ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

ECOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

MASTER THESIS

Evaluation of Optical Aberrations using Phase Diversity

Author:
Jordan VOIRIN

Supervisors:
Dr. Laurent JOLISSAINT
Dr. Jean-Paul KNEIB

A thesis submitted in fulfillment of the requirements for the degree of Master in Applied Physics

in the

Astrophysics laboratory
Basic Sciences

March 14, 2018

Declaration of Authorship

I, Jordan VOIRIN, declare that this thesis titled, "Evaluation of Optical Aberrations using Phase Diversity" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"For the sake of persons of ... different types, scientific truth should be presented in different forms, and should be regarded as equally scientific, whether it appears in the robust form and the vivid coloring of a physical illustration, or in the tenuity and paleness of a symbolic expression. "

James Clerk Maxwell

ECOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Abstract

Physics
Basic Sciences

Master in Applied Physics

Evaluation of Optical Aberrations using Phase Diversity

by Jordan VOIRIN

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too...

Acknowledgements

The acknowledgements and the people to thank go here, don't forget to include your project advisor...

Contents

List of Figures

List of Tables

Introduction

This master thesis treats of one of the major challenge in astronomy and in optics in general, the aberrations and their corrections. We will study and develop a method to identify the static aberrations present in an optical system using the data produced by it.

Ground based observation is as old as the first men. But since the Galilei Galileo telescope in 1609, the resolution power of the telescopes is far greater. Many inventions lead to the construction of the largest telescope ever built, the Very Large Telescope (VLT). Now that we are able to build enormous telescopes with mirror diameter up to 8m, soon to be bitten by the Extremely Large Telescope (ELT) with its 18m, the limitation of the resolution is not the diameter anymore, but the atmosphere. Indeed, the turbulence present in the atmosphere creates a turbulent distribution of refractive index due to the distribution of temperature principally. This alters the optical path of the light through the Earth's atmosphere which finally deteriorates the images given by the telescope.

The first system to correct aberrations in telescopes is the active optics system. It consisted of actuators placed under the telescope mirrors capable of correcting the telescope deformation under the gravity effect, vibration, etc... Those systems are not fast enough to correct for the turbulence but they already improve the images. The state-of-the-art technique is the adaptive optics, developed to take into account the effects of the atmosphere. This technique characterizes the aberrations present in the wavefront and correct them in real time. The schema of principle is shown in Figure 1. It uses a wavefront sensor to characterize its deformation and then passes the information to a deformable mirror to correct it, it is called an adaptive optic loop. It works well, but the drawback of this method is that it does not correct the aberrations up to the scientific detector. Those uncorrected aberrations are called Non-Common Path Aberration (NCPA), they are generally static or slowly evolving over time. The adaptive optic also requires a complex optical system before the scientific detector. The beam is split in two. One arm goes to the scientific detector and the other is used in the adaptive optic loop.

We present a method called phase diversity that uses images of a scientific detector to deduct the aberration present in the wavefront. In our case, we work on the static aberrations that deform the wavefront. The phase diversity was first introduced by Gonsalves (1982). The idea is to use one focused and one defocused image to retrieve the aberrations. Two images at least are needed due to the property of light acquisition. There is an indetermination, because the detector is only sensitive to the intensity of light and not the light wave itself. It is raised adding a phase diversity. Unlike adaptive optic system, the phase diversity requires nearly no additional optical component depending on how it is implemented. The final aim of this project would be to integrate the developed phase diversity into the design of the DAG telescope in Turkey, see Figure 2.

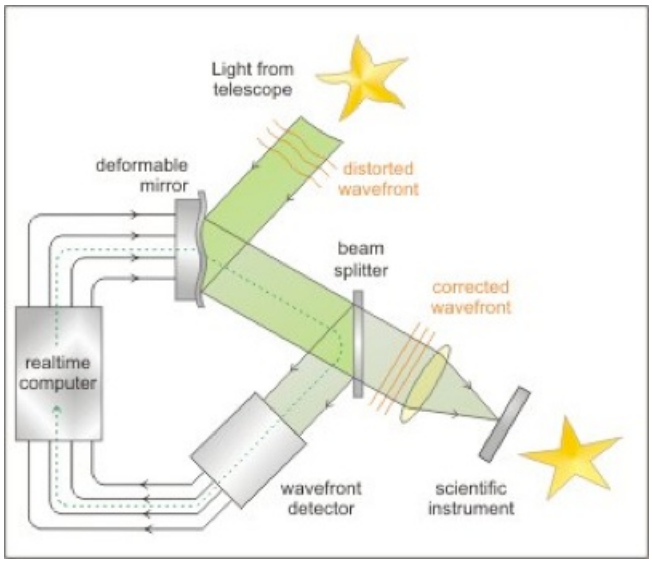


FIGURE 1: Adaptive optics schema
source : http://www.bo.astro.it/ter5/ter5/ao.html.

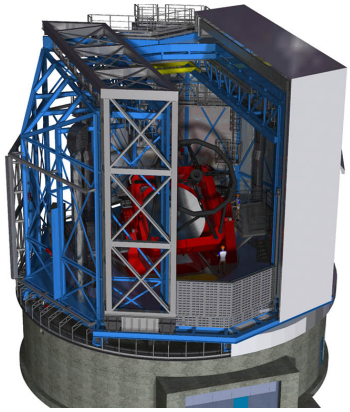


FIGURE 2: DAG 4m telescope
source : http://www.eie.it/en/progetti/dag-dogu-anadolu-gozlemevi.

This report is organized as following. First, a review of the necessary theoretical background is reminded, going from the scalar theory of diffraction to the description of phase retrievals methods. Then we present an experiment performed at the optic laboratory at HEIG-VD. The ONERA (Office National d'Etudes et de Recherches Aerospatiales) algorithm of phase diversity is tested and used in comparison with a Shack-Hartmann wavefront sensor. And finally, the development of an analytical phase diversity algorithm is presented and tested with simulated Point Spread Functions (PSFs).

Chapter 1

Theoretical background

In this chapter, we present the theory upon which this work is based. First, the light propagation formalism is reminded through the scalar diffraction theory based on the Goodman (1988). Then we describe in general an imaging system and its properties. And finally we discuss the wavefront aberration theory, and correction method to identify and correct the aberrations.

1.1 Scalar Diffraction Theory

1.1.1 Scalar Field and Helmholtz equation

A monochromatic wave, at position P and time t, can be represented by a scalar field $u(P,t)$ written as :

$$u(P,t) = A(P)exp\left[-j2\pi\nu t - j\phi(P)\right], \tag{1.1}$$

where $A(P)$ and $\phi(P)$ are the amplitude and phase, respectively, of the wave at position P and ν is the wave frequency.

The spatial part of eqt. (1.1), also called phasor in the literature,

$$U(P) = A(P)e^{-j\phi(P)}, \tag{1.2}$$

must verify the Helmotz equation :

$$(\nabla^2 + k^2)U = 0, \tag{1.3}$$

where k is the wave number given by

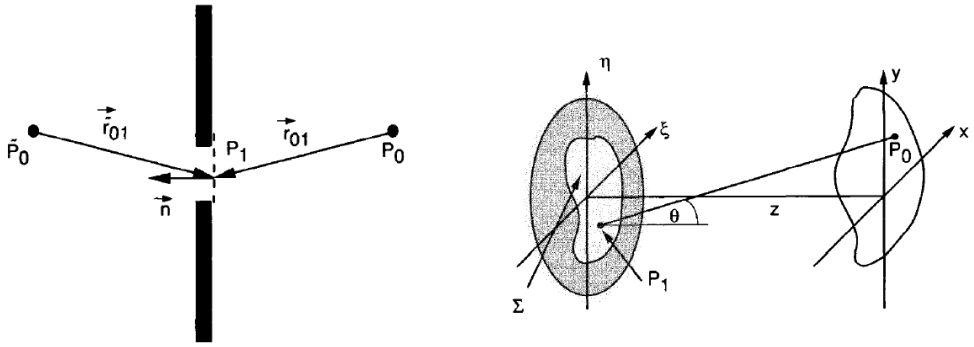
$$k = 2\pi n\frac{\nu}{c} = \frac{2\pi}{\lambda}, \tag{1.4}$$

and λ is the wavelength in the dielectric medium.

1.1.2 Rayleigh-Sommerfeld integral

Rayleigh and Sommerfeld developed a formalism using the Helmholtz equation and Green's Theorem to compute the induced diffraction by a plane screen. Let's suppose that we have a monochromatic source at $\widetilde{P_0}$ on the left of a plane screen with aperture Σ, the Rayleigh-Sommerfeld formula allows to compute the complex amplitude at P_0 on the right of the plane screen (see Figure 1.1a).

$$U(P_0) = \frac{1}{j\lambda}\iint_{\Sigma} U'(P_1)\frac{exp(jkr_{01})}{r_{01}}cos(\mathbf{n}, \mathbf{r_{01}})\mathbf{ds} \tag{1.5}$$



(A) Rayleigh-Sommerfeld formulation of diffraction by a plane screen, (Goodman, 1988, Chapter 3.5).

(B) Diffraction geometry, (Goodman, 1988, Chapter 4.1).

FIGURE 1.1: Diffraction Schemas

$U'(P_1)$ is the complex amplitude on the screen, $cos(\mathbf{n}, \mathbf{r_{01}})$ is the cosine of the angle between the aperture plane normal toward the source and the vector $\mathbf{r_{01}} = \mathbf{P_0P_1}$ given by

$$r_{01} = \sqrt{z^2 + (x-\xi)^2 + (y-\eta)^2}. \tag{1.6}$$

We can rewrite eqt. (1.5) using $cos(\mathbf{n}, \mathbf{r_{01}}) = cos(\theta) = \frac{z}{r_{01}}$ and the coordinate systems (ξ, η) and (x, y), see Figure 1.1b,

$$U(x,y) = \frac{z}{j\lambda} \iint_{-\infty}^{\infty} U(\xi,\eta) \frac{exp(jkr_{01})}{r_{01}^2} d\xi d\eta. \tag{1.7}$$

We can integrate from $-\infty$ to ∞, using $U(\xi,\eta) = P(\xi,\eta)U'(\xi,\eta)$ where $P(\xi,\eta)$ is the pupil function. The latter equals to one in the pupil and zero outside.

1.1.3 Fresnel approximation

To reduce eqt. (1.7), also known as the Huygens-Fresnel principle, one can approximate the distance r_{01} using the taylor expansion of the square root :

$$r_{01} = z\sqrt{1 + \frac{x-\xi}{z} + \frac{y-\eta}{z}} \approx z\left[1 + \frac{1}{2}\left(\frac{x-\xi}{z}\right)^2 + \frac{1}{2}\left(\frac{y-\eta}{z}\right)^2\right] \tag{1.8}$$

To obtain the Fresnel approximation, one has to replace r_{01} by eqt. (1.8) in eqt. (1.7). At the denominator, only the first term z is kept, since the introduced error is small, but in the exponential everything is kept. Then the final expression is given by,

$$U(x,y) = \frac{e^{jkz}}{j\lambda z} \iint_{-\infty}^{\infty} U(\xi,\eta) exp\left\{j\frac{k}{2z}\left[(x-\xi)^2 + (y-\eta)^2\right]\right\} d\xi d\eta. \tag{1.9}$$

In this form, the Fresnel approximation can be seen as a convolution between $U(\xi,\eta)$ and $h(x,y) = \frac{e^{jkz}}{j\lambda z} exp\left[\frac{jk}{2z}\left(x^2+y^2\right)\right]$.

Another form is found by developing $\left[(x-\xi)^2 + (y-\eta)^2\right]$,

$$U(x,y) = \frac{e^{jkz}}{j\lambda z} e^{j\frac{k}{2z}(x^2+y^2)} \iint_{-\infty}^{\infty} \left\{ U(\xi,\eta) e^{j\frac{k}{2z}(\xi^2+\eta^2)} \right\} e^{-j\frac{2\pi}{\lambda z}(x\xi+y\eta)} d\xi d\eta, \tag{1.10}$$

it is the Fourier transform of the complex field in the pupil multiplied by a quadratic phase exponential.

1.1.4 Fraunhofer approximation

In addition to the Fresnel approximation, we can introduce another approximation using the condition,

$$z >> \frac{k(\xi^2+\eta^2)_{max}}{2}. \tag{1.11}$$

If eqt. (1.11) is satisfied the Fresnel approximation simplifies, since the quadratic phase factor in (ξ,η) is approximately one on the entire pupil, as

$$U(x,y) = \frac{e^{jkz}}{j\lambda z} e^{j\frac{k}{2z}(x^2+y^2)} \iint_{-\infty}^{\infty} U(\xi,\eta) e^{-j\frac{2\pi}{\lambda z}(x\xi+y\eta)} d\xi d\eta. \tag{1.12}$$

For instance, at a wavelength of 637.5 nm and a pupil diameter of 3.6 mm the Fraunhofer approximation constrain z to be greater than 63 meters to be valid.

1.1.5 Converging lens introduction

The Fraunhofer conditions are severe as shown above, but one can reduce the distance z by observing at the focal plane of a converging lens. Indeed, using the paraxial approximation, i.e. small angles with respect to the optical axis, the lens transmission function is given by,

$$\begin{aligned} t_l(\xi,\eta) &= exp\left[jkn\Delta_0\right] exp\left[-jk\left(n-1\right)\frac{\xi^2+\eta^2}{2}\left(\frac{1}{R_1}-\frac{1}{R_2}\right)\right] \\ &= exp\left[-j\frac{k}{2f}(\xi^2+\eta^2)\right], \end{aligned} \tag{1.13}$$

where n is the refractive index of the lens material, R_1 and R_2 are the radii of curvature of the front and back surface of the lens, respectively, and f is the focal length of the lens defined as,

$$\frac{1}{f} \equiv (n-1)\left(\frac{1}{R_1}-\frac{1}{R_2}.\right) \tag{1.14}$$

We can define $U_l(\xi,\eta) = U(\xi,\eta)t_l(\xi,\eta)$, which represents the complex amplitude passing through a lens. Finally, replacing $U(\xi,\eta)$ by $U_l(\xi,\eta)$ in Fresnel approximation and setting the observing distance to the focal length of the converging lens, we recover the Fraunhofer approximation,

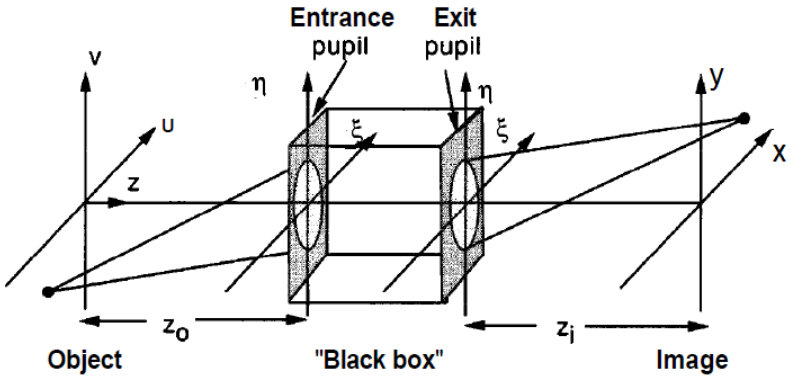


FIGURE 1.2: Schema of a imaging instrument, (Goodman, 1988, Chapter 6.1)

$$U(x,y) = \frac{e^{jkz}}{j\lambda z} e^{j\frac{k}{2z}(x^2+y^2)} \mathcal{F}\left\{ U(\xi,\eta) exp\left[j\frac{k}{2}(\xi^2+\eta^2)(\frac{1}{z}-\frac{1}{f}) \right] \right\}$$
$$\overset{z=f}{=} \frac{e^{jkz}}{j\lambda z} e^{j\frac{k}{2z}(x^2+y^2)} \mathcal{F}\left\{ U(\xi,\eta) \right\}. \tag{1.15}$$

1.2 Imaging system

An imaging system, such as a telescope, is used to acquire images of an object as perfectly as possible. An optical system, forming an instrument, is composed by lenses, mirrors, etc... and a detector (can be the human eye). A complex optical system can be reduced to a pupil, $P(\xi,\eta)$, and a focal length, f. The diffraction of the wave can be determined by the Fraunhofer approximation as long as the paraxial approximation is valid, see subsection 1.1.5. And the observed image of an incoherent object at the focal plane of the system is proportional to the square modulus of the complex amplitude $U(x,y)$,

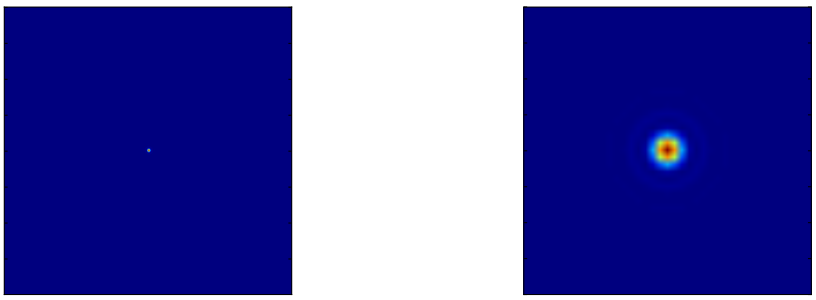
1.2.1 Impulse Response (IR)

The impulse response or point spread function (PSF), $h(x,y;u,v)$, of an optical system is the field amplitude induced at coordinates (x,y) by a unit-amplitude point source at object coordinates (u,v). An example of PSF for a incoherent point source object is shown in Figure 1.3. Using the linearity of the wave propagation, we can write the imaged amplitude as a superposition integral,

$$U(x,y) = \iint_{-\infty}^{\infty} h(x,y;u,v) U(u,v) du dv. \tag{1.16}$$

1.2.2 Optical Transfer Function (OTF)

The optical transfer function, OTF, is defined as the Fourier transform of the impulse response, an example for an incoherent point source is shown in Figure 1.4 and it correspond to the OTF of the PSF shown in Figure 1.3. As we will use it later, for



(A) PSF (B) PSF zoom x10

FIGURE 1.3: PSF of a perfect imaging system composed by a 3.6 mm pupil and a focal length of 80 mm at a wavelength of 637.5 nm. The size, N, of the PSF is 400 and the pixel size is 5.3 μm.

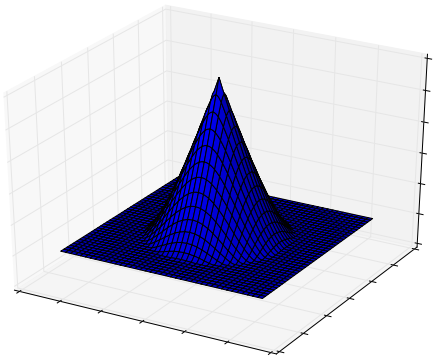


FIGURE 1.4: OTF of a perfect imaging system composed by a 3.6 mm pupil and a focal length of 80 mm at a wavelength of 637.5 nm.

incoherent illumination and using the Fourier transform properties it can also be given by the autocorrelation of the pupil function as we will see in section 1.2.3 that the impulse response is given by the Fourier transform of the squared pupil function,

$$\widetilde{h}_{optical}(\xi,\eta) = \mathcal{F}\left\{h_{optical}(x,y)\right\} = (P \otimes P)(\xi,\eta), \tag{1.17}$$

where ξ and η are the conjugate variables of x and y with respect to the Fourier transform.

1.2.3 Image formation

A detector only senses the energy distribution produced by an electromagnetic wave. Therefore, an image is given by the square modulus of the complex amplitude at the focal plane,

$$i(x,y) = |U(x,y)|^2 = |\iint_{-\infty}^{\infty} h(x,y;u,v)U(u,v)dudv|^2. \tag{1.18}$$

This integral simplifies differently depending on the type of object we are observing. For a **coherent object**, Goodman (1988, Chapter 6.2) showed that the imaging is linear in complex amplitude, thus eqt. (1.18) becomes,

$$i(x,y) = |\iint_{-\infty}^{\infty} h(x-\widetilde{u},y-\widetilde{v})U(\widetilde{u},\widetilde{v})d\widetilde{u}d\widetilde{v}|^2, \tag{1.19}$$

where $(\widetilde{u} = Mu, \widetilde{v} = Mv)$ are the normalized object coordinates and M is the magnification of the imaging system. The image is given by the squared modulus of the convolution of the impulse response and the object complex amplitude.

For an **incoherent object**, Goodman (1988, Chapter 6.2) showed that the imaging is linear in intensity,

$$i(x,y) = \iint_{-\infty}^{\infty} |h(x-\widetilde{u},y-\widetilde{v})|^2 o(\widetilde{u},\widetilde{v})d\widetilde{u}d\widetilde{v} = (h_{optical} \otimes o)(x,y), \tag{1.20}$$

where $o(x,y) = |U(x,y)|^2$. We can recognize the convolution of an object with an intensity impulse response, $h_{optical}(x,y) = |h(x,y)|^2$.

In this study, we will study stars radiation which are object with an incoherent emission. The stars are point source objects given there distances. The image that an optical system gives of a point source is called the point spread function, PSF, or impulse response, IR, of the system, see Figure 1.3. A point source is characterized by an infinite distance to the instrument and therefore the wave is planar, which means that the phasor is reduced to $U(\xi,\eta) = P(\xi,\eta)$. The PSF or IR is given by,

$$h_{optical}(x,y) = |\left[\mathcal{F}\left\{P(\xi,\eta)\right\}\right](x,y)|^2 \tag{1.21}$$

The domain where the impulse response is invariant under translation is called the **isoplanatic domain**.

In presence of aberrations, which will be discussed in section 1.3, the wavefront is deformed with respect to the perfect planar or spheric form. The pupil function at the exit of the imaging system is modified as following,

$$\mathcal{P}(\xi,\eta) = P(\xi,\eta)e^{-j\phi_{Ab}(\xi,\eta)}, \tag{1.22}$$

where $\phi_{Ab}(\xi, \eta)$ is the dephasing caused by the aberration present between the object and the image planes. Replacing the new pupil function in eqt. (1.21), we obtain the PSF of an imaging system having aberrations on the optical path.

1.3 Aberrations

The aberrations present on the optical path between the object and the image plane decrease the quality of the resulting image. Indeed, they induce fluctuations of the amplitude and phase of the wave in the pupil plane. Since the amplitude fluctuations are negligible with respect to the phase fluctuations, we focus only on the latter. In figure 1.5, one can see the effect of the aberrations on an hypothetical spherical wavefront at the exit pupil of an imaging system. And in Figure 1.6, one can see the effect of aberrations on the PSF of an imaging system. The PSF is blurred due to the defocus and we can well recognize the astigmatism introduced as the shape of the PSF tends towards the ellipse. The Strehl ratio is often used to quantify the importance of the aberrations, it is given by the following expression,

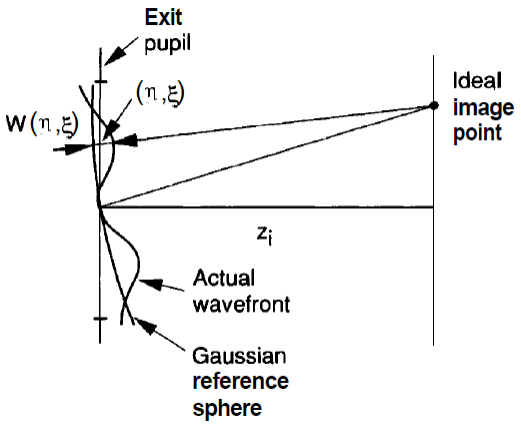


FIGURE 1.5: Gaussian reference sphere vs. aberrated Wavefront, (Goodman, 1988, Chapter 6.4). The dephasing is equal to the wave number multiplied by the aberration function, $\phi_{Ab}(\xi, \eta) = kW(\xi, \eta)$.

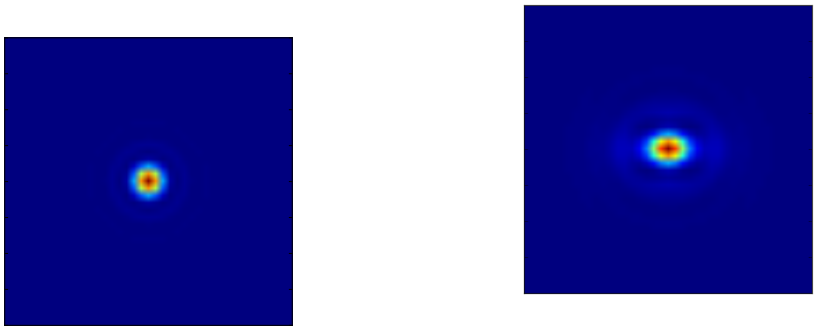
$$SR = \frac{\int \widetilde{h}_{optical}(\xi, \eta) \mathrm{d}\xi \mathrm{d}\eta}{\int \widetilde{h}_{perfect}(\xi, \eta) \mathrm{d}\xi \mathrm{d}\eta}, \tag{1.23}$$

where $\widetilde{h}_{perfect}(\xi, \eta)$ is the OTF of the perfect system with the same pupil as the optical system with the OTF $\widetilde{h}_{optical}(\xi, \eta)$.

1.3.1 Sources of aberration

The phase fluctuations introduced by aberrations are due to different kind of perturbations on the optical path. In ground based astronomy, the main source of aberrations is the **atmosphere**. Indeed, the atmosphere's temperatures fluctuations coupled with turbulent airflows are at the origin of the variations of the refractive index. Those variations induce then different optical paths, in other words they generate perturbations on an electromagnetic wave passing through the atmosphere. The theory of the Atmospheric Turbulence is beyond the scope of this work, but we redirect the interested reader to Obukhov (1949), V.I. Tatarski (1961), and Kolmogorov (1968).

The other source of aberrations are the **defects of the instrument** itself. The defaults can limit the resolution and decrease the quality of a diffraction-limited imaging system. The defects can have multiple origins as described by Blanc (2002). The first one is a **default during the fabrication process**, such as mirror polishing. This kind of defect is constant over time and of high spatial frequency. Perturbation of the



(A) perfect PSF (zoomed x10)

(B) PSF with aberrations (a_j = 50nm for a_4, $a_6$and a_{11}, see subsec 1.3.2 for the definition of a_j, zoomed x10)

FIGURE 1.6: Comparison of perfect PSF and PSF with aberrations of an imaging system composed by a 3.6 mm pupil and a focal length of 80 mm at a wavelength of 637.5 nm. The size, N, of the PSF is 400 and the pixel size is 5.3 μm.

electromagnetic wave can also be due to **misalignment of the optical components of the instrument**. For instance, during the first stages of operation of the Hubble telescope, the mirrors were not correctly aligned and the resulting images were blurry. These defects are of low spatial frequencies and vary slowly over time. Finally, there are defects due to **mechanical stresses of the instrument**. The mirrors have to be held in place by different mechanical components. And under the influence of the gravity, a mirror deformation can arise resulting in an aberration introduction. This kind of defect also evolves slowly through time but has a large spatial frequency domain.

We focus on a method to correct the effect of the instrument defects. The method is particularly suited since it does not require a different path than the path to the scientific instrument, which means that we can correct the aberrations on the entire optical path.

1.3.2 Zernike polynomials

In order to study the aberrations present in an imaging system, F. Zernike (1934) introduced an orthonormal basis on which we can decompose the phase on a circular pupil, such as a telescope pupil. Those polynomials are the product of a trigonometric function and a radial polynomial function (Noll, 1976).

$$Z_j(\mathbf{r}) = R_n^m(r)\Theta_n^m(\theta), \tag{1.24}$$

where $r = \frac{r'}{R_{pup}}$ is the normalized radius on the unit circle and θ is the azimuthal angle on the unit circle. j correspond to the Zernike index in the Noll order, a specific j corresponds only to one couple (n, m). The values of n and m satisfy the conditions $|m| \leq n$ and $n - |m| =$ even.

The trigonometric function is given by,

$$\Theta_n^m(\theta) = \begin{cases} \sqrt{n+1} & if\, m = 0, \\ \sqrt{2(n+1)}cos(m\theta) & \text{if } m \neq 0 \text{ and } i \text{ even}, \\ \sqrt{2(n+1)}sin(m\theta) & \text{if } m \neq 0 \text{ and } i \text{ odd}, \end{cases} \tag{1.25}$$

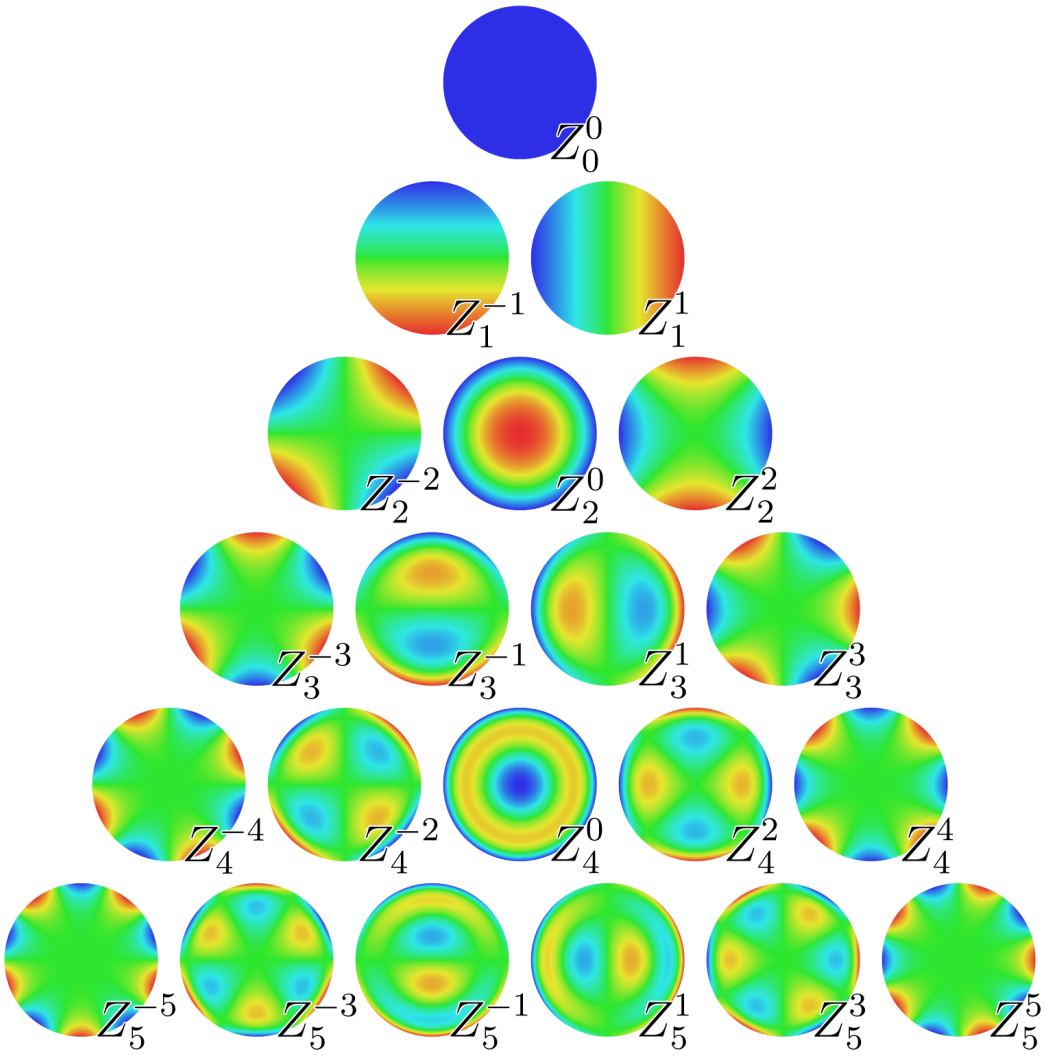


FIGURE 1.7: Representation of the 21^{st} Zernike polynomials (Wikipedia, 2018)

and the radial function is given by,

$$R_n^m(r) = \sum_{s=0}^{(n-m)/2} \frac{(-1)^s (n-s)!}{s![(n+m)/2-s]![(n-m)/2-s]!} r^{n-2s}. \tag{1.26}$$

The decomposition of the phase onto the Zernike polynomials is given by,

$$\phi(\mathbf{r}) = \sum_{i=1}^{+\infty} a_i Z_i(\mathbf{r}), \tag{1.27}$$

where the a_i's are the Zernike coefficient. And we characterize a wavefront by its root mean squared error (RMS error or RMSE) which is given by,

$$\sigma_\phi = \sqrt{\frac{1}{S}\int_S \phi^2(\mathbf{r}) d\mathbf{r}} = \sqrt{\sum_{i=2}^{+\infty} a_i^2}, \tag{1.28}$$

where S is the surface of the pupil.

1.4 Phase retrieval

The phase retrieval is a complicated process since the detectors are only sensitive to the intensity of a wave and not the wave itself, which prevents to obtain a direct relation between phase and image. Thus, we only have an indirect measurement of the wavefront. Nevertheless, it is important to be able to estimate the aberrations of a system in order to correct them in real-time (Adaptive Optics systems) or in post-processing (Image restoration).

There are a couple of ways to characterize the form of a wavefront, i.e. to determine the amplitude of the aberrations present in an imaging system. Some uses the optical geometric approximation, which says that locally the light rays are perpendicular to the wavefront. These achromatic methods measure the gradient of the wave surface, such as Shack-Hartmann (Hartmann, 1900; Shack and Platt, 1971; Fontanella, 1985), Curvature sensing (Roddier, 1988) and Pyramid sensing (Ragazzoni, 1996). Another kind of methods are called interferometric or focal plane methods. They use the interference patterns of the pupil to determine the form of the wavefront. The **Phase Diversity** is part of those kind of methods. It was first proposed by Gonsalves (1982). The most popular method nowadays are the Shack-Hartmann, the curvature and the phase diversity. We explain the principle of a Shack-Hartmann sensor and phase diversity in the two next subsections.

1.4.1 Shack-Hartmann (Hartmann, 1900; Shack and Platt, 1971)

The Shack-Hartmann wavefront sensor measures the gradient of an aberrant phase. Figure 1.8 shows the principle. A micro-lenses array samples the wavefront passing through the pupil in a conjugated plane of the entrance pupil. Each micro-lens produces a dot on a CCD placed at the foci of the micro-lenses. The deformations of the wavefront induce a displacement of the dot with respect to a reference position obtained with a perfectly planar or spherical wavefront. By measuring these displacements, Δx and Δy, it gives directly the local slope of the wavefront (Fontanella, 1985),

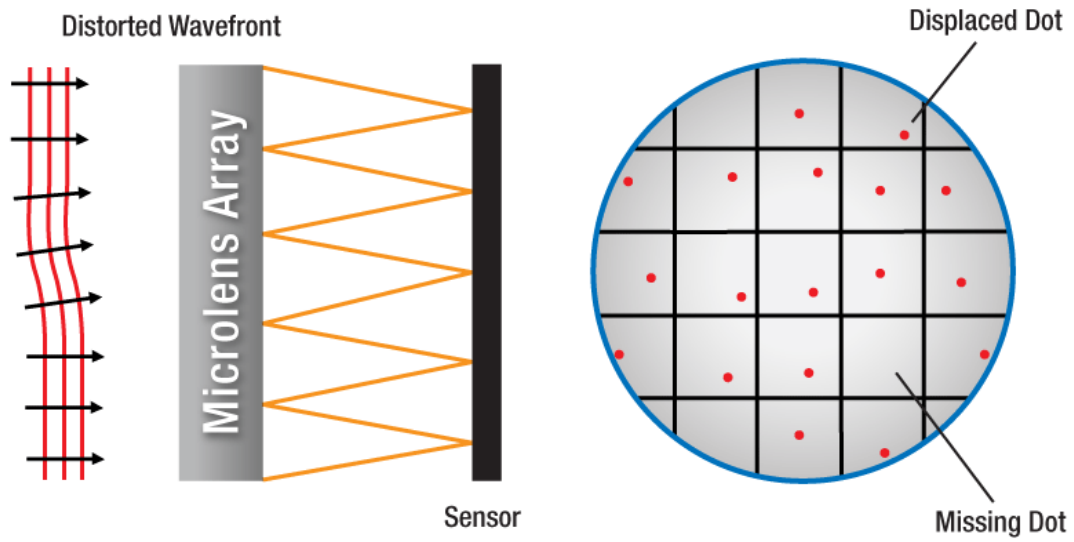


FIGURE 1.8: Shack-Hartmann principle (Thorlabs, 2018)

$$\frac{2\pi}{\lambda}\frac{\Delta s}{f} = \frac{1}{S}\iint_S \frac{\delta\phi(x,y)}{\delta s}dxdy, \tag{1.29}$$

where s is either x or y and S is the surface of a micro-lens and f is the focal length of the micro-lenses. The wavefront is reconstructed by integrating over all the local slope measurements. The advantage of this technique is that it does not require a lot of computation since it is a direct measurement. But it requires an important optical system to acquire the data. It is used in many fields, especially in adaptive optics system to correct for the atmospheric turbulence.

1.4.2 Phase Diversity

The phase diversity was first implemented by R. A. Gonsalves in 1976 (Gonsalves, 1976; Gonsalves, 1982) to retrieve the phase of a wavefront coming from a point source. It uses two images, one at the focal plane and another one with a diversity introduced, such as defocus, in order to recover the phase of the wavefront.

Unlike Shack-Hartmann wavefront reconstruction, which is a pupil plane technique, the phase diversity uses data acquired at the focal plane. Using the non-linear relation between the phase of the wavefront and the image,

$$i(x,y) = (h_{optical} \otimes o)(x,y),\ with\ h_{optical}(x,y) = |\left[\mathcal{F}\left\{A(\xi,\eta)e^{j\phi(\xi,\eta)}\right\}\right](x,y)|^2, \tag{1.30}$$

one can determine the phase, i.e. the aberrations present in the imaging system, by solving an inverse problem. The major difficulty of this technique is that, as one can see in eqt. (1.30), there is not a unique solution to the problem at hand. This indetermination comes from the fact that the available detectors can only sense the intensity of the wave, and not the wave itself, which is the modulus squared of the phasor as exposed in section 1.2. Thus, $\phi(\xi,\eta)$ and $\phi'(\xi,\eta) = -\phi(-\xi,-\eta)$ give the same PSF. More specifically by decomposing the phase in its even and odd part and

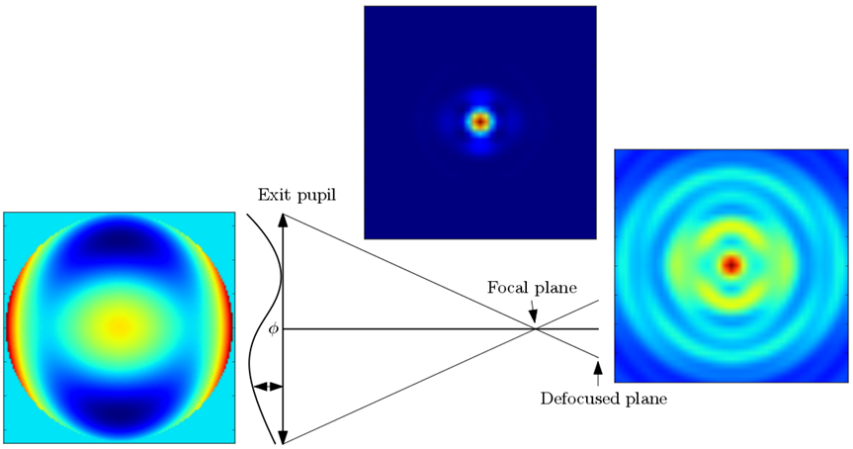


FIGURE 1.9: Schema of the phase diversity principle. The images from left to right are : the phase arriving on the exit pupil, the focused image and the defocused (2π) image.

using the autocorrelation properties ($\Gamma_A = \Gamma_{A'}$ with $A'(t) = A^*(-t)$), with only one image, one cannot determine the sign of the phase even part. This leads to the introduction of a phase diversity to raise the indetermination. The idea is to add a known aberration $\delta\phi$ to the system and to use two images to retrieve the phase of the wavefront.

Figure 1.9 shows the principle of the phase diversity. Two images are acquired, one at the focal plane and another with a defocus. We introduce the defocus by sliding the detector along the z-axis, others use a beamsplitter and another detector (Mugnier, Blanc, and Idier, 2006). We notice that having a more complex system, such has a deformable mirror, one could introduce any other even aberration such as an astigmatism, the only requirement is that the diversity introduced must have an even radial and azimuthal order.

The phase diversity is a technique that is sensitive to chromatism, because it is based on diffraction. The non-linear relation that links the image and the phase depends also on the object. This renders the inverse problem to solve more complicated, but it allows to retrieve the phase, the object or both, when the object is unknown (most of the time). Furthermore, it is very simple optically, does not require a complex optical components to acquire the data, one just uses the detector in place. Thus it depends directly on the images, without any NCPA between the adaptive optic system and the scientific detector.

The final application of the phase diversity algorithm is to correct for static aberration in an optical system. We know the object, since it is a point source that we introduce with a laser or choose in the sky (a star).

Chapter 2

Phase Diversity Experiment

Here, we describe the experiment put in place in the optical laboratory at the HEIG-VD to reconstruct wavefronts with unknown static aberrations introduced using phase screens. At first, we study the behaviour of the phase diversity algorithm put in place by Mugnier, Blanc, and Idier (2006) at ONERA with respect to number of averaging images, in other words noise level, number of Zernike coefficients retrieved, etc... Then we test the algorithm using an calibrated aberration introduced by a parallel plane plate in the beam comparing the result to a Zemax simulation. And finally, we introduce the phase screen to have random aberrations in the pupil and try to compare the phase diversity results with the Shack Hartman wavefront sensor results.

2.1 ONERA algorithm, Mugnier, Blanc, and Idier (2006)

In this section, we briefly present the phase diversity algorithm developed at ONERA by Mugnier, Blanc, and Idier (2006).

2.1.1 Algorithm description

As explained in section **??**, the phase diversity determines the phase of the wavefront, as well as the unknown object when needed, using two images of the same object with a phase diversity between each image. This gives the following equation system (Mugnier, Blanc, and Idier, 2006, p.11),

$$\mathbf{i}_f = \mathbf{h}_f * \mathbf{o} + \mathbf{n}_f \tag{2.1}$$
$$\mathbf{i}_d = \mathbf{h}_d * \mathbf{o} + \mathbf{n}_d, \tag{2.2}$$

where the bold letters means that it is the sampled quantities, the indexes f and d means respectively focused and defocused and $\mathbf{n}$ regroups the photon and detector noise present on the image. Using the data available, this algorithm approaches the problem at hand with a statistic point of view. They estimate jointly the aberrations and the object (Paxman, Schulz, and Fienup, 1992), which consist to compute the joint maximum *a posteriori* (JMAP) estimator (Mugnier, Blanc, and Idier, 2006, p.17),

$$\begin{aligned}(\hat{\mathbf{o}}, \hat{\boldsymbol{\phi}})_{MAP} &= \arg\max_{\mathbf{o},\boldsymbol{\phi}} p(\mathbf{i}_f, \mathbf{i}_d, \mathbf{o}, \boldsymbol{\phi}; \boldsymbol{\theta}) \\ &= \arg\max_{\mathbf{o},\boldsymbol{\phi}} p(\mathbf{i}_f|\mathbf{o}, \boldsymbol{\phi}; \boldsymbol{\theta}_n) p(\mathbf{i}_d|\mathbf{o}, \boldsymbol{\phi}; \boldsymbol{\theta}_n) p(\mathbf{o}; \boldsymbol{\theta}_o) p(\boldsymbol{\phi}; \boldsymbol{\theta}_\phi),\end{aligned} \tag{2.3}$$

where $p(\mathbf{i}_f, \mathbf{i}_d, \mathbf{o}, \boldsymbol{\phi}; \boldsymbol{\theta})$ is the joint probability density function of the two images $(\mathbf{i}_f, \mathbf{i}_d)$, the object $\mathbf{o}$ and the phase $\boldsymbol{\phi}$. It can also depend on a set of hyperparameters $\boldsymbol{\theta} = (\boldsymbol{\theta}_n, \boldsymbol{\theta}_o, \boldsymbol{\theta}_\phi)$. $p(\mathbf{i}_k|\mathbf{o}, \boldsymbol{\phi}; \boldsymbol{\theta}_n)$ is the likelihood of the image $\mathbf{i}_k$. $p(\mathbf{o}; \boldsymbol{\theta}_o)$ and $p(\boldsymbol{\phi}; \boldsymbol{\theta}_{\boldsymbol{\phi}})$ are the *a priori* probability density functions of $\mathbf{o}$ and $\boldsymbol{\phi}$.

They assume that the noise is white and stationary with a variance σ^2 on each image. They take Gaussian prior probability distributions for the object and for the phase which they decompose on the Zernike polynomial basis, $\boldsymbol{\phi}(\mathbf{a})$, with $\mathbf{a}$ the vector containing the Zernike coefficients from a_4 to $a_{j_{max}}$, see Mugnier, Blanc, and Idier (2006, p.18-19) for the detailed expressions. Finally, the phase and the object are retrieved by maximizing the joint probability density function $p(\mathbf{i}_f, \mathbf{i}_d, \mathbf{o}, \mathbf{a}; \boldsymbol{\theta})$ or taking the logarithm of the latter they retrieve them by minimizing the following criterion,

$$
\begin{aligned}
&L_{JMAP}(\mathbf{o}, \mathbf{a}, \boldsymbol{\theta}) \\
&\quad = -\ln\, p(\mathbf{i}_f, \mathbf{i}_d, \mathbf{o}, \mathbf{a}; \boldsymbol{\theta}) \\
&\quad = N^2 \ln \sigma^2 + \frac{1}{2}\ln \det(R_0) + \frac{1}{2}\ln \det(R_a) \\
&\quad + \frac{1}{2\sigma^2}(\mathbf{i}_f - H_f\mathbf{o})^t(\mathbf{i}_f - H_f\mathbf{o}) + \frac{1}{2\sigma^2}(\mathbf{i}_d - H_d\mathbf{o})^t(\mathbf{i}_d - H_d\mathbf{o}) \\
&\quad + \frac{1}{2}(\mathbf{o} - \mathbf{o}_m)^t R_o^{-1}(\mathbf{o} - \mathbf{o}_m) + \frac{1}{2}\mathbf{a}^t R_a^{-1}\mathbf{a} + A,
\end{aligned}
\tag{2.4}
$$

where N^2 is the number of pixels in the image, $\mathbf{o}_m$ and R_o are the mean object and its covariance matrix, R_a is the covariance matrix of the aberrations, H_k is the matrix representing the discrete convolution by the sampled $\mathbf{h}$ and A is a constant.

In order to simplify and fasten the computation, they rewrite the criterion replacing $\mathbf{o}$ by its estimator $\hat{\mathbf{o}}(\mathbf{a}, \boldsymbol{\theta})$ obtained by cancelling the derivative of L_{JMAP} with respect to $\mathbf{o}$, and they move to the Fourier domain, see eqt. (24) of Mugnier, Blanc, and Idier (2006, p.21).

2.1.2 Implementation

The implementation of the algorithm was done by Mugnier, Blanc, and Idier (2006), the code is called `Deco_ConjMarg`. Dr. Laurent Jolissaint wrote the IDL code to run the ONERA phase diversity, it is called `Diversity.pro` and visible in Appendix B.1.1. The code takes as input an array of at least two PSFs, one focused and one defocused, the displacement Δz of the PSFs with respect to the focus position, the wavelength of the incoming light, the focal distance of the system, the pixel size in arc second, the threshold of the minimization and the mode of reconstruction. It takes also the pupil diameter, pupil central obscuration diameter if needed and finally j_{max}. The code can run in two different mode, *Modal* and *Zonal*. The modal method reconstruct the phase using the Zernike basis. The zonal mode uses the pixels of the phase as the basis to reconstruct and the projection on the Zernike is done after the reconstruction. Both method uses the JMAP estimator.

2.2 Experimental Setup

The design of the experiment was already done by Bouxin (2017). The system is built according to her plans and specifications. Figure 2.1 shows the schema of the experimental setup.

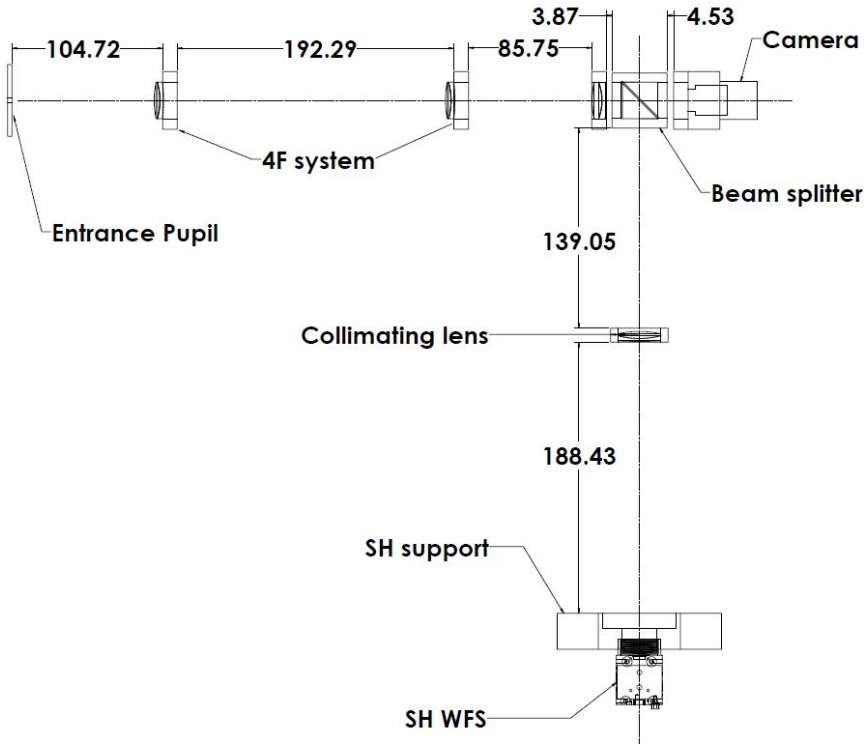


FIGURE 2.1: Experimental setup schema with the relevant distances, (Bouxin, 2017).

The experiment is mounted on a pressurized legs optical table. The assembly contains six main components : a light source, an entrance pupil, an imaging system, a converging lens to focus the beam on the camera, a camera and a wavefront sensor.

2.2.1 Light source

Regarding the final application of the phase diversity, the light source has to simulate a distant star aberration-free wavefront. A distant star wavefront is considered planar since the object distance, z, is far greater than the telescope size, r, see Fig. 2.2. The source of our experiment must then be characterized by a planar wavefront.

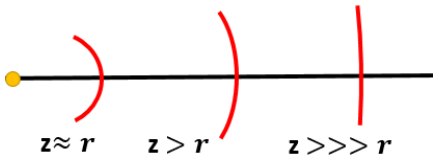
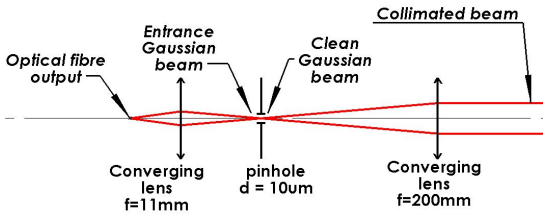
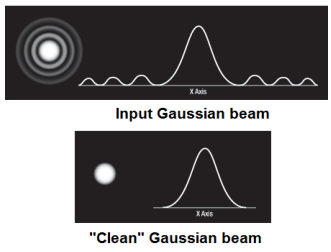


FIGURE 2.2: Wavefront curvature for different point source's distances, z. r represents the characteristic size of the arc of interest.

In order to obtain such a planar wavefront at the entrance pupil, the light source consist of a "pigtailed laser diode", a f=11mm converging lens, a pinhole and a f=200 mm converging lens, see Table 2.1. The pigtailed laser diode emits a Gaussian beam centred at 637.5 nm slightly diverging. The converging lens concentrates the beam at the center of the 10μm pinhole to filter the noise. The second converging lens collimates the beam, obtaining a collimated beam with a planar wavefront, see Fig. 2.3a and 2.3b.



(A) Source ray tracing, (Bouxin, 2017).



(B) Beam view before and after the pinhole, (Thorlabs, 2017).

FIGURE 2.3: Source schema and pinhole effect on the beam.

2.2.2 Entrance pupil

The entrance pupil of our optical system is a circular aperture of 3.2 mm diameter placed after the collimating lens of the light source. It is milled in a metal plate and centred in his support, to avoid positioning with a XY table. The diameter is chosen in available material to fit in the different detector's surfaces.

2.2.3 Pupil imaging system

The phase diversity technique requires PSFs images as input, which means that the beam as to be focused onto the detector surface. To analyse the aberration in the pupil plane, one needs to focus an image of the beam passing through the entrance pupil. The simplest assembly to achieve this goal is the 4F system, which consist of two converging lenses of focal 100 mm. The two lenses are separated by 200 mm, see Fig. 2.1. This places the image of the entrance pupil 100 mm after the second converging lens.

2.2.4 Detectors

The image of the entrance pupil, obtained with the 4F system, is focused onto a CMOS Ximea camera by a f = 80 mm converging lens to acquire the PSFs for the phase diversity wavefront retrieval. The camera has a surface composed by 1280x1024 pixels of 5.3 μm, see Appendix C.6. It is mounted on sliding support in order to be able to acquire in/out-of-focus images. A beam splitter is placed in the converging beam to separate it in two. The second beam is collimated and a Shack-Hartman WFS is placed on the entrance pupil image plane, to check the results of the phase diversity wavefront retrieval. The Shack-Hartman WFS has a 39 X 31 lenslets grid and a CCD with a resolution of 1280x1024 pixels of 4.65 μm, see Appendix C.7.

2.3 Data Acquisition

2.3.1 Ximea Camera

The ONERA algorithm takes at least one focused and one defocused PSFs, as described in section 2.1.2. The PSFs are acquired using a python script which uses an open-source library to control the ximea camera, `pyXimea`[1], available on GitHub. The acquisition is done following these steps. The first step in order to acquire

[1] https://github.com/pupil-labs/pyximea

TABLE 2.1: Optical Components

#	Components	Model	Reference
1	Pigtailed laser diode	Thorlabs, LPS-635-FC	C.1
2	Converging lens, f = 11 mm	Thorlabs, A220TM-A	C.2
3	Pinhole, 10 μm	Thorlabs, P10S	C.3
4	Converging lens, f = 200 mm	Thorlabs, AL100200	C.4
5	3.2 mm Hole milled in metal sheet	...	...
6	Converging lens, f = 100 mm	Thorlabs, AC254-100-A	C.5
7	Converging lens, f = 80 mm		
8	Camera CMOS	Ximea, MQ013MG-E2	C.6
9	Converging lens, f = 100 mm		
10	Shack-Hartman WFS	Thorlabs, WFS150-5C	C.7

PSFs is to determine the position of the camera's focus point using the python script `AlignementScriptXimeaCamera.py`, see Appendix A.2.1. This script let's you acquire consecutively PSFs at different camera's positions and computes their FWHM. It finally returns the minimum FWHM and the camera's position, see Figure 2.4a and 2.4b.

Once the focus point position of the camera is determined, the acquisition of the data is possible. The acquisition script is called `AcquisAndSaveXimea.py`, see Appendix A.2.2.

The user needs to set the main parameters before acquiring. They are the number of images on which to average, the size of the final PSFs, the position of the focus point on the sliding system and the initial guess to fit the 2D Gaussian on the PSF to find its center. The initial guess can be made using the Ximea camera software, called `xiCamTool`, with which one gets a live image of the camera.

Then the running program ask the user what he needs to do after having made a sound. The first thing the user needs to do is to place the camera at the focus point and turn on the LED in order to set the optimal exposure time in order to avoid saturation.

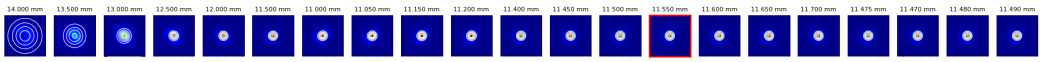
Then the acquiring sequence begins, the program ask the user to shut down and turn on the source as the user acquire the different images to get the dark images and the PSF images. The user needs to manually displace the camera to get the defocused PSFs. The program computes the positions to get a 2π P2V defocus dephasing. The PSFs are finally saved in *.fits* file.

Many functions, used in the two scripts described above, are coded in the script `functionsXimea.py`, see Appendix A.2.3.

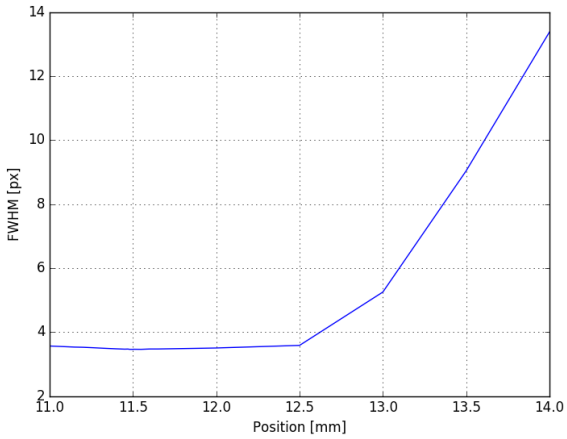
2.3.2 Shack-Hartmann WFS

The Shack-Hartmann wavefront sensor is delivered with a software which does the numerical integration to compute the wavefront. The GUI shows the spot field (focal points), the beam view (irradiance on the CCD), the wavefront measured or reconstructed (where you can choose the Zernike coefficients to consider for the integration up to $j_{max} = 66$) and the Zernike coefficients.

The acquisition is done with the company software. The data of interest are the Zernike coefficients and the reconstructed wavefront computed following the principle described in section 1.4.1. Their is a few parameters that the user can set :



(A) PSFs taken during the alignment procedure, each image is taken at an other camera position.



(B) FWHM of the PSFs as a function of the camera's position. The minimum is at 11.55 mm

FIGURE 2.4: Example of the results of an alignment procedure

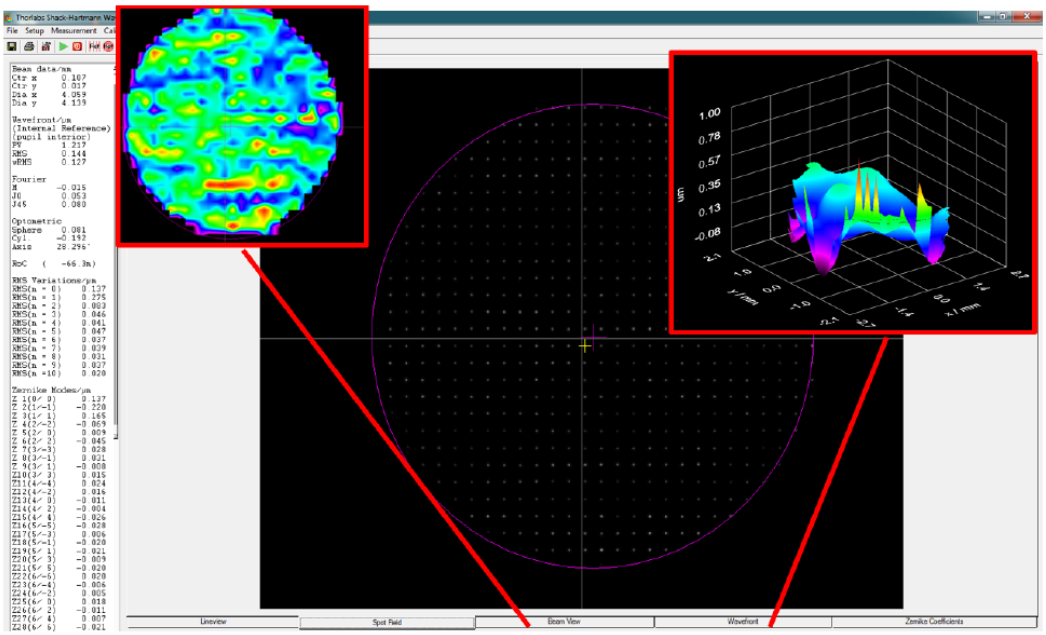


FIGURE 2.5: Shack-Hartmann Software GUI, with the beam view on the top left, the spot field in the middle and the reconstructed wavefront on the top right.

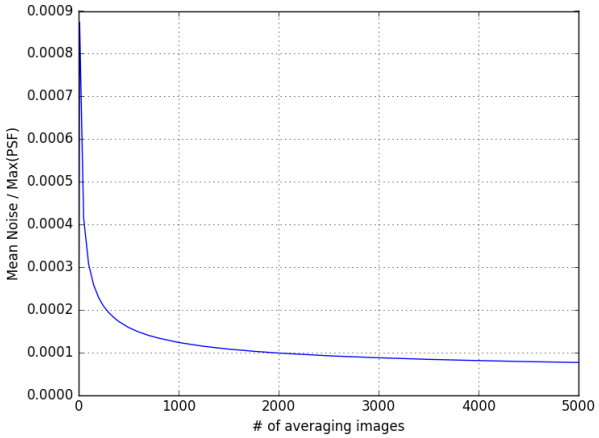


FIGURE 2.6: Noise level as a function of the number of averaging images acquired with $\sim$ 300 μs exposition time. The noise level is computed as the mean of the standard deviation of every pixel divided by the maximum of the focused PSF.

the exposure time (there is also an auto set), the number of averaging images (1 up to 300) and the focus points reference of the micro-lenses array.

The data are saved manually in a *.csv* file. I coded an IDL script to read and average the Shack-Hartmann data, in order to be able to analyse them and compare the results with the phase diversity, see Appendix B.2.1 and B.2.2.

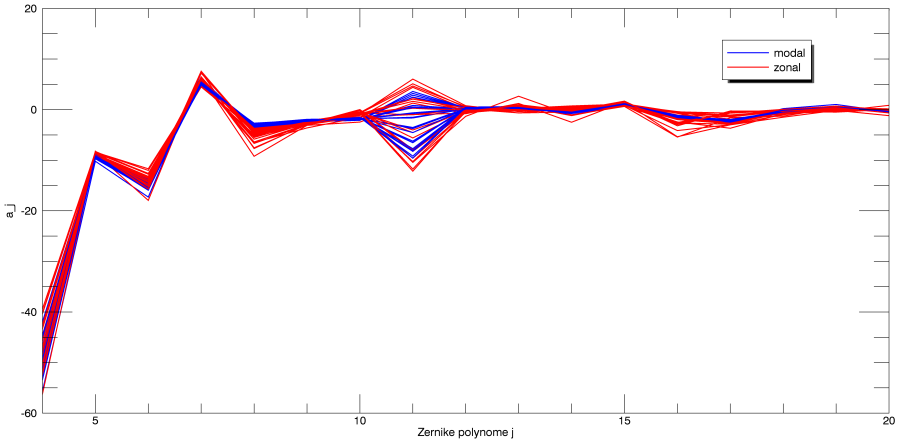
2.4 Results

This section presents the results of the phase diversity experiment, with the introduction of different sources of aberration. We will first present the results of the ONERA algorithm test, then we will compare the phase diversity retrieval with a calibrated aberration and finally we introduce random static aberrations using a phase screen and compare the results with the Shack-Hartmann wavefront sensor results.

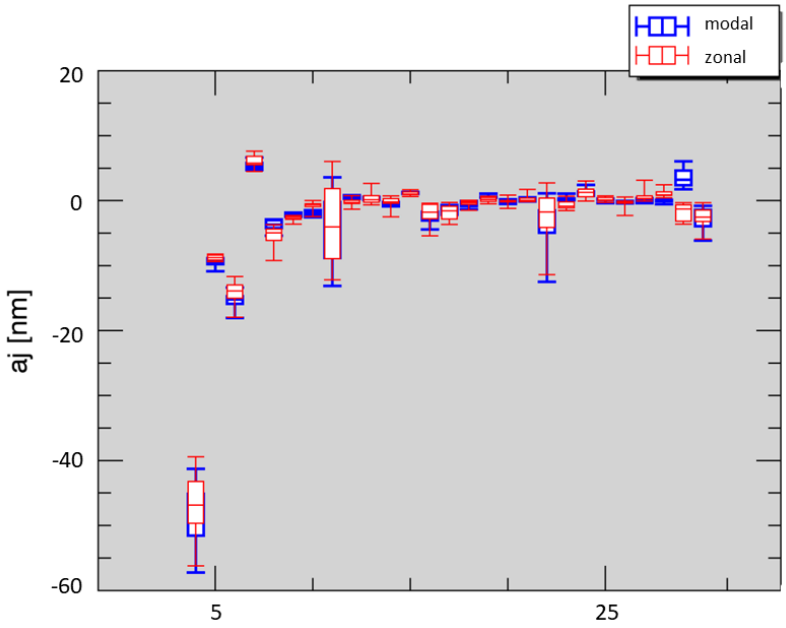
2.4.1 ONERA Phase Diversity test

The motivation of this test is to better understand the behaviour of the ONERA phase diversity algorithm with respect to the different parameters that could influence its results, such as the noise present in the PSFs, the number of Zernike coefficients returned for the modal mode and the error on the position of the Ximea camera.

To study the noise, we acquire PSFs with 25 different numbers of averaging images going from 10 to 5000. The noise level is defined as the ratio between the mean pixel standard deviation and the PSF maximum. The Ximea detector has a noise curve visible in Figure 2.6. The noise curve is computed with the noise level of the focused PSFs. It follows an exponential law with the number of averaging images. The noise level varies from $\sim 1e-3$ to $\sim 8e-5$, for 10 and 5000 images respectively. Having the noise curve of the detector, we can study empirically how it influences the phase diversity results.



(A) Zernike coefficients a_j as a function of j the Zernike index of the 25 modal and zonal phase retrievals, in blue and red respectively.



(B) Boxplot of the 25 Zernike coefficients a_j of the different phase retrievals computed with the different averaging numbers of images as a function of the Zernike index j.

FIGURE 2.7: Results of the phase retrievals computed with the 25 different noise levels. The phase retrievals are done with a $j_{max} = 30$

Figures 2.7a and 2.7b presents the results of the 25 different retrievals. As one can see there is some spread due to the different noise levels present in the PSFs given to the algorithm. The biggest standard deviation on a Zernike coefficient is smaller than 6 nm. And there can be up to 20 nm of maximal difference between the a_j's as one can see on Figure 2.7b. The spherical aberration, a_{11}, has the biggest standard deviation and range of values. This shows that the PSFs noise levels have an impact on the retrieval and that not all Zernike coefficient are affected in the same way. Two important points are the good correspondence between the modal and zonal retrievals and the relatively small spread of the Zernike coefficients.

Furthermore, looking at the reconstructed wavefronts in Figure 2.8, one can see the effect of noise in the PSFs and how important is the choice of j_{max} for the modal reconstruction. Indeed, for the wavefronts reconstructed using $j_{max} = 30$, the retrieved structures are similar with 10 or 3500 averaging images, however for $j_{max} = 200$ there are more structures in the retrieval done on the PSFs with 10 averaging images than with 3500 images. This shows that the noise has a strong signal at high spatial frequencies. The zonal retrieval shows it clearly, the reconstruction is significantly different between 10 and 3500 averaging images. The averaging over a large number of acquisition smooth out the noise high frequencies that perturbs the reconstruction. Looking at the 25 reconstructed wavefronts with the zonal method, the noise threshold is at 3500 averaging images, corresponding to a noise level of 0.0009, above it the effect of the noise are reduced. Also as said above for the Zernike coefficients and still valid for the wavefront, the two retrieval method gives similar results, as one can compare the three retrievals with 3500 images on the left column of Figure 2.8. There is structures present on the wavefront reconstructed on 200 Zernike coefficients, but the footprint of the beam is similar to the two other reconstructions.

In order to be sure that the number of Zernike coefficient retrieved only influenced the wavefront reconstruction and not the a_j's values, we computed them varying j_{max} from 4 to 231 and found that the spread is negligible, see Figure 2.9. This confirmation is expected since the Zernike polynomials are orthonormal, thus not correlated.

Finally, the last test is the control of the impact of a measurement error $\sigma_{\Delta z}$, the error position of the Ximea camera, on the phase retrieval. To do so the latter is run with 125 different permutations of the errors $[-2\sigma_{\Delta z}, -1\sigma_{\Delta z}, 0, 1\sigma_{\Delta z}, 2\sigma_{\Delta z}]$ on the 3 PSFs position. $\sigma_{\Delta z}$ is set to represent the best precision we have on the position of the camera. As explained in section 2.2.4, the camera is mounted on a sliding element moved by a micro metric screw, so $\sigma_{\Delta z} = 5e-3$ mm. Figures 2.10a and 2.10b present the results. The position error has nearly no effect on the phase retrieval, the median standard deviations are ~ 0.05 nm and ~ 0.07 nm for the modal and zonal method respectively.

In conclusion, the ONERA algorithm tests shows that first the two different methods, to retrieve the phase using the JMAP estimator, modal and zonal give similar results. Then, we have seen that the noise levels present in the PSFs influence the retrieval, see Figures 2.7a, 2.7b and 2.8. We can consider the noise as an added aberration that touches all the Zernike coefficients. The spherical aberration is the most sensitive one. Also looking at the wavefronts, we see that there is a number of averaging images threshold at 3500 images for the zonal retrieval. Above this threshold the noise seems to be smooth and the retrieval looses the high spatial frequency components. The same thing is true for the modal retrieval, but the smoothing is not as effective at 3500 images. For the rest of this work each PSFs will be acquired with 5000 averaging images. Also we showed that the number of Zernike coefficient

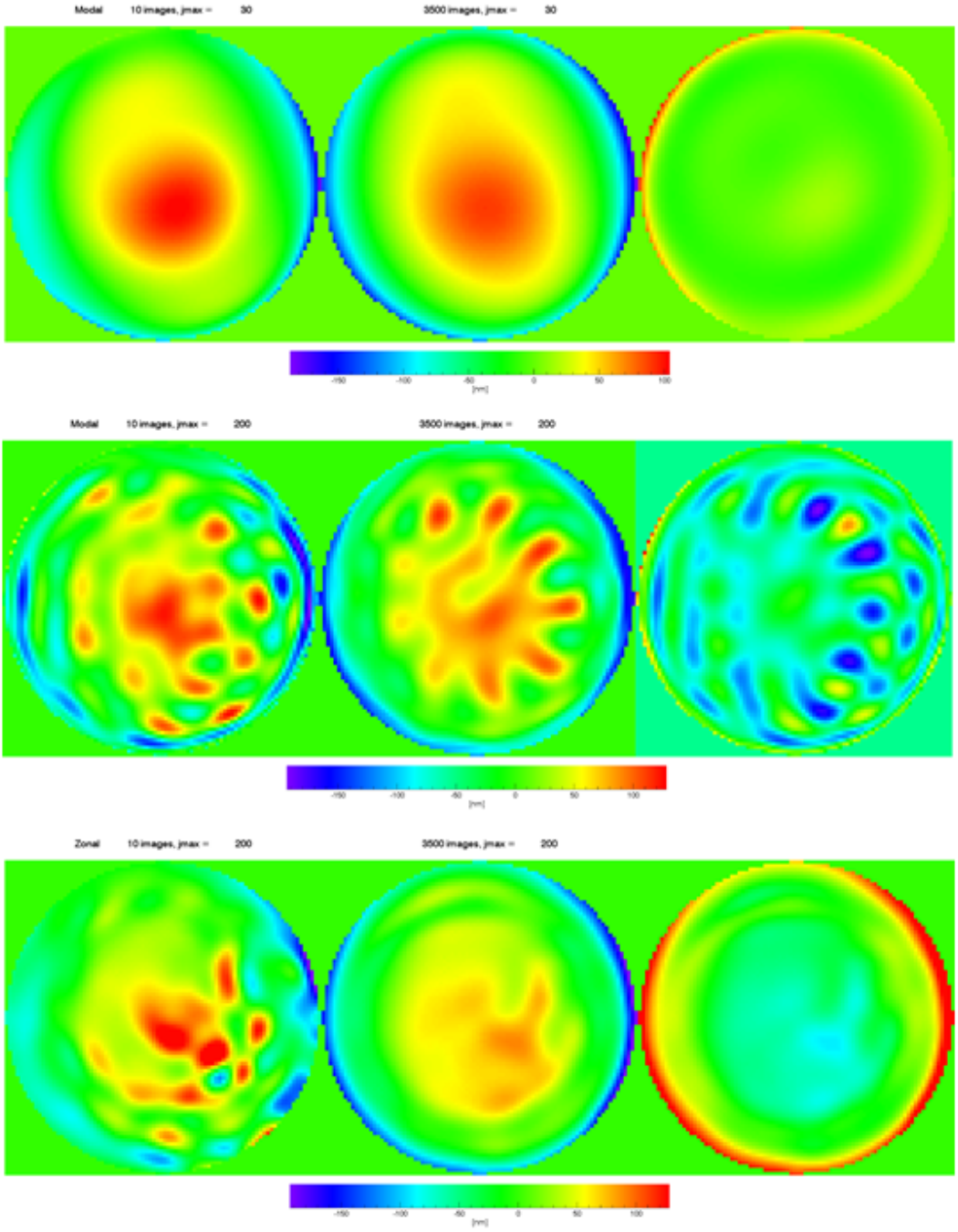


FIGURE 2.8: Reconstructed wavefronts for two different numbers of averaging images, 10 and 3500, and their difference are on the first, second and third column respectively. The two first line are modal retrieval with $j_{max} = 30$ and $j_{max} = 200$ and the last line is the zonal retrieval.

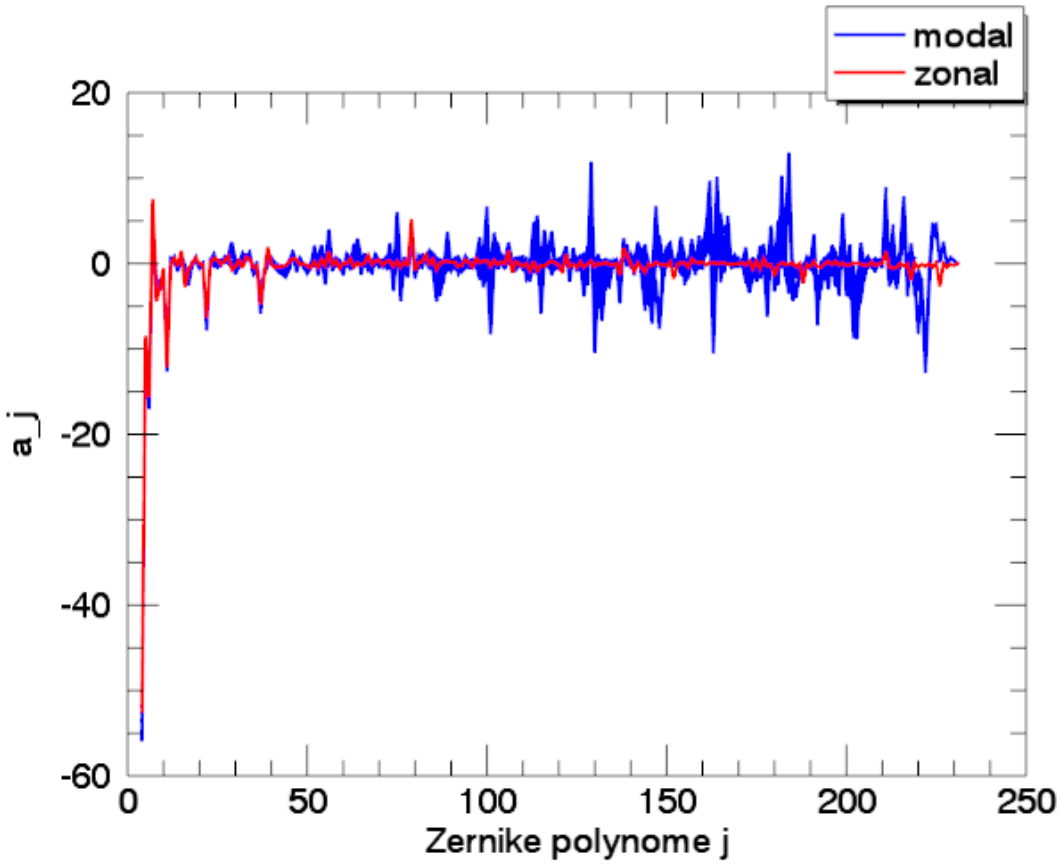
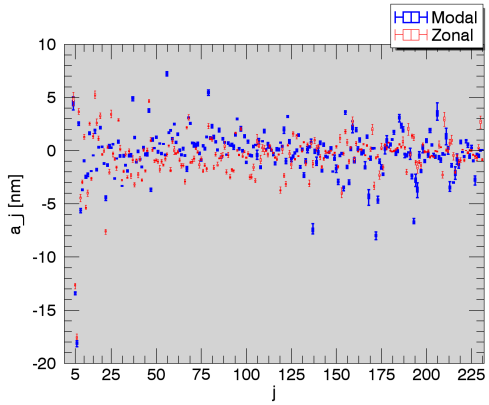
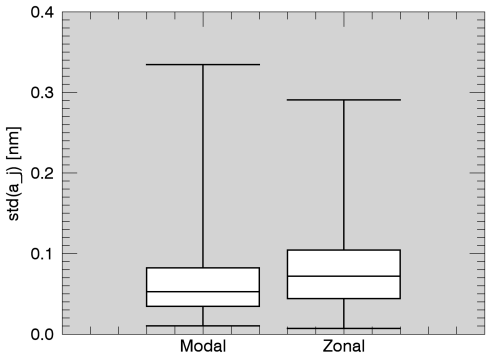


FIGURE 2.9: Zernike coefficients a_j as a function of j for the 227 retrievals done for 231 different j_{max} from 4 to 231.



(A) Boxplots of the 125 a_j's retrieved with the modal method (blue) and zonal method (red). The PSFs are acquired with 5000 averaging images.



(B) Boxplots of the 125 standard deviations on the Zernike coefficient for the modal and zonal method.

FIGURE 2.10: Results of the Δz measurement error test. The plots represent the results of the 125 phase retrievals run with all the permutations of errors, $[-2\sigma_{\Delta z}, -1\sigma_{\Delta z}, 0, 1\sigma_{\Delta z}, 2\sigma_{\Delta z}]$ with $\sigma_{\Delta z} = 5e-3$ mm, on the three PSFs position Δz

.

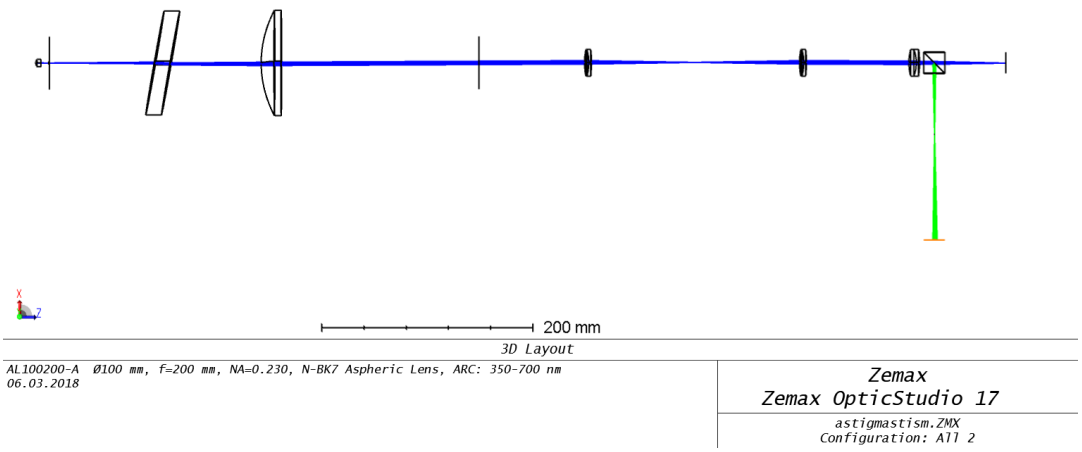


FIGURE 2.11: Zemax simulation of our optical system with the parallel plane plate introducing astigmatism.

j_{max} does not alter the modal retrieval. And finally, the error on the position of the camera also has a negligible impact.

2.4.2 Parallel plane plate

Knowing how the algorithm behaves with respect to noise, j_{max}, etc... We compare its results to a calibrated aberration. The calibration is dome by simulating the optical system using Zemax[2]. We introduce astigmatism using a plexiglas parallel plane plate with an angle with respect to the optical axis, see Figure 2.12. The astigmatism is due to the non symmetric incident angle on the left and right side of the optical axis with respect to x. This induces a difference in optical path between the symmetrical rays on the left and right side of the optical axis and thus induces an aberration, called the astigmatism.

In order to have the best correspondence between the simulation and the experiment, the results are the difference between the Zernike coefficient retrieved with the parallel plane plate and the Zernike coefficient without the plate. Indeed, the Zemax simulation do not have alignment issues and other man introduced imperfections. So by subtracting the aberrations present in the system without the parallel plane plate, in theory we should remove any component that do not come from the plate itself in order to only retrieve the plate effect.

Figure 2.13 shows the vertical astigmatism Zernike coefficient a_6 as a function of the parallel plane plate angle from 10° to 50°. The two coloured lines correspond to the two phase retrieval methods and the dashed line is the Zemax simulation of the parallel plane plate. As one can see, there is an offset of $\sim 10 - 15$ nm between the simulation and the experiment, which correspond to an error of $\sim \frac{\lambda}{64}$. The behaviours of the simulation and the experiment are the same which is good, despite the overestimation of the astigmatism by the phase diversity retrievals. The system without the parallel plane plate has a $a_6 = 2.1$ nm. This might show that our system is not perfectly aligned, but it does explain the $\sim 10 - 15$ nm offset.

The vertical coma, Zernike coefficient a_8, as a function of the parallel plane plate angle is shown in Figure 2.14. The optical system in presence of the parallel plane

[2] https://www.zemax.com/opticstudio

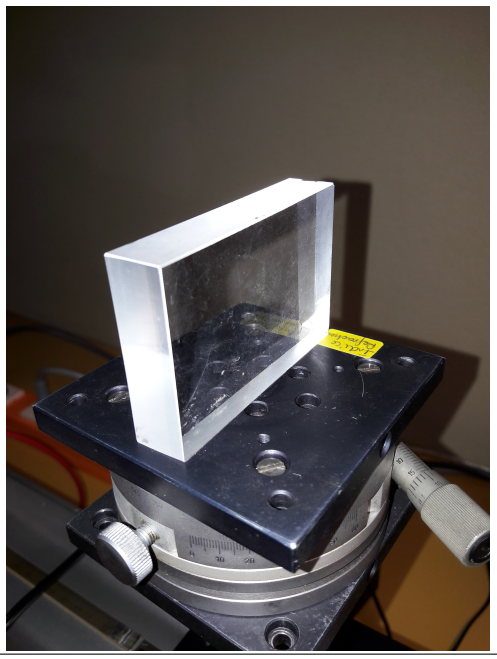


FIGURE 2.12: Plexiglas parallel plane plate used to introduce calibrated aberrations. The width is equal to 14 mm and the refractive index is $n = 1.49$.

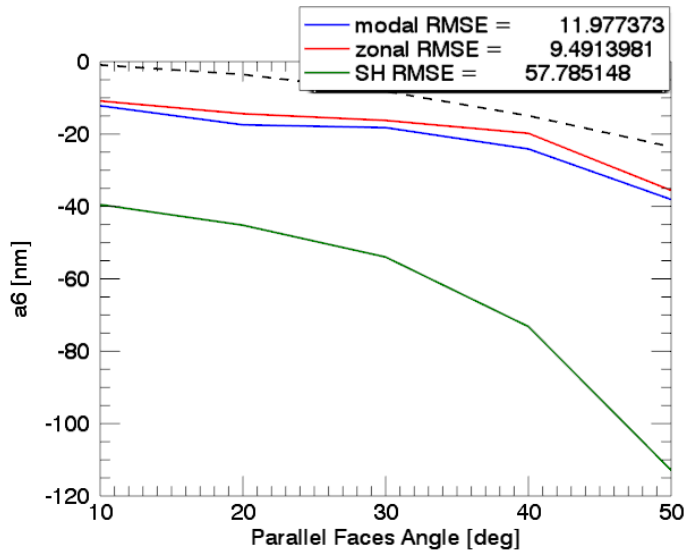


FIGURE 2.13: Zernike coefficient of the vertical astigmatism a_6 as a function of the angle of the plexiglas parallel plane plate. The dashed line correspond to the Zemax simulation result, the blue line to the modal retrieval method, the red line to the zonal retrieval and the green line to the Shack-Hartmann reconstruction

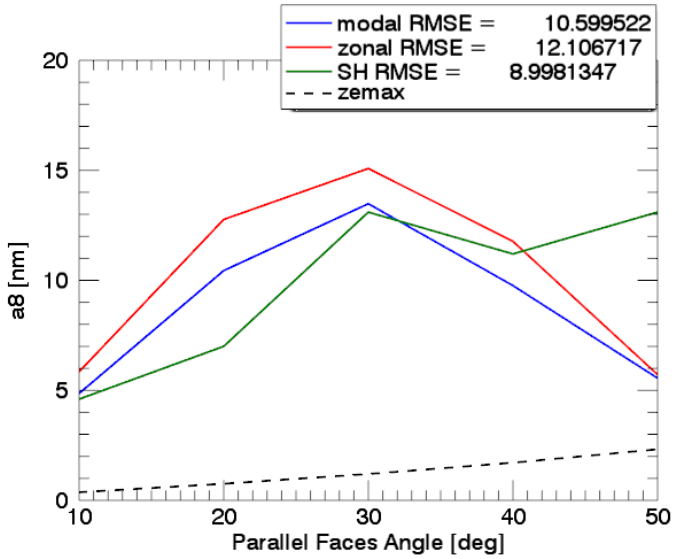


FIGURE 2.14: Zernike coefficient of the horizontal coma a_8 as a function of the angle of the plexiglas parallel plane plate. The dashed line correspond to the Zemax simulation result, the blue line to the modal retrieval method , the red line to the zonal retrieval and the green line to the Shack-Hartmann reconstruction.

plate has a significant coma aberration, up to ~ 15 nm according to the phase diversity. Without an aberration introduction, the system has a vertical coma coefficient $a_8 = 6.4$ nm, which also support the fact that our system is not perfectly aligned.

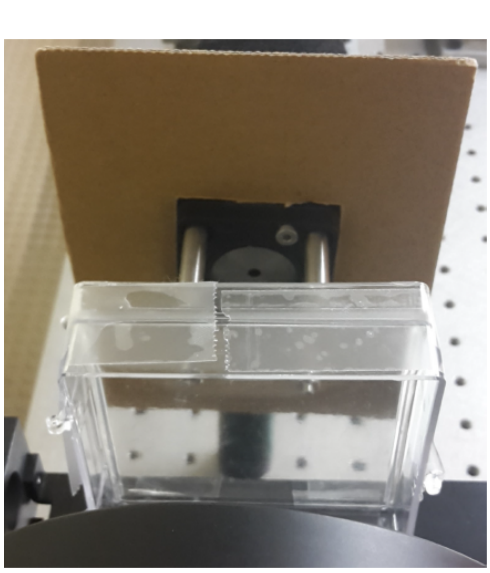
2.4.3 Shack-Hartmann comparison

Even with the offset and alignment problems, we compared the phase diversity to the Shack-Hartmann. The comparison is performed with random static aberrations introduced by a phase screen. A phase screen is a transparent plastic box, see Figure 2.15a, whose fabrication process gives it the property of static turbulent aberrations, see Figure 2.15b. Indeed, the plastic injection follows the typical flux that we can find in the atmosphere and the solidification fix this distribution.

The results of the comparison are presented in Figures 2.16 and 2.17. According to both instrument, the phase screen introduces significant aberrations, but the comparison between the two does not seem to support any correlation between the two methods. There is more than a 25 nm difference in average between the phase diversity and the Shack-Hartmann. Also the comparison of the wavefronts is not good. Furthermore, the comparison of the parallel plane plate results of the Shack-Hartmann, the phase diversity and the Zemax model reveals that there is differences between the three, see Figures 2.13 and 2.14. However, the vertical coma comparison between the phase diversity and the Shack-Hartmann is better, but it might be a coincidence.

2.4.4 Discussion

The algorithm test, section 2.4.1, is interesting. We learned a lot of things about the properties and behaviour of the ONERA phase diversity algorithm. The noise



(A) Phase screen used in the experiment.

(B) Neon light with (up) and without (down) phase screen static turbulence.

FIGURE 2.15: Phase screen introduction.

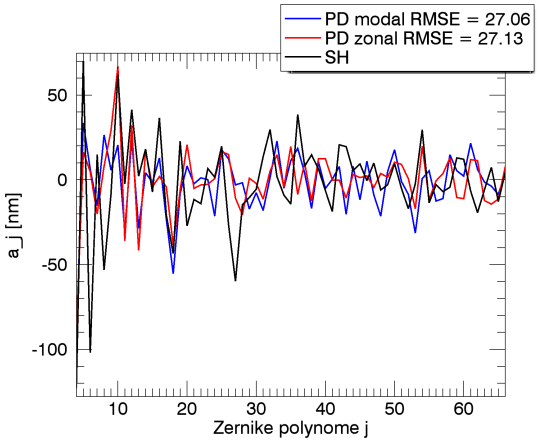


FIGURE 2.16: Zernike coefficients retrieved as a function of j. The red and blue line correspond to the zonal and modal retrieval method respectively. The black line represent the Shack-Hartmann results. $j_{min} = 4$ and $j_{max} = 66$.

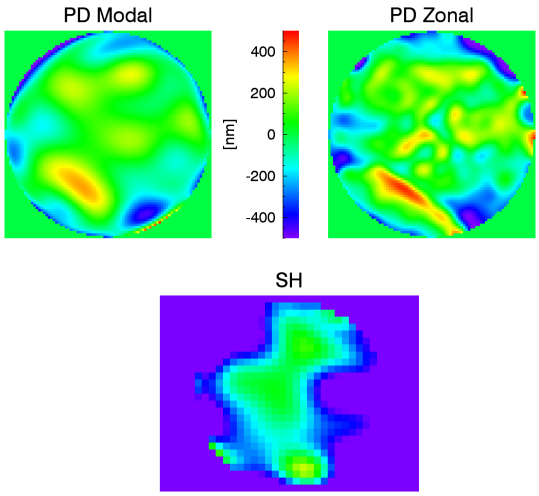


FIGURE 2.17: Reconstructed wavefront by the Modal and Zonal method of the phase diversity on 66 Zernike mode on the first line and reconstructed wavefront by the Shack-Hartmann on 66 mode as well, on the second line.

present in the PSFs has its importance especially when trying to reconstruct the wavefronts. There seems to be a threshold above which the high spatial frequency component of the noise are neutralized. This threshold is at 3500 images corresponding to a noise level of ~ 0.00009 as observed in the Zonal reconstruction. Also the number of Zernike over which the retrieval is done, j_{max}, does not really matters to recover the Zernike coefficient, but as seen in Figure 2.8, it has a significant influence on the reconstruction of the wavefronts due to the high spatial frequency component of the noise. Finally the test also confirms that a small error on the position of the camera would not perturbed the phase retrieval.

The parallel plane plate experiment used to compare the phase diversity with a calibrated aberration does not work as expected. The astigmatism introduced by the Plexiglas plate follows the right behaviour, but there is an offset that is not expected of $\sim 10 - 15$ nm. Comparing the vertical coma from the model with the retrieved ones, we see that that a significant amount of coma is present even though it is negligible in the Zemax model. The coma is interesting because it is typical in system with alignment problem. In amateur telescopes with two mirrors, it arises when the M2 is not correctly aligned on the M1. Thus, we conclude that there is an alignment problem. The alignment problem surely comes from the complex setup collimating the beam after the pinhole. One should use a f=200mm lens that goes onto the 4-rods mounting system and attach the source to the rods also. That way any aberrations due to a misalignment of the source with its collimating lens and the rest of the setup could be avoided.

Now, the problem is the fact that our calibrated aberration modelled by Zemax cannot be reproduced in reality. Indeed, the offset is unexpected, because subtracting the results of the phase retrieval on the PSFs acquired without introducing aberrations to the results with the parallel plane plate should have eliminated the aberrations present in the optical system. Either the Zemax model is not representing our system correctly. Or the phase diversity algorithm does not respect the linearity of the image formation process. This explanation would also mean that the Shack-Hartmann has a linearity problem. The most plausible explanation is the

first one, but we do not know why since all the components introduced in the model correspond to the one we have on the optical bench. An argument supporting the second explanation could be that since we retrieve a limited number of Zernike, non linear effects could arise as the available Zernike could take a bigger amount of aberrations that they truly have.

The second problem is the fact that the phase diversity and the Shack-Hartmann are not similar. This might be due to a misalignment of the wavefront sensor, in other words it might not be on a plane conjugated with the pupil. Indeed, as we argue above, the system seems to have alignment problem. The Shack-Hartmann is placed according to the distance on the plan (Figure 2.1) and cancelled the defocus coefficient a_4 to find the correct position. Thus, since the systems already has aberrations the position with no defocus might not be the correct one, explaining the difference between the two retrieval methods. A better way to check the conjugation between the planes would have been to put a rule on the milled entrance pupil and determine the position where it was sharp on the Shack-Hartmann.

Finally, in order to improve our results and be able to use correctly the ONERA phase diversity algorithm, we would need to be more prudent at the beginning. We should first have a solid way to describe the system independently, without the phase diversity. It was the idea behind the use of the Shack-Hartmann. But we went to quickly into trying the phase diversity algorithm and testing it. We need to first be sure of the data collected by the Shack-Hartmann, be able to recover the Zemax predictions with the Shack-Hartmann. And once we know how our system behaves and we have a clear way to characterize it, we can try to compare the phase diversity and the Shack-Hartmann.

Chapter 3

Analytical phase diversity algorithm

After having tested and used the ONERA algorithm, we can pass to the main goal of this work, the development of our phase diversity algorithm. We describe the algorithm and its implementation. We present the result of its testing with simulated PSFs. A comparison between our algorithm and the ONERA algorithm is reported. And finally an approach to expand the validity domain of the method is presented, that could be used on a telescope equipped with a deformable mirror.

3.1 Analytical algorithm

3.1.1 Algorithm description

This algorithm uses an analytical approach, developed by Dr. Laurent Jolissaint, to retrieve the phase of the wavefront **induced by a known point source object**. We assume the object known, because we want to correct for the static aberrations present in the optical system to the scientific detector. Thus we use a point source to illuminate the optical system, either a star or a laser.

As we have seen in section 1.2, the PSF of an optical system correspond to the image it gives of a point source,

$$PSF(x,y) = \frac{1}{S_p^2} \left| \left[\mathcal{F} \left\{ P(\xi,\eta) A(\xi,\eta) e^{-j\phi(\xi,\eta)} \right\} \right] (x,y) \right|^2, \tag{3.1}$$

where $P(\xi,\eta)$ is the exit pupil function, $A(\xi,\eta)$ is the amplitude of the wave through the exit pupil, $\phi(\xi,\eta)$ is the phase of the wavefront and S_p is the exit pupil surface. In the following, we omit the coordinates to simplify the notation. The unit of the PSF is directly the Strehl ratio. Under the assumption that we have weak aberrations, we can expand the exponential term,

$$exp(-j\phi) \approx 1 - j\phi - \frac{\phi^2}{2} + O(\phi^3), \tag{3.2}$$

replacing the exponential by its expansion in eqt.(3.1) leads to,

$$S_p^2 PSF \cong \left| \mathcal{F} \left\{ PA(1 - j\phi - \frac{\phi^2}{2}) \right\} \right|^2 \tag{3.3}$$

Developing eqt. (3.3), keeping only the terms up to the second order, assuming that the amplitude through the pupil $A(\xi,\eta)$ is constant and unitary since we have a point source object and using the well known complex relations,

$$a + a^* = 2\Re\{a\}$$
$$a - a^* = 2j\Im\{a\},$$

we obtain the following relation,

$$S_p^2 PSF \cong |\widetilde{P}|^2 + |\widetilde{P\phi}|^2 + 2\Im\{\widetilde{P^*}\widetilde{P\phi}\} - 2\Re\{\widetilde{P^*}\widetilde{P\phi^2}\} \tag{3.4}$$

Defining ΔPSF as the difference between eqt. (3.4) for an arbitrary optical system and its perfect equivalent, we obtain the following expression,

$$\Delta PSF = S_p^2 PSF - S_p^2 PSF_{perfect} = |\widetilde{P\phi}|^2 + 2\Im\{\widetilde{P^*}\widetilde{P\phi}\} - 2\Re\{\widetilde{P^*}\widetilde{P\phi^2}\}, \tag{3.5}$$

where $S_p^2 PSF_{perfect}$ is equal to $|\widetilde{P}|^2$ for an equivalent perfect system with the same pupil. One can decompose ϕ into its even and odd phase, ψ and γ respectively,

$$\phi = \psi + \gamma \tag{3.6}$$

Developing eqt. (3.5) after replacing ϕ by its decomposition and using the properties of the Fourier transform of real and purely even or odd functions, we get the following expression,

$$\Delta PSF = |\widetilde{P\psi}|^2 + |\widetilde{P\gamma}|^2 + 2\Im\{\widetilde{P^*}\widetilde{P\gamma}\} - \Re\{\widetilde{P^*}\widetilde{P\psi^2}\} - \Re\{\widetilde{P^*}\widetilde{P\gamma^2}\} \tag{3.7}$$

We can decompose ΔPSF into its even and odd components,

$$\Delta PSF_{even} = |\widetilde{P\psi}|^2 + |\widetilde{P\gamma}|^2 - \Re\{\widetilde{P^*}\widetilde{P\psi^2}\} - \Re\{\widetilde{P^*}\widetilde{P\gamma^2}\}, \tag{3.8}$$
$$\Delta PSF_{odd} = 2\Im\{\widetilde{P^*}\widetilde{P\gamma}\}, \tag{3.9}$$

This equation system shows that we can retrieve the odd part of the phase easily with eqt. (3.9). But eqt. (3.8) clearly reveals the indetermination of the phase retrieval with only one image, as the sign of the even part of ϕ can not be determine. In order to raise this indetermination, as exposed in section **??**, we need to introduce a phase diversity $\delta\phi$. We can modify the pupil function P in order to take into account this introduced diversity,

$$P_\delta \equiv Pe^{-j\delta\phi} = P(cos(\delta\phi) - jsin(\delta\phi)) = P(C - iS) \tag{3.10}$$

The expression of $\Delta PSF_{\delta\phi}$, which is the ΔPSF at the defocus plane, is found by replacing P by P_δ in eqt. (3.7), we give directly the expressions of the even and odd components by taking into account that the phase is only define on the pupil ($P\phi = \phi$) to simplify the reading,

$$\begin{aligned}\Delta PSF_{\delta\phi,even} = {} & |\widetilde{C\psi}|^2 + |\widetilde{C\gamma}|^2 + |\widetilde{S\psi}|^2 + |\widetilde{S\gamma}|^2 - 2\widetilde{PC}^*\widetilde{S\psi} + 2\widetilde{PS}^*\widetilde{C\psi} \\ & - \widetilde{PC}^*\widetilde{C\psi^2} - \widetilde{PC}^*\widetilde{C\gamma^2} - \widetilde{PS}^*\widetilde{S\psi^2} - \widetilde{PS}^*\widetilde{S\gamma^2}\end{aligned} \tag{3.11}$$

$$\begin{aligned}\Delta PSF_{\delta\phi,odd} = {} & 2\widetilde{C\psi}^*\Im\{\widetilde{S\gamma}\} + 2\Im\{\widetilde{C\gamma}^*\}\widetilde{S\psi} + 2\widetilde{PC}^*\Im\{\widetilde{C\gamma}\} + 2\widetilde{PS}^*\Im\{\widetilde{S\gamma}\} \\ & + 2j\widetilde{PC}^*\widetilde{S\psi\gamma} - 2j\widetilde{PS}^*\widetilde{C\psi\gamma}.\end{aligned} \tag{3.12}$$

Eqt. (3.9), eqt. (3.11) and eqt. (3.12) allow to retrieve the complete phase of the optical system under the assumption of weak aberrations. The numerical retrieval method uses the decomposition of the even and odd part of the phase on the Zernike polynomials,

$$\psi = \sum_{js\ even} a_j Z_j \tag{3.13}$$

$$\gamma = \sum_{js\ odd} a_j Z_j. \tag{3.14}$$

This allows to have a linear system of equations with respect to the Zernike coefficient a_j using eqts. (3.9) and (3.12), but a problem arises when the phase is purely even. The equations gives an odd phase part equals to zero, as expected, but also an even phase part equals to zero. This comes from the fact that in eqt. (3.12), each term is multiplied by the odd phase component. And we can not use eqts. (3.8) or (3.11), due to the squared modulus of the even and odd phase component, which rendered our system of equation non-linear.

One way to get around this issue is to add another diversity, which is equal in amplitude to the first one but its inverse. So we have two diversities given by,

$$\delta\phi_+ = \delta\phi \text{ and } \delta\phi_- = -\delta\phi \tag{3.15}$$

Using the new diversity, eqt. (3.9) allows to determine the odd part of the phase as before, but now for the even part of the phase we compute the difference between $\Delta PSF_{\delta\phi_+,even}$ and $\Delta PSF_{\delta\phi_-,even}$. This gives us the following system of equation,

$$\Delta PSF_{odd} = 2\Im\{\widetilde{P^*}\widetilde{P\gamma}\} \tag{3.16}$$

$$\Delta PSF_{\delta\phi_+,even} - \Delta PSF_{\delta\phi_-,even} = -4\widetilde{PC}^*\widetilde{S\psi} + 4\widetilde{PS}^*\widetilde{C\psi}, \tag{3.17}$$

which we can use to determine the complete phase of the wavefront. We can rewrite it using the decomposition of ψ and γ on the Zernike basis,

$$\Delta PSF_{odd} = \sum_{js\ odd} a_j 2\Im\{\widetilde{P^*}\widetilde{PZ_j}\} \tag{3.18}$$

$$\Delta PSF_{\delta\phi_+,even} - \Delta PSF_{\delta\phi_-,even} = \sum_{js\ even} a_j\{-4\widetilde{PC}^*\widetilde{SZ_j} + 4\widetilde{PS}^*\widetilde{CZ_j}\}. \tag{3.19}$$

To solve these two equations and find the a_j's, we use a linear regression method. We can rewrite the equations under a vectorial form to clarify the equations by flattening the images,

$$\overrightarrow{\Delta PSF}_{odd} = \underline{Z}_f \vec{a}_{odd} \tag{3.20}$$

$$\overrightarrow{\Delta PSF}_{\delta\phi_+,even} - \overrightarrow{\Delta PSF}_{\delta\phi_-,even} = \underline{Z}_d \vec{a}_{even}. \tag{3.21}$$

where $\underline{Z}_f$ is the N^2xk_{odd} matrix regrouping all the terms of $2\Im\{\widetilde{P^*}\widetilde{PZ_j}\}$, k_{odd} is the number of odd Zernike polynomials between j_{min} and j_{max}, and $\underline{Z}_d$ is the N^2xk_{even}

matrix regrouping all the terms of $\{-4\widetilde{PC}^{*}\widetilde{SZ_j} + 4\widetilde{PS}^{*}\widetilde{CZ_j}\}$, k_{even} is the number of even Zernike polynomials between j_{min} and j_{max}.

3.1.2 Implementation

The resolution of the equations as said above is done with a linear regression. The code is structured as following :

The Class phaseDiversity3PSFs is the main class of the program, see Appendix A.1.1. It takes as inputs : the 3 PSFs needed to retrieve the phase as exposed above (inFoc, outFocpos and outFocneg), the displacement of the detector with respect to its focused position (Δz),

$$\Delta z = \frac{4\Delta\phi}{\pi}\lambda\left(\frac{D}{F}\right)^2, \tag{3.22}$$

where $\Delta\phi$ is the Peak To Valley (P2V) dephasing that we want to introduce in the defocused image in the following it will be equal to 2π, D is the pupil diameter and F the focal length of the optical system. It takes also as inputs the wavelength of the incoming light, the pixel size of the detector, the focal length of the optical system, the radius of the pupil and the boundary on $jmin < j < jmax$ the Zernike index.

The instantiation of all the elements of eqts. (3.20) and (3.21) is done by calling the class method `initiateMatrix1()` and `initiateMatrix2()`. These two methods compute the left and right members of the equations by calling methods present in the script `fs.py`.

The script fs is gathering functions needed to retrieve the phase of the wavefront, see Appendix A.1.2. ΔPSF_{odd} and $\Delta PSF_{\delta\phi_+,even} - \Delta PSF_{\delta\phi_-,even}$ are computed by the two methods `y1(params)` and `y2even(params)`. The matrix elements of $\underline{Z}_f$ are computed by the method `f1j(params)` and the elements of $\underline{Z}_d$ are computed by `f2jeven(params)`. Those two functions needs the Zernike polynomial basis which is coded in `zernike.py`, see Appendix A.1.4.

The linear regression is done after having initiated all the elements of the equations. We use the `numpy.linalg` class, especially its `lstq` function[1]. This functions takes a vector $\vec{b}$ corresponding to the flattened ΔPSF_{odd} or $\Delta PSF_{\delta\phi_+,even} - \Delta PSF_{\delta\phi_-,even}$ and a matrix $\underline{a}$ corresponding $\underline{Z}_f$ or $\underline{Z}_d$. Then it determines the vector $\vec{x}$ corresponding to the vector $\vec{a}$ that minimizes the euclidean 2-Norm $\|b - ax\|$ by computing the Moore-Penrose pseudo inverse of $\underline{a}$.

3.2 Results

This section presents the results of the test of the analytical algorithm of phase diversity developed in the frame of this project and described above.

3.2.1 Algorithm test

To test the new algorithm, we use simulated PSFs, see Figure 3.1. The simulated PSFs are computed using eqt. (3.1). We assume a unitary amplitude over the entire

[1]Documentation found at https://docs.scipy.org/doc/numpy/reference/generated/numpy.linalg.lstsq.html#numpy.linalg.lstsq

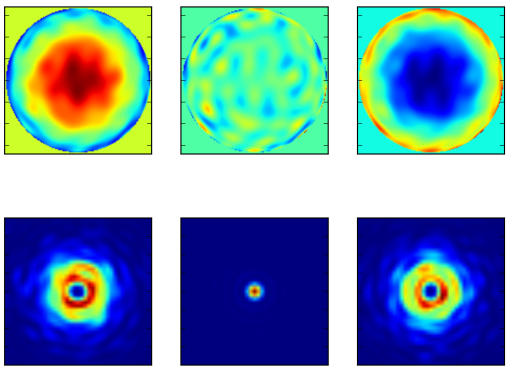


FIGURE 3.1: Example of simulated phases and their PSFs to test the analytical algorithm, $j_{min} = 4$ and $j_{max} = 30$. On the phases plot, first line, the image is not entirely shown, only from pixel 132 to 268 out of 400 pixels, for clarity. The PSFs are zoomed x10 (pixels 180 to 220) in order to be able to compare the spots. From left to right the PSFs are the defocused one, $-\delta\phi$, the focused one and the other defocused one, $\delta\phi$. The wavefront RMS error is set to 30 nm. The imaging system simulates the one we have on the optical bench composed by a 3.6 mm pupil and a focal length of 80 mm at a wavelength of 637.5 nm. The size, N, of the PSF is 400 and the pixel size is 5.3 μm.

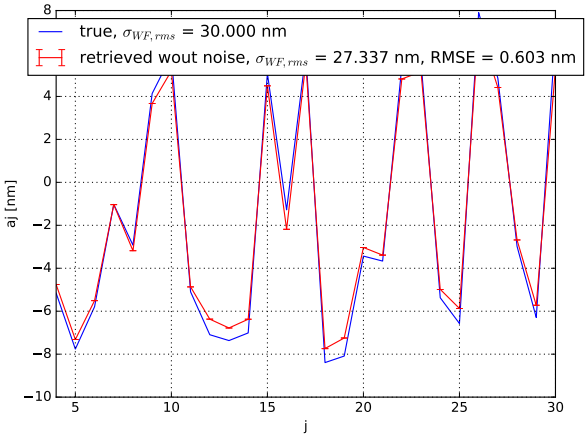


FIGURE 3.2: Zernike coefficient a_j in nm as a function of j. The blue line are the true a_j's and the red line represent the retrieved one. The phase and the 3 PSFs used to run the retrieval are the ones presented in Figure 3.1. $j_{min} = 4$ and $j_{max} = 30$.

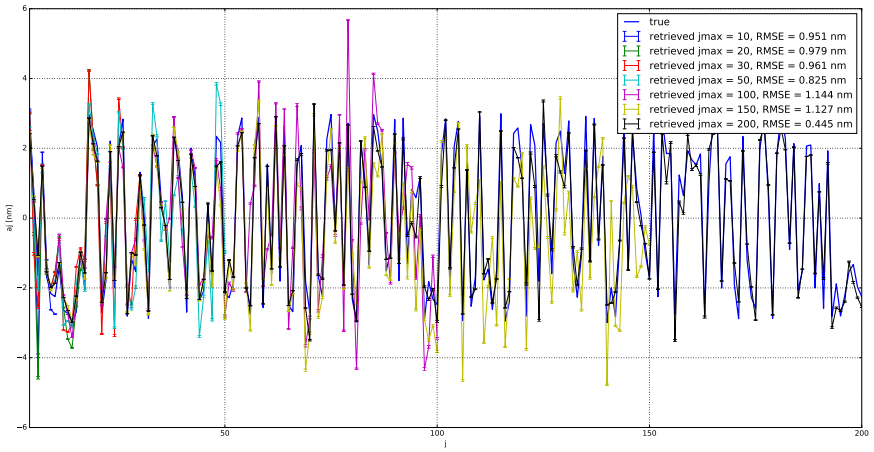


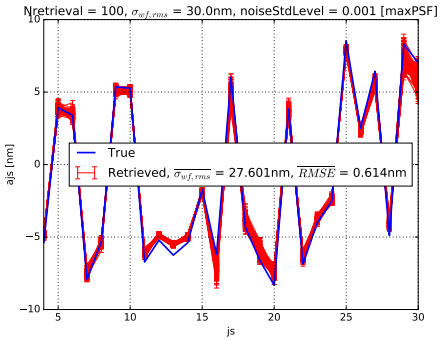
FIGURE 3.3: Zernike coefficient a_j in nm as a function of j. The blue thick line are the true a_j's, the wavefront has a RMS error of 30 nm and $j_{max} = 200$. The other lines are the retrieved Zernike coefficients with different j_max values.

pupil, as explained above and we construct the phase by attributing random values to Zernike coefficients from j_{min} to j_{max} and the two defocused PSFs have a 2π P2V dephasing. The number of pixel is N = 400 and the pixel size is 5.3 μm, same as the Ximea Camera. The focal length of the exit pupil is 80 mm and its diameter is set to 3.2 mm. The system simulates the optical system we used in the phase diversity experiment. The implementation can be seen in Appendix A.1.6.

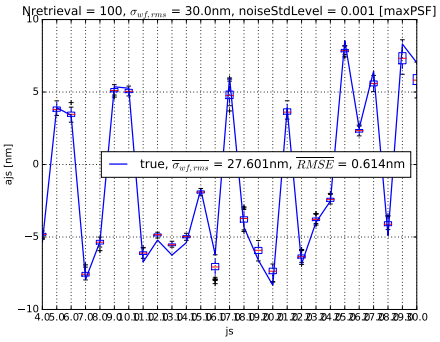
Using the PSFs shown in Figure 3.1, we are able to reconstruct the phase of the incoming wavefront onto the pupil. Figure 3.2 shows the true Zernike coefficient as well as the retrieved one. The correspondence is good. There is a RMS error of 0.603 nm between the retrieved coefficient and the true ones. And the wavefront RMS error retrieved is 2.663 nm to small. This is due to the fact that our algorithm is based on an approximation that causes the retrieval to underestimate the aberrations present.

As for the ONERA algorithm, we want to know if j_{max} has an influence on the results. In order to study its effect, a phase with $j_{max} = 200$ is used. Retrievals are done for $j_{max} = 10, 20, 30, 50, 100, 150, 200$. The results are presented in Figure 3.3. There is two interesting thing with those results. First, we can see that j_{max} has a negligible influence on the retrieval. The RMSE is similar for all retrieval except for $j_{max} = 200$. This leads to the second interesting result. The phase diversity algorithm is more precise if j_{max} used for the retrieval is equal to the one used to create the simulated PSFs. Indeed, when j_{max} used for the retrieval is smaller than the one used to create the PSFs, in other words when we retrieve less Zernike coefficients than the number of aberrations present in the wavefront, the results tends to overestimate the a_j's. This might be due to the fact that there is not enough polynomials to amount for the total budget of aberrations.

Figure 3.2 and 3.3 show the results of phase retrievals computed with noiseless PSFs. Given that in reality detectors have acquisition noise. To be as close to reality as possible with the simulated PSFs, we need to introduce noise into them and study its effect on the retrieval. We assume that the noise is white Gaussian as it is simpler



(A) Zernike coefficient as a function of j. The blue line represents the true a_j's . The red lines correspond to the 100 different retrievals computed with the 100 noisy PSFs. $j_{min} = 4$ and $j_{max} = 30$.



(B) Boxplots representing the statistics of the 100 Zernike coefficients as a function of j. The blue line represents the true ones. $j_{min} = 4$ and $j_{max} = 30$.

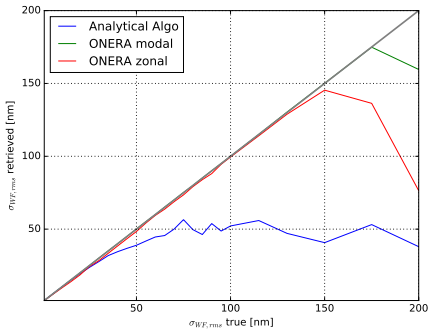
FIGURE 3.4: Noise study results. Noise level is 0.001. The wavefront RMS error is 30 nm.

than adding Poisson noise and it is a good approximation. The random introduction of noise is done with a numpy function called `numpy.random.normal()`[2].

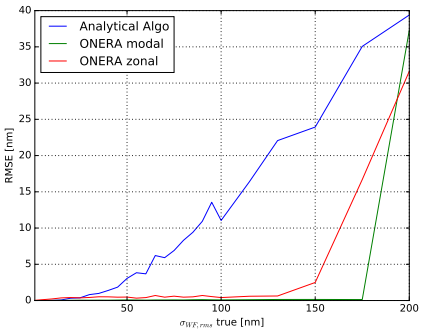
Figure 3.4a and 3.4b presents the results of the 100 phase retrievals computed with the 100 different noisy PSFs. The comparison between the true Zernike coefficients and the retrieved ones is good but as expected the performance of the algorithm is decreased. The mean RMSE of the 100 retrievals is equal to 0.715 nm, which is comparable to the one we have using noiseless PSFs. Plus, the boxplots in Figure 3.4b shows that the spread due to the noise is clearly less than 1 nm, $\bar{\sigma}_{ajs} = 0.22$ nm, which demonstrates that the noise do not affect the retrieval method. The results are similar for other wavefront RMS error and different j_{max}.

Another important point to look at is the domain of validity of the phase retrieval method. Indeed as explained in section 3.1, the analytical development used is possible only by assuming weak phase aberrations in order to enable the expansion of the phase term. In order to constrain the validity domain, a series of phase retrievals is computed on noise less PSFs with random phase having a RMS wavefront error going from 1 to 200 nm. The comparison between the wavefront RMS errors retrieved and the true ones is visible in Figure 3.5a. We focus first on the blue lines, which represent the analytical algorithm. As expected, the method is underestimating the phase aberrations due to the approximation on which the phase retrieval is based, see eqt. (3.3). The retrieval method is off by more than 10% and saturates at $\sim$ 50 nm when retrieving the aberrations of a wavefront having a true RMS wavefront error above 50 nm. Also the $\sim$ 50 nm threshold is the RMS wavefront error above which the RMSE of the retrieved Zernike becomes significant as seen in Figure 3.5b. To have an idea of the performance of the analytical algorithm, the ONERA algorithm modal and zonal RMS wavefront errors retrieved are also shown in Figure 3.5a. They are both far better than the analytical algorithm. They go up to $\sim$ 150 nm really accurately, but above the two retrieval methods also underestimate the RMS wavefront error. So the approximation limits considerably the range of validity of the method.

[2] https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.random.html



(A) RMS wavefront error retrieved as a function of the true one. The grey line is the one to one line. The blue line correspond to the analytical algorithm and the green and red lines correspond to the ONERA algorithm, modal and zonal retrieval respectively. $j_{min} = 4$ and $j_{max} = 30$.



(B) RMSE of the Zernike coefficient retrieved with respect to the true ones as a function of the true RMS wavefront error. The blue line correspond to the analytical algorithm and the green and red lines correspond to the ONERA algorithm, modal and zonal retrieval respectively. $j_{min} = 4$ and $j_{max} = 30$.

FIGURE 3.5: Validity domain study results.

3.2.2 Recursive approach

The tests have shown that the analytical phase diversity algorithm works well for wavefront having weak aberrations up to 50 nm of RMS error. But in order to use it on a real optical system to correct the static aberrations present, the algorithm should be able to at least correct wavefront with 100-120 nm RMS error. An idea comes then to mind to use the phase retrievals recursively as the aberration phenomenon is linear. Indeed, identifying aberrations in the system and correct them using the deformable mirror present on the telescope, we could reduce the aberrations in the system recursively.

To simulate the deformable mirror, the retrieved Zernike coefficients at each iteration are subtract to the ones used to compute the PSFs as a deformable mirror would remove the aberrations retrieved by the phase diversity. The results of the recursive approach on a wavefront having a RMS error of 130 nm are presented in Figure 3.6. The dashed line confounded with the thick blue line represent the final result of the recursive phase retrieval. The other thin lines present the different Zernike coefficients retrieved at each iteration. For a 130 nm RMS wavefront, the method needs 6 iterations to converge to the solution. The comparison between the wavefront RMS errors retrieved and the true ones as well as the RMSE as a function of the wavefront RMS error are visible in Figures 3.7a and 3.7b. The recursive phase diversity allows to extend the validity domain by 100 nm. The wavefront RMS errors retrieved is correct up to 150 nm and then the method does not work systematically any more. The RMSE shows it also clearly as it is nearly zero and then explodes at 150 nm.

3.2.3 Discussion

The analytical algorithm works as expected. The phase retrieval is correctly done for weak aberrations using simulated PSFs, see Figure 3.2. The test of the algorithm were also satisfying. As for the ONERA algorithm, the number of Zernike over which the reconstruction is done do not influence significantly the results. There is

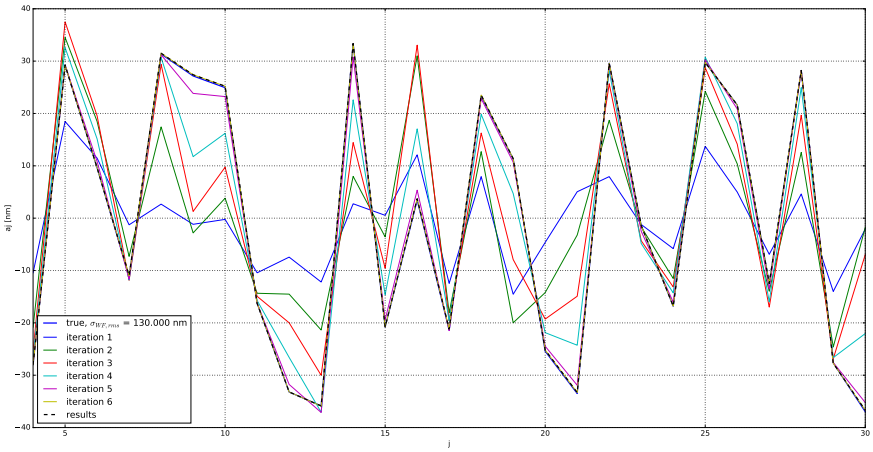
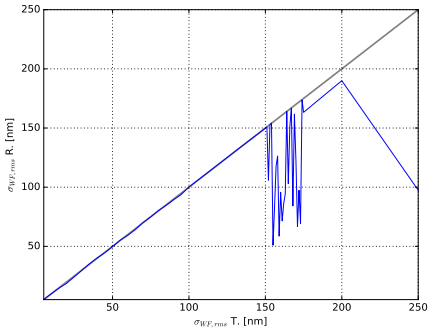
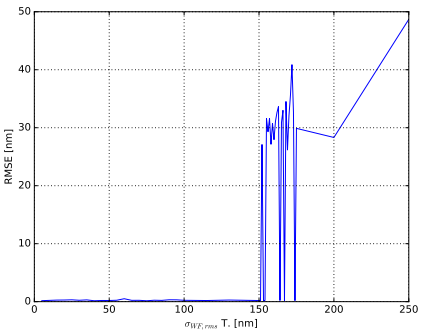


FIGURE 3.6: Zernike coefficient as a function of j. The thick blue line represents the true a_j's, the dashed line the retrieved ones after the 6 iterations. The other coloured thin lines represent the retrieved coefficient at each iteration.



(A) RMS wavefront error retrieved using the recursive approach as a function of the true one. The grey line is the one to one line. $j_{min} = 4$ and $j_{max} = 30$.



(B) RMSE of the Zernike coefficient retrieved, using the recursive approach, with respect to the true ones as a function of the true RMS wavefront error. $j_{min} = 4$ and $j_{max} = 30$.

FIGURE 3.7: Results of the recursive approach

still an interesting behaviour intrinsic to the retrieval method. If the number of Zernike is over which the reconstruction is done is smaller than the number of aberrations present in the PSFs, the algorithm tends to overestimate the Zernike coefficient to counter-balance the lost of components available. The introduction of noise in the PSFs to simulate the effect of the detector does not influence significantly the retrieval, and the mean spread on the a_j values is 0.22 nm. So the algorithm works well and its behaviour is really good with respect to noise and j_{max}.

The only down side of the algorithm is its validity domain. The approximation used to expand the phase exponential in the phasor limits significantly the range of RMS wavefront error, the algorithm can reconstruct. Figure 3.5a shows that it is usable for $\sigma_{WF,rms} < 50$ nm. This is a third of the ONERA algorithm validity domain.

To increase the validity domain the method, a new approach is looked into. Since the telescope on which this algorithm will be used has a deformable mirror, the latter allows to correct the aberrations present in the system recursively, using the linearity of the phenomenon. Which means that we use the algorithm more than once to retrieve bigger aberrations. The recursive phase diversity simulation gives promising results as we triple the validity domain. At 150 nm, non linear effect enter in play and disturb the retrieval and the algorithm do not converge each time. This result is really good since we are now as good as the ONERA algorithm on simulated PSFs.

The next step would be to test the algorithm using PSFs acquired on the optical bench and if possible introduce a deformable mirror to also test the recursive approach. But as explained in section 2.4.4, the experiment does not work as expected and some preliminary work needs to be undertaken before any phase diversity can be tested.

Conclusion

The wavefront reconstruction is at the forefront of the development of astronomical instruments nowadays. Having bigger and bigger telescope will not bring a lot without methods to counter and correct the effect of the atmosphere. Both the VLT and the ELT as well as the DAG telescope use and will use wavefront sensing and reconstruction to correct the aberrations present and improve the resolution up to the diffraction limit. At the moment phase diversity is still not widely use but its advantages are certain regarding the NCPA and its principle simplicity.

In our work, we conducted an experiment to test the phase diversity algorithm developed at ONERA by Mugnier, Blanc, and Idier (2006). The results are mixed. The test of the algorithm regarding noise level and j_{max} are interesting. We discovered that, as expected, the noise level in the PSFs has its importance especially in order to reconstruct wavefronts. The zonal method shows that under a noise level of ~ 0.00009, the high spatial frequencies of the noise are smoothed out and cancelled. For the values of the Zernike coefficient retrieved, the spread is generally small and the noise level is not as critical as to reconstruct the wavefront.

We then computed a Zemax model of our optical system to introduce calibrated astigmatism with a parallel plane plate in Plexiglas. The results are not as expected. The astigmatism retrieved by the phase diversity follows the right behaviour, but there is an offset. Trying to explain it we compared the vertical coma retrieved with the Zemax model, and once again the results were off. We concluded that the most plausible explanation is either that there is an alignment problem or that there is a problem with the usage of the ONERA phase diversity.

The comparison between the Shack-Hartmann and the phase diversity is also not successful. The two wavefront retrieval method shows aberrations of the same amplitude but we have more than 25 nm of difference between the two, which is significant. And we can note conclude anything about the phase diversity as we are unsure about the Shack-Hartmann results after comparing them with the Zemax astigmatism model.

In order to improve the experiment, we would need to take a step back and be sure that we have a way to characterize the system aberrations before trying to use the phase diversity. This means that in the future, we would need to be sure of the Shack-Hartmann results.

After having tested the ONERA algorithm, the aim was to develop our own phase diversity. It works as expected. And the behaviour with respect to noise and j_{max} are satisfying. An interesting result is the overestimation of the Zernike coefficients when the number of Zernike used by the retrieval is smaller than the one used to construct the simulated PSFs. This overestimation comes from the fact that the algorithm counter-balance for the lost of available components as there is a definite budget of aberrations to spread over a smaller number of Zernike. The validity

domain of the method is limited as the method is based on a strong approximation. The method is usable up to a wavefront RMS error of 50 nm. Above the difference becomes too important to be trustworthy.

A recursive approach has been tested in order to increase the validity domain of the algorithm. This method could be used on a system having a deformable mirror. Th results are really satisfying and we succeeded to multiply by three the validity domain the analytical phase diversity. This is similar to the validity domain of the ONERA algorithm.

Now, before implementing this algorithm on any telescope, we would need to first test it on real PSFs and characterize it as we did for the ONERA algorithm. This could be done after having improved the phase diversity experiment as said above.

In conclusion, this work was really interesting and I learned the hard way that in science research most of the time it does not go as planned. It was really amazing to do experimental work, as it was the first time I worked as much in the lab. The development of the phase diversity algorithm was a bit more common. But I learned a lot about discrete Fourier transform and their specificity. It has been a great experience. There is still plenty to do especially improving the laboratory experiment in order to correctly test the ONERA algorithm and especially test our own phase diversity on real data.

Appendix A

Python Code

A.1 Phase Diversity analytical algorithm code

A.1.1 phaseDiversity3PSFs.py

```
#Class phaseDiversity object to retrieve the phase in the pupil from two psfs in/out of focus

import numpy as np
import fs
import myExceptions
import PSF as psf

class phaseDiversity3PSFs(object):

    def __init__(self,inFoc,outFocpos,outFocneg,deltaZ,lbda,pxsize,F,pupilRadius,jmin,jmax):
#
#       input:
#       inFoc,outFocpos,outFocneg are the 3 squared PSFs data, (focused, defocused positiv and defocused negative)
#       deltaZ is the displacement of the detector to acquire the two defocused PSFs
#       lbda is the wavelength of the incoming light
#       pxsize is the pixel size of the detector
#       F is the focal length of the imaging system
#       pupilRadius is the radius of the exit pupil
#       jmin and jmax gives the boundary on the js to retrieve

        print 'phaseDiversity ...'

        #PSF
        self.inFoc = inFoc
        self.outFocpos = outFocpos
        self.outFocneg = outFocneg
        shapeinFoc = np.shape(self.inFoc)
        shapeoutFocpos = np.shape(self.outFocpos)
        shapeoutFocneg = np.shape(self.outFocneg)
        if shapeinFoc == shapeoutFocpos and shapeinFoc == shapeoutFocneg:
            self.shape = shapeinFoc
        else:
            raise myExceptions.PSFssizeError('the shape of the in/out PSFs is not the same',[shapeinFoc,shapeoutFocpos])
        if shapeinFoc[0]==shapeinFoc[1] and np.mod(shapeinFoc[0],2)==0:
            self.N = shapeinFoc[0]
```

```
36          else:
37              raise myExceptions.PSFssizeError('Either PSF is not square or
                    mod(N,2) != 0',shapeinFoc)
38
39          # properties
40          self.deltaZ = deltaZ
41          self.lbda = lbda
42          self.pxsize = pxsize
43          self.F = F
44          self.pupilRadius = pupilRadius
45          self.dxp = self.F*self.lbda/(self.N*self.pxsize)
46          self.rad = int(np.ceil(self.pupilRadius/self.dxp))
47          if 2*self.rad > self.N/2.:
48              raise myExceptions.PupilSizeError('Npupil (2*rad) is bigger than
                    N/2 which is not correct for the fft computation',[])
49          self.NyquistCriterion()
50          self.jmin = jmin
51          self.jmax = jmax
52          self.oddjs = fs.getOddJs(self.jmin,self.jmax)
53          self.evenjs = fs.getEvenJs(self.jmin,self.jmax)
54
55          # result computation
56          self.result = self.retrievePhase()
57
58      def NyquistCriterion(self):
59          deltaXfnyq = 0.5*self.lbda/(2*self.pupilRadius)
60          deltaXf = self.pxsize/self.F
61          if deltaXfnyq < deltaXf : raise myExceptions.NyquistError('the
                system properties do not respect the nyquist criterion',[])
62
63      def retrievePhase(self):
64          y1,A1 = self.initiateMatrix1()
65          results1 = np.linalg.lstsq(A1,y1)
66          ajsodd = results1[0]
67          A1tA1inv= np.linalg.inv(np.matmul(np.transpose(A1),A1))
68          n_p = y1.size-1
69          ste1 = np.sqrt(results1[1]/n_p * np.diagonal(A1tA1inv))
70
71          deltaphi =
                fs.deltaPhi(self.N,self.deltaZ,self.F,2*self.pupilRadius,self.lbda,self.dxp)
72          y2,A2 = self.initiateMatrix2(deltaphi)
73          results2 = np.linalg.lstsq(A2,y2)
74          ajseven = results2[0]
75          A2tA2inv= np.linalg.inv(np.matmul(np.transpose(A2),A2))
76          n_p = y2.size-1
77          ste2 = np.sqrt(results2[1]/n_p * np.diagonal(A2tA2inv))
78
79          js = np.append(self.oddjs,self.evenjs)
80          ajs = np.append(ajsodd,ajseven)
81          ajsSte = np.append(ste1,ste2)
82
83          Ixjs = np.argsort(js)
84          result = {'js': js[Ixjs], 'ajs': ajs[Ixjs], 'ajsSte': ajsSte[Ixjs]}
                #,'wavefront':phase}
85          return result
86
87      def initiateMatrix1(self):
```

```
88          deltaPSFinFoc = self.CMPTEdeltaPSF()
89          y1 = fs.y1(deltaPSFinFoc)
90          A1 = np.zeros((self.N**2,len(self.oddjs)))
91          for ij in np.arange(len(self.oddjs)):
92              phiJ = fs.f1j(self.oddjs[ij],self.N,self.dxp,self.pupilRadius)
93              A1[:,ij] = phiJ
94          return y1,A1
95
96      def initiateMatrix2(self,deltaphi):
97          deltaPSFoutFocpos,deltaPSFoutFocneg = self.CMPTEdeltaPSF(self.deltaZ)
98          y2 =
                fs.y2even(fs.getEvenPart(deltaPSFoutFocpos),fs.getEvenPart(deltaPSFoutFocneg))
99          A2 = np.zeros((self.N**2,len(self.evenjs)))
100         for ij in np.arange(len(self.evenjs)):
101             phiJ =
                    fs.f2jeven(self.evenjs[ij],self.N,deltaphi,self.dxp,self.pupilRadius)
102             A2[:,ij] = phiJ
103         return y2,A2
104
105     def CMPTEdeltaPSF(self,deltaZ=[]):
106         if not deltaZ:
107             PSF = psf.PSF([1],[0],self.N,self.dxp,self.pupilRadius)
108             return PSF.Sp**2*self.inFoc - PSF.Sp**2*PSF.PSF
109         else:
110             P2Vdephasing =
                    np.pi*self.deltaZ/self.lbda*(2*self.pupilRadius/self.F)**2/4.
111             a4 = P2Vdephasing/2./np.sqrt(3)
112             PSF = psf.PSF([4],[a4],self.N,self.dxp,self.pupilRadius)
113             return [PSF.Sp**2*self.outFocpos -
                    PSF.Sp**2*PSF.PSF,PSF.Sp**2*self.outFocneg -
                    PSF.Sp**2*PSF.PSF]
```

A.1.2 fs.py

```
1  #functions to compute the matrix element of the system of equations Ax = b
2  import zernike as Z
3  import phasor as ph
4  import numpy as np
5
6  def f1j(j,N,dxp,pupilRadius): # 1: phij's of matrix A to find a_j odd
7      Zj = Z.calc_zern_j(j,N,dxp,pupilRadius)
8      FFTZj = scaledfft2(Zj,dxp)
9
10     Lp = N*dxp
11     xp = np.arange(-Lp/2,Lp/2,dxp)
12     yp = xp
13
14     [Xp,Yp]=np.meshgrid(xp,yp)
15
16     pupil = np.float64(np.sqrt(Xp**2+Yp**2)<=pupilRadius)
17     FFTPupil = scaledfft2(pupil,dxp)
18
19     return np.ravel(2 * np.real(FFTPupil) * np.imag(FFTZj))
20
```

```
21 def f2j(j,N,jsodd,ajsodd,deltaphi,dxp,pupilRadius): # 2: phij's of matrix A
       to find a_j even
22     #Get the jth zernike polynomials values on a circular pupil of radius rad
23     Zj = Z.calc_zern_j(j,N,dxp,pupilRadius)
24
25     #compute the different 2Dfft given in the equations of deltaPSF
26     cosZj = np.cos(deltaphi)*Zj
27     sinZj = np.sin(deltaphi)*Zj
28     FFTcosZj = scaledfft2(cosZj,dxp)
29     FFTsinZj = scaledfft2(sinZj,dxp)
30
31     #odd phase
32     oddPhasor = ph.phasor(jsodd,ajsodd,N,dxp,pupilRadius)
33     oddPhase = oddPhasor.phase
34     pupil = oddPhasor.pupil
35     cosOddPhase = pupil*np.cos(deltaphi)*oddPhase
36     sinOddPhase = pupil*np.sin(deltaphi)*oddPhase
37     FFTcosOddPhase = scaledfft2(cosOddPhase,dxp)
38     FFTsinOddPhase = scaledfft2(sinOddPhase,dxp)
39
40     #2Dfft of pupil function times sin(deltaPhi) and cos(deltaPhi)
41     pupilSin = pupil*np.sin(deltaphi)
42     pupilCos = pupil*np.cos(deltaphi)
43     FFTPupilSin = scaledfft2(pupilSin,dxp)
44     FFTPupilCos = scaledfft2(pupilCos,dxp)
45
46     FFTsinZjOddPhase = scaledfft2(sinZj*oddPhase,dxp)
47     FFTcosZjOddPhase = scaledfft2(cosZj*oddPhase,dxp)
48
49     return np.ravel(2*np.imag(np.conj(FFTcosZj) * FFTsinOddPhase + FFTsinZj
           * np.conj(FFTcosOddPhase)
50             - np.conj(FFTPupilCos) * FFTsinZjOddPhase + np.conj(FFTPupilSin)
                 * FFTcosZjOddPhase))
51
52 def f2jeven(j,N,deltaphi,dxp,pupilRadius): # 2: phij's of matrix A to find
       a_j even
53     #Get the jth zernike polynomials values on a circular pupil of radius rad
54     Zj = Z.calc_zern_j(j,N,dxp,pupilRadius)
55     Phasor = ph.phasor([1],[0],N,dxp,pupilRadius)
56     pupil = Phasor.pupil
57     #compute the different 2Dfft given in the equations of deltaPSF
58     cosZj = pupil*np.cos(deltaphi)*Zj
59     sinZj = pupil*np.sin(deltaphi)*Zj
60     FFTcosZj = scaledfft2(cosZj,dxp)
61     FFTsinZj = scaledfft2(sinZj,dxp)
62     #2Dfft of pupil function times sin(deltaPhi) and cos(deltaPhi)
63     pupilSin = pupil*np.sin(deltaphi)
64     pupilCos = pupil*np.cos(deltaphi)
65     FFTPupilSin = scaledfft2(pupilSin,dxp)
66     FFTPupilCos = scaledfft2(pupilCos,dxp)
67
68     return
           np.ravel(-4*np.real(np.conj(FFTPupilCos)*FFTsinZj-np.conj(FFTPupilSin)*FFTcosZj))
69
70 def y1(deltaPSFinFoc): #1: yi's of y to find a_j odd
71     #compute the odd part of delta PSF
72     oddDeltaPSF = getOddPart(deltaPSFinFoc)
```

```
73      return np.ravel(oddDeltaPSF)
74
75  def y2(deltaPSFoutFoc,N,jsodd,ajsodd,deltaphi,dxp,pupilRadius): # 2: yi's
        of y to find a_j even
76      oddDeltaPSF = getOddPart(deltaPSFoutFoc)
77      oddPhasor = ph.phasor(jsodd,ajsodd,N,dxp,pupilRadius)
78      oddPhase = oddPhasor.phase
79
80      pupilSin = oddPhasor.pupil*np.sin(deltaphi)
81      pupilCos = oddPhasor.pupil*np.cos(deltaphi)
82      FFTPupilSin = scaledfft2(pupilSin,dxp)
83      FFTPupilCos = scaledfft2(pupilCos,dxp)
84      cosOddPhase = oddPhasor.pupil*np.cos(deltaphi)*oddPhase
85      sinOddPhase = oddPhasor.pupil*np.sin(deltaphi)*oddPhase
86      FFTcosOddPhase = scaledfft2(cosOddPhase,dxp)
87      FFTsinOddPhase = scaledfft2(sinOddPhase,dxp)
88
89      return np.ravel(oddDeltaPSF -
            2*np.real(np.conj(FFTPupilCos))*np.imag(FFTcosOddPhase)
90              - 2*np.real(np.conj(FFTPupilSin))*np.imag(FFTsinOddPhase))
91
92  def y2even(deltaPSFoutFocpos,deltaPSFoutfocneg): #3: yi's of y to find a_j
        even
93      return np.ravel(deltaPSFoutFocpos-deltaPSFoutfocneg)
94
95  #Other
        functions----------------------------------------------------------------------
96  def flipMatrix(M):
97      #flip 2D matrix along x and y
98      dimM = (np.shape(M))[0]
99      Mflipped = np.flipud(np.fliplr(M))
100     if np.mod(dimM,2)==0:
101         Mflipped = np.roll(Mflipped,1,axis=0)
102         Mflipped = np.roll(Mflipped,1,axis=1)
103     return Mflipped
104 def getOddPart(F):
105     oddF = (F - flipMatrix(F))/2.
106     return oddF
107 def getEvenPart(F):
108     evenF = (F + flipMatrix(F))/2.
109     return evenF
110 def getOddJs(jmin,jmax):
111     js = []
112     for j in np.arange(jmin,jmax+1):
113         [n,m] = Z.noll_to_zern(j)
114         if np.mod(m,2) == 1:
115             js.append(j)
116         else:
117             continue
118     return np.array(js)
119 def getEvenJs(jmin,jmax):
120     js = []
121     for j in np.arange(jmin,jmax+1):
122         [n,m] = Z.noll_to_zern(j)
123         if np.mod(m,2) == 0:
124             js.append(j)
125         else:
```

```
            continue
    return np.array(js)
def deltaPhi(N,deltaZ,F,D,wavelength,dxp):
    Zj = Z.calc_zern_j(4,N,dxp,D/2.)
    P2Vdephasing = np.pi*deltaZ/wavelength*(D/F)**2/4.
    a4defocus = P2Vdephasing/2/np.sqrt(3)
    return a4defocus*Zj
def cleanZeros(A,threshold):
    A[np.abs(A) < threshold] = 0.
    return A
def scaledfft2(f,dxp):
    return np.fft.ifftshift(np.fft.fft2(np.fft.fftshift(f)))*dxp**2
def RMSE(estimator,target):
    return np.sqrt(np.mean((estimator-target)**2))
def BIAS(estimator,target):
    return np.mean((estimator-target))

def RMSwavefrontError(js,ajs):
    if 1 in js:
        return np.sqrt(np.sum(ajs**2)-ajs[js==1]**2)
    else:
        return np.sqrt(np.sum(ajs**2))
```

A.1.3 myExceptions.py

```
class PSFssizeError(ValueError):
   '''Raise when the size of the PSFs are not correct'''
   def __init__(self, message, foo, *args):
       self.message = message
       self.foo = foo
       super(PSFssizeError, self).__init__(message, foo, *args)

class NyquistError(ValueError):
   '''Raise when the PSFs properties do not respect the nyquist criterion'''
   def __init__(self, message, foo, *args):
       self.message = message
       self.foo = foo
       super(NyquistError, self).__init__(message, foo, *args)

class PupilSizeError(ValueError):
    '''Raise when Npupil is bigger than the size of the PSF N'''
    def __init__(self, message, foo, *args):
        self.message = message
        self.foo = foo
        super(PupilSizeError, self).__init__(message, foo, *args)
```

A.1.4 zernike.py

```
import numpy as np

def calc_zern_j(j, N, dxp, pupilRadius):

    Lp = N*dxp

```

```
    if (j <= 0):
        return {'modes':[], 'modesmat':[], 'covmat':0, 'covmat_in':0,
            'mask':[[0]]}
    if (N <= 0):
        raise ValueError("N should be > 0")
    if (dxp <= 0 or dxp >= N):
        raise ValueError("dxp should be > 0 or < N")

    xp = np.arange(-Lp/2,Lp/2,dxp)
    yp = xp
    [Xp,Yp]=np.meshgrid(xp,yp)
    r = np.sqrt(Xp**2+Yp**2)
    r = r*(r<=pupilRadius)/pupilRadius
    pup = np.float64(np.sqrt(Xp**2+Yp**2)<=pupilRadius)
    theta = np.arctan2(Yp,Xp)

    Zj = zernike(j,r,theta)*pup
    return Zj

def zernike(j,r,theta):
    n,m = noll_to_zern(j)
    nc = (2*(n+1)/(1+(m==0)))**0.5
    #nc = (2*(n+1)/(1+(m==0)))**0.5
    if (m > 0): return nc*zernike_rad(m, n, r) * np.cos(m * theta)
    if (m < 0): return nc*zernike_rad(-m, n, r) * np.sin(-m * theta)
    return nc*zernike_rad(0, n, r)

def zernike_rad(m, n, r):
    if (np.mod(n-m, 2) == 1):
        return r*0.0
    wf = r*0.0
    for k in range((n-m)/2+1):
        wf += r**(n-2.0*k) * (-1.0)**k * fac(n-k) / ( fac(k) * fac(
            (n+m)/2.0 - k ) * fac( (n-m)/2.0 - k ) )
    return wf

def noll_to_zern(j):
    j = int(j)
    if (j == 0):
        raise ValueError("Noll indices start at 1, 0 is invalid.")
    n = 0
    j1 = j-1
    while (j1 > n):
        n += 1
        j1 -= n
    m = (-1)**j * ((n % 2) + 2 * int((j1+((n+1)%2)) / 2.0 ))
    return (n, m)
def fac(n):
    if n == 0:
        return 1
    else:
        return n * fac(n-1)
```

A.1.5 phasor.py

```python
#class phasor
import numpy as np
import zernike as Z

class phasor(object):

    def __init__(self,js=[1],ajs=[0],N=800,dxp=1,pupilRadius = 200):
        self.js = js
        self.ajs = ajs
        self.N = N
        self.dxp = dxp
        self.pupilRadius = pupilRadius

        self.pupil = self.constructPupil()

        self.phase = self.constructPhase()
        self.phasor = self.pupil*np.exp(-1j*self.phase)

    def constructPhase(self):
        phase = np.zeros((self.N,self.N))
        for ij, j in enumerate(self.js):
            Zj = Z.calc_zern_j(j,self.N,self.dxp,self.pupilRadius)
            phase += self.ajs[ij]*Zj
        return phase

    def constructPupil(self):
        Lp = self.N*self.dxp

        xp = np.arange(-Lp/2,Lp/2,self.dxp)
        yp = xp

        [Xp,Yp]=np.meshgrid(xp,yp)

        pup = np.float64(np.sqrt(Xp**2+Yp**2)<=self.pupilRadius)
        return pup
```

A.1.6 PSF.py

```python
import numpy as np
import phasor as ph
import fs

class PSF(object):

    def __init__(self,js=[1],ajs=[0],N=400,dxp=1.,pupilRadius = 67.):
        self.phasor = ph.phasor(js,ajs,N,dxp,pupilRadius) #phasor in the
            pupil
        self.Sp = np.sum(self.phasor.pupil)*dxp**2 #pupil surface
        self.FFTphasor = fs.scaledfft2(self.phasor.phasor,dxp) #fourier
            transform of the phasor
        self.PSF = np.abs(self.FFTphasor)**2/self.Sp**2 #PSF on the focal
            plane
        self.FFTpupil = fs.scaledfft2(self.phasor.pupil,dxp) #fourier
            transform of the pupil function which gives the perfect PSF
        self.perfectPSF = np.abs(self.FFTpupil)**2/self.Sp**2 #perfect PSF
```

```
14        self.deltaPSF = self.Sp**2*self.PSF-self.Sp**2*self.perfectPSF
              #deltaPSF
```

A.2 Acquisition Code: Ximea Camera

A.2.1 AlignementScriptXimeaCamera.py

```
1  ##Script to compute the FWHM of the beam on the camera averaging
2   #over "nbrImgAveraging" images and see which position minimizes it.
3
4  from ximea import xiapi
5  import numpy as np
6  from matplotlib import pyplot as plt
7  import scipy.optimize as opt
8  import datetime
9  import functionsXimea as fX
10 import seaborn as sns
11 import os
12 sns.set()
13 #%% instanciation
14
15 dataFolderPath = '...'
16 plotFolderPath = '...'
17 #create the matrix grid of the detector CCD
18 x = np.linspace(0,1280,1280)
19 y = np.linspace(0,1024,1024)
20 x, y = np.meshgrid(x, y)
21
22 #initial guess for the fit depending on the position of the beam in the CCD
23 initial_guess = [250, 481, 706, 3, 3] # [max PSF,y,x,sigmay,sigmax]
24
25 #number of image to average
26 nbrImgAveraging = 10
27
28 #%%data acquisition and treatment
29
30 #create instance for first connected camera
31 cam = xiapi.Camera()
32 #start communication
33 print('Opening camera...')
34 cam.open_device()
35 #settings
36 cam.set_imgdataformat('XI_MONO8') #XIMEA format 8 bits per pixel
37 cam.set_gain(0)
38 #create instance of Image to store image data and metadata
39 img = xiapi.Image()
40 #start data acquisition
41 print('Starting data acquisition...')
42 if cam.get_acquisition_status() == 'XI_OFF':
43         cam.start_acquisition()
44
45 cam.set_exposure(fX.determineUnsaturatedExposureTime(cam,img,[60,10000],1))
46
47 #instanciation for the while loop
48 answer ='y'
```

```
49 i=0
50 relativePos = []
51 data = []
52 data_fitted = []
53 FWHMx = []
54 FWHMy = []
55 x0 = []
56 y0 = []
57 sigmaX0 = []
58 sigmaY0 = []
59
60 while answer == 'y':
61
62     try:
63         relativePos.append(float(raw_input('What is the position on the
               screw [mm] ? ')))
64     except ValueError:
65         print('Not a float number')
66
67     [tmpdata,stdData] = fX.acquireImg(cam,img,nbrImgAveraging)
68     data.append(tmpdata)
69     #Fit the img data on the 2D Gaussian to compute the FWHM
70     print('Fitting 2D Gaussian...')
71     popt, pcov = opt.curve_fit(fX.TwoDGaussian, (x,y), data[i].ravel(), p0 =
           initial_guess)
72     print('Fitting done')
73
74     FWHMx.append(2*np.sqrt(2*np.log(2))*popt[3])
75     FWHMy.append(2*np.sqrt(2*np.log(2))*popt[4])
76     x0.append(popt[2])
77     sigmaX0.append(popt[4])
78     y0.append(popt[1])
79     sigmaY0.append(popt[3])
80
81     print 'Fig %d : (x,y) = (%3.2f,%3.2f), FWHM x = %3.2f, FWHM y = %3.2f'
           %(i,x0[i],y0[i],FWHMx[i],FWHMy[i])
82
83     data_fitted.append(fX.TwoDGaussian((x, y),
           popt[0],popt[1],popt[2],popt[3],popt[4]).reshape(1024, 1280))
84
85     #plot the beamspot
86     fig, ax = plt.subplots(1, 1)
87     ax.imshow(data[i], cmap=plt.cm.jet,origin='bottom',
88         extent=(x.min(), x.max(), y.min(), y.max()))
89     ax.contour(x, y, data_fitted[i], 5, colors='w',linewidths=0.8)
90     plt.xlim( (popt[2]-4*popt[4], popt[2]+4*popt[4]) )
91     plt.ylim( (popt[1]-4*popt[3], popt[1]+4*popt[3]) )
92     plt.show()
93
94     #ask if the person wants to acquire a new image to improve the alignement
95     pressedkey = raw_input('Do you want to acquire an other image [y (yes)
           or n (no)]: ')
96     if (pressedkey =='n'):
97         answer = pressedkey
98     #increase i
99     i+=1
100
```

```
#stop data acquisition
print('Stopping acquisition...')
cam.stop_acquisition()

#stop communication
cam.close_device()

#convert list to np.array
relativePos = np.array(relativePos)
data = np.array(data)
FWHMx = np.array(FWHMx)
FWHMy = np.array(FWHMy)
x0 = np.array(x0)
y0 = np.array(y0)
sigmaX0 = np.array(sigmaX0)
sigmaY0 = np.array(sigmaY0)

#plot the FWHM vs. relPos
fig, ax = plt.subplots(1,1)
ind = np.argsort(relativePos)
ax.plot(relativePos[ind],(np.sqrt(FWHMx**2+FWHMy**2))[ind])
ax.set_xlabel('Position [mm]')
ax.set_ylabel('FWHM [px]')
ax.grid()
date = datetime.datetime.today()
if not os.path.isdir(plotFolderPath):
    os.makedirs(plotFolderPath)
plt.savefig(plotFolderPath+date.strftime('%Y%m%d%H%M%S')+'FWHM_pos.pdf')
plt.savefig(plotFolderPath+date.strftime('%Y%m%d%H%M%S')+'FWHM_pos.png')


indOfMinFWHM = np.argmin(np.sqrt(FWHMx**2+FWHMy**2))

fig, axarr = plt.subplots(1,np.size(data,0))
#plot all the images besides each other
for iImg in ind:
    axarr[iImg].imshow(data[iImg], cmap=plt.cm.jet,origin='bottom',
        extent=(x.min(), x.max(), y.min(), y.max()))
#    axarr[iImg].contour(x, y, data_fitted[iImg], 5,
    colors='w',linewidths=0.8)
    axarr[iImg].set_xlim( (x0[iImg]-12, x0[iImg]+12) )
    axarr[iImg].set_ylim( (y0[iImg]-12, y0[iImg]+12) )
    axarr[iImg].set_yticklabels('',visible=False)
    axarr[iImg].set_xticklabels('',visible=False)
    axarr[iImg].set_title('%5.3f mm'%relativePos[iImg],fontsize=8)
    if iImg == indOfMinFWHM:
        axarr[iImg].set_frame_on(True)
        for pos in ['top', 'bottom', 'right', 'left']:
            axarr[iImg].spines[pos].set_edgecolor('r')
            axarr[iImg].spines[pos].set_linewidth(2)
    else:
        axarr[iImg].set_frame_on(False)
plt.show()
date = datetime.datetime.today()

plt.savefig(plotFolderPath+date.strftime('%Y%m%d%H%M%S')+'ImgPSF.pdf')
plt.savefig(plotFolderPath+date.strftime('%Y%m%d%H%M%S')+'ImgPSF.png')
```

```
157
158
159 #save data
160 if not os.path.isdir(dataFolderPath):
161     os.makedirs(dataFolderPath)
162 date = datetime.datetime.today()
163 np.save(dataFolderPath+date.strftime('%Y%m%d%H%M%S')+'data.npy',data)
164 np.save(dataFolderPath+date.strftime('%Y%m%d%H%M%S')+'relativePos.npy',relativePos)
```

A.2.2 AcquisAndSaveXimea.py

```
1  #%% Script to acquire images average over nbrImgAveraging images and save
       them into fits file
2
3  from ximea import xiapi
4  import datetime
5  import functionsXimea as fX
6  import winsound
7  import numpy as np
8
9  #%%instanciation
       ----------------------------------------------------------------
10 #number of image to average
11 nbrImgAveraging = 5000
12 numberOfFinalImages = 1
13
14 #Cropping information
15 sizeImg = 256
16
17 #Parameter of camera and saving
18 folderPathCropped = 'data//cropped/20/'
19 darkFolderPathCropped = '.data/dark//cropped/20/'
20 folderPathFull = 'data/full/'
21 darkFolderPathFull = '/data/dark//full/'
22 nameCamera = 'Ximea'
23 focusPos = 11.63
24
25 #Sound
26 duration = 1000 # millisecond
27 freq = 2000 # Hz
28
29 #initial guess for the fit depending on the position of the beam in the CCD
30 initial_guess = [250, 468, 954, 3, 3] # [max PSF,y,x,sigmay,sigmax]
31
32 #-------------------------------------------------------------------------------
33 #%% data acquisition
       ------------------------------------------------------------
34
35 #Opening the connection to the camera
36 cam = xiapi.Camera()
37 cam.open_device()
38 cam.set_imgdataformat('XI_MONO8') #XIMEA format 8 bits per pixel
39 cam.set_gain(0)
40
41 img = xiapi.Image()
```

```
42 if cam.get_acquisition_status() == 'XI_OFF':
43     cam.start_acquisition()
44 #%% exposition
45 cond = 1
46 while bool(cond):
47     source = ''
48     winsound.Beep(freq, duration)
49     source = int(raw_input('Is the source turned on and at focus point
           (usually %5.3f mm) (yes = 1) ? '%focusPos))
50     if source == 1:
51         cond = 0
52     else:
53         print 'Please turn on the source and place the camera on the focus
               point (%5.3f mm)'%focusPos
54
55 if bool(source):
56     #Set exposure time
57     cam.set_exposure(fX.determineUnsaturatedExposureTime(cam,img,[1,10000],1))
58     #get centroid
59     centroid = fX.acquirePSFCentroid(cam,img,initial_guess)
60     print 'centroid at (%d, %d)' %(centroid[0],centroid[1])
61
62 #%%Acquire images at different camera position
63
64 acquire = 1
65 while bool(acquire):
66     cond = 1
67     while bool(cond):
68         dark = ''
69         winsound.Beep(freq, duration)
70         dark = int(raw_input('Is the source turned off (yes = 1) ? '))
71         if dark == 1:
72             cond = 0
73         else:
74             print 'Please shut down the source.'
75
76     winsound.Beep(freq, duration)
77     pos = float(raw_input('What is the position of the camera in mm focused
           (%5.3f mm) dephase 2Pi (pos+ = %5.3f mm, pos- = %5.3f) ?
           '%(focusPos,focusPos+3.19,focusPos-3.19)))
78
79     if bool(dark):
80         print 'Acquiring dark image...'
81         # Acquire dark images
82         [darkData,stdDarkData] = fX.acquireImg(cam,img,nbrImgAveraging)
83         print 'Cropping'
84         [darkdataCropped,stddarkDataCropped] =
               fX.cropAroundPSF(darkData,stdDarkData,centroid,sizeImg,sizeImg)
85         print 'saving'
86         fX.saveImg2Fits(datetime.datetime.today(),darkFolderPathCropped,nameCamera,darkdataCropped,
87         fX.saveImg2Fits(datetime.datetime.today(),darkFolderPathFull,nameCamera,darkData,stdDarkDat
88
89     #Acquire images -------------------------
90     cond = 1
91     while bool(cond):
92         source = ''
93         winsound.Beep(freq, duration)
```

```
94          source = int(raw_input('Is the source turned on (yes = 1) ? '))
95          if source == 1:
96              cond = 0
97          else:
98              print 'Please place turn on the camera'
99
100     if bool(source):
101         print 'Acquiring images...'
102         # Acquire focused images
103         for iImg in range(numberOfFinalImages):
104             imgNumber = iImg+1
105             print 'Acquiring Image %d'%imgNumber
106             [data,stdData] = fX.acquireImg(cam,img,nbrImgAveraging)
107             print 'Cropping'
108             [dataCropped,stdDataCropped] =
                    fX.cropAndCenterPSF(data-darkData,stdData+stdDarkData,sizeImg,initial_guess)
109             print 'Saving'
110             fX.saveImg2Fits(datetime.datetime.today(),folderPathCropped,nameCamera,dataCropped,
111             fX.saveImg2Fits(datetime.datetime.today(),folderPathFull,nameCamera,data-darkData,s
112
113     cond = 1
114     while bool(cond):
115         acquire = ''
116         winsound.Beep(freq, duration)
117         acquire = int(raw_input('Do you want to acquire at an other camera
                position (yes = 1, no = 0) ? '))
118         if acquire == 1:
119             cond = 0
120         elif acquire == 0:
121             cond = 0
122         else:
123              print 'please answer with 0 or 1 for no or yes, respectively'
124
125
126 ##Stop the acquisition
127 cam.stop_acquisition()
128 cam.close_device()
129
130 print 'Acquisition finished'
```

A.2.3 functionsXimea.py

```
1  import numpy as np
2  import pyfits
3  import os
4  import scipy.optimize as opt
5  #%% Functions
       ----------------------------------------------------------------------
6
7
8  #Create and save .fits from numpy array
9  def saveImg2Fits(date,folderPath,Detector,data,stdData,pos,nbrAveragingImg):
10
11     #date : datetime at which the data where taken
12     #folderPath : where to save the data$
```

```
13      #Detector : name of detector (ex:Ximea)
14      #data : np.array containing the image
15      #stdData: np.array containing the error on each pixel
16      #pos : the position of the camera on the sliding holder in mm
17
18      imgHdu = pyfits.PrimaryHDU(data)
19      stdHdu = pyfits.ImageHDU(stdData,name = 'imgStdData')
20      hdulist = pyfits.HDUList([imgHdu,stdHdu])
21
22      if not os.path.isdir(folderPath):
23          os.makedirs(folderPath)
24
25      hdulist.writeto(folderPath +
            date.strftime('%Y%m%d%H%M%S')+'_'+Detector+'_'+pos+'.fits')
26
27  def acquireImg(cam,img,nbrImgAveraging):
28      imgData = np.zeros([1024,1280])
29      stdData = np.zeros([1024,1280])
30      for iImg in range(nbrImgAveraging) :
31          #print iImg
32          cam.get_image(img)
33          imgTmpData = img.get_image_data_numpy()
34          imgData[:,:] += imgTmpData
35          stdData += imgTmpData*imgTmpData
36
37      stdData =
            np.sqrt((stdData-imgData*imgData/nbrImgAveraging)/(nbrImgAveraging-1))/(np.sqrt(nbrImgAvera
38      imgData = imgData/nbrImgAveraging
39
40      return [imgData,stdData]
41
42
43  def determineUnsaturatedExposureTime(cam,img,exposureLimit,precision):
44       exposureTimes = exposureLimit
45       if cam.get_acquisition_status() == 'XI_OFF':
46           cam.start_acquisition()
47
48       while np.absolute(np.diff(exposureTimes))>precision:
49           expTime2check = int(np.round(np.nanmean(exposureTimes)))
50           print 'Try expTime : %d [us]\n' %expTime2check
51           cam.set_exposure(expTime2check)
52           data = acquireImg(cam,img,10)[0]
53
54           if np.sum(data>250)>1:
55               exposureTimes[1] = int(np.ceil(np.nanmean(exposureTimes)))
56           else:
57               exposureTimes[0] = int(np.floor(np.nanmean(exposureTimes)))
58           print 'exposure time between %d and %d \n'
                %(exposureTimes[0],exposureTimes[1])
59
60       return int(np.floor(np.nanmean(exposureTimes)))
61
62  def TwoDGaussian((x, y), A, yo, xo, sigma_y, sigma_x):
63      g = A*np.exp( - ((x-xo)**2/(2*sigma_x**2) + ((y-yo)**2)/(2*sigma_y**2)))
64      return g.ravel()
65
66  def acquirePSFCentroid(cam,img,initial_guess):
```

```
    #create the matrix grid of the detector CCD

    data = acquireImg(cam,img,200)[0]
    centroid = getPSFCentroid(data,initial_guess)
    return centroid

def getPSFCentroid(data,initial_guess):

    x = np.linspace(0,1280,1280)
    y = np.linspace(0,1024,1024)
    x, y = np.meshgrid(x, y)
    print 'fitting'
    popt,pcov = opt.curve_fit(TwoDGaussian, (x,y), data.ravel(), p0 =
        initial_guess)
    print 'fitting done'
    return [popt[2],popt[1]]


def cropAndCenterPSF(data,stdData,size,initial_guess):
    Xextent = np.size(data,1)
    Yextent = np.size(data,0)

    centroid = getPSFCentroid(data,initial_guess)

    minMarge =
        np.min([centroid[0],centroid[1],Xextent-centroid[0],Yextent-centroid[1]])

    if minMarge>size/2:
        return cropAroundPSF(data,stdData,centroid,size,size)
    elif minMarge<size/2:
        return cropAroundPSF(data,stdData, centroid,2*minMarge,2*minMarge)


def cropAroundPSF(data,stdData,centroid,sizeX,sizeY):

    pxX =
        [int(np.floor(centroid[0])-np.ceil(sizeX/2)),int(np.floor(centroid[0])+np.ceil(sizeX/2
    pxY =
        [int(np.floor(centroid[1])-np.ceil(sizeY/2)),int(np.floor(centroid[1])+np.ceil(sizeY/2

    dataCropped = data[pxY[0]:pxY[1],pxX[0]:pxX[1]]

    stdDataCropped = stdData[pxY[0]:pxY[1],pxX[0]:pxX[1]]

    return [dataCropped,stdDataCropped]
```

Appendix B

IDL Code

B.1 ONERA phase diversity

B.1.1 Diversity.pro

```
;+
;FUNCTION
    DIVERSITY(PSF,DEFOCUS,LAMBDA,FDIST,PXSIZE,THRESHOLD,MODE[,D1=,D2=,JMAX=,/KECK,/LARGEHEX,/SHOW]
;
;  A function to retrieve the phase from a set of several in-focus and
;  out-of-focus PSF (at least two PSF) using a phase diversity algorithm.
;
;  ============================================================================
;  ======== Copyrights : PLEASE DO NOT DISTRIBUTE OUTSIDE OF WMKO ========
;  ======== The central tool, decoconj_marg.pro is a property of =========
;  ========== Laurent Mugnier et al. at ONERA, Paris, France =============
;  ============================================================================
;
;INPUTS name | type | unit
;
;  PSF | REAL 2D SQUARE ARRAY DIM = [N,N,M] | FREE
;  A suite of M PSF acquired along the optical axis. It is best to have
;  at least one PSF close to the focal plane. The PSF must have been
;  cleaned from bad pixels, cosmic etc, and back-ground
;  subtracted. Each PSF must be square (DIM NxN) and an even size.
;
;  NOTE ON PSF MATRIX SIZE N : N must be at least equal to 118. This is
;  required to ensure the minimum spatial resolution in the pupil plane
;  when reconstructing the phase. Indeed, there is a specification on
;  this spatial resolution which is coming from the AO OTF
;  computation, and the smallest size for the PSF is the one above.
;
;  DEFOCUS | REAL VECTOR WITH M COMPONENTS | MILLIMETERS
;  Defocus distance for PSF above. Must be of course of the same length as
;  the number M of PSF. For instance [0,3,5] for a set of 3 PSF. Positive
;  value means the PSF is after the telescope focal plane, along the
;  optical axis, in the direction of light propagation.
;  ************************
;  THE IDEAL VALUE FOR DZ IS DZ = LAMBDA * 8 * (F/D)^2, IN ABSOLUTE
;  VALUE, BUT IT DOES NOT HAVE TO BE EXACTLY THIS VALUE. HAVING PSF IN
;  INTERMEDIATE PLANES CAN HELP BUT IS NOT MANDATORY.
;  NOTE THAT DZ CAN BE NEGATIVE (PSF TAKEN ON BOTH SIDES OF THE FOCAL PLANE).
;  ************************
;
;  LAMBDA | REAL SCALAR > 0 | MICRONS
```

```
40 ;  The central wavelength of the filter associated with the PSF
41 ;  acquisition. We ignore the filter width here, so we assume
42 ;  it's a rather narrow filter. Of course if we have a laser
43 ;  beam then the lambda to give is the laser lambda, and there is
44 ;  no filter.
45 ;  >>>>>>>> DO NOT USE ANY OTHER VALUE OTHERWISE THE COMPUTATION <<<<<<<<
46 ;  >>>>>>>> IS TOTALLY MESSED UP - EVEN IF IT WILL CONVERGE !! <<<<<<<<<<
47 ;
48 ;  FDIST | REAL SCALAR > 0 | METERS
49 ;  Telescope optics focal distance. (150 m for Keck telescopes).
50 ;
51 ;  PXSIZE | REAL SCALAR > 0 | ASEC/PX
52 ;  Focal plane camera angular pixel size.
53 ;
54 ;  THRESHOLD | REAL SCALAR > 0 | 1
55 ;  A relative convergence threshold for the phase reconstruction.
56 ;  This value must be at least 1e-3.
57 ;
58 ;  MODE | CHAR STRING | -
59 ;  The computation basis mode. There are two options:
60 ;
61 ;  > MODAL, where the basis of the reconstructed phase are the Zernike
62 ;    polynomials. This mode assumes that the telescope pupil is
63 ;    circular (or an annulus), and you will be asked to enter the
64 ;    pupil diameters, and the number of Zernike polynomials you want
65 ;    to take into account.
66 ;
67 ;  > ZONAL, where the basis are the pixels of the matrix that is being
68 ;    used as the support to reconstruct the phase. The pixel size
69 ;    depends on the PSF matrix size (DIMPSF), the sky pixel size
70 ;    (PXSIZE), and the wavelength, and is computed from
71 ;         dxp = LAMBDA*1e-6/(DIMPSF*PXSIZE)
72 ;    where of course units have to be homogeneous.
73 ;
74 ;    This mode works in principle with any sort of pupil
75 ;    shape. Now for the Keck application, we only allow for either an
76 ;    annular pupil (for instance we used circular pupil masks in the
77 ;    camera) or the full hexagonal telescope pupil.
78 ;
79 ;  It is interesting to run the code on both modes (modal/zonal)
80 ;  when the pupil is circular. Do not be surprised if the
81 ;  reconstructed phase does not look exactly the same in the two
82 ;  cases. The phase reconstruction is always a procedure which is very
83 ;  sensitive to a lots of sometime badly known parameters, for instance
84 ;  the noise level, and other funny things...
85 ;
86 ;  *************************
87 ;  THE OTHER INPUTS DEPEND ON THE MODE
88 ;  *************************
89 ;
90 ;OPTIONAL INPUTS name | type | unit | default value
91 ;
92 ;  *****************
93 ;  If MODE = 'MODAL' the following conditions and inputs are required:
94 ;  *****************
95 ;
96 ;  Pupil shape must be a disk or an annulus.
```

```
97  ;
98  ;  D1 = value | REAL SCALAR > 0 | METERS | none
99  ;  Telescope primary mirror external diameter.
100 ;  Syntax: D1=value.
101 ;
102 ;  D2 = value | REAL SCALAR >= 0 | METERS | none
103 ;  Telescope primary mirror central obscuration diameter.
104 ;  Syntax: D2=value.
105 ;
106 ;  JMAX = value | REAL SCALAR > 0 | 1 | 45
107 ;  The last Zernike polynomial j-index to consider in the reconstruction.
108 ;  Syntax: JMAX=value.
109 ;
110 ;  *****************
111 ;  If MODE = 'ZONAL' the following conditions and inputs are required:
112 ;  *****************
113 ;
114 ;  -------------------------
115 ;  If the pupil is an annulus
116 ;  -------------------------
117 ;
118 ;  D1 = value | REAL SCALAR > 0 | METERS | none
119 ;  Telescope primary mirror external diameter.
120 ;  Syntax: D1=value.
121 ;
122 ;  D2 = value | REAL SCALAR >= 0 | METERS | none
123 ;  Telescope primary mirror central obscuration diameter.
124 ;  Syntax: D2=value.
125 ;
126 ;  JMAX = value | REAL SCALAR > 0 | 1 | 45
127 ;  **** OPTIONAL INPUT ****
128 ;  The last Zernike polynomial j-index to consider in the modal
129 ;  reconstruction of the zonal reconstructed phase. Do not try to
130 ;  project on too many modes !
131 ;  Syntax: JMAX = value.
132 ;
133 ;  ------------------------------------------------
134 ;  If the pupil is the Keck telescope primary mirror
135 ;  ------------------------------------------------
136 ;
137 ;  1/ Set the keyword /KECK,
138 ;  2/ make the parameter file 'KeckSegmentedPupilArchitecture.sav'
139 ;     available in the directory where you run the code.
140 ;
141 ;KEYWORD
142 ;
143 ;  /KECK reconstruct the phase for Keck hexagonal pupil.
144 ;
145 ;  /LARGEHEX set if the pupil is the large hexagonal mask of NIRC2 @ KECK.
146 ;
147 ;  /SHOW to display the in/out-of-focus OTF and the final phase.
148 ;
149 ;OUTPUT NAME | TYPE | UNIT
150 ;
151 ;  A structure variable, with the following components (depending on
152 ;  the computation mode and the pupil)
153 ;
```

```
154 ;  .PHASE | REAL 2D ARRAY [DIMREC,DIMREC] | RAD
155 ;  The reconstructed phase, in the pupil plane, projected in M2 plane,
156 ;  and without tip-tilt.
157 ;
158 ;  .LAMBDA | REAL SCALAR | MICROMETERS
159 ;  The wavelength associated with the phase diversity PSF.
160 ;
161 ;  .WAVEFRONT | REAL 2D ARRAY [DIMREC,DIMREC] | MICROMETERS
162 ;  The reconstructed wavefront = PHASE*LAMBDA/(2*!dpi),
163 ;  in the primary mirror plane. We need it when we want to use the
164 ;  static phase for another LAMBDA.
165 ;
166 ;  .DIMREC | INTEGER SCALAR | 1
167 ;  Phase matrix size.
168 ;
169 ;  .DXP | REAL SCALAR | M
170 ;  Phase matrix pixel size.
171 ;
172 ;  .MASKPUPIL | BYTE 2D ARRAY | 1
173 ;  The pupil mask, 1 inside, 0 outside.
174 ;
175 ;  >>>>>> IF /KECK KEYWORD NOT SET
176 ;  .A_J | REAL VECTOR DIM = 187 | MICROMETERS
177 ;  The Zernike coefficients of the reconstructed phase, starting
178 ;  with the defocus coefficient a4, up to a_JMAX.
179 ;  PISTON AND TIP-TILT COEFFICIENTS ARE NOT GIVEN.
180 ;
181 ;  .J | INTEGER VECTOR OF LENGTH (JMAX-3) | 1
182 ;  The Zernike j-indexes associated with .AJ, from j=4 to j=JMAX.
183 ;  PISTON AND TIP-TILT COEFFICIENTS ARE NOT GIVEN.
184 ;
185 ;  >>>>>>> Only in MODAL mode :
186 ;  .HIGHPHASE | REAL 2D ARRAY | RAD
187 ;  The reconstructed phase without the defocus polynomial (Z4).
188 ;
189 ;EXTERNAL CUSTOM ROUTINES AND OTHER REQUIRED FILES
190 ;
191 ;  coogrid.pro
192 ;  discft.pro
193 ;  hexaft.pro
194 ;  hexagon.pro
195 ;  indzer.pro
196 ;  mathft.pro
197 ;  pixmatsize.pro
198 ;  psfotftsc.pro
199 ;  polzer.pro
200 ;  rectft.pro
201 ;  tvsg.pro
202 ;  valid_input.pro
203 ;
204 ;  deco_conjmarg-IDLv7.0.6.sav - the phase diversity codes compiled.
205 ;
206 ;  for KECK project : KeckSegmentedPupilArchitecture.sav
207 ;  for KECK project : LargeHex.sav
208 ;
209 ;DEV NOTES
210 ;
```

```
211 ;  Laurent Jolissaint, HEIG-VD, Switzerland, May 2, 2013.
212 ;  Oct 01, 2013 LJ included Keck telescope pupil option.
213 ;  Oct 18, 2013 LJ added pupil mask to the output structure variable
214 ;  Dec 18, 2013 LJ added projection into the Zernike basis, in zonal mode.
215 ;  May 16, 2014 LJ rotation of the hexagonal pupil to have the
216 ;                 elevation axis oriented horizontally, as it should.
217 ;  Jun 17, 2014 LJ there was a typo and a unit error for DEFOCUS.
218 ;  Jun 17, 2014 LJ added a reconstruction of the phase map from the
219 ;                 projected Zernike coefficients, for checking.
220 ;  Jul 14, 2014 LJ made it possible to use more than 2 PSF on input.
221 ;  Jul 14, 2014 LJ improved projection onto annular Zernike basis.
222 ;  Jul 15, 2014 LJ changed input PXSIZE unit from asec to microns.
223 ;  Sep 30, 2015 LJ changed KECK pupil orientation, now aligned with
224 ;                 focal plane detector (NIRC2 for instance).
225 ;  Sep 30, 2015 LJ removed projection onto the Zernike basis, replaced
226 ;                 with a match of the reconstructed phase resolution
227 ;                 with the AO structure function resolution.
228 ;  Oct 01, 2015 LJ reorganized code, added a reconstructed wavefront
229 ;                 to the output.
230 ;  May 12, 2016 LJ added LARGEHEX option for KECK.
231 ;  May 12, 2016 LJ restored PXSIZE unit to asec instead of mu (that was
     wrong).
232 ;  May 18, 2016 LJ reset the setting of the phase matrix size to an
233 ;                 optimum, different from KECK requirement.
234 ;  May 18, 2016 LJ reintroduced projection onto the Zernike basis (but not
235 ;                 in KECK mode).
236 ;  Jun 08, 2016 LJ fully removed the constraint on the spatial
237 ;                 frequency sampling in the pupil plane. It is better
238 ;                 to use the value adapted to the input PSF and adapt
239 ;                 the scaling of the final phase, if needed, using REBIN.
240 ;  Jun 09, 2016 LJ simplified options in MODAL mode.
241 ;  Jun 12, 2016 LJ allowed JMAX input in ZONAL mode.
242 ;  Jun 12, 2016 LJ set the optical axis in between the 4 central pixels in
     order
243 ;                 to follow the same convention than deco_conjmarg.
244 ;  Jun 13, 2016 LJ removed the segments gaps for Keck pupil mask.
245 ;  Jun 16, 2016 LJ implemented PSF noise reduction by filtering out the
246 ;                 OTF above the telescope OTF domain definition.
247 ;  Jul 11, 2016 LJ added /SHOW to display the OTF and the final phase.
248 ;  Jul 11, 2016 LJ made the MODAL mode compatible with more than 2
249 ;                 defocused planes.
250 ;  Sep 10, 2016 LJ LARGEHEX option had an error in the pupil
251 ;                 definition.
252 ;  Sep 21, 2016 LJ I forgot to set a central obscuration in /KECK mode.
253 ;  Nov 17, 2016 LJ computation of the defocus in radian done in double
254 ;                 precision, because w/o this it was making a little
255 ;                 but disturbing difference with L. Mugnier root
256 ;                 code deco_conjmarg.pro
257 ;  Nov 30, 2016 LJ added a few warnings for pixel size and wavelength.
258 ;
259 ;BUGS write to laurent.jolissaint@heig-vd.ch
260 ;-
261 FUNCTION
     DIVERSITY,PSF,DEFOCUS,LAMBDA,FDIST,PXSIZE,THRESHOLD,MODE,D1=D1,D2=D2,JMAX=JMAX,KECK=KECK,LARGE
262
263   ;************
264   ;CHECK INPUTS
```

```
265 ;************
266 if n_params() ne 7 then message,'THIS FUNCTION REQUIRES 7 INPUTS AND
        SEVERAL OPTIONAL INPUTS'
267 VALID_INPUT,'DIVERSITY.PRO','PSF',PSF,'real',3,'no','free','free'
268 ss=size(PSF)
269 if ss[1] ne ss[2] then message,'PSF MATRICES MUST BE SQUARE AND EVEN SIZES'
270 if ss[1] mod 2 ne 0 then message,'PSF MATRICES MUST BE SQUARE AND EVEN
        SIZES'
271 VALID_INPUT,'DIVERSITY.PRO','DEFOCUS',DEFOCUS,'real',[1,ss[3]],'no','free','free'
272 VALID_INPUT,'DIVERSITY.PRO','LAMBDA',LAMBDA,'real',0,'no','++','free'
273 VALID_INPUT,'DIVERSITY.PRO','FDIST',FDIST,'real',0,'no','++','free'
274 VALID_INPUT,'DIVERSITY.PRO','PXSIZE',PXSIZE,'real',0,'no','++','free'
275 VALID_INPUT,'DIVERSITY.PRO','THRESHOLD',THRESHOLD,'real',0,'no','++','free'
276 VALID_INPUT,'DIVERSITY.PRO','MODE',MODE,'string',0,'no',['zonal','modal','ZONAL','MODAL'],'f
277 if size(D1,/type) ne 0 then
        VALID_INPUT,'DIVERSITY.PRO','D1',D1,'real',0,'no','++','free'
278 if size(D2,/type) ne 0 then
        VALID_INPUT,'DIVERSITY.PRO','D2',D2,'real',0,'no','0+','free'
279 if size(JMAX,/type) ne 0 then
        VALID_INPUT,'DIVERSITY.PRO','JMAX',JMAX,'integer',0,'no','++',1
280 if strlowcase(MODE) eq 'modal' then if size(JMAX,/type) eq 0 then
        message,'JMAX INPUT IS MANDATORY IF MODE=MODAL'
281 if strlowcase(MODE) eq 'zonal' then begin
282   if not keyword_set(KECK) and not keyword_set(LARGEHEX) then if
          fix(size(D1,/type) ne 0)+fix(size(D2,/type) ne 0) ne 2 then $
283       message,'IF MODE = ZONAL, EITHER GIVE THE PUPIL DIAMETERS D1 & D2
              **OR** SET THE KEYWORDS /KECK OR /LARGEHEX'
284   if keyword_set(KECK) or keyword_set(LARGEHEX) then if fix(size(D1,/type)
          ne 0)+fix(size(D2,/type) ne 0) ne 0 then $
285       message,'IN /KECK OR /LARGEHEX MODES, D1 AND D2 ARE NOT REQUIRED'
286 endif
287 if PXSIZE gt 100 or PXSIZE lt 0.01 then print,'WARNING: THE PIXEL SIZE
        LOOKS ODD - CHECK IT. COMPUTATION CONTINUES ANYWAY.'
288
289 ;********
290 ;SETTINGS
291 ;********
292 rad2asec=180*3600.d/!dpi
293 asec2rad=1.d/rad2asec
294 dimmat=n_elements(PSF[*,0,0])
295 dfp=PXSIZE*asec2rad/(LAMBDA*1e-6)
296 dxp=1.d/dfp/dimmat
297 xyrt=COOGRID(dimmat,dimmat,scale=(dimmat-1)*0.5*dxp)
298
299 ;****************************************
300 ;BUILD PUPIL MASK
301 ;****************************************
302 if keyword_set(KECK) then begin
303   restore,'KeckSegmentedPupilArchitecture.sav'
304   D1=mir.DEXTMAX
305   D2=2.65d
306   hexa=HEXAGON(mir.ssz,xyrt.x,xyrt.y)
307   maskpupil=bytarr(dimmat,dimmat) ; Below, I have an 'or' instead of a '+'
          because I want maskpupil to be 0 or 1
308   for i=0,mir.tns-1 do maskpupil = maskpupil or
          shift(hexa,mir.pos[0,i]/dxp,mir.pos[1,i]/dxp)
309   maskpupil=rotate(maskpupil,6)
```

```
310   hexa=HEXAGON(10.39,xyrt.x,xyrt.y) ; this is to remove the segment gaps
311   maskpupil=maskpupil or hexa
312   maskpupil=maskpupil*(xyrt.r ge 0.5*D2)
313 endif
314 if keyword_set(LARGEHEX) then begin
315   restore,'LargeHex.sav'
316   D1=2*b
317   maskpupil=hexagon(2*b,xyrt.x,xyrt.y)*(1-rotate(hexagon(2*a*2/sqrt(3),xyrt.x,xyrt.y),6))
318 endif
319 if not keyword_set(KECK) and not keyword_set(LARGEHEX) then begin
320   maskpupil=xyrt.r le 0.5*D1
321   if D2 gt dxp then maskpupil=maskpupil*(xyrt.r ge 0.5*D2)
322 endif
323 dimrec=fix(D1/dxp)
324 if dimrec mod 2 eq 1 then dimrec=dimrec+1
325 masklarge=maskpupil
326 if dimrec gt dimmat then message,'THERE IS AN ISSUE WITH THE PIXEL SIZE OR
        THE WAVELENGTH - CHECK THE UNITS'
327 maskpupil=maskpupil[dimmat/2-dimrec/2:dimmat/2+dimrec/2-1,dimmat/2-dimrec/2:dimmat/2+dimrec/2-1]
328
329 ;***********
330 ;PREPARE PSF
331 ;***********
332 ; setting the sky OTF frequencies beyond the telescope OTF domain to 0
333 ; to minimize the noise content in the PSF
334 if keyword_set(KECK) then begin
335   ps=PIXMATSIZE(mir0,PXSIZE,dimmat,LAMBDA)
336   tscpsf=(PSFOTFTSC(mir,ps)).psf
337   tscpsf[1:dimmat-1,1:dimmat-1]=rotate(tscpsf[1:dimmat-1,1:dimmat-1],1)
338 endif else tscpsf=abs(MATHFT(masklarge,dx=dxp,ic=dimmat/2,jc=dimmat/2))^2
339 tscotf=MATHFT(tscpsf,dx=PXSIZE*asec2rad,ic=dimmat/2,jc=dimmat/2)
340 ww=where(abs(tscotf) lt max(abs(tscotf))*1e-4) ; Telescope OTF footprint
        selection
341 ;sky PSF filtering
342 images=PSF
343 skyotf=PSF*dcomplex(0,1)
344 for i=0,n_elements(images[0,0,*])-1 do begin
345   tmp=MATHFT(images[*,*,i],dx=PXSIZE*asec2rad,ic=dimmat/2,jc=dimmat/2)
346   tmp[ww]=0
347   skyotf[*,*,i]=tmp
348   images[*,*,i]=double(MATHFT(skyotf[*,*,i],dx=PXSIZE*asec2rad,ic=dimmat/2,jc=dimmat/2,/inverse))
349   images[*,*,i]=images[*,*,i]/mean(images[*,*,i])
350 endfor
351
352 ;optionally displaying the OTF
353 if keyword_set(SHOW) then begin
354   rw,800,400,0
355   tmp=abs(skyotf[*,*,0])
356   for i=1,n_elements(images[0,0,*])-1 do tmp=[tmp,abs(skyotf[*,*,i])]
357   shade_surf,tmp
358 endif
359
360 ;************
361 ;STARTING RECONSTRUCTION
362 ;************
363 restore,'C:\Users\Jojo\Desktop\PdM-HEIG\Science\OneraCode\deco_conjmarg-IDLv7.0.6.sav'
364 defoc_array_a4=1d3*DEFOCUS/(16.d*sqrt(3.d)*(FDIST/D1)^2)*2.d*!dpi/LAMBDA
```

```
365 pupdiamPX=D1/dxp
366 ;
367 ;************
368 ;OPTION MODAL RECONSTRUCTION
369 ;************
370 if strlowcase(MODE) eq 'modal' then begin
371
372   ai_align = dblarr((n_elements(images[0,0,*])-1)*2)
373   ai_init = dblarr(JMAX-3) ; from Z_4 to Z_{nbmodes+1}
374   sigma2 = 1.0           ; noise variance initial guess / pixel
375
376   DECO_CONJMARG, objet_rec, a_rec, $
377                 IMAGES = images, $
378                 SIGMA2 = sigma2, $
379                 OBJ_REGUL_TYPE = 'none', $
380                 PSF_TYPE = 'ai', $
381                 PARAMPSF_GUESS = ai_init, $
382                 DEFOC_ARRAY_A4 = defoc_array_a4, $
383                 PUP_DIAM = pupdiamPX, $
384                 DOUBLE = 1B, $
385                 METHODE = 'conj', $
386                 PUP_MASQ = maskpupil, $
387                 AI_ALIGN = ai_align, $
388                 /VMLM, FSEUIL = THRESHOLD ,/LEQ
389
390   ;build the phase
391   ;
392   ;get Zernike basis
393   zernike=POLZER(dimrec,D1/dxp,4,JMAX-3)
394   ;
395   phase=dblarr(dimrec,dimrec)
396   for i=0,JMAX-4 do phase=phase+a_rec[i]*zernike[*,*,i]*maskpupil
397   highphase=dblarr(dimrec,dimrec)
398   for i=1,JMAX-4 do highphase=highphase+a_rec[i]*zernike[*,*,i]*maskpupil
399
400   return,{phase:phase,highphase:highphase,a_j:a_rec*LAMBDA/2/!dpi,j:indgen(JMAX-3)+4,dxp:dxp
401          maskpupil:maskpupil,lambda:LAMBDA,wavefront:LAMBDA/2/!dpi*phase,dimrec:n_elements(p
402
403 endif
404 ;
405 ;************
406 ;OPTION ZONAL RECONSTRUCTION
407 ;************
408 if strlowcase(MODE) eq 'zonal' then begin
409
410   psf_hyper=600.0 ; this parameter can be adjusted to optimize the
        computation but this values works best
411   phase_init = dblarr(dimrec, dimrec)
412   sigma2 = 1.0D
413   itmax = 5000L
414
415   DECO_CONJMARG, objet_rec, phase, $
416                 IMAGES = images, $
417                 SIGMA2 = sigma2, $
418                 NOISE_TYPE = 'ls', $
419                 OBJ_REGUL_TYPE = 'none', $
420                 PSF_TYPE = 'phase', $
```

```
                PSF_REGUL_TYPE = 'expifi', $
                PSF_HYPER = psf_hyper, $
                PARAMPSF_GUESS = phase_init, $
                DEFOC_ARRAY_A4 = defoc_array_a4, $
                PUP_DIAM = pupdiamPX, $
                DOUBLE = 1B, $
                ITMAX = itmax, $
                METHODE = 'conj', $
                PUP_MASQ = maskpupil, $
                /VMLM, FSEUIL = THRESHOLD

  if not keyword_set(KECK) and not keyword_set(LARGEHEX) then begin
    ;project reconstructed phase onto the Zernike basis to get the
        coefficients
    ;
    ;get Zernike basis
    if not keyword_set(JMAX) then JMAX = 45
    n_modes=JMAX-3
    zernike=POLZER(dimrec,D1/dxp,4,n_modes)
    ;
    ;compute annular pupil Zernike polynomials covariance matrix
    ;
    zercov=ZERNIKE_SPATIAL_COVARIANCE_ANNULAR_PUPIL(D2/D1,4,JMAX)
    ;get B vector
    b_rec=dblarr(n_modes)
    for i=0,n_modes-1 do
        b_rec[i]=total(maskpupil*phase*zernike[*,*,i])*(dxp/(0.5*D1))^2/!dpi
    ;get solution
    a_rec=invert(zercov,/double)#b_rec*LAMBDA/2/!dpi
    return,{phase:phase,dxp:dxp,maskpupil:maskpupil,dimrec:dimrec,lambda:LAMBDA,wavefront:LAMBDA/
  endif

  return,{phase:phase,dxp:dxp,maskpupil:maskpupil,dimrec:dimrec,lambda:LAMBDA,wavefront:LAMBDA/2

 endif

end
```

B.2 Shack-Hartmann Acquisition Code

B.2.1 readAndAverageSHdata.pro

```
function readAndAverageSHdata, folderPath

fileExt='*.csv'

files = file_search(folderPath+fileExt)

r = readshwfsdata(files[0])

NFiles = n_elements(files)

for ifile = 1,Nfiles-1 do begin

 rtmp = readshwfsdata(files[ifile])
```

```
14
15   r.wavefront = r.wavefront+rtmp.wavefront
16   r.zernike[3,*] = r.zernike[3,*]+rtmp.zernike[3,*]
17
18 endfor
19
20 r.wavefront = r.wavefront / NFiles
21 r.zernike[3,*] = r.zernike[3,*] / Nfiles
22
23 return, r
24 end
```

B.2.2 readSHWFSdata.pro

```
1  function readSHWFSdata, filePath
2
3  openr, f, filePath, /GET_LUN
4
5  iLine = 0
6  line = ''
7
8
9  coefficient = []
10 index = []
11 order = []
12 frequency = []
13 wavefront = []
14
15 while ~EOF(f) do begin
16   readf, f, line
17   iLine += 1
18
19   ;get the zernike coefficient
20   if strmatch(line,'* ZERNIKE FIT *') then begin
21     subheaderNbrLines = 5
22     for isHd =1,subheaderNbrLines do begin
23       readf, f, line
24       iLine += 1
25     endfor
26
27     readf, f, line
28     iLine += 1
29     sLine = strsplit(line,',',/EXTRACT)
30
31     while stregex(sLine[0],'[0-9]+') ne -1 and ~EOF(f) do begin
32       index = [[index],[long(sLine[0])]]
33       order = [[order],[long(sLine[1])]]
34       frequency = [[frequency],[long(sLine[2])]]
35       coefficient = [[coefficient],[double(sLine[3])]]
36       readf, f, line
37       iLine += 1
38       sLine = strsplit(line,',',/EXTRACT)
39     endwhile
40   endif
41
```

```
42   if strmatch(line,'\*\*\* WAVEFRONT \*\*\*') then begin
43     subheaderNbrLines = 11
44     for isHd =1,subheaderNbrLines do begin
45       readf, f, line
46       iLine += 1
47     endfor
48     readf, f, line
49     iLine += 1
50     sLine = strsplit(line,',',/EXTRACT)
51     nel = n_elements(sLine)
52     while stregex(sLine[0],'[0-9]+') ne -1 and ~EOF(f) do begin
53       wavefront = [[wavefront],[double(sLine[1:nel-1])]]
54       readf, f, line
55       iLine += 1
56       sLine = strsplit(line,',',/EXTRACT)
57     endwhile
58
59   endif
60 endwhile
61 free_lun, f
62
63 zernike = [index,order,frequency,coefficient]
64
65 return, {zernike:zernike,wavefront:wavefront}
66
67 end
```

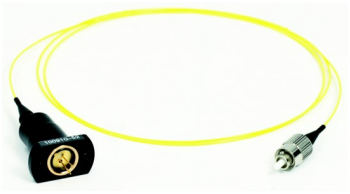
Appendix C

Optical Component Datasheets

C.1 Pigtailed laser diode

THORLABS

Pigtailed Laser Diode, SMF



Description

Thorlabs' Single Mode Pigtailed Laser Diodes are standard TO-packaged diodes that have been pigtailed to a 1 m long single mode fiber with an FC/PC connector. Each unit is tested before shipment. Please refer to the unit-specific test datasheet for optimal operating parameters.

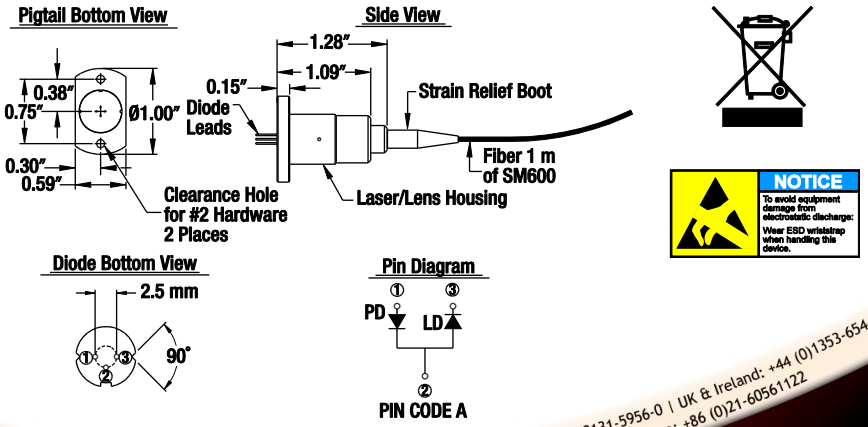
Specifications

LPS-635-FC Specifications		
LD Reverse Voltage (Max)	2 V	
PD Reverse Voltage (Max)	30 V	
Optical Output Power	2.5 mW (Typ.)	3.5 mW (Max)
Operating Temperature	0 to 50 °C	
Storage Temperature	-10 to 65 °C	
Pin Code	9A	
Laser Diode	HL6320G	
Fiber	SM600	
Connector	FC/PC	

LPS-635-FC Specifications			
	Min	Typ.	Max
Wavelength	625 nm	635 nm	640 nm
Threshold Current*	20 mA	50 mA	75 mA
Slope Efficiency*	0.13 mW/mA	0.18 mW/mA	-
Operating Current @ P_O = 2.5 mW*	-	70 mA	95 mA
Operating Voltage @ P_O = 2.5 mW*	-	2.2 V	2.7 V
Monitor Current @ P_O = 2.5 mW*	0.05 mA	0.17 mA	0.3 mA

*Temperature = 25 °C

Drawing



US, Canada, & South America: +1-973-300-3000 | France: +33 (0) 970 444 844 | Europe: +49 (0) 8131-5956-0 | UK & Ireland: +44 (0)1353-654440
Brazil: +55-16-3413 7062 | Scandinavia: +46-31-733-30-00 | Japan & Asia: +81-3-5979-8889 | China: +86 (0)21-60561122
www.thorlabs.com

June 6, 2013
5175-S01, Rev B

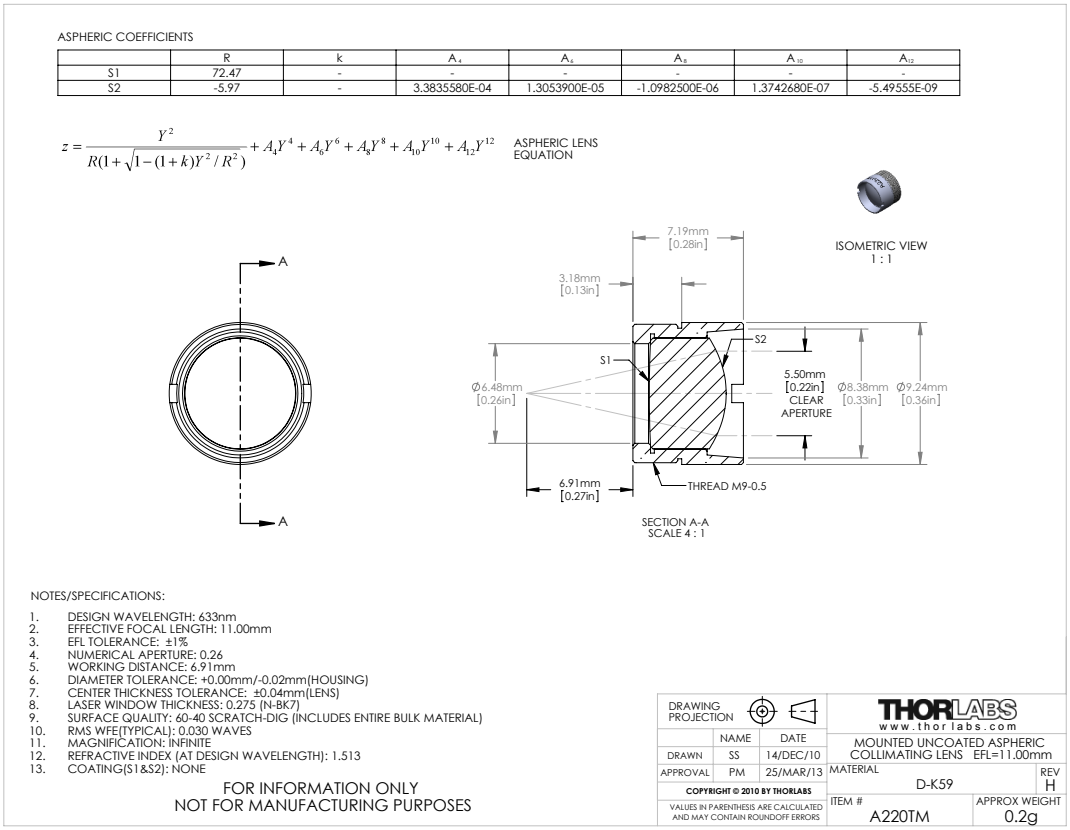
Source : www.thorlabs.com

C.1.1 Power supply modification

The former student, ??? its name ???, mounted a diode driver card to power it. Unfortunately, for the phase diversity experiment, chapter , the power was too high and the Ximea camera was always saturated. So I modified the driver circuit and added two resistances to lower the current so that the detector do not reach the saturation.

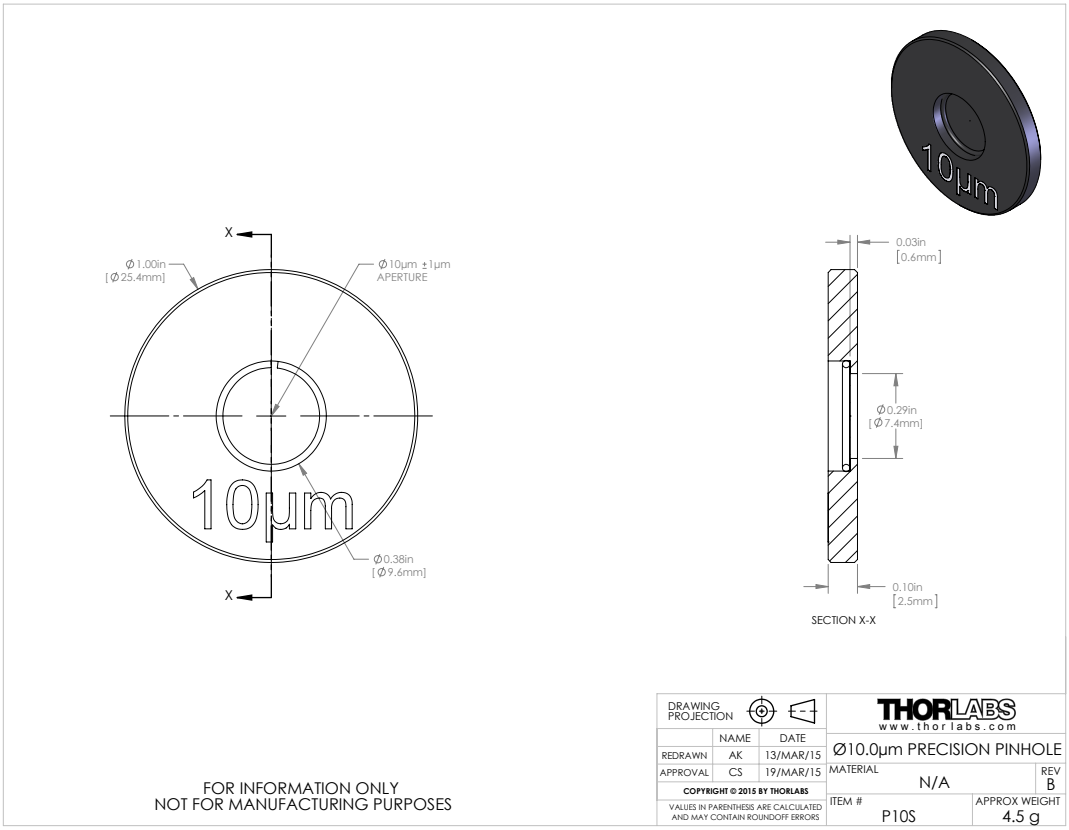
!!! mettre la photo du driver et de la modif !!!

C.2 Converging lens A220TM-A, f = 11 mm



Source : www.thorlabs.com

C.3 Pinhole 10 μm



Source : www.thorlabs.com

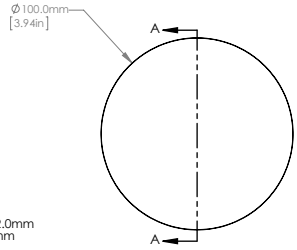
C.4 Converging lens AL100200, f = 200 mm

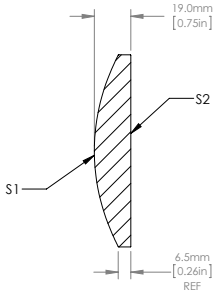
ASPHERIC COEFFICIENTS

	R	k	A_4	A_6	A_8
S1	102.24	-1	4.9646003e-08	7.4017872e-13	9.4141703e-18
S2	INFINITE	-	-	-	-

ASPHERIC LENS EQUATION

$$z = \frac{Y^2}{R\left(1+\sqrt{1-(1+k)Y^2/R^2}\right)} + A_4Y^4 + A_6Y^6 + A_8Y^8$$





NOTES/SPECIFICATIONS:
1. CLEAR APERTURE (COLLIMATION): 92.0mm
2. CLEAR APERTURE (FOCUSING): 89.4mm
3. FOCAL LENGTH: EFL=200mm ±1%
4. Nd: 1.517 ±0.001 Vd: 64.167 ±1.0%
5. DESIGN WAVELENGTH: 780nm
6. NUMERICAL APERTURE: 0.23
7. WORKING DISTANCE (REF): 187.4mm
8. DIAMETER TOLERANCE: +0.00/-0.10mm
9. SURFACE QUALITY: 40-20 SCRATCH-DIG
10. RMS WAVEFRONT ERROR: <0.5µm
11. SAG DEVIATION (POWER)(S1): ±7.5µm
12. RMS IRREGULARITY (S1): <3 FRINGES
13. CENTRATION: <3arcmin
14. MAGNIFICATION: INFINITE

FOR INFORMATION ONLY
NOT FOR MANUFACTURING PURPOSES

DRAWING PROJECTION		
	NAME	DATE
DRAWN	NE	10/OCT/14
APPROVAL	BW	18/NOV/14

VALUES IN PARENTHESIS ARE CALCULATED AND MAY CONTAIN ROUNDOFF ERRORS

THORLABS
www.thorlabs.com

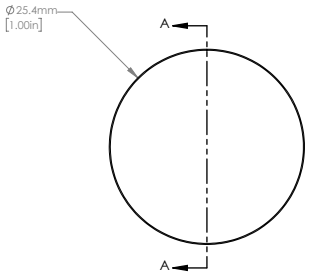
Ø100.0mm ASPHERIC LENS, NA=0.23, f=200mm

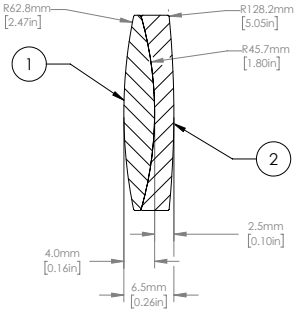
MATERIAL	REV
N-BK7	A
ITEM #	APPROX WEIGHT
AL100200	0.6 kg

Source : www.thorlabs.com

C.5 Converging lens AC254-100-A, f = 100 mm

	MATERIAL
1	N-BK7
2	SF5





NOTES/SPECIFICATIONS:

1. DESIGN WAVELENGTH: 486.1nm, 587.6nm, & 656.3nm
2. FOCAL LENGTH: 100.1mm ±1%
3. BACK FOCAL LENGTH (REF): 97.2mm
4. CLEAR APERTURE: > Ø22.86mm
5. SURFACE QUALITY: 40-20 SCRATCH-DIG
6. CENTRATION: <3arcmin
7. DIAMETER TOLERANCE: +0.0/-0.1mm
8. THICKNESS TOLERANCE: ±0.20mm
9. COATING: BBAR Ravg <0.5% FROM 400-700nm, 0° AOI, ON OUTER OPTICAL SURFACES

FOR INFORMATION ONLY
NOT FOR MANUFACTURING PURPOSES

DRAWING PROJECTION		
	NAME	DATE
DRAWN	KMR	20/SEP/16
APPROVAL	DD	26/SEP/16

VALUES IN PARENTHESIS ARE CALCULATED AND MAY CONTAIN ROUNDOFF ERRORS

THORLABS
www.thorlabs.com

Ø25.4mm, f=100.0mm
VISIBLE ACHROMAT, -A COAT

MATERIAL	REV
SEE TABLE	C
ITEM #	APPROX WEIGHT
AC254-100-A	9.47 g

Source : www.thorlabs.com

C.6 Ximea Camera, MQ013MG-E2

Specifications:	
Resolution:	1.3 MP 1280 × 1024 pixels
Sensor type:	CMOS Matrix B/W
Sensor model:	e2V EV76C560 ABT-EQV
Sensor size:	1/1.8"
Sensor active area:	6.9 × 5.5 mm
Pixel size:	5.3 µm
Bits per pixel:	8, 10
Dynamic range:	60 dB
Frame rates:	60 fps
On-chip binning:	1x1, 2x2
Image data interface:	USB 3.0
Data I/O:	GPIO IN, OUT
Power requirements:	0.9 Watt
Lens mount:	C or CS Mount
Weight:	26 grams
Dimensions WxHxD:	26 x 26 x 26 mm
Operating environment:	50 °C
Customs tariff code:	8525.80 30 (EU) / 8525.80 40 (USA)
ECCN:	EAR99

Source : www.ximea.com/en/products/usb3-vision-cameras-xiq-line/mq013mg-e2

C.7 Shack-Hartmann wavefront sensor, WFS150-5C

8.1 Technical Data

8.1.1 WFS150/300

Item #	WFS150-5C	WFS150-7AR	WFS300-14AR
Microlenses			
Microlens Array	MLA150M-5C	MLA150M-7AR	MLA300M-14AR
Substrate Material	Fused Silica (Quartz)		
Number of Active Lenslets	Software Selectable		
Max. Number of Lenslets	39 x 31		19 x 15
Camera			
Sensor Type	CCD		
Resolution	max. 1280 x 1024 pixels, Software Selectable		
Aperture Size	5.95 mm x 4.76 mm		
Pixel Size	4.65 µm x 4.65 µm		
Shutter	Global		
Exposure Range	79 µs - 65 ms		
Frame Rate	max. 15 Hz		
Image Digitization	8 bit		
Wavefront Measurement			
Wavefront Accuracy [1]	λ/15 rms @ 633 nm		λ/50 rms @ 633 nm
Wavefront Sensitivity [2]	λ/50 rms @ 633 nm		λ/150 rms @ 633 nm
Wavefront Dynamic Range [3]	> 100 λ @ 633 nm		> 50 λ @ 633 nm
Local Wavefront Curvature [4]	> 7.4 mm	> 10.0 mm	> 40.0 mm
External Trigger Input			
Save Static Voltage level	0 to 30 V DC		
LOW Level	0.0 V to 2.0 V		
HIGH Level	5.0 V to 24 V		
Input current	> 10 mA		
Min Pulse Width	100 µs		
Min. Slew Rate	35 V / msec		
Common Specifications			
Optical Input	C-Mount		
Power Supply	<1.5 W, via USB		
Operating Temperature Range [5]	+5 to +35 °C		
Storage Temperature Range	-40 to 70 °C		
Warm-Up Time for Rated Accuracy	15 min		
Dimensions (W x H x D)	32.0 mm x 40.4 mm x 45.5 mm		
Weight	0.1 kg		

[1]) Absolute accuracy using internal reference. Measured for spherical wavefronts of known RoC.
[2]) Typical relative accuracy. Achievable after, and with respect to a user calibration, 10 image averages
[3]) Over entire aperture of wavefront sensor
[4]) Radius of wavefront curvature over single lenslet aperture
[5]) non-condensing

All technical data are valid at 23 ± 5°C and 45 ± 15% rel. humidity (non condensing)

Source : WFS Series Operation Manual, www.thorlabs.com

Bibliography

Blanc, Amandine (2002). "Identification de réponse impulsionnelle et restauration d'images: apports de la diversité de phase". PhD thesis. Université Paris XI UFR Scientifique d'Orsay.

Bouxin, A. (2017). "Phasor diversity to measure the static aberrations of an optical system". MA thesis. HEIG-VD.

F. Zernike, von (1934). "Beugungstheorie des schneidenver-fahrens und seiner verbesserten form, der phasenkontrastmethode". In: *Physica* 1.7, pp. 689 –704. ISSN: 0031-8914. DOI: https://doi.org/10.1016/S0031-8914(34)80259-5. URL: http://www.sciencedirect.com/science/article/pii/S0031891434802595.

Fontanella, J C (1985). "Wavefront sensing deconvolution and adaptive optics". In: *Journal of Optics* 16.6, p. 257. URL: http://stacks.iop.org/0150-536X/16/i=6/a=002.

Gerchberg, R. W. and W. O. Saxton (1972). "A practical algorithm for the determination of phase from image and diffraction plane pictures". In: *Optik* 35.2, 237–250.

Gonsalves, R. A. (1976). "Phase retrieval from modulus data". In: *J. Opt. Soc. Am.* 66.9, pp. 961–964. DOI: 10.1364/JOSA.66.000961. URL: http://www.osapublishing.org/abstract.cfm?URI=josa-66-9-961.

Gonsalves, Robert A. (1982). "Phase Retrieval And Diversity In Adaptive Optics". In: *Optical Engineering* 21, pp. 21 –21 –4. DOI: 10.1117/12.7972989. URL: http://dx.doi.org/10.1117/12.7972989.

Goodman, Joseph W. (1988). *Introduction to Fourier Optics*. Ed. by L. Cox and John M. Morris. 2nd. McGraw-Hill Companies, Inc.

Hartmann (1900). "Bemerkungen über den Bau und die Justirung von Spektrographen". In: *Z. Instrumentenkd* 20, p. 47.

Kolmogorov, A N (1968). "Local Strucure of Turbulence in an Incompressible Viscous Fluid at Very High Reynolds numbers". In: *Soviet Physics Uspekhi* 10.6, p. 734. URL: http://stacks.iop.org/0038-5670/10/i=6/a=R02.

Mugnier, Laurent M., Amandine Blanc, and Jérôme Idier (2006). "Phase Diversity: A Technique for Wave-Front Sensing and for Diffraction-Limited Imaging". In: ed. by Peter Hawkes. Vol. 141. Advances in Imaging and Electron Physics. Elsevier, pp. 1 –76. DOI: https://doi.org/10.1016/S1076-5670(05)41001-0. URL: http://www.sciencedirect.com/science/article/pii/S1076567005410010.

Noll, Robert J. (1976). "Zernike Polynomials and Atmospheric Turbulence". In: *J. Opt. Soc. Am.* 66.3, pp. 207–211. DOI: 10.1364/JOSA.66.000207. URL: http://www.osapublishing.org/abstract.cfm?URI=josa-66-3-207.

Obukhov, A. M. (1949). "Structure of the temperature field in turbulent flow". In: *Ser. Geograf. Geofiz.* 13.1, pp. 58–69.

Paxman, Richard G., Timothy J. Schulz, and James R. Fienup (1992). "Joint estimation of object and aberrations by using phase diversity". In: *J. Opt. Soc. Am. A* 9.7, pp. 1072–1085. DOI: 10.1364/JOSAA.9.001072. URL: http://josaa.osa.org/abstract.cfm?URI=josaa-9-7-1072.

Ragazzoni, Roberto (1996). "Pupil plane wavefront sensing with an oscillating prism". In: *Journal of modern Optics* 43.2, pp. 289–293.

Roddier, François (1988). "Curvature sensing and compensation: a new concept in adaptive optics". In: *Appl. Opt.* 27, pp. 1223–1225.

Shack, R. V. and B. C. Platt (1971). "Production and use of a lenticular Hartmann screen". In: *Spring Meeting of the Optical Society of America*. Vol. 61, 656–660.

Thorlabs (2017). *Principles of Spatial Filters*. Thorlabs. URL: https://www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=1400.

— (2018). *Shack-Hartmann Wavefront Sensor*. Thorlabs. URL: https://www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=2946.

V.I. Tatarski, Richard A. Silverman (1961). *Wave Propagation in a Turbulent Medium*. Ed. by McGraw-Hill. McGraw-Hill.

Wikipedia (2018). *Zernike polynomials*. Wikipedia. URL: https://en.wikipedia.org/wiki/Zernike_polynomials.