

ÉCOLE 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

January 21, 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

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

Contents

Declaration of Authorship	iii
Abstract	vii
Acknowledgements	ix
1 Introduction	1
2 Phase Diversity Experiment	3
2.1 ONERA algorithm	3
2.2 Experimental Setup	3
2.2.1 Light source	3
2.2.2 Entrance pupil	4
2.2.3 Pupil imaging system	4
2.2.4 Detectors	5
2.3 Data Acquisition	5
2.3.1 Ximea Camera	5
2.3.2 Shack-Hartman WFS	5
2.4 Results	5
2.4.1 Parallel plane plate	6
A Optical Component Datasheets	7
A.1 Pigtailed laser diode	8
A.1.1 Power supply modification	9
A.2 Converging lens A220TM-A, $f = 11$ mm	9
A.3 Pinhole $10\ \mu\text{m}$	10
A.4 Converging lens AL100200, $f = 200$ mm	11
A.5 Converging lens AC254-100-A, $f = 100$ mm	12
A.6 Ximea Camera, MQ013MG-E2	13
A.7 Shack-Hartmann wavefront sensor, WFS150-5C	14
B Python Code	15
B.1 AlignementScriptXimeaCamera.py	15
B.2 AcquisAndSaveXimea.py	18
Bibliography	21

List of Figures

2.1	Schema of the experimental setup	3
2.3	Source schema and pinhole effect on the beam.	4
2.2	Wavefront curvature	4
2.4	PSFs example of an alignment procedure	5

List of Tables

2.1 Optical Components	4
----------------------------------	---

List of Abbreviations

FWHM	F ull W idth H alf M aximum
PD	P hase D iversity
PSF	P oint S read F unction
WFS	W ave F ront S ensor

Physical Constants

Speed of Light $c_0 = 2.997\,924\,58 \times 10^8 \text{ m s}^{-1}$ (exact)

List of Symbols

a	distance	m
P	power	W (J s ⁻¹)
ω	angular frequency	rad

For/Dedicated to/To my...

Chapter 1

Introduction

???

Chapter 2

Phase Diversity Experiment

In this chapter, we will 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 behavior 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, and number of Zernike coefficients retrieved. Then we test the algorithm using a known aberration introduced by a parallel plane plate in the beam comparing the result to 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

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.

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

The final application of the phase diversity will be to characterize the optical aberrations induced by the imperfect optical path to a scientific detector of a telescope. For this reason, 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.

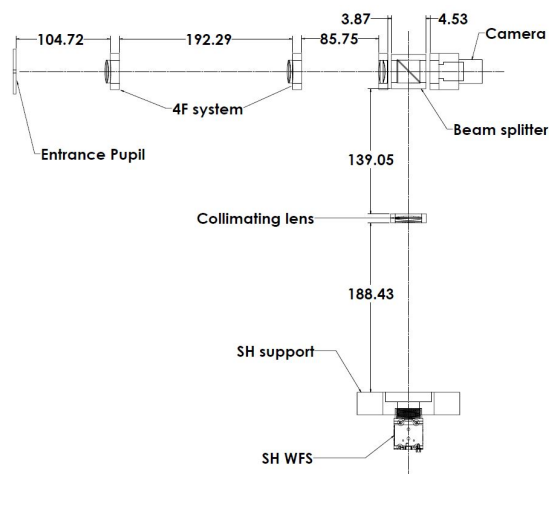


FIGURE 2.1: Experimental setup schema with the relevant distances, (Bouxin, 2017).

TABLE 2.1: Optical Components

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

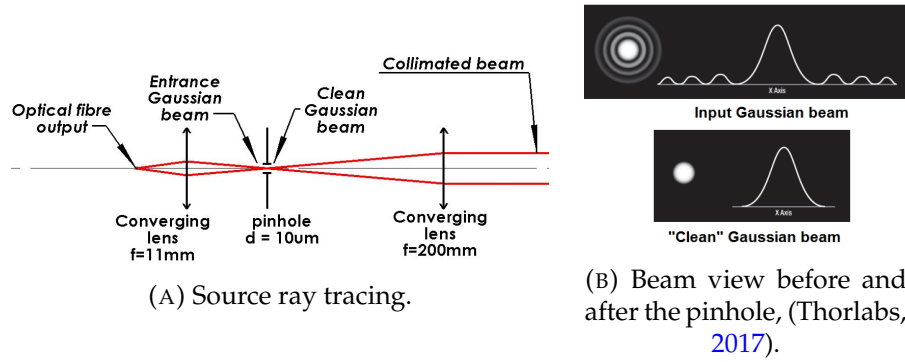
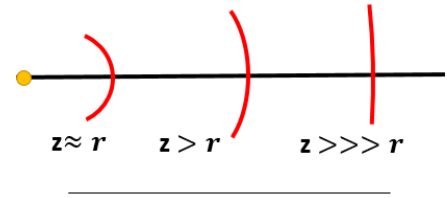


FIGURE 2.3: Source schema and pinhole effect on the beam.

In order to obtain such a planar wavefront at the entrance pupil, the light source consists of a "pigtailed laser diode", a $f=11\text{mm}$ converging lens, a pinhole and a $f=200\text{ mm}$ 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 $10\mu\text{m}$ 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.

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

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 its 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 has 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

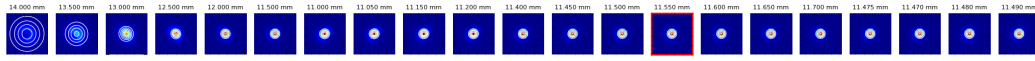


FIGURE 2.4: PSFs example of an alignment procedure

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 1280×1024 pixels of $5.3 \mu\text{m}$, see Appendix A.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 39×31 lenslets grid and a CCD with a resolution of 1280×1024 pixels of $4.65 \mu\text{m}$, see Appendix A.7.

2.3 Data Acquisition

2.3.1 Ximea Camera

The ONERA algorithm takes one focused and one defocused PSFs, as said in section 2.1. 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 :

1. The first step in order to acquire PSFs is to determine the position of the camera's focus point using the python script `AlignementScriptXimeaCamera.py`, see Appendix B.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 2.4.
2. Once you have the focus point position

2.3.2 Shack-Hartman WFS

2.4 Results

This section presents the results of the phase diversity experiment, with the introduction of different sources of aberration.

¹<https://github.com/pupil-labs/pyximea>

2.4.1 Parallel plane plate

The first source of aberration studied in this work is a tilted parallel plane plate which is used as a calibrated source of astigmatism.

Appendix A

Optical Component Datasheets


A.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 ₀ = 2.5 mW*	-	70 mA	95 mA
Operating Voltage @ P ₀ = 2.5 mW*	-	2.2 V	2.7 V
Monitor Current @ P ₀ = 2.5 mW*	0.05 mA	0.17 mA	0.3 mA

*Temperature = 25 °C

Drawing

Pigtail Bottom View



Diode Bottom View



Side View



Pin Diagram





NOTICE

To avoid equipment damage from electrostatic discharge: Wear ESD wriststrap when handling this device.

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

June 6, 2013
5175-S01, Rev B

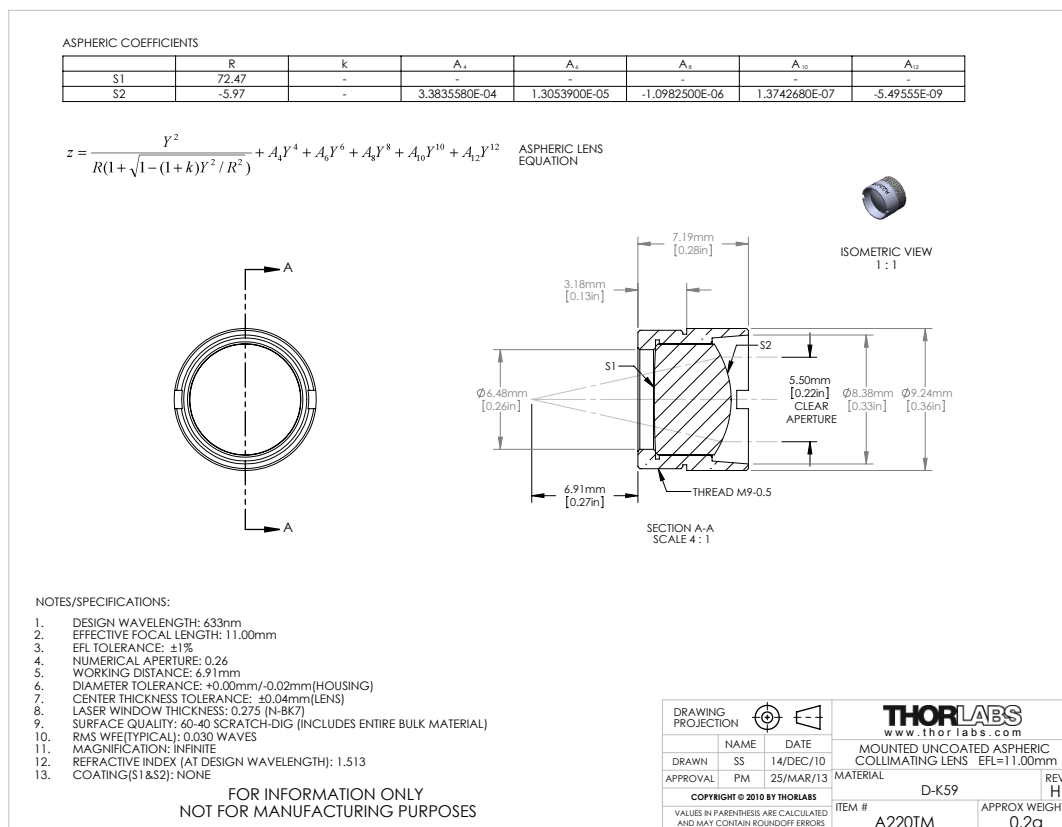
www.thorlabs.com

A.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.

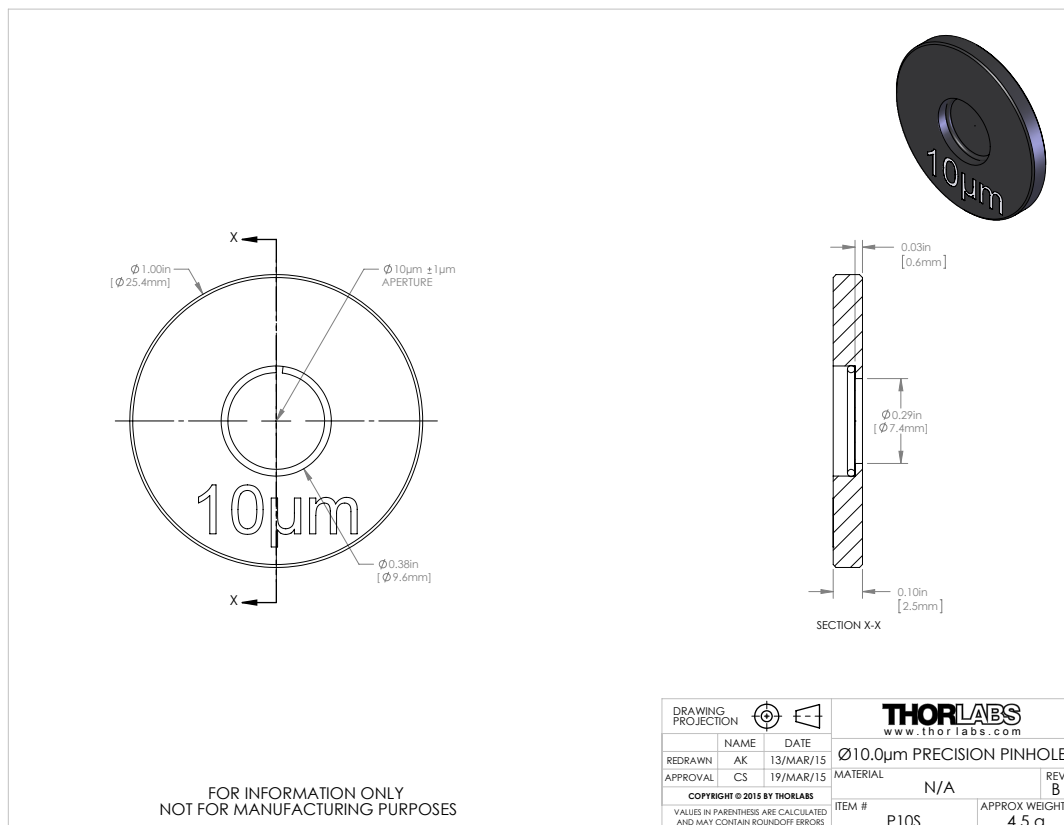
!!! mettre la photo du driver et de la modif !!!

A.2 Converging lens A220TM-A, $f = 11 \text{ mm}$



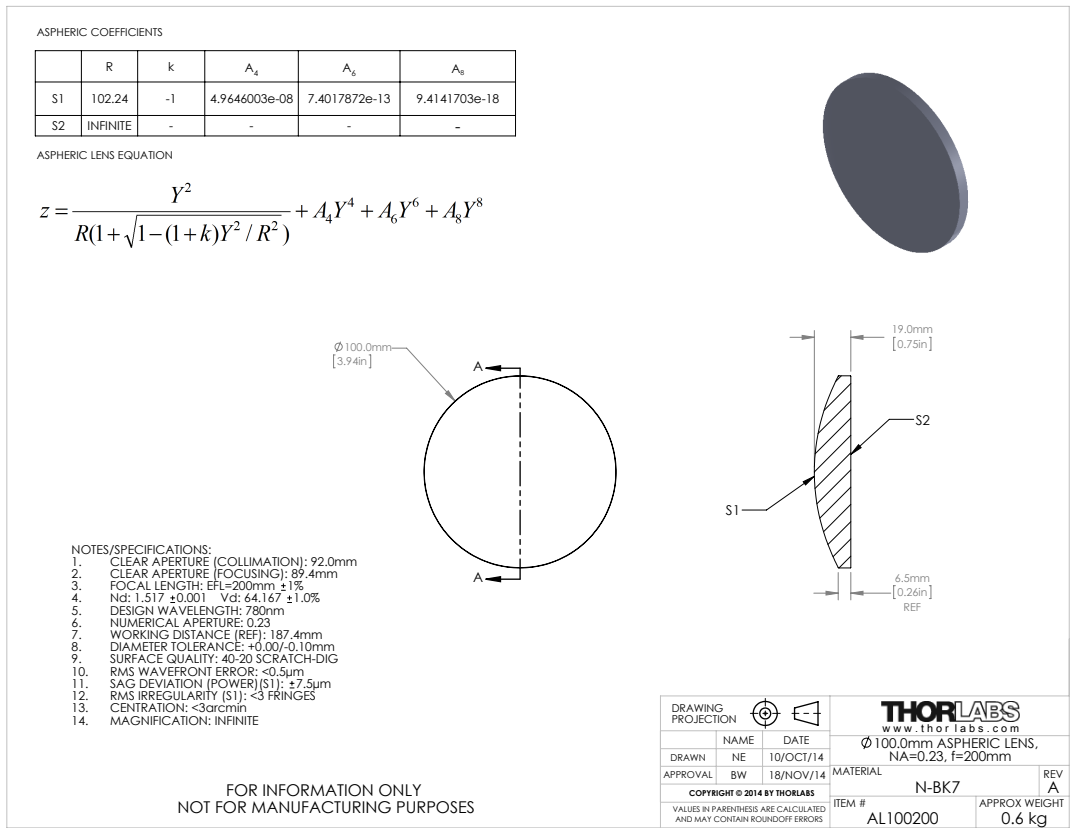
Source : www.thorlabs.com

A.3 Pinhole 10 μm



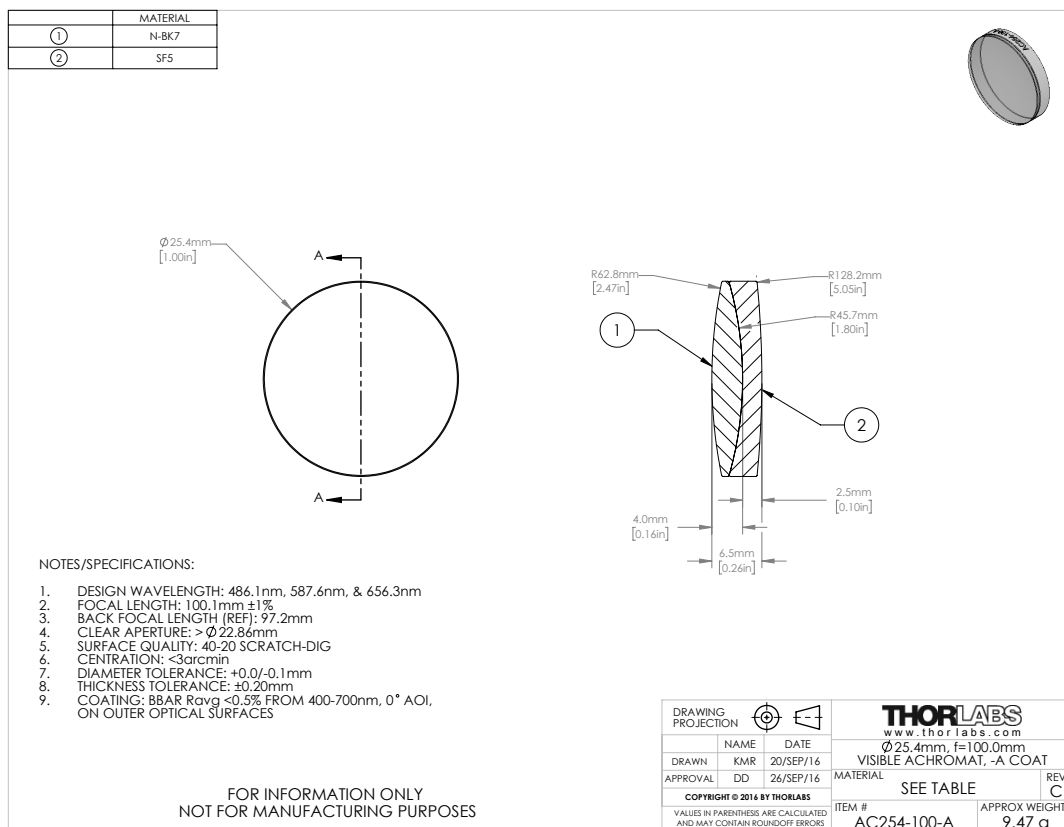
Source : www.thorlabs.com

A.4 Converging lens AL100200, f = 200 mm



Source : www.thorlabs.com

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



Source : www.thorlabs.com

A.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

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

8 Appendix

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⁾	$\lambda/15$ rms @ 633 nm		$\lambda/50$ rms @ 633 nm
Wavefront Sensitivity ²⁾	$\lambda/50$ rms @ 633 nm		$\lambda/150$ 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 $^{\circ}\text{C}$		
Storage Temperature Range	-40 to 70 $^{\circ}\text{C}$		
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 \pm 5°C and 45 \pm 15% rel. humidity (non condensing)

Appendix B

Python Code

B.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 ### instantiation
14
15 dataFolderPath =
16     'C:/Users/Jojo/Desktop/PdM-HEIG/Science/data/PD/phaseScreen/alignement/'
17 plotFolderPath =
18     'C:/Users/Jojo/Desktop/PdM-HEIG/Science/fig/PD/phaseScreen/alignement/'
19 #create the matrix grid of the detector CCD
20 x = np.linspace(0,1280,1280)
21 y = np.linspace(0,1024,1024)
22 x, y = np.meshgrid(x, y)
23
24 #initial guess for the fit depending on the position of the beam in the CCD
25 initial_guess = [250, 481, 706, 3, 3]
26
27 #number of image to average
28 nbrImgAveraging = 10
29
30 ###data acquisition and treatment
31
32 #create instance for first connected camera
33 cam = xiapi.Camera()
34 #start communication
35 print('Opening camera...')
36 cam.open_device()
37 #settings
38 cam.set_imgdataformat('XI_MONO8') #XIMEA format 8 bits per pixel
39 cam.set_gain(0)
40 #create instance of Image to store image data and metadata
41 img = xiapi.Image()
42 #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
64             screw [mm] ? ')))
65     except ValueError:
66         print('Not a float number')
67
68     [tmpdata,stdData] = fX.acquireImg(cam,img,nbrImgAveraging)
69     data.append(tmpdata)
70     #Fit the img data on the 2D Gaussian to compute the FWHM
71     print('Fitting 2D Gaussian...')
72     popt, pcov = opt.curve_fit(fX.TwoDGaussian, (x,y), data[i].ravel(), p0 =
73         initial_guess)
74     print('Fitting done')
75
76     FWHMx.append(2*np.sqrt(2*np.log(2))*popt[3])
77     FWHMy.append(2*np.sqrt(2*np.log(2))*popt[4])
78     x0.append(popt[2])
79     sigmaX0.append(popt[4])
80     y0.append(popt[1])
81     sigmaY0.append(popt[3])
82
83     print 'Fig %d : (x,y) = (%3.2f,%3.2f), FWHM x = %3.2f, FWHM y = %3.2f'
84         %(i,x0[i],y0[i],FWHMx[i],FWHMy[i])
85
86     data_fitted.append(fX.TwoDGaussian((x, y),
87         popt[0],popt[1],popt[2],popt[3],popt[4]).reshape(1024, 1280))
88
89     #plot the beamspot
90     fig, ax = plt.subplots(1, 1)
91     ax.imshow(data[i], cmap=plt.cm.jet,origin='bottom',
92         extent=(x.min(), x.max(), y.min(), y.max()))
93     ax.contour(x, y, data_fitted[i], 5, colors='w',linewidths=0.8)
94     plt.xlim( (popt[2]-4*popt[4], popt[2]+4*popt[4]) )
95     plt.ylim( (popt[1]-4*popt[3], popt[1]+4*popt[3]) )
96     plt.show()
97

```



```

94     #ask if the person wants to acquire a new image to improve the alignment
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
101 #stop data acquisition
102 print('Stopping acquisition...')
103 cam.stop_acquisition()
104
105 #stop communication
106 cam.close_device()
107
108 #convert list to np.array
109 relativePos = np.array(relativePos)
110 data = np.array(data)
111 FWHMx = np.array(FWHMx)
112 FWHMy = np.array(FWHMy)
113 x0 = np.array(x0)
114 y0 = np.array(y0)
115 sigmaX0 = np.array(sigmaX0)
116 sigmaY0 = np.array(sigmaY0)
117
118 #plot the FWHM vs. relPos
119 fig, ax = plt.subplots(1,1)
120 ind = np.argsort(relativePos)
121 ax.plot(relativePos[ind], (np.sqrt(FWHMx**2+FWHMy**2))[ind])
122 ax.set_xlabel('Position [mm]')
123 ax.set_ylabel('FWHM [px]')
124 ax.grid()
125 date = datetime.datetime.today()
126 if not os.path.isdir(plotFolderPath):
127     os.makedirs(plotFolderPath)
128 plt.savefig(plotFolderPath+date.strftime('%Y%m%d%H%M%S')+ 'FWHM_pos.pdf')
129 plt.savefig(plotFolderPath+date.strftime('%Y%m%d%H%M%S')+ 'FWHM_pos.png')
130
131
132 indOfMinFWHM = np.argmin(np.sqrt(FWHMx**2+FWHMy**2))
133
134 fig, axarr = plt.subplots(1,np.size(data,0))
135 #plot all the images besides each other
136 for iImg in ind:
137     axarr[iImg].imshow(data[iImg], cmap=plt.cm.jet,origin='bottom',
138         extent=(x.min(), x.max(), y.min(), y.max()))
139     # axarr[iImg].contour(x, y, data_fitted[iImg], 5,
140         colors='w',linewidths=0.8)
141     axarr[iImg].set_xlim( (x0[iImg]-12, x0[iImg]+12) )
142     axarr[iImg].set_ylim( (y0[iImg]-12, y0[iImg]+12) )
143     axarr[iImg].set_yticklabels(' ',visible=False)
144     axarr[iImg].set_xticklabels(' ',visible=False)
145     axarr[iImg].set_title('%5.3f mm'%relativePos[iImg],fontsize=8)
146     if iImg == indOfMinFWHM:
147         axarr[iImg].set_frame_on(True)
148         for pos in ['top', 'bottom', 'right', 'left']:
149             axarr[iImg].spines[pos].set_edgecolor('r')

```

```

149         axarr[iImg].spines[pos].set_linewidth(2)
150     else:
151         axarr[iImg].set_frame_on(False)
152 plt.show()
153 date = datetime.datetime.today()
154
155 plt.savefig(plotFolderPath+date.strftime('%Y%m%d%H%M%S')+'ImgPSF.pdf')
156 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)

```

B.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/PD/astigmatism/angle_study_3/wth/cropped/20/'
19 darkFolderPathCropped =
   '...../data/dark/astigmatism/angle_study_3/wth/cropped/20/'
20 folderPathFull = '...../data/PD/astigmatism/angle_study_3/wth/full/20/'
21 darkFolderPathFull =
   '...../data/dark/astigmatism/angle_study_3/wth/full/20/'
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]
31

```

```

32 #-----
33 ### data acquisition
34 -----
35
36 #Opening the connection to the camera
37 cam = xiapi.Camera()
38 cam.open_device()
39 cam.set_imgdataformat('XI_MONO8') #XIMEA format 8 bits per pixel
40 cam.set_gain(0)
41
42 img = xiapi.Image()
43 if cam.get_acquisition_status() == 'XI_OFF':
44     cam.start_acquisition()
45 ### exposition
46 cond = 1
47 while bool(cond):
48     source = ''
49     winsound.Beep(freq, duration)
50     source = int(raw_input('Is the source turned on and at focus point
51                           (usually %5.3f mm) (yes = 1) ? '%focusPos))
52     if source == 1:
53         cond = 0
54     else:
55         print 'Please turn on the source and place the camera on the focus
56               point (%5.3f mm) '%focusPos
57
58 if bool(source):
59     #Set exposure time
60     cam.set_exposure(fX.determineUnsaturatedExposureTime(cam,img,[1,10000],1))
61     #get centroid
62     centroid = fX.acquirePSFCentroid(cam,img,initial_guess)
63     print 'centroid at (%d, %d)' %(centroid[0],centroid[1])
64
65 ###Acquire images at different camera position
66
67 acquire = 1
68 while bool(acquire):
69     cond = 1
70     while bool(cond):
71         dark = ''
72         winsound.Beep(freq, duration)
73         dark = int(raw_input('Is the source turned off (yes = 1) ? '))
74         if dark == 1:
75             cond = 0
76         else:
77             print 'Please shut down the source.'
78
79 winsound.Beep(freq, duration)
80 pos = float(raw_input('What is the position of the camera in mm focused
81                      (%5.3f mm) dephase 2Pi (pos+ = %5.3f mm, pos- = %5.3f) ?
82                      '%(focusPos,focusPos+3.19,focusPos-3.19)))
83
84 if bool(dark):
85     print 'Acquiring dark image...'
86     # Acquire dark images
87     [darkData,stdDarkData] = fX.acquireImg(cam,img,nbrImgAveraging)
88     print 'Cropping'

```

```

84     [darkdataCropped, stddarkDataCropped] =
        fX.cropAroundPSF(darkData, stdDarkData, centroid, sizeImg, sizeImg)
85     print 'saving'
86     fX.saveImg2Fits(datetime.datetime.today(), darkFolderPathCropped, nameCamera, darkdataCropped, stddarkDataCropped)
87     fX.saveImg2Fits(datetime.datetime.today(), darkFolderPathFull, nameCamera, darkData, stdDarkData)
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, stdDataCropped)
111            fX.saveImg2Fits(datetime.datetime.today(), folderPathFull, nameCamera, data-darkData, stdDarkData+stdDataCropped)
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    ##Stop the acquisition
126    cam.stop_acquisition()
127    cam.close_device()
128
129    print 'Acquisition finished'
130

```

Bibliography

- Bouxin, A. (2017). "Phasor diversity to measure the static aberrations of an optical system". MA thesis. HEIG-VD.
- 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](https://doi.org/10.1016/S1076-5670(05)41001-0). URL: <http://www.sciencedirect.com/science/article/pii/S1076567005410010>.
- Thorlabs (2017). *Principles of Spatial Filters*. Thorlabs. URL: https://www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=1400.