This could have something to do with the units of the spot displacements exported by the software of the SHS or the conventions used by the SHS. This could be subject to further investigation.

Finally, in further research more Zernike polynomials could be evaluated instead of just five, i.e. defocus, astigmatism $0°$, astigmatism $45°$, coma X and coma Y.

A

Calculation of results of the measurements

In this appendix, the numerical values of Zernike coefficients of the measurements are shown in tables. Note that the calculation of the results of measurement 11 is already shown in table 5.2.

For every measurement, the first table refers to the input addressed to the SLM. The subsequent tables contain the results corresponding to the different wavefront reconstruction methods. The reference measurement, i.e. a black screen addressed to the SLM, is analyzed by every method separately. The obtained raw Zernike coefficients, in units of λ, are listed in the second column. The third column shows the raw Zernike coefficients, again in units of λ, of the measurement when *input* was addressed to the SLM. The fourth column equals the third subtracted by the second column. The last column equals the fourth divided by the Euclidean norm.

A.1. Measurement 1

Table A.1: Calculation of results of measurement 1

(a) Input

Aberration	*Raw* (λ)	*Normalized*
Defocus	0.5	1
Ast. 0°	0	0
Ast. 45°	0	0
Coma X	0	0
Coma Y	0	0

(b) Analytical

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	0.047	0.261	0.214	0.999
Ast. 0°	0.115	0.110	−0.004	−0.021
Ast. 45°	−0.068	−0.065	0.003	0.0140
Coma X	0.007	−0.003	−0.010	−0.046
Coma Y	−0.015	−0.016	0.000	−0.001

(c) Iterative

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	0.047	0.258	0.211	0.999
Ast. 0°	0.112	0.107	−0.005	−0.022
Ast. 45°	−0.067	−0.063	0.004	0.017
Coma X	0.008	−0.001	−0.009	−0.042
Coma Y	−0.016	−0.017	−0.001	−0.005

(d) SHS

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	0.119	0.683	0.564	0.998
Ast. 0°	0.303	0.293	−0.009	−0.017
Ast. 45°	−0.177	−0.167	0.010	0.017
Coma X	0.021	0.025	0.004	0.007
Coma Y	−0.024	−0.052	−0.028	−0.049

A.2. Measurement 2

Table A.2: Calculation of results of measurement 2

(a) Input

Aberration	*Raw* (λ)	*Normalized*
Defocus	0	0
Ast. 0°	0.5	1
Ast. 45°	0	0
Coma X	0	0
Coma Y	0	0

(b) Analytical

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	0.047	0.048	0.001	0.006
Ast. 0°	0.115	0.311	0.196	0.998
Ast. 45°	−0.068	−0.078	−0.010	−0.051
Coma X	0.007	0.003	−0.004	−0.022
Coma Y	−0.015	−0.013	0.002	0.012

(c) Iterative

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	0.047	0.049	0.002	0.012
Ast. 0°	0.112	0.309	0.197	0.998
Ast. 45°	−0.067	−0.077	−0.010	−0.052
Coma X	0.008	0.004	−0.004	−0.022
Coma Y	−0.016	−0.014	0.003	0.013

(d) SHS

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	0.119	0.127	0.008	0.015
Ast. 0°	0.303	0.826	0.523	0.998
Ast. 45°	−0.177	−0.206	−0.029	−0.056
Coma X	0.021	0.019	−0.002	−0.003
Coma Y	−0.024	−0.014	0.011	0.020

A.3. Measurement 3

Table A.3: Calculation of results of measurement 3

(a) Input

Aberration	*Raw* (λ)	*Normalized*
Defocus	0	0
Ast. 0°	0	0
Ast. 45°	0.5	1
Coma X	0	0
Coma Y	0	0

(b) Analytical

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	0.047	0.055	0.009	0.046
Ast. 0°	0.115	0.113	−0.001	−0.006
Ast. 45°	−0.068	0.119	0.187	0.999
Coma X	0.007	0.007	0.000	0.002
Coma Y	−0.015	−0.012	0.003	0.017

(c) Iterative

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	0.047	0.057	0.010	0.055
Ast. 0°	0.112	0.103	−0.008	−0.044
Ast. 45°	−0.067	0.123	0.190	0.997
Coma X	0.008	0.009	0.001	0.007
Coma Y	−0.016	−0.021	−0.004	−0.023

(d) SHS

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	0.119	0.145	0.027	0.052
Ast. 0°	0.303	0.283	−0.020	−0.039
Ast. 45°	−0.177	0.329	0.506	0.998
Coma X	0.021	0.026	0.005	0.010
Coma Y	−0.024	−0.034	−0.010	−0.019

A.4. Measurement 4

Table A.4: Calculation of results of measurement 4

(a) Input

Aberration	*Raw* (λ)	*Normalized*
Defocus	0	0
Ast. 0°	0	0
Ast. 45°	0	0
Coma X	0.5	1
Coma Y	0	0

(b) Analytical

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	0.047	0.067	0.021	0.089
Ast. 0°	0.115	0.110	−0.004	−0.018
Ast. 45°	−0.068	−0.066	0.002	0.009
Coma X	0.007	0.240	0.233	0.996
Coma Y	−0.015	−0.012	0.003	0.013

(c) Iterative

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	0.047	0.069	0.022	0.097
Ast. 0°	0.112	0.106	−0.006	−0.026
Ast. 45°	−0.067	−0.064	0.003	0.012
Coma X	0.008	0.231	0.223	0.995
Coma Y	−0.016	−0.017	0.000	−0.001

(d) SHS

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	0.119	0.144	0.025	0.040
Ast. 0°	0.303	0.253	−0.050	−0.082
Ast. 45°	−0.177	−0.140	0.037	0.061
Coma X	0.021	0.626	0.606	0.994
Coma Y	−0.024	−0.031	−0.007	−0.011

A.5. Measurement 5

Table A.5: Calculation of results of measurement 5

(a) Input

Aberration	*Raw* (λ)	*Normalized*
Defocus	0	0
Ast. 0°	0	0
Ast. 45°	0	0
Coma X	0	0
Coma Y	0.5	1

(b) Analytical

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	0.047	0.057	0.010	0.045
Ast. 0°	0.115	0.107	−0.007	−0.032
Ast. 45°	−0.068	−0.056	0.012	0.052
Coma X	0.007	0.010	0.003	0.014
Coma Y	−0.015	0.213	0.229	0.997

(c) Iterative

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	0.047	0.058	0.011	0.052
Ast. 0°	0.112	0.103	−0.009	−0.041
Ast. 45°	−0.067	−0.055	0.012	0.057
Coma X	0.008	0.012	0.004	0.018
Coma Y	−0.016	0.198	0.214	0.996

(d) SHS

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	0.119	0.168	0.049	0.087
Ast. 0°	0.303	0.261	−0.041	−0.073
Ast. 45°	−0.177	−0.174	0.003	0.005
Coma X	0.021	0.037	0.016	0.029
Coma Y	−0.024	0.541	0.566	0.993

A.6. Measurement 6

Table A.6: Calculation of results of measurement 6

(a) Input

Aberration	*Raw* (λ)	*Normalized*
Defocus	0.299	0.707
Ast. 0°	0	0
Ast. 45°	0	0
Coma X	0	0
Coma Y	0.299	0.707

(b) Analytical

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	−0.007	0.121	0.128	0.726
Ast. 0°	0.129	0.118	−0.011	−0.063
Ast. 45°	−0.158	−0.151	0.007	0.038
Coma X	0.013	0.013	0.001	0.003
Coma Y	−0.004	0.117	0.121	0.684

(c) Iterative

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	−0.007	0.122	0.129	0.732
Ast. 0°	0.126	0.122	−0.004	−0.021
Ast. 45°	−0.157	−0.151	0.006	0.035
Coma X	0.013	0.016	0.003	0.018
Coma Y	−0.005	0.116	0.120	0.680

(d) SHS

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	−0.021	0.298	0.319	0.743
Ast. 0°	0.303	0.262	−0.042	−0.097
Ast. 45°	−0.375	−0.373	0.001	0.003
Coma X	0.029	0.025	−0.003	−0.008
Coma Y	−0.006	0.278	0.284	0.662

A.7. Measurement 7

Table A.7: Calculation of results of measurement 7

(a) Input

Aberration	*Raw* (λ)	*Normalized*
Defocus	0	0
Ast. 0°	0.354	0.707
Ast. 45°	0.354	0.707
Coma X	0	0
Coma Y	0	0

(b) Analytical

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	−0.007	−0.005	0.002	0.011
Ast. 0°	0.129	0.272	0.143	0.699
Ast. 45°	−0.158	−0.012	0.146	0.714
Coma X	0.013	0.006	−0.007	−0.035
Coma Y	−0.004	0.000	0.004	0.019

(c) Iterative

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	−0.007	−0.004	0.003	0.014
Ast. 0°	0.126	0.271	0.145	0.698
Ast. 45°	−0.157	−0.009	0.148	0.716
Coma X	0.013	0.007	−0.006	−0.029
Coma Y	−0.005	−0.002	0.003	0.013

(d) SHS

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	−0.021	−0.015	0.007	0.013
Ast. 0°	0.303	0.647	0.343	0.698
Ast. 45°	−0.375	−0.023	0.351	0.715
Coma X	0.029	0.011	−0.018	−0.037
Coma Y	−0.006	0.003	0.009	0.019

A.8. Measurement 8

Table A.8: Calculation of results of measurement 8

(a) Input

Aberration	*Raw* (λ)	*Normalized*
Defocus	0.281	0.686
Ast. 0°	0.141	0.343
Ast. 45°	0.211	0.514
Coma X	0.141	0.343
Coma Y	0.070	0.171

(b) Analytical

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	−0.007	0.110	0.118	0.664
Ast. 0°	0.129	0.184	0.055	0.310
Ast. 45°	−0.158	−0.057	0.101	0.568
Coma X	0.013	0.071	0.058	0.327
Coma Y	−0.004	0.029	0.033	0.185

(c) Iterative

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	−0.007	0.111	0.118	0.660
Ast. 0°	0.126	0.186	0.060	0.333
Ast. 45°	−0.157	−0.055	0.102	0.571
Coma X	0.013	0.069	0.056	0.314
Coma Y	−0.005	0.025	0.030	0.168

(d) SHS

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	−0.021	0.252	0.273	0.669
Ast. 0°	0.303	0.427	0.124	0.303
Ast. 45°	−0.375	−0.139	0.235	0.575
Coma X	0.029	0.154	0.126	0.307
Coma Y	−0.006	0.072	0.077	0.190

A.9. Measurement 9

Table A.9: Calculation of results of measurement 9

(a) Input

Aberration	*Raw* (λ)	*Normalized*
Defocus	0.281	0.686
Ast. 0°	0.141	0.343
Ast. 45°	−0.211	−0.514
Coma X	−0.141	−0.343
Coma Y	0.070	0.171

(b) Analytical

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	−0.007	0.109	0.116	0.684
Ast. 0°	0.129	0.182	0.053	0.312
Ast. 45°	−0.158	−0.251	−0.093	−0.550
Coma X	0.013	−0.040	−0.053	−0.312
Coma Y	−0.004	0.028	0.032	0.187

(c) Iterative

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	−0.007	0.110	0.117	0.681
Ast. 0°	0.126	0.184	0.058	0.339
Ast. 45°	−0.157	−0.253	−0.095	−0.553
Coma X	0.013	−0.038	−0.051	−0.294
Coma Y	−0.005	0.024	0.029	0.169

(d) SHS

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	−0.021	0.272	0.293	0.676
Ast. 0°	0.303	0.461	0.158	0.364
Ast. 45°	−0.375	−0.607	−0.232	−0.535
Coma X	0.029	−0.105	−0.133	−0.308
Coma Y	−0.006	0.069	0.075	0.173

A.10. Measurement 10

Table A.10: Calculation of results of measurement 10

(a) Input

Aberration	*Raw* (λ)	*Normalized*
Defocus	0.191	0.447
Ast. 0°	0.191	0.447
Ast. 45°	0.191	0.447
Coma X	0.191	0.447
Coma Y	0.191	0.447

(b) Analytical

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	−0.007	0.075	0.083	0.475
Ast. 0°	0.129	0.197	0.068	0.390
Ast. 45°	−0.158	−0.067	0.090	0.518
Coma X	0.013	0.086	0.073	0.417
Coma Y	−0.004	0.070	0.074	0.425

(c) Iterative

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	−0.007	0.074	0.082	0.461
Ast. 0°	0.126	0.198	0.071	0.405
Ast. 45°	−0.157	−0.066	0.091	0.516
Coma X	0.013	0.088	0.075	0.425
Coma Y	−0.005	0.070	0.074	0.421

(d) SHS

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	−0.021	0.162	0.184	0.469
Ast. 0°	0.303	0.422	0.118	0.302
Ast. 45°	−0.375	−0.173	0.202	0.515
Coma X	0.029	0.204	0.176	0.448
Coma Y	−0.006	0.179	0.185	0.472

A.11. Measurement 12

Table A.11: Calculation of results of measurement 12

(a) Input

Aberration	*Raw* (λ)	*Normalized*
Defocus	0.299	0.707
Ast. 0°	0	0
Ast. 45°	0	0
Coma X	0	0
Coma Y	0.299	0.707

(b) Analytical

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	−0.026	0.104	0.131	0.707
Ast. 0°	0.086	0.071	−0.015	−0.08
Ast. 45°	0.017	0.035	0.018	0.098
Coma X	0.014	0.014	0.000	−0.003
Coma Y	−0.007	0.122	0.129	0.696

(c) Iterative

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	−0.026	0.104	0.130	0.707
Ast. 0°	0.086	0.072	−0.014	−0.077
Ast. 45°	0.018	0.037	0.019	0.101
Coma X	0.015	0.015	0.000	−0.003
Coma Y	−0.007	0.122	0.128	0.696

(d) SHS

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	−0.058	0.239	0.298	0.730
Ast. 0°	0.178	0.110	−0.068	−0.167
Ast. 45°	0.041	0.056	0.015	0.038
Coma X	0.031	0.029	−0.002	−0.005
Coma Y	−0.007	0.263	0.270	0.662

(e) Interferometer

Aberration	*Reference*	*Measurement*		
	Raw (λ)	*Raw* (λ)	*Corrected* (λ)	*Normalized*
Defocus	0.036	0.328	0.292	0.691
Ast. 0°	0.162	0.154	−0.007	−0.017
Ast. 45°	0.183	0.204	0.020	0.047
Coma X	0.000	0.009	0.009	0.020
Coma Y	0.012	0.317	0.305	0.721

B
Matlab scripts: wavefrontsensor

B.1. MethodsComparison

```
%MethodsComparison
% Reads data(derivatives) given by the SHS and runs it trhough the
scipts of
% the Janssen method and the itearive integration method. Also reads
the Zernike
% coefficients given by SHS itself based on those data.
% prints out the vectors containing the coefficients and can give
barplot or
% plot of pupil distribution based on the calculated Zernikes.
```

reading data

```
folder_to_analyze = ('/Users/jellesalverda/Documents/MATLAB/Data');
addpath(folder_to_analyze)

%Read spotdisplacements given by SHS
[datax,xm,ym]=readzernikedata('black_spot_x.txt');
[datay,xm,ym]=readzernikedata('black_spot_y.txt');

%Recorrect matrices if they are not a square
[M,N]=size(datax);
if M<N
    ym(1:M)=ym(N-M+1:N); ym(M+1:N)=[];
end
```

Read coeffcients given by SHS

```
SHScoef=readSHScoef('black_zernike.txt');
```

rotation of 90 degrees clockwise

```
datax=datax*(-1);
datay=datay*(-1);

%Mirror data wih respect to x-axix, because of different conventions
datax=flipud(datax);
datay=flipud(datay);

datax2=datax;
datay2=datay;
datay=rot90(datax2,-1);
```

```
datax=rot90(datay2,-1);
datay=datay*(-1);
datax=datax*(1);

xm2=xm;
ym2=ym;
xm=ym2;
ym=xm2;
```

Normalizing to unit circle

```
%variables specifying some properties of the data. These are needed to
run
%the scipts of both methods
s=length(xm);
kmax=length(SHScoef);

%some variables that might be needed to correct the data to the right
units
lambda = 635e-9;
pixel_sz = 11e-6;
f = 4.63; %focal length

%normalizing to the unit disk
x=xm/max([max(abs(xm)) max(abs(ym))]);
y=ym/max([max(abs(xm)) max(abs(ym))]);
```

spotdisp to derivative

```
dWdx = datax./sqrt(f^2+datax.^2+datay.^2);
dWdy = datay./sqrt(f^2+datax.^2+datay.^2);
```

Possible to remove tilt

tiltx=mean2(dWdx(find(1-isnan(dWdx)))); tilty=mean2(dWdy(find(1-isnan(dWdy)))); dWdx=dWdx-tiltx; dWdy=dWdy-tilty;

further analysis

```
%Janssen
[Alpha_est] = EstimateAlphasFromDerivatives_flipud(dWdx, dWdy, x, y,
kmax)./(2*pi);

%It-Int
[Alpha_est2] = AlphasFromDerivativesItInt_flipud(dWdx, dWdy, x, y,
kmax)./(2*pi);

Coef=[Alpha_est', Alpha_est2', SHScoef'];

%Leave out piston and tilt X and Y
SHScoef(1:3)=0;
Alpha_est(1:3)=0;
Alpha_est2(1:3)=0;
SHScoef(9)=0;
Alpha_est(9)=0;
Alpha_est2(9)=0;

%nomalizing the vectors containing the given Zernikes
SHScoef=SHScoef/norm(SHScoef);
Alpha_est=Alpha_est/norm(Alpha_est);
Alpha_est2=Alpha_est2/norm(Alpha_est2);

%print out coefficients
```

```
Coef_norm_notilt=[Alpha_est', Alpha_est2', SHScoef'];
```

Plots

```
%Plot comparison
figure;bar([SHScoef; Alpha_est; Alpha_est2].');
xlabel('Zernike'); ylabel('coefficient');legend('SHS','Analytic' ,
'Iterative Int' );

%plotting the pupil
%pints of axis be plotted
[X,Y]=meshgrid(x,y);
[th,p]=cart2pol(X,Y);

%cenverting Zrnikes from single index to double index convention
%needed to use Alpha2pupil
A1=Zk2BetaSet(Alpha_est);
A2=Zk2BetaSet(Alpha_est2);
S=Zk2BetaSet(SHScoef);

%calculate pupil distribution
pupilA1=Alpha2Pupil(A1,p,th);
pupilA2=Alpha2Pupil(A2,p,th);
pupilS=Alpha2Pupil(S,p,th);

%plots
figure;
subplot(2,2,1)
surf(X,Y,angle(pupilA1));
title('analytic method');

subplot(2,2,2)
surf(X,Y,angle(pupilA2));
title('Iterative integration');

subplot(2,2,3)
surf(X,Y,angle(pupilS));
title('SHS');
```

B.2. Readzernikedata

```
%----modified version of the one given by Ruud Bokdam, Leendert van
Veen,
%---Ike Mulder, Joost Wooning and Michel van der Kaay
% Loads data in file specified by filename from Shack Hartmann sensor
% into matlab. Also includes the x and y values specified by the
sensor.
%
% Example:
% [data, x, y] = readzernikedata('datax.txt')
```

Function declaration

```
function [data, x, y] = readzernikedata(filename)
% read file
file = fileread(filename);

% preprocess filedata
strrep(file, '"', '');
strrep(file, 'UNDEFINED', 'NaN');
rows = strsplit(file, '\n');
rows = rows(7:end-3); % strip metadata
```

```
% declare variables with sizes
data = zeros(length(rows)-1, length(strsplit(cell2mat(rows(1)),
';'))-2);
y = zeros(1, length(strsplit(cell2mat(rows(1)), ';'))-2);

% read first row as x positions
row = strsplit(cell2mat(rows(1)), ';');
x = str2double(row(2:end-1));

% read actual data and y postions
for i=2:length(rows)
    row = strsplit(cell2mat(rows(i)), ';');

    y(i-1) = str2double(row(1));
    data(i-1, :) = str2double(row(2:end-1));
end

data = flipud(data);
y = fliplr(y);
```

B.3. ReadSHScoef

Reads the Zernike coefficeints from a .txt file format given by the SHS.

Input:

filename = file given by SHS whic conatins coefficients

output:

coef = row vector with the read coefficients

```
function [coef]=readSHScoef(filename)

file=fileread(filename);

rows = strsplit(file, {'\n',':'});
rows=rows(22:end-1);
rows=str2double(rows);
rows=rows(2:2:18);
coef=rows;
```

B.4. EstimateAlphasFromDerivatives

```
%By Sven van haver
%Phase retrieval method by A.J.E.M. Janssen
% DEv script WF reconstruction from cartesian derivatives via Zernikes
function [out] = EstimateAlphasFromDerivatives_flipud(dWdx, dWdy,
x_list, y_list, kmax)

[X, Y] = meshgrid(x_list, y_list);
Y=flipud(Y);
[THETA, RHO] = cart2pol(X, Y);

Kmax = power(ceil(sqrt(kmax)),2);
k=2:Kmax;
[n, m] = ZernikeNumber2NM(k);
Kmax = max(NM2ZernikeNumber(n+1,abs(m)+1));

%Get complex Zernikes from the pupil
Betap = Pupil2Zernike(dWdx + 1i.*dWdy, RHO, THETA, Kmax);
Betam = Pupil2Zernike(dWdx - 1i.*dWdy, RHO, THETA, Kmax);
```

```
Bsz = size(Betap);
Asz = [2,max(n)+1,max(abs(m)+1)];
Alpha_est = zeros(Asz);

%apply formula from Janssen
Alpha_est(sub2ind(Asz,1+(m<0), n+1, abs(m)+1)) = ((Betap(sub2ind(Bsz,1+((m+1)<0),
n, abs(m+1)+1))./2) + ...
(Betam(sub2ind(Bsz,1+((m-1)<0), n, abs(m-1)+1))./2))
./(n.*NeumannSymbol((n-abs(m))/2)) - ...
((Betap(sub2ind(Bsz,1+((m+1)<0), n+2, abs(m+1)+1))./2) + ...
(Betam(sub2ind(Bsz,1+((m-1)<0), n+2, abs(m-1)+1))./2))
./((n+2).*NeumannSymbol((n+2-abs(m))/2));

% Convert from complex to classical
Alphas(k(m==0)) = Alpha_est(sub2ind(Asz,1+(m(m==0)<0), n(m==0)+1,
abs(m(m==0))+1));
Alphas(k(m>0)) = Alpha_est(sub2ind(Asz,1+(m(m>0)<0), n(m>0)+1,
abs(m(m>0))+1)) + ...
    Alpha_est(sub2ind(Asz,2+(m(m>0)<0), n(m>0)+1, abs(m(m>0))+1));
Alphas(k(m<0)) = -1i.*Alpha_est(sub2ind(Asz,1+(m(m<0)<0), n(m<0)+1,
abs(m(m<0))+1)) + ...
    1i.*Alpha_est(sub2ind(Asz,(m(m<0)<0), n(m<0)+1, abs(m(m<0))+1));

out = real(Alphas(1:kmax));

function [out] = NeumannSymbol(k)

out = 1 + double(k~=0);
```

B.5. AlphasFromDerivativesItInt

```
%AlphasFromDerivativesItInt
% function that calculates the Zernikes coefficients from a set of
derivatives,
% using the iterative integraion method.
%
% Input:
%
% dWdx   = matrix containing x-derivatives
% dWdy   = matrix containing y-derivatives
% x-list = vector containing the x points
% y-list = vector containing the y points
%
% Output:
%
% out   = vector containing the calculated Zernikes coefficients.

function [out] = AlphasFromDerivativesItInt_flipud(dWdx, dWdy, x_list,
y_list, kmax)

addpath('NewImplementationBySvH');
addpath('studenten');

[X, Y] = meshgrid(x_list, y_list);
Y=flipud(Y);

[THETA, RHO] = cart2pol(X, Y);

Kmax = power(ceil(sqrt(kmax)),2);
k=2:Kmax;
[n, m] = ZernikeNumber2NM(k);
```

```
Asz = [2,max(n)+1,max(abs(m)+1)];

%Construct wavefront using iterative integration
wavefront  = iterativeIntegration(dWdx, dWdy, x_list, y_list, 10^(-5));
WaveFront=wavefront(:);

%Delete Nans
nans=isnan(WaveFront);
WaveFront(nans)=[];
THETA(nans)=[];
RHO(nans)=[];

%Calculate complex Zernikes from wavefront
Beta=Pupil2Zernike(WaveFront,RHO,THETA,kmax);

% Convert from complex to classical
Alphas(k(m==0)) = Beta(sub2ind(Asz,1+(m(m==0)<0), n(m==0)+1,
abs(m(m==0))+1));
Alphas(k(m>0)) = Beta(sub2ind(Asz,1+(m(m>0)<0), n(m>0)+1,
abs(m(m>0))+1)) + ...
    Beta(sub2ind(Asz,2+(m(m>0)<0), n(m>0)+1, abs(m(m>0))+1));
Alphas(k(m<0)) = -1i.*Beta(sub2ind(Asz,1+(m(m<0)<0), n(m<0)+1,
abs(m(m<0))+1)) + ...
    1i.*Beta(sub2ind(Asz,(m(m<0)<0), n(m<0)+1, abs(m(m<0))+1));

out = real(Alphas(1:kmax));
```

B.6. Zk2BetaSet

```
function [out] = Zk2BetaSet(Zk)
% BETASET2ZK
%
% This function converts a single index Zernike set, Zk, into a double
% index BetaSet (or AlphaSet) in 3 dimensional matrix notation:
%
%   Z(k) -> BetaSet(s,n+1,|m|+1)
%
% Syntax:   [BetaSet] = Zk2BetaSet(Zk)
%
% Sven van Haver 20131010
%

if max(size(Zk)) ~= numel(Zk)
    error('Input argument should be a Z(k) vector!')
end

k = find(Zk);
BetaSet = zeros(2,2,2);

if ~isempty(k)
    [n,m] = ZernikeNumber2NM(k);
    sZk =  [2 max([2 n(end) + abs(m(end))+1]) max([2 1+(n(end) +
abs(m(end)))/2])];
    BetaSet = zeros(sZk);
    BetaSet(sub2ind(sZk,1+(m<0),n+1,abs(m)+1))=Zk(k);
end

if ~(isreal(Zk) ~= isreal(BetaSet))
    out = (BetaSet);
else
    out = complex(BetaSet);
```

```
end
```

B.7. Alpha2Pupil

```
Generate Phase-only pupil from classical set of Zernike coefficients

This function generates the pupil defined by a set of classical Zernike
coefficients (Alpha's). It thus assumes the pupil amplitude to be
uniform and valued One. The function generates pupil values for the
specified radial and azymuthal coordinates.

Syntax:
        [Pupil] = Alpha2Pupil(Alpha,RHO,THETA)

Input:

Alpha = Matrix of 3 dimensions containing standard Zernike
coefficients.
        Matrix element Alpha(s,n+1,m+1) pertains to a Zernike
polynomial
        of degree, n, and order, \ensuremath{|}m\ensuremath{|}.
Parameter s indicates the angular
        dependance of the Zernike polynomial, where s=1 pertains to a
        Cosine and s=2 to a Sine dependance. [Radians]

RHO =   Matrix with radial positions in circular coordinate space of
        all pupil positions for which the field distribution should be
        generated. [length normalized to pupil radius]

THETA = Matrix with azymuthal positions in circular coordinate space of
        all pupil positions for which the field distribution should be
        generated. [Radians]

Output:

Pupil = Scalar eletric field distribution in the pupil, with uniform
amplitude of magnitude one and phase dsitribution defined by the
classical Zernike expansion Alpha.

Example(s):

     Generate a pupil with trefoil aberration
         >> Alpha=zeros(2,4,4);Alpha(1,4,4)=0.25;
         >> rho=0:0.5:1; theta=0:pi/2:2*pi;
         >> [RHO,THETA]=meshgrid(rho,theta);
         >> P=Alpha2Pupil(Alpha,RHO,THETA)
     Output:
         P =
              1.0000            0.9995 + 0.0312i   0.9689 + 0.2474i
              1.0000            1.0000 - 0.0000i   1.0000 - 0.0000i
              1.0000            0.9995 - 0.0312i   0.9689 - 0.2474i
              1.0000            1.0000 + 0.0000i   1.0000 + 0.0000i
              1.0000            0.9995 + 0.0312i   0.9689 + 0.2474i

Dependencies:
    Name:               Version:    Author:         Toolbox:
    Zernike.m           1.1         S. van Haver    NONE

Author: Dr. Ir. S. van Haver
Email:  svenvanhaver@gmail.com

Project: Extended Nijboer-Zernike theory
Keywords: Classical Zernike coefficients, Phase aberration,
          Phase only pupil, generate pupil

Version: v2.0 (26-07-2012)
```

Change-log

v1.0 (19-06-2010) * First operational version. v1.1 (08-11-2010) * Function has been fully documented v2.0 (26-07-2012) * The internal meshgrid command has been removed for compatibility reasons. If the function is now called with a vector input, the output is now also a vector. * Now using more efficient DCT recipe to calculate Zernikes

Function declaration

```
function [out] = Alpha2Pupil(arg1,arg2,arg3)
```

Function description

This function generates and plots the pupil distribution defined by a set of classical Zernike-coefficents (Alpha's) and coordinate data.

Syntax: [E] = Zernike2Pupil(Alpha,rho,theta)

```
Alpha is a 3D matrix containing the values of the non-zero Zernike
coefficients Alpha(s,n+1,|m|+1). The indices of the various matrix
elements are given by the degree and order of the Zernike coefficients.
n and m are the degree and order, respectively, and s indicates whether
the coefficient belongs to the cosine (s=1) or sine (s=2) version of
the Zernike function.

rho and theta are the sampling points in the radial and azymuthal
directions, respectively.
```

Sven van Haver (June 2010).

Function Input Handling

```
% Check number of input arguments
if nargin~=3
    error('Wrong numer of input arguments!')
end

% Check if arg2 and 3 are the same size
if isequal(size(arg2),size(arg3))==1

    % Assign computation variables:
    Alpha = arg1;     % Zernike coefficients
    RHO=arg2;          % Matrix with radial postions [normalized]
    THETA=arg3;         % Matrix with azimuthal postitions [radians]

    if isvector(Alpha)
        Alpha = Zk2BetaSet(Alpha);
    end

else

    % Display error if arg2 and arg3 are not the same size
    error('Argument 2 and 3 should be the same size!')

end
```

Function Part1: Analyze input

Extract additional info from input arguments

```
sB = size(Alpha);    % Determine size of 3D matrix Alpha. Size of the
                     % dimensions give the maximum degree and order of
                     % Zernike values included.
```

```
nmax = sB(1,2)-1;   % Determine maximum value of n (Zernike degree) for
                    % which a Zernike coefficient is included in Alpha.

mmax = sB(1,3)-1;   % Determine maximum value of m (Zernike order) for
                    % which a Zernike coefficient is included in Alpha.
```

Function Part2: Prepare calculation variables

Compute all Zernike functions needed for the execution of this function and allocate memory for output quantity

```
ZerDCT = ZernikeDCT(nmax,RHO(:));       % Generate all required Zernike
polynomials in
                                        % polynomial form.

PHI=zeros(size(RHO));                   % Allocate memory
```

Function Part3: generate pupil distribution

Calculate individual contributions of Zernike coefficients to phase function and sum them.

```
% Loop over all possible Zernike terms:
for n=0:nmax
    for m=min(n,mmax):-1:0
        if sum(abs(Alpha(:,n+1,abs(m)+1))) ~= 0

            % Select required Zernike polynomial, angular dependence
and
            % corresponding coefficient. Add their combined
contribution to
            % the total field E.
            PHI(:) = PHI(:) + (Alpha(1,n+1,abs(m)+1).*cos(m*THETA(:)) +
...
                Alpha(2,n+1,abs(m)+1).*sin(m*THETA(:))).*...
                squeeze(ZerDCT(n+1,abs(m)+1,:));

        end
    end
end

E = exp(1i*PHI);      % Compute Pupil field from phase distribution
```

Output handling

The output is a matrix containing the electric field at all positions [RHO,THETA] in the pupil.

```
out = E;    % The electric field distribution in the pupil as defined
by
            % the input Beta coefficients. Output is a 2D matrix where
            % every element E(x1,x2) gives the field at position
            % [RHO(x1,x2), THETA(x1,x2)] in circular coordinate space.
```

B.8. ZernikeNumber2NM

Convert Zernike single index, j, into double index, n and m.

This function converts the Zernike function numbering according to the FRINGE convention into a double index based numbering according to the radial and azymuthal orders, n and m, of the Zernike function [1]. We distinguish between Zernike functions having a cosine or sine dependence by adressing them a positive or negative azymuthal index, m, respectively.

```
[1] Handbook of Optical Systems, Vol 2, by Herbert Gross, WILEY-VCH
    2005, page 215.

Syntax:

     [n,m] = ZernikeNumber2NM(j)
     [n,m] = ZernikeNumber2NM(j,option)

Input:

   j  =  Vector with Single indices defining Zernike functions
according
         to the FRINGE convention.
Option =  A string defining which algorithm to use:
             - 'DEFAULT'     Recent, most efficient algorithm
             - 'LEGACY'      Algorithm used in previous toolbox version

Output:

  n  =   Corresponding degrees of the Zernike function pertaining to
the
         input indices j.

  m  =   Corresponding orders of the Zernike function pertaining to the
         input indices j. Note that positive m values pertain to
Zernike
         functions having a cosine angular dependence and negative m a
         sine dependence.

Example(s):
    [n,m] = ZernikeNumber2NM([1 7 36 94 12345])

         n =
                 0     3    10    15   122

         m =
                 0     1     0     3  -100

Dependencies: NONE

Author: Dr. Ir. S. van Haver
Email:  svenvanhaver@gmail.com

Project: Extended Nijboer-Zernike (ENZ) theory
Keywords: Zernike polynomial numbering, FRINGE convention, single index
          double index, conversion.

Version: v2.0 20131010
```

Change-log

```
v1.0 (23-12-2010)
    * First operational version.
    * Full documentation added
v2.0 20131010
    * Faster algorithm implemented
    * Old algorithm available under option 'legacy'
```

Function declaration

```
function [out1,out2] = ZernikeNumber2NM(arg1,arg2)
```

Function description

No additional information available.

Input handling

Assign input variables to computation variables and check if they are consistant.

```
switch nargin

    case 2
        index = arg1;            % FRINGE single index numbering
        option = arg2;           % string defining which algorithm to
use

    case 1
        index = arg1;            % FRINGE single index numbering
        option = 'DEFAULT';      % string defining which algorithm to
use

    otherwise
        error('Wrong number of input arguments!')
end

% Check if index only contains integers larger than Zero
if numel(index(index<1))>0 || numel(index(~(round(index)==index)))
    error('Input arg1 should be a vector with positive integers')
end

switch upper(option)

    case 'DEFAULT'

        k = index;

        q = floor(sqrt(k-1));
        p = floor((k-q.^2-1)./2);
        r = k-q.^2-2.*p;

        n = q+p;
        m = power(-1,r+1).*(q-p);

    case 'LEGACY'
```

Function Part1: Convert index to n and m

First intermediate quantities p and q are determined after which n and m are derived.

```
        n=0*index;
        m=0*index;
        for t = 1:numel(index)

            k = index(t);

            % Determine p = (n+|m|)/2 :
            tmp1 = 0;
            p = 0;
            while tmp1 < k-1
                p = p+1;
                tmp1 = max(cumsum(1:2:(2*p+1)))-1;
            end

            % Determine q = (n-|m|)/2 :
            tmp2 = tmp1;
            q = p;
            while tmp2 > k-1
```

```
                q = q-1;
                tmp2 = tmp1 - 2*(p-q);
            end

            % Derive n and m
            n(t) = p + q;
            if tmp2 == k-1
                m(t) = p - q;
            else
                m(t) = q - p;
            end
        end

    otherwise
        error('Unknown "option" provided as second argument!')
end
```

Output handling

Sent the desired output data to the output variables

```
out1 = n;   % Zernike degree
out2 = m;   % Zernike order
```

B.9. NM2ZernikeNumber

```
Convert double Zernike index, n and m, into a single index, j.

This function converts the Zernike function numbering according to the
Zernike degree and order, n and m, into a single index number according
to the FRINGE convention [1]. Note that we distinguish between Zernike
functions having a cosine or sine dependence by adressing them a
positive or negative azymuthal index, m, respectively.

[1] Handbook of Optical Systems, Vol 2, by Herbert Gross, WILEY-VCH
    2005, page 215.

Syntax:

     [j] = NM2ZernikeNumber(n,m)
     [j] = NM2ZernikeNumber(n,m,option)

Input:

  n    =  Degree of the Zernike function for which the index j should be
          generated.

  m    =  Order of the Zernike function for which the index j should be
          generated. Positive or negative values of m adress the cosine
          and sine dependence respectively.
```

Option = A string defining which algorithm to use: - 'DEFAULT' Recent, most efficient algorithm - 'LEGACY' Algorithm used in previous toolbox version

```
Output:

  j   =   Single index defining a Zernike function according to the
          FRINGE convention.

Example(s):
    \ensuremath{>}\ensuremath{>} n = [0 3 10 15 122];
    \ensuremath{>}\ensuremath{>} m = [0 1 0 3 -100];
    \ensuremath{>}\ensuremath{>} [j] = NM2ZernikeNumber(n,m)
```

```
       j =
            1    7   36   94   12345

Dependencies: NONE

Author: Dr. Ir. S. van Haver
Email:  svenvanhaver@gmail.com

Project: Extended Nijboer-Zernike (ENZ) theory
Keywords: Zernike polynomial numbering, FRINGE convention, single index
          double index, conversion.

Version: v2.0 20131010
```

Change-log

```
v1.0 (23-12-2010)
    * First operational version.
    * Full documentation added
v2.0 20131010
    * Faster algorithm implemented
    * Old algorithm available under option 'legacy'
```

Function declaration

```
function [out] = NM2ZernikeNumber(arg1,arg2,arg3)
```

Function description

No additional information available.

Input handling

Assign input variables to computation variables and check if they are consistant.

```
switch nargin

    case 3
        n = arg1;    % Zernike degree
        m = arg2;    % Zernike order
        option = arg3;     % string defining which algorithm to use

    case 2
        n = arg1;    % Zernike degree
        m = arg2;    % Zernike order
        option = 'DEFAULT';     % string defining which algorithm to
use

    otherwise
        error('Wrong number of input arguments!')
end

if numel(n(n<0))>0 || numel(n(~(round(n)==n)))>0
    error('Input arg1 should be a vector with positive integers')
elseif numel(m) ~= numel(n) || numel(m(abs(m)>n))>0 ||
numel(m(~(round(abs(m))==abs(m))))>0
    error('Input arg2 should be of same lentgh as arg1, and |arg2| >=
arg1 for all elements')
elseif numel(n((rem(n+abs(m),2)==1)))
    error('Wrong combination of n and m, (n + m) should be even!')
end
```

Function Part1: Convert n and m to FRINGE index

```
switch upper(option)

    case 'DEFAULT'
        p = (n+abs(m))./2;
        index = p.^2 + n - abs(m) + 1 + (m<0);

    case 'LEGACY'
        index = 0*n;
        for t = 1:numel(n)
            index(t) = max(cumsum(1:2:(n(t)+abs(m(t))+1)))
-(2*abs(m(t)))+((m(t)<0));
        end

    otherwise
        error('Unknown "option" provided as second argument!')
end
```

Output handling

Sent the desired output data to the output variables

```
out = index;   % Zernike degree
```

B.10. Pupil2Zernike

```
Represent a given pupil distribution (scalar) as a Zernike expansion

This function is used to generate the Zernike expansion coefficients
(Beta's) representing a given (scalar) pupil distribution. It thus
provides the inverse operation of the function "Zernike2Pupil.m". For
more info see Eq.(3.10) subsection 3.2.1 of Ph.D. Thesis Sven van Haver
(http://www.nijboerzernike.nl/\_PDF/Thesis\_S\_vanHaver\_optimized.pdf)

Note that for vector distributions in the pupil one can use this
function for every field component seperately.

Syntax:

     [Beta] = Pupil2Zernike(Pupil,RHO,THETA,nmax,mmax,option)

     [Beta] = Pupil2Zernike(Pupil,RHO,THETA,nmax,mmax)

     [Beta] = Pupil2Zernike(Pupil,RHO,THETA,kmax)

     [Beta] = Pupil2Zernike(Pupil,RHO,THETA,tol)

     [Beta] = Pupil2Zernike(Pupil,RHO,THETA)

Input:

Pupil = Field distribution in the pupil which should be represented as
        a Zernike expansion.

RHO =   Matrix with radial positions in circular coordinate space of
        all pupil positions for which the field distribution should be
        generated. [length normalized to pupil radius]

THETA = Matrix with azymuthal positions in circular coordinate space of
        all pupil positions for ehich the field distribution should be
        generated. [Radians]

nmax =  Maximum Zernike degree used in the expansion (Integer)
        [Dimensionless]
```

```
mmax =  Maximum Zernike order used in the expansion (Integer)
        [Dimensionless]

kmax =  Maximum single Zernike index used in the expansion (Integer)
        [Dimensionless]

option =  optional string (value = 'force one') that can be used to
          automatically rescale the amplitude, and shift the phase, such
          that the Zernike expansion representing the distribution has
          Beta^0_0 equal to One.

tol   = Maximum difference allowed between the input pupil field and
        the Zernike representation constructed by this function.

Output:

Beta =  Matrix of 3 dimensions containing the complex Zernike
        coefficients describing the input pupil distribution. Matrix
        element Beta(s,n+1,m+1) pertains to a Zernike polynomial
        of degree, n, and order, \ensuremath{|}m\ensuremath{|}.
Parameter s indicates the sign of
        parameter m, where s=1 corresponds to positive m and s=2 to
        negative m. [Radians]

Example(s):

     Generate a pupil with astigmatic aberration
         >> [RHO,THETA]=meshgrid(0:0.05:1,0:pi/10:2*pi);nmax=6;mmax=2;
         >> Pupil = exp(0.25i.*(RHO.^2));
         >> Beta=Pupil2Zernike(Pupil,RHO,THETA,nmax,mmax,'force one');
         >> squeeze(Beta(1,:,:))
     Output:
         ans =

              1.0000 - 0.0000i        0                  0
                   0             0.0000 - 0.0000i        0
              0.0000 + 0.1251i        0             -0.0000 + 0.0000i
                   0             0.0000 - 0.0000i        0
             -0.0052 + 0.0000i        0             -0.0000 + 0.0000i
                   0            -0.0000 + 0.0000i        0
             -0.0000 - 0.0001i        0              0.0000 - 0.0000i

Dependencies:
    Name:                      Version:    Author:         Toolbox:
    Sol2Beta.m                 1.1         S. van Haver    NONE
    Pupil\_inversion\_Matrix.m    1.1         S. van Haver    NONE

Author: Dr. Ir. S. van Haver, Axel Wiegmann
Email:  svenvanhaver@gmail.com, axel.wiegmann@web.de

Project: Extended Nijboer-Zernike (ENZ) theory
Keywords: Pupil fitting, Zernike expansion, Transmission function,
          Pupil representation, Zernike coefficients

Version: v2.0 (26-07-2012)
```

Change-log

v1.0 (19-06-2010) * First operational version. v1.1 (27-09-2010) * Function has been fully documented v1.2 (16-11-2010) * Allow function to run without specification of nmax and mmax v1.3 (13-12-2010) * NaN values in the Input values are ignored v2.0 (26-07-2012) * Added the option to specify the upper Zernike single index used in the expansion

Function declaration

```
function [Beta] = Pupil2Zernike(varargin)
```

Function description

This m-file calculates the Beta coefficients that represent the input pupil using a Least-Square fit.

[Beta] = Pupil2Beta(Pupil,r,th,nmax,mmax)

By: S. van Haver (May 2010).

Adjusted 07/2010 to force Beta(1,1,1) to be One!

Input handling

Assign input variables to computation variables and check if they are consistant.

```
switch nargin

    case 6

        % Assign arguments to calculation variables:
        Pupil = cell2mat(varargin(1));    % Input (scalar) pupil
distribution
        RHO = cell2mat(varargin(2));      % Radial coordinate of pupil
points
        THETA = cell2mat(varargin(3));    % Azymuthal coordinate of
pupil points
        nmax = cell2mat(varargin(4));     % Maximum Zernike degree used
in the
        % expansion
        mmax = cell2mat(varargin(5));     % MAximum Zernike order used
in the
        % expansion
        option = cell2mat(varargin(6));   % Optional parameter
indicating if
        % distribution should be scaled in order to
        % have Beta^0_0 == 1. (either: 'no scaling'
        % or 'force one')

    case 5

        % Assign arguments to calculation variables:
        Pupil = cell2mat(varargin(1));  % Input (scalar) pupil
distribution
        RHO = cell2mat(varargin(2));    % Radial coordinate of pupil
points
        THETA = cell2mat(varargin(3));  % Azymuthal coordinate of pupil
points
        nmax = cell2mat(varargin(4));   % Maximum Zernike degree used
in the
                                        % expansion
        mmax = cell2mat(varargin(5));   % MAximum Zernike order used in
the
                                        % expansion
        option = 'no scaling';          % Optional parameter set to
Default:
                                        % NO SCALING

    case 4

        % Assign arguments to calculation variables:
        Pupil = cell2mat(varargin(1));    % Input (scalar) pupil
distribution
```

```
        RHO = cell2mat(varargin(2));      % Radial coordinate of pupil
points
        THETA = cell2mat(varargin(3));    % Azymuthal coordinate of
pupil points
        arg4 = cell2mat(varargin(4));
        if arg4 < 1
            tol = arg4;                   % Maximum Error tolerance in
the Zernike
                                          % fit of the pupil
distribution
        else
            kmax = arg4;                  % Maximum Zernike single
index to
        end                               % use in the reconstruction

        option = 'no scaling';            % Optional parameter set to
Default:
                                          % NO SCALING

    case 3

        % Assign arguments to calculation variables:
        Pupil = cell2mat(varargin(1));    % Input (scalar) pupil
distribution
        RHO = cell2mat(varargin(2));      % Radial coordinate of pupil
points
        THETA = cell2mat(varargin(3));    % Azymuthal coordinate of
pupil points
        tol = 1e-3;                       % Default Error tolerance in
the Zernike
                                          % fit of the pupil
distribution
        option = 'no scaling';            % Optional parameter set to
Default:
                                          % NO SCALING

    otherwise

        error('Wrong number of input arguments!!!')

end

% Check consistancy:
if nargin>4
    if nmax<mmax
        error('Nmax should always be equal or larger than mmax!')
    end
end

if (numel(RHO)~=numel(THETA)) || (numel(Pupil)~=numel(RHO))
    error('Number of elements in E, RHO and Theta are not consistant!')
end

% Remove all points where a coordinate or the pupilvalue is NaN and
convert
% input parameters to vectors
RHO=RHO(:);
THETA=THETA(:);
Pupil=Pupil(:);
idx=isnan(RHO+THETA+Pupil);
RHO(idx)=[];
```

```
THETA(idx)=[];
Pupil(idx)=[];

if nargin > 4 || exist('kmax')
```

Function Part1: generate fitting matrix

We start by generating a linear system of equations, one for every point in the pupil, that upon solving provides the best-fit Zernike expansion in a Least-Square (LS) sense.

```
    % Generate fitting matrix:
    if nargin > 4
        [LSFM] = Pupil_inversion_Matrix(RHO,THETA,mmax,nmax);
    else
        [LSFM] = Pupil_inversion_Matrix(RHO,THETA,kmax);
    end
```

Function Part2: determine Zernike expansion

Solve linear system of equation and write solution vector in the formar of a standard set of Zernike coefficients.

```
    % Solve lineair system:
    Sol= LSFM\Pupil;

    % Write solution as Beta's:
    %Beta=Sol2Beta(mmax,nmax,Sol);
    Beta=Sol2Beta(Sol);

    % Generate temp pupil from Beta's
    Pupiltemp = Zernike2Pupil(Beta,RHO,THETA);

    % Determine maximum error
    toltemp = max(abs(Pupil-Pupiltemp));
```

Function Part3: Rescale results

If the option 'force one' is specified, we rescale our input Pupil distribution such that the reconstructed Beta^0_0 Zernike coefficients has value One.

```
    if strcmp('force one',option)

        % Scale Pupil and again solve linear system:
        Sol= LSFM\(Pupil./Beta(1,1,1));

        % Write solution as scaled Beta's
        Beta=Sol2Beta(Sol);

    end

else
```

Function Part4: Adaptive Zernike fit

When no upper bound for the Zernikes is specified, increase kmax step by step until the resulting Zernike expansion satisfies the specified or default fitting tolerance.

```
    % Set initial values for conditional loop
    toltemp = 99;
    nmax=0;

    % Determine Zernike expansion iteratively
    while tol < toltemp

        % Adjust adaptive parameters:
        nmax=nmax+2;
        kmax = NM2ZernikeNumber(nmax,0);

        % Generate fitting matrix:
        [LSFM] = Pupil_inversion_Matrix(RHO,THETA,kmax);

        % Solve lineair system:
        Sol= LSFM\Pupil;

        % Write solution as Beta's:
        Beta=Sol2Beta(Sol);

        % Generate temp pupil from Beta's
        Pupiltemp = Zernike2Pupil(Beta,RHO,THETA);

        % Determine maximum error
        toltemp = max(abs(Pupil-Pupiltemp));


    end
```

Function Part5: remove non-significant coefficients

Set all Beta coefficients that are too small to contribute to zero, and adjust the min and max size of the 3D Beta matrix accordingly

```
    % Only execute this part of the function when number of coeffients
is
    % variable


    % Set all coefficients too small to contribute to zero
    Beta(abs(Beta)<tol/10)=0;

    % check if dimensions of Beta can be reduced (Remove dimensions
    % containing only zeros
    redo=1;
    while redo==1 % Repeat while redo==1

        redo=0;

        % Check if mmax can be reduced by one
        temp1=squeeze(max(max(abs(Beta(:,:,:)))));

        % Check if nmax can be reduced by one
        temp2=squeeze(max(max(abs(permute(Beta(:,:,:),[1 3 2])))));

        % reduce mmax contained in Beta by one
        if (temp1(end)<tol/10) && (size(Beta,3)>2)
            Beta=squeeze(Beta(:,:,1:end-1));
            redo=1;
        end

        % reduce nmax contained in Beta by one
        if temp2(end)<tol/10 && (size(Beta,2)>2)
            Beta=squeeze(Beta(:,1:end-1,:));
            redo=1;
        end
    end

end
```

Output handling

Sent the desired output data to the output variables

```
% Provide exactly the subset of coefficients that was requested
if nargin > 4
        Beta = Beta(:,1:nmax+1,1:mmax+1);
end

% Provide output to user:
%display(['The maximum error between ' inputname(1) ' and its Zernike
fit is: ' num2str(toltemp)])

% The Output Beta is a Matrix of 3 dimensions containing the complex
% Zernike coefficients describing the input pupil distribution.
% Matrix element Beta(s,n+1,m+1) pertains to a Zernike
% polynomial of degree, n, and order, |m|. Parameter s
% indicates the sign of parameter m, where s=1 corresponds
% to positive m and s=2 to negative m. [Radians]
```

B.11. IterativeIntegration

```
%----modified version of the one given by Ruud Bokdam, Leendert van
Veen,
%---Ike Mulder, Joost Wooning and Michel van der Kaay
% Calculates waveFront in the same way the SHS does, using iterative
% integration.
%
% Syntax:
%
% * [waveFront] = iterativeIntegration(datax, datay, xVals, yVals)
% * [waveFront] = iterativeIntegration(datax, datay, xVals, yVals,
errorMargin)
%
% Input:
%
% * datax        = matrix specifying x derivatives at positions
% * datay        = matrix specifying y derivatives at positions
% * xVals        = x components of positions
% * yVals        = y components of positions
% * (errorMargin)= optional argument, specifying the maximum absolute
total
%                  difference between waveFront and the optimal solution
%
% Output:
%
% * waveFront    = matrix containing function values at positions, whose
x
%                  and y derivatives should match datax and datay
respecively
```

Function declaration

```
function [waveFront] = iterativeIntegration(datax, datay, xVals, yVals,
errorMargin)
if (nargin < 5)
    errorMargin = 10^(-3);
end
```

```
% synchronize NaN values for safety
[datax, datay] = syncNans(datax, datay);
[ny, nx] = size(datax);

% Creating appropriate scaling factor
P = xVals(2) - xVals(1);

% To get the value of higher x/y, we need to circshift back
% Rows are y-coordinates
yUp = [-1, 0];
yDown = [1,0];
% Columns are x-coordinates
xUp = [0, -1];
xDown = [0,1];

% We need to store which values are relevant
hasValue = ~isnan(datax);
hasXUp = circshift(hasValue,xUp);
hasXDown = circshift(hasValue,xDown);
hasYUp = circshift(hasValue,yUp);
hasYDown = circshift(hasValue,yDown);

% circshift causes an unwanted jump, which we manually remove
hasXUp(:,nx)=false;
hasXDown(:,1)=false;
hasYUp(ny,:)=false;
hasYDown(1,:)=false;

% We also need to know how many relevant values each position has
divisor = (hasXUp+hasXDown+hasYUp+hasYDown).*hasValue;

% Starting the iteration
waveFrontPrev = zeros(ny,nx);
waveFrontPrev(~hasValue) = NaN;
a1=P/2*(circshift(datax,xUp)+datax);
a2=P/2*(circshift(datax,xDown)+datax);
a3=P/2*(circshift(datay,yUp)+datay);
a4=P/2*(circshift(datay,yDown)+datay);
while true
    % Each adjacent position's value and corresponding derivative are
an
    % estimate for the actual value
    estimXUp = circshift(waveFrontPrev,xUp) - a1;
    estimXDown = circshift(waveFrontPrev,xDown) + a2;
    estimYUp = circshift(waveFrontPrev,yUp) - a3;
    estimYDown = circshift(waveFrontPrev,yDown) + a4;

    % Filtering out the irrelevant values (mostly NaNs)
    estimXUp(~hasXUp) = 0;
    estimXDown(~hasXDown) = 0;
    estimYUp(~hasYUp) = 0;
    estimYDown(~hasYDown) = 0;

    % Making the estimate using all relevant values
    waveFront =(estimXUp + estimXDown + estimYUp + estimYDown)./divisor;

    % Checking break conditions
    if(nansum(nansum(abs(waveFront-waveFrontPrev))) < errorMargin)
        break;
    end
    if(exist('waveFrontPrevPrev', 'var'))
```

```
        if(nansum(nansum(abs(waveFront-waveFrontPrevPrev))) <
errorMargin)
            break;
        end
    end

    % Shifting values
    waveFrontPrevPrev = waveFrontPrev;
    waveFrontPrev = waveFront;
end
end
```

Bibliography

[1] A. J. E. M. Janssen, *Zernike expansion of derivatives and Laplacians of the Zernike circle polynomials,* Journal of the Optical Society of America A **31**, 1604 (2014).

[2] F. Zernike and F. Stratton, *Diffraction Theory of the Knife-Edge Test and its Improved Form, The Phase-Contrast Method,* Monthly Notices of the Royal Astronomical Society **94**, 377 (1934).

[3] Nobel Media AB 2014, *The Nobel Prize in Physics 1953,* https://www.nobelprize.org/nobel_prizes/physics/laureates/1953/ (2016).

[4] A. B. Bhatia and E. Wolf, *On the circle polynomials of Zernike and related orhtogonal sets,* Mathematical Proceedings of the Cambridge Philosophical Society **50**, 40 (1954).

[5] MathWorks, *Four-quadrant inverse tangent - MATLAB atan2,* http://nl.mathworks.com/help/matlab/ref/atan2.html (2016).

[6] J. C. Wyant and K. Creath, *Basic Wavefront Aberration Theory for Optical Metrology,* Applied Optics and Optical Engineering **11**, 2 (1992).

[7] OPTOCRAFT, *SHSLab Technical manual* (2013).

[8] W. Lukosz, *Der einfluÃ der aberrationen auf die optische Ãbertragungsfunktion bei kleinen orts-frequenzen,* Optica Acta: International Journal of Optics **10**, 1 (1963), http://dx.doi.org/10.1080/713817744 .

[9] *Figure of Michelson interferometer,* https://skullsinthestars.files.wordpress.com/2008/09/michelson.jpg (2016).

[10] HAMAMATSU, *Phase spatial light modulator LCOS-SLM* (2015), available at https://www.hamamatsu.com/resources/pdf/ssd/e12_handbook_lcos_slm.pdf.

[11] A. Alfalou and C. Brosseau, *Optical image compression and encryption methods,* Adv. Opt. Photon. **1**, 589 (2009).

[12] N. Kumar, L. Cisotto, S. Roy, G. K. P. Ramanandan, S. F. Pereira, and H. P. Urbach, *Determination of the full scattering matrix using coherent Fourier scatterometry,* Applied Optics **55**, 4408 (2016).

[13] J. Schwider, R. Burow, K.-E. Elssner, J. Grzanna, R. Spolaczyk, and K. Merkel, *Digital wave-front measuring interferometry: some systematic error sources,* Applied Optics **22**, 3421 (1983).

[14] M. A. Herráez, D. R. Burton, M. J. Lalor, and M. A. Gdeisat, *Fast two-dimensional phase-unwrapping algorithm based on sorting by reliability following a noncontinuous path,* Appl. Opt. **41**, 7437 (2002).