Milestone M7.5:

Wavefront Software Tests at ESRF

Elena-Ruxandra Cojocaru[1], Sebastien Berujon[1], Eric Ziegler[1], Jonas Schenke[2], Michael Bussmann[2]

[1]ESRF (WP7: PUCCA), [2]HZDR (WP5: UFDAC)

March 28, 2018

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220.

Contents

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220.

1 Introduction

X-rays define a specific region of the electromagnetic spectrum characterized by a very short wavelength ranging from 0.01 nm to 10 nm. This makes them suitable for probing structures much smaller than those usually accessible with visible light. X-ray photons carry a sufficient amount of energy to ionize atoms and disrupt molecular bonds. Due to these properties, they constribute to a vast range of applications, in fields such as material science, chemistry, medicine and life sciences, environmental and earth sciences and even in cultural heritage studies.

The first ever and still the most commonly used X-ray sources are X-ray tubes, initially developed at the end of the 19th century. The next major source for X-rays is synchrotron radiation, demonstrated experimentally in the forties using a table-top system followed by the appearance in the seventies of machines dedicated to applied science. Although synchrotrons represent a far larger and more expensive alternative to regular laboratory X-ray sources, they offer the advantage of a highly collimated and much brighter beam with tunable energy. Currently, diffraction-limited storage rings have opened (MAX IV) or are planned (ESRF, Sirius, PETRA IV, Diamond, APS, Soleil, SPring8) around the world. They provide a rise in performance with respect to previous synchrotron accelerator lattice by at least one order of magnitude in brightness, horizontal transverse coherence or pulse duration [1]. On the other hand, other technologies are employed to develop new types of X-ray sources, such as laser-powered based sources or X-ray free electron lasers (large scale facilities such as SACLA, LCLS, European XFEL). Both kind of sources aim at producing ultrashort pulses, in order to enable the probing of ultrafast processes, such as the formation or breakup of chemical bonds. In total, 60 synchrotrons and free electron lasers facilities are currently in operation or under construction in the world.

One challenge in the case of imaging applications, whatever the technology employed for generating the X-rays, is finding a suitable tool for wavefront metrology. Online metrology is crucial to characterize the influence of optical components present along the beam and optimize its properties. Some of the current most popular choices for wavefront sensing are the Hartmann sensor and grating based sensors, e.g. [2–5]. Although effective, these instruments suffer from a number of drawbacks such as a limited spatial resolution (dependent on the grid

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220.

used or the pitch of the grating) and the need for a delicate calibration to account for wavefront modulator imperfections. An alternative technology suitable for the hard X-ray regime (energy $>\sim 10$ keV, or wavelength < 0.12 nm) is a speckle-based wavefront sensing approach. A device based on such principle can overcome some of the disadvantages of competing technologies [6].

In the frame of the EUCALL PUCCA WP7 project, the ESRF is designing a wavefront sensor prototype for the hard X-ray energy regime based on the principle of X-ray speckle tracking. Alongside this instrument, we also developed a dedicated and versatile software package for wavefront retrieval and reconstruction from speckle experimental data.

This report presents milestone advances and results achieved in the last 18 months that were dedicated to the development of this code. Chapter 2 shortly summarizes the characteristics of the instrument for which the code was designed. Chapter 3 describes the main components of the software package and the theoretical framework for wavefront retrieval and reconstruction. Chapter 4 presents experimental results obtained from testing the code on different types of data sets.

Finally, Chapters 5 and 6 are contributions from HZDR colleagues, members of EUCALL WP5: UFDAC. These sections introduce the benchmarking of the Python code and a study of potential improvements through parallelization work. Conclusions and future outlook are presented in Chapter 7.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220.

2 Description of the Instrument

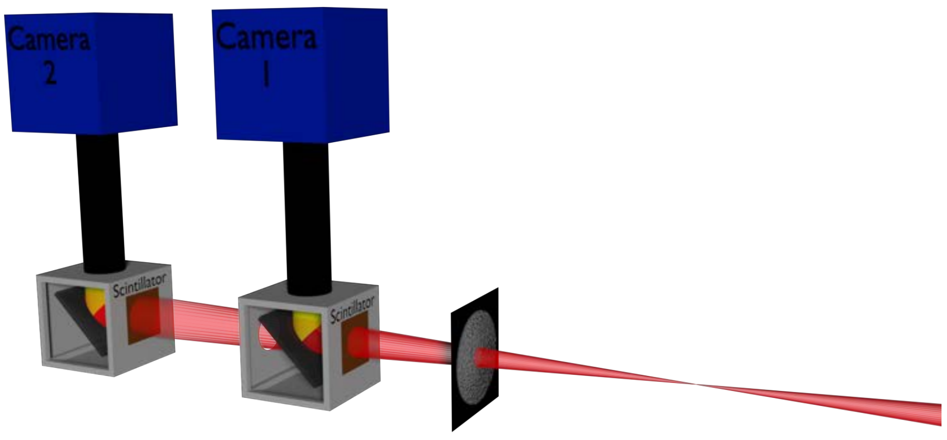


Figure 2.1: Sketch of experimental setup for absolute wavefront metrology

The proposed prototype is a hard X-ray wavefront sensor based on the speckle tracking principle, and whose elements can be adapted to work with different types of X-ray sources and experimental configurations. We aim to provide a metrology instrument usable in two modes:

(i) The differential mode permits the characterization of wavefront variations between consecutive measurements (e.g. for a high-repetition pulsed source) or variations generated by the introduction of new optical elements in the beam path. This mode allows for instance for the individual characterization of optical elements and was used at the ESRF for the individual characterization of each X-ray refractive lens composing an optical system [7]. This configuration requires only one detector and one speckle membrane. When further X-ray measurements are desired downstream from the wavefront sensor, the optical arrangement composing the detector system has to be semi-transparent. Otherwise and more generally, the detector acts as a beamstop by absorbing fully the X-ray beam.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220.

(ii) The absolute wavefront measurement mode characterizes the global wavefront obtained from the contribution of all optical elements present along the beam. This mode is useful for overall beamline optimization measurements. However it is also more demanding as it requires the use of two synchronized detectors plus a speckle membrane (see Fig. 2.1). In addition, the first detector optics must be semi-transparent and designed so that both detectors receive similar flux levels.

At a large scale facility, the partial coherence of the beam is sufficient for applying the speckle tracking principle. The generation of near-field speckle as a random intensity pattern is then made possible by the interference between the directly transmitted beam and the waves scattered from a membrane that plays the role of a random phase object. The resulting speckle grains act as trajectory markers of the X-ray trajectories, eventually permitting the extraction of the two wavefront gradients from one single measurement. In a more general sense, the speckle membrane plays the same role of wavefront modulator as the grating does in other sensors. Nonetheless, the speckle membrane offers the advantage of being economical and much easier to source for instance in case of beam damage or different feature size requirements (speckle size/grating pitch) for the experimental configurations, e.g. when using detectors with different pixel sizes.

Other speckle based approaches and processing methods exist for wavefront retrieval. These alternative methods can provide higher resolution or/and sensitivity but at the cost of longer data collection times. Our code was designed for the X-ray speckle tracking method (XST), i.e. comparing two speckle images obtained either in differential or absolute metrology modes. However, future work could permit the adaptation of the code to handle the X-ray speckle vector tracking method (XSVT) [8].

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220.

3 Wavefront Recovery Software Developed in Python and Theoretical Framework

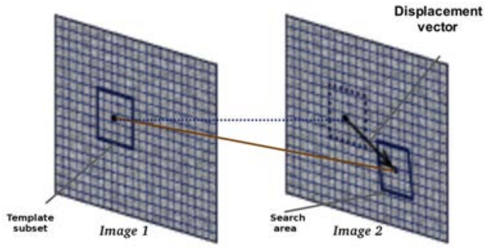


Figure 3.1: XST principle [9]

The XST principle relies on using speckle grains as markers of the trajectory of the X-rays. By considering two speckle images, a cross-correlation algorithm can identify small subsets of pixels from the first image into the second, from which the wavefront gradient is eventually recovered. Then, through bidimensional integration, the wavefront is reconstructed.

A key component of the XST method is the cross-correlation algorithm. To fulfil this function, we use the TemplateMatching algorithm implemented in C++, interfaced for Python through the OpenCV2 library [10]. This algorthim looks for a small template subset $\boldsymbol{T}$, of size $a \cdot b$, in a larger search area $\boldsymbol{A}$, of size $m \cdot n$, and return a correlation matrix, of size $(m - a + 1) \cdot (n - b + 1)$. This matrix contains numerical values from -1 to 1, where values close to 1 represent a strong correlation and values close to -1 represent a strong anti-correlation. The computation of the cross-correlation matrix calls for a cross-correlation criterion. In XST, we use a normalized correlation coefficient-based criterion (”method = cv2.TM_CCOEFF_NORMED” in the TemplateMatching code [11]):

$$R(x,y) = \frac{\sum_{x',y'}(T'(x',y') \cdot A'(x+x',y+y'))}{\sqrt{\sum_{x',y'}(T'(x',y'))^2 \cdot \sum_{x',y'}(A'(x+x',y+y'))^2}} \tag{3.1}$$

where:

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220.

$$\begin{aligned} T'(x',y') &= T(x',y') - \tfrac{1}{w\cdot h}\textstyle\sum_{x'',y''} T(x'',y'') \\ A'(x',y') &= A(x+x',x+y') - \tfrac{1}{w\cdot h}\textstyle\sum_{x'',y''} A(x+x'',x+y'') \end{aligned} \tag{3.2}$$

The correlation matrix returned by the template matching algorithm is then used to compute the relative displacement of the central position of the template subset relative to the central position of the search area. To obtain the full displacement map, all template subsets centered onto pixels of a grid of Image 1 must be searched for in corresponding search areas in Image 2. This is the role of the ”crossSpot” subroutine (illustrated in Fig. 3.2) in our software package. The process involves a ”first” and a ”second pass”. In the ”first pass”, a coarse grid is applied to the image, centering a template subset in each pixel of this grid in Image 1 with a corresponding larger search area in Image 2. 2. Assuming a rigid translation between the two images, it is necessary to correct for this when defining the position of the search area. This is the role of the parameter called ”mode”. Next, all template subsets and corresponding search areas are run through the cross-correlation algorithm. An initial pixel-accurate displacement map is computed and is interpolated to obtain a value for all image pixels and not only those of the coarse grid. Later, the ”second pass” is run, defined on a finer grid. This second pass consists of a loop where parameters, such as the template subset size, are optimized. In practice, the template subset size is gradually increased from one iteration to the other to gain in robustness: the cross-correlation is run several times for certain points with weak correlation peak values, until the errors in each control point are satisfactorily minimized. Finally, the obtained displacement maps are again interpolated for all the pixels in the image to recover the original sampling resolution.

The crossSpot subroutine being one of the most time consuming parts of the program, it will be the object of the optimization efforts presented in Chap. 6.

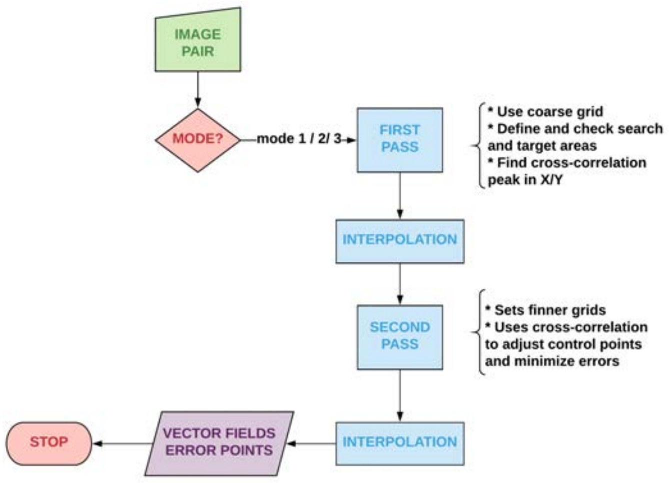


IMAGE PAIR
MODE?
mode 1 / 2/ 3
FIRST PASS
* Use coarse grid
* Define and check search and target areas
* Find cross-correlation peak in X/Y
INTERPOLATION
SECOND PASS
* Sets finner grids
* Uses cross-correlation to adjust control points and minimize errors
INTERPOLATION
VECTOR FIELDS ERROR POINTS
STOP

Figure 3.2: crossSpot subroutine workflow diagram

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220.

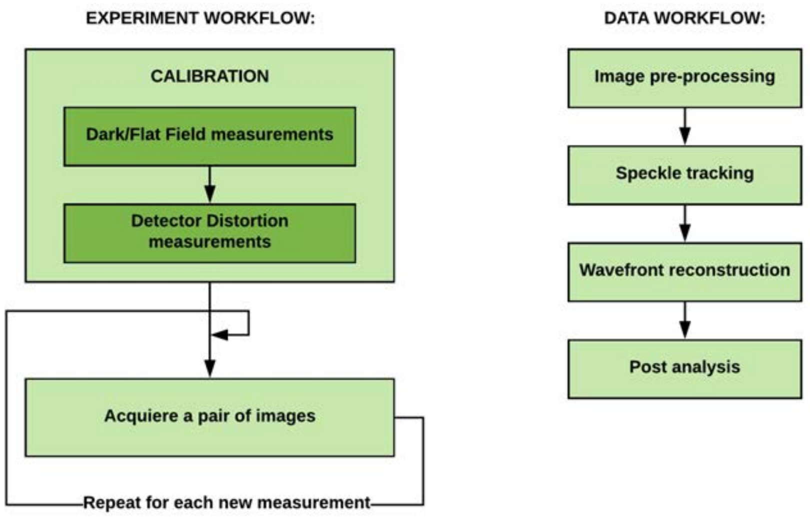


Figure 3.3: Experiment and data workflows

The experiment can be divided in two main parts (see Fig. 3.3); an initial calibration phase and then a measurement cycle. The calibration requires first the collection of dark images, i.e. without X-ray beam, with the same duration as for the future measurements that will be averaged. The resulting image, I_{dark}, is used to account and partially correct for the electronic noise inherent to the detector camera. Afterwards, a set of flat images are recorded at different transverse positions detector with respect to the beam. This time, the averaged flat image, I_{flat}, is used to remove the defects caused by the presence of the X-ray to the visible light scintillator. When two detectors are used, these steps have to be performed for each detector. Later on, each measured speckle image is corrected as follows:

$$I_{speckle} = \frac{I_{raw} - I_{dark}}{I_{flat} - I_{dark}} \tag{3.3}$$

An important step for detector calibration is the characterization of its distortion, in order to correct for the aberrations arising from the optical components of the imaging detector. This well-defined detector distortion calibration protocol represents one of the important advantages offered by a speckle-based wavefront sensor over its competitors that tend to elude the issue [6]. In practice, this calibration process requires images collected by moving each detector in a equally-spaced two-dimensional mesh grid in the transversal plane with respect to the beam. Mathematically, the distortion-free image, (x_r, y_r), can be defined by a function, $f = (f_x, f_y)$, of its distorted equivalent, (x_d, y_d):

$$\begin{aligned} x_r &= x_d + f_x(x_d, y_d) \\ y_r &= y_d + f_y(x_d, y_d) \end{aligned} \tag{3.4}$$

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220.

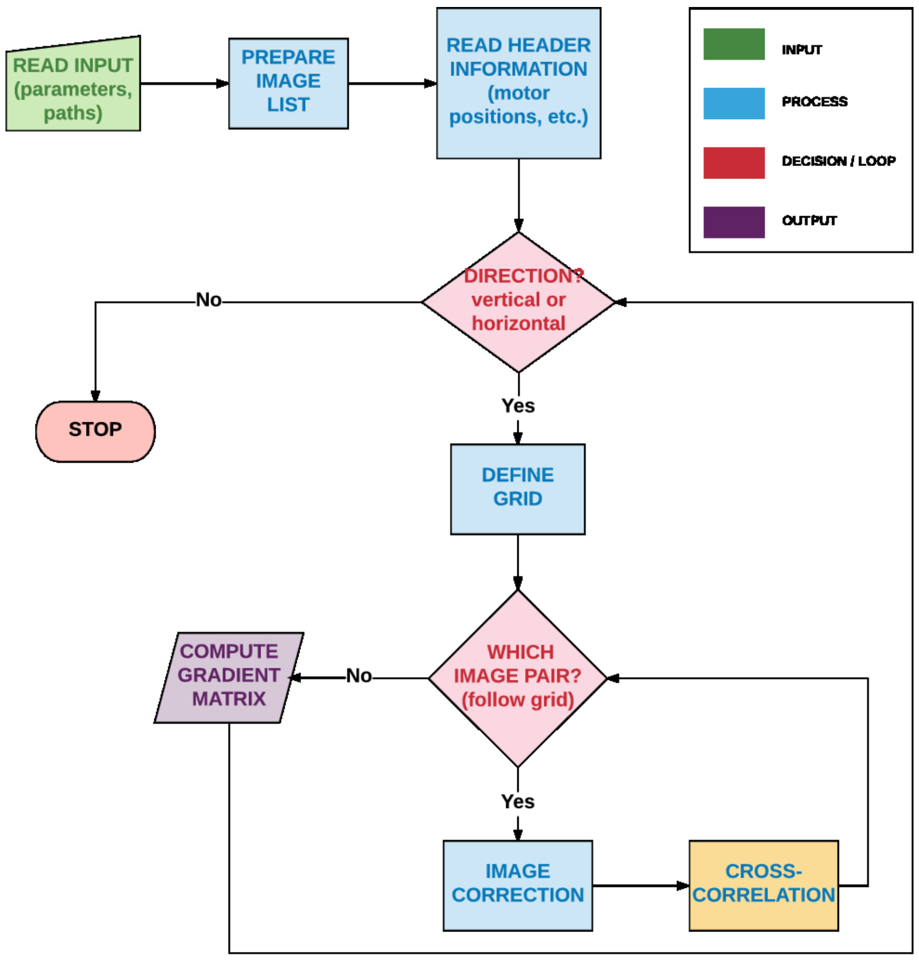

READ INPUT (parameters, paths)
PREPARE IMAGE LIST
READ HEADER INFORMATION (motor positions, etc.)
INPUT
PROCESS
DECISION / LOOP
OUTPUT
DIRECTION? vertical or horizontal
No
STOP
Yes
DEFINE GRID
WHICH IMAGE PAIR? (follow grid)
No
COMPUTE GRADIENT MATRIX
Yes
IMAGE CORRECTION
CROSS-CORRELATION

Figure 3.4: Detector Distortion workflow diagram

When subjecting the detector to a translation (corresponding to the step of the grid in each direction), for example in the x direction, $\boldsymbol{h}_x = h \cdot \boldsymbol{e}_x$, this translates into a displacement vector, $\boldsymbol{v}_{xy}$:

$$\begin{aligned} \boldsymbol{v}_{xy} \cdot \boldsymbol{e}_x &= \tfrac{1}{s_{pix}}[h + f_x(x_d + h, y_d) - f_x(x_d, y_d)] \\ \boldsymbol{v}_{xy} \cdot \boldsymbol{e}_y &= \tfrac{1}{s_{pix}}[f_y(x_d + h, y_d) - f_y(x_d, y_d)] \end{aligned} \tag{3.5}$$

where s_{pix} is the detector pixel size. In the case of a translation in the y direction, $\boldsymbol{h}_y = h \cdot \boldsymbol{e}_y$, the displacement vector, $\boldsymbol{v}'_{xy}$, will be:

$$\begin{aligned} \boldsymbol{v}'_{xy} \cdot \boldsymbol{e}_x &= \tfrac{1}{s_{pix}}[f_x(x_d, y_d + h) - f_x(x_d, y_d)] \\ \boldsymbol{v}'_{xy} \cdot \boldsymbol{e}_y &= \tfrac{1}{s_{pix}}[h + f_y(x_d, y_d + h) - f_y(x_d, y_d)] \end{aligned} \tag{3.6}$$

By assuming $s_{pix} = \frac{h}{<|\boldsymbol{v}_{xy}|>_{(x,y)}}$, the approximate directional derivatives of (f_x, f_y), corresponding to the four gradient maps, is obtained by forward difference:

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220.

$$\begin{aligned}
\nabla_x f_x(x,y) &= \frac{\partial f_x}{\partial x}\Big|_{(x,y)} = \frac{\boldsymbol{v}_{xy}\cdot\boldsymbol{e}_x - <|\boldsymbol{v}_{xy}|>}{<|\boldsymbol{v}_{xy}|>} \\
\nabla_x f_y(x,y) &= \frac{\partial f_y}{\partial x}\Big|_{(x,y)} = \frac{\boldsymbol{v}_{xy}\cdot\boldsymbol{e}_y}{<|\boldsymbol{v}_{xy}|>} \\
\nabla_y f_x(x,y) &= \frac{\partial f_x}{\partial y}\Big|_{(x,y)} = \frac{\boldsymbol{v}'_{xy}\cdot\boldsymbol{e}_x}{<|\boldsymbol{v}'_{xy}|>} \\
\nabla_y f_y(x,y) &= \frac{\partial f_y}{\partial y}\Big|_{(x,y)} = \frac{\boldsymbol{v}'_{xy}\cdot\boldsymbol{e}_y - <|\boldsymbol{v}'_{xy}|>}{<|\boldsymbol{v}'_{xy}|>}
\end{aligned} \tag{3.7}$$

Finally, the distortion functions are computed through the numerical integration of the gradient map pairs $(\nabla_x f_x, \nabla_y f_x)$ and $(\nabla_y f_x, \nabla_y f_y)$ to find the intermediate function (f'_x, f'_y) and then setting the integration constants to 0:

$$\begin{aligned}
J_1 &= \int\int \left(\frac{\partial f'_x}{\partial x} - \nabla_x f_x\right)^2 + \left(\frac{\partial f'_x}{\partial y} - \nabla_y f_x\right)^2 dxdy \\
J_2 &= \int\int \left(\frac{\partial f'_y}{\partial x} - \nabla_x f_y\right)^2 + \left(\frac{\partial f'_y}{\partial y} - \nabla_y f_y\right)^2 dxdy
\end{aligned} \tag{3.8}$$

By computing these two distortion maps, one can undistort the speckle images as if they were recorded by perfect detector grids. That calibration protocol also allows for a precise calculation of the detector effective pixel size.

Within the software package, the computing of the detector distortion maps is the role of one of the two core modules, named ”detectorDistortion.py”, illustrated as a flow-diagram in Fig. 3.4. This module operates as a double loop through the images collected when moving the detector in the mesh grid and compares pairs of images in the horizontal and vertical direction using the ”crossSpot” subroutine. Next, the four resulting displacement maps are integrated two by two resulting in two distortion maps, corresponding to the horizontal and vertical directions.

The other main core module, ”waveFront.py”, is dedicated to processing each measured pair of images in order to retrieve and reconstruct the wavefront, either in differential or absolute modes. This module is illustrated in Fig. 3.5. The main steps consist of:

(i) Image corrections, namely dark and flat field correction, then undistorting the images.

(ii) The computing of the displacement maps between the pair of images, again using the ”crossSpot” subroutine.

(iii) Finally, the resulting displacement maps are numerically integrated to obtain the reconstructed wavefront.

For numerical integration purposes, our software package offers two options: an implementation of the Frankot-Challappa algorithm [12] using the Fast Fourier Transform (FFT) or the grad2surf algorithm [13, 14], translated to the Python language [15].

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220.

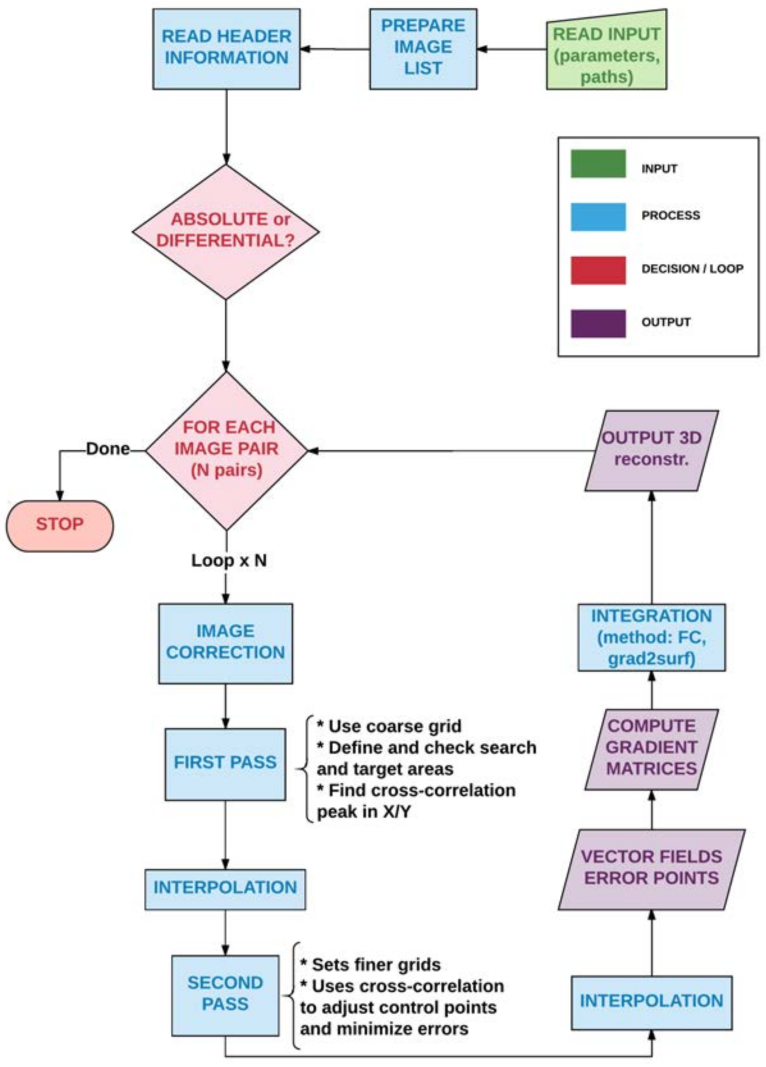

READ HEADER INFORMATION
PREPARE IMAGE LIST
READ INPUT (parameters, paths)
INPUT
PROCESS
DECISION / LOOP
OUTPUT
ABSOLUTE or DIFFERENTIAL?
FOR EACH IMAGE PAIR (N pairs)
Done
STOP
Loop x N
OUTPUT 3D reconstr.
IMAGE CORRECTION
INTEGRATION (method: FC, grad2surf)
FIRST PASS
* Use coarse grid
* Define and check search and target areas
* Find cross-correlation peak in X/Y
COMPUTE GRADIENT MATRICES
INTERPOLATION
VECTOR FIELDS ERROR POINTS
SECOND PASS
* Sets finer grids
* Uses cross-correlation to adjust control points and minimize errors
INTERPOLATION

Figure 3.5: Wavefront Reconstruction workflow diagram

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220.

4 Tests Performed on the Python Software Package: Current Results

The wavefront recovery and reconstruction software package was developed such as to present several key features:

- open-sourced language, the reason for choosing the Python language;
- flexibility, to facilitate the use of different data formats;
- versatility, for the code to be usable for different metrology modes and expandable for different wavefront recovery methods;
- completeness, to address both the calibration and reconstruction steps;
- potential optimization in order to achieve near real-time processing (see Chap. 6 for discussion)

As explained in Chap. 3, the code has two core modules, one for computing the detector distortion for each detector used and the other one for wavefront recovery. Our first tests focused on the detector distortion module, for debugging purposes. The module was then tested using data acquired in different experiments performed at the ESRF between November 2016 and December 2017. Fig. 4.1 shows a successful computation of the integrated distortion maps for the second detector. We have tested other datasets from different experiments with slightly different detectors configurations and different types of cameras and mirrors, for which we obtained equally good results. The resulting distortion maps produced by this module are then used as input for the wavefront recovery module, to undistort the images before further computations.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220.

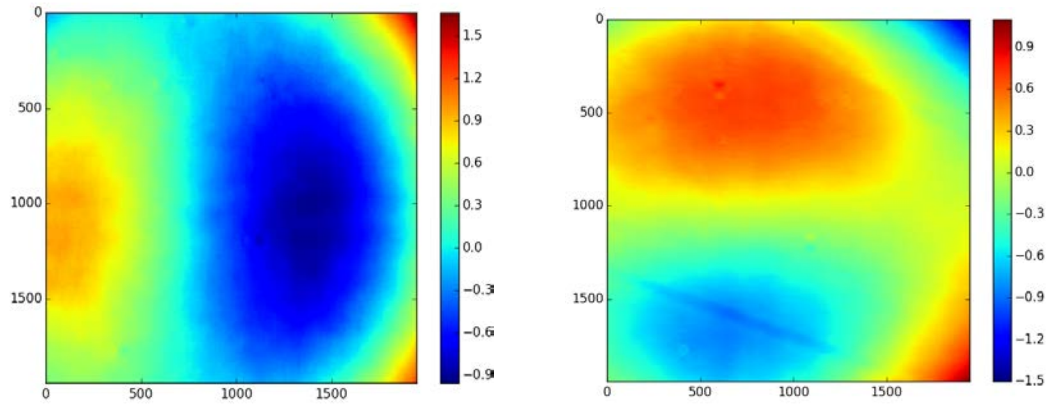


Figure 4.1: Integrated distortion maps for the second detector in a configuration using a PCO EDGE 4.2 camera and 5x magnification (all axes in pixels)

Concerning the data formats, two formats are supported so far: EDF (ESRF Data Format) and TIFF. Support for other image data formats can be implemented (e.g. HDF5), with the condition of having an existing Python library for extracting the image information from the raw data file. Another aspect to take into account is that the EDF format has a header that contains many experimental parameters needed during data processing, whereas the TIFF data does not possess one. Yet, the need for a header can be waived as most of the necessary information can just be fed to the code as input parameters before the data processing begins. The most delicate case concerns the computing of the detector distortion, for which it is necessary to specify the horizontal (x-direction) and vertical (y-direction) motor positions for each image used. However, this can be replaced by the manual input of the two motor step parameters under the condition that the displacements during data acquisition are generating an equally spaced grid.

The following tests were focused on the wavefront recovery and reconstruction module, with an emphasis on its behavior for data measured in both differential and absolute metrology modes, using different cameras and data types. Four test cases are presented. Most datasets selected for these test cases contain data obtained after having introduced an X-ray refractive lens (or CRL) in the beam, as the resulting wavefronts are convenient for result control.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220.

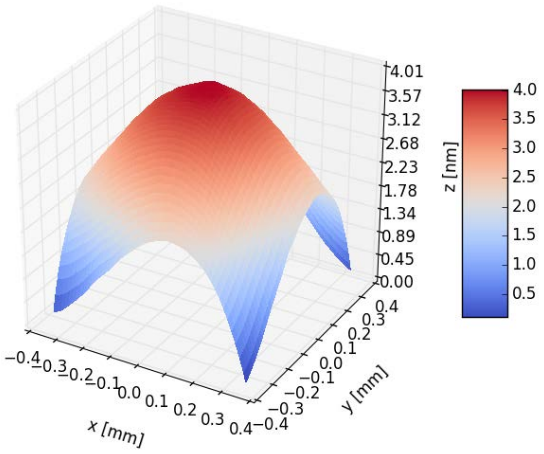


Figure 4.2: Test Case 1 – Reconstruction of a 500 μm 2D CRL measured in absolute metrology mode at the BM05 beamline of the ESRF

Test Case 1 uses data measured in the absolute metrology mode using two detectors, both based on PCO EDGE 4.2 cameras with data saved in the EDF format. The reconstructed wavefront for a 500 μm 2D CRL can be seen in Fig. 4.2. As one can noticed, the entire reconstructed wavefront is dominated by the lens, as the usable region of interest (ROI) in the case of this experiment was less than 1.5 mm $\times$ 1.5 mm, i.e. smaller than the lens aperture. This was due to the fact that we were using a pierced mirror with a 1.5 mm diameter hole in the first detector optics to ensure a higher flux on the second detector in the case of lower photon energies. A wider ROI would have allowed to see the contribution of the rest of the beamline optical elements to the global wavefront. This is more noticeable in Fig. 4.5 discussed later in this chapter.

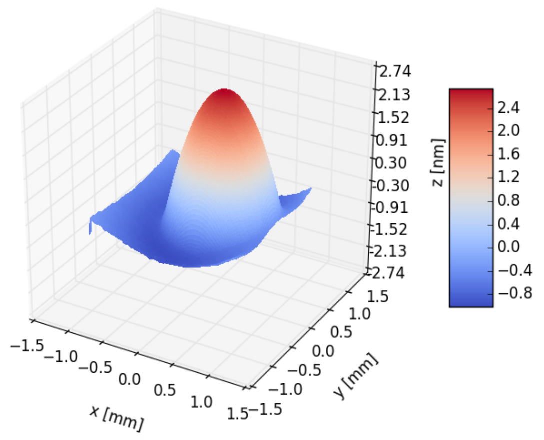
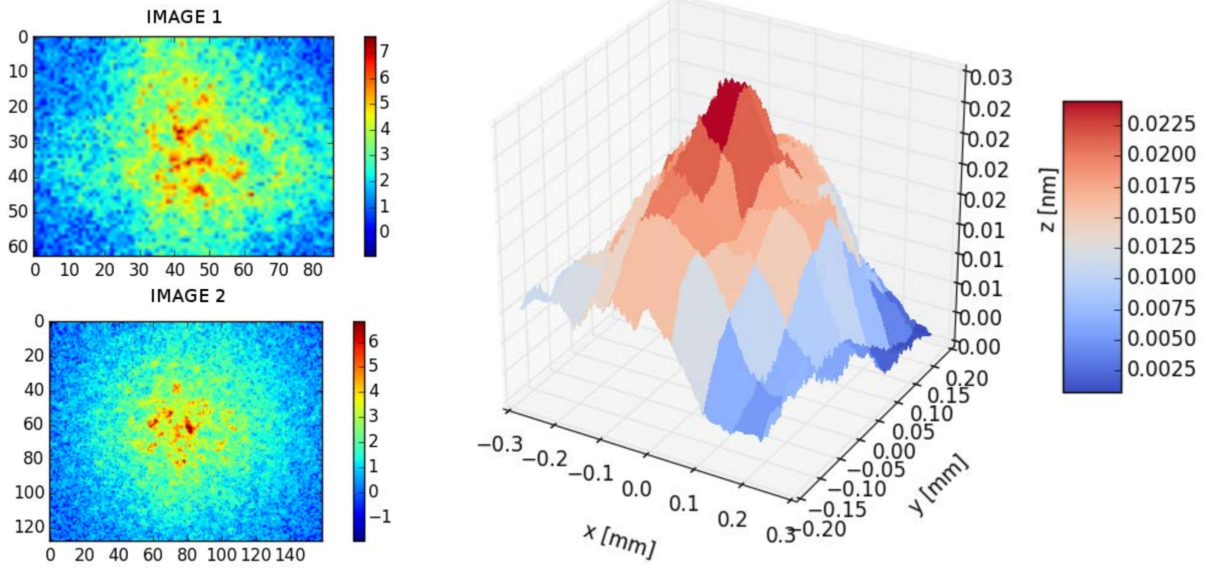


Figure 4.3: Test Case 2 – Reconstruction of a 2D CRL measured in differential metrology mode at the BM05 beamline of the ESRF [7]

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220.

Next, Test Case 2 uses data obtained in the differential mode with a 50 μm 2D CRL [7]. In Fig. 4.3 the lens makes the only contribution to the wavefront difference, even though the ROI is wider than the area occupied by the lens.

Test Case 3 used data measured in differential mode at the FXE beamline of the European XFEL (EuXFEL). The two input images in Fig. 4.4 correspond to data with and without the presence of lanthanum hexaboride (LaB_6) calibration powder in the beam, acquired with the ultrafast Shimadzu HPV-X camera. Each image corresponds to an ultrashort pulse, <100 fs, produced by the EuXFEL.



(a) Images after correction and ROI crop, measurements with and without LaB_6 callibration powder in the beam

(b) Reconstructed wavefront

Figure 4.4: Test Case 3 – Wavefront reconstruction using measurements in differential mode acquired with a fast Shimadzu HPV-X camera at the FXE beamline of the EuXFEL

The reconstruction shows only the LaB_6 contribution to the wavefront. Further tests comparing two consecutive pulses, taken in otherwise identical conditions, with no alterations along the beam path, show no detectable wavefront variations from one pulse to the other. Since the trains of pulses were short, i.e. of only 2 pulses/train, thermal effects were probably too low to affect significantly the optical properties of the X-ray optics. For longer trains, e.g. when the EuXFEL will reach 27000 pulses per train, one may expect wavefront variations from the first to the last pulse of a train.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220.

Lastly, Test Case 4 uses data measured at the FXE beamline of the EuXFEL using the absolute mode, therefore requiring the instrumental configuration using two detectors. Since the flux on the second detector at the working photon energy of 9.1 keV was rather low, several images were used applying the XSVT method.

Thus the resulting wavefront shown in Fig. 4.5 is actually an averaged wavefront over several pulses. Even so, an averaged wavefront can prove to be useful in terms for beamline optimization efforts. In this example one can observe not only the contribution of the 2D CRL inserted in the beam but also that of the rest of the optical elements in the beam path. Including the XSVT method in the software package is still an ongoing effort as it was not initially planned.

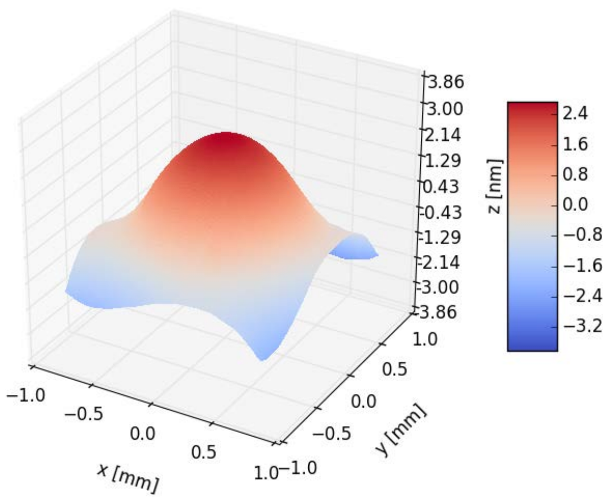


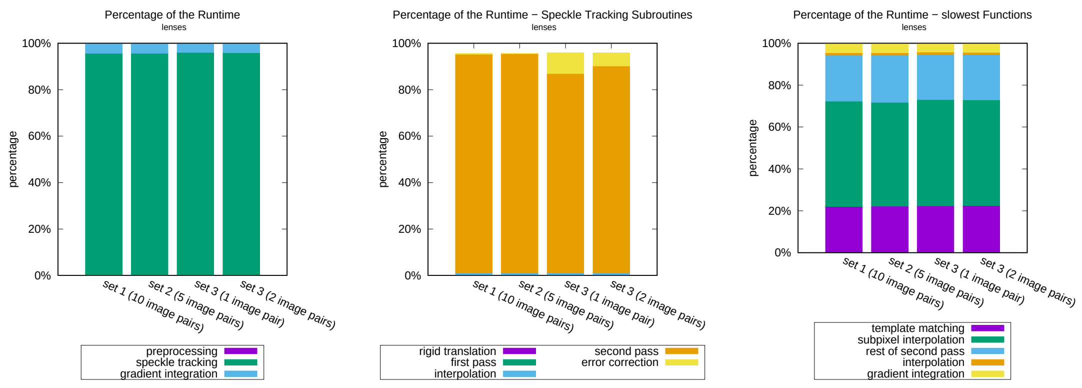
Figure 4.5: Test Case 4 – Reconstruction of a 2D CRL inserted in the beam from absolute wavefront measurements at the FXE beamline of the EuXFEL using the XSVT method

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220.

5 Benchmarking Results

All benchmarks were ran on the Haswell partition of the Taurus supercomputer at the Technische Universität Dresden. Technical details on how these benchmarking computations were performed can be found in [16].

Each configuration used consisted of a dataset and core count. The times presented here for each configuration represent an average over five runs, which started after four initial warm-up runs. In each case, the pure execution times of the scripts and individual functions were recorded. From these, input/output (I/O) times were excluded. The number of cores was varied from one to twenty-four and each benchmark ran exclusively on one node. To evaluate the performance of the present implementation, several types of data sets were used, however here we will only show the results obtained using the data sets from [7] measured in differential mode.



(a) (b) (c)

Figure 5.1: Percentage of runtime consumed by main subroutines in the base Python code

To identify bottlenecks and particularly time-consuming functions, the program execution times and total number of function calls were logged. An overview of the complete program with its subroutines and their share of the total computation time consumed is shown in Fig. 5.1a. This clearly shows that most of the computational time is needed for speckle tracking. A more detailed temporal distribution of the time consumed by each step involved

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220.

in the speckle tracking process is shown in Fig. 5.1b. This shows that the second pass is the most demanding section of the code. The cumulative time of the top five most computational-intensive functions from all configurations (shown in Fig. 5.1c) adds up to more than 95% of the total time.

The main reason why such long computation times are needed for the template matching process and subpixel interpolation is due to the elevated number of function calls. For example, a run involving 10 pairs of images (lenses - Set 1) resulted in over 22.5 million calls. In each of these calls, the template matching and subpixel interpolation are used only once. In addition, except for the template matching, the second run makes only minor use of already optimized libraries like numpy, that however increase the Python overhead. In the speckle tracking process, the calls within the second pass are parallelized by means of joblib. This uses the standard multiprocessing library, which creates a fork of the entire Python environment for each thread [17]. The high computation time needed for gradient integration is related to the size of the image used. Overall, the program has a poor CPU utilization of only 19.635% on average [18], which often means that some of the cores are not used at times or are only slightly used. This shows that much could be gained with further optimization.

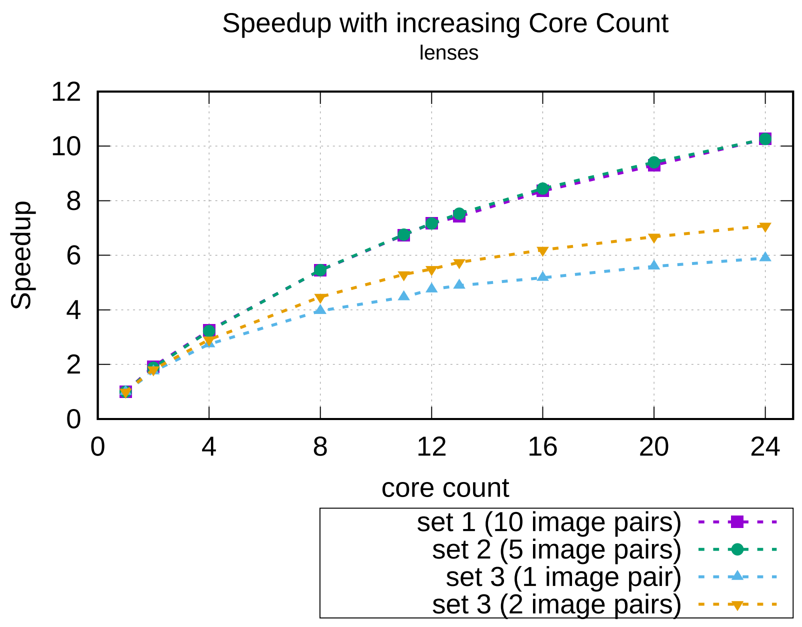


Figure 5.2: Speedup of base Python implementation relative to the increase in the number of processing cores used

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220.

6 Parallelization of the Wavefront Reconstruction Algorithm through Multi-core Processing

As part of his Bachelor thesis [16], J. Schenke studied several possibilities for parallelizing and then optimizing the code to ensure faster processing times. This included:

- parallelizing the processing of a batch of image pairs by means of MPI, by distributing each pair over a computational core.
- the use of parallel computing within the processing of an individual image pair by means of MPI.
- optimizing performance by eliminating bottlenecks within the Python version.

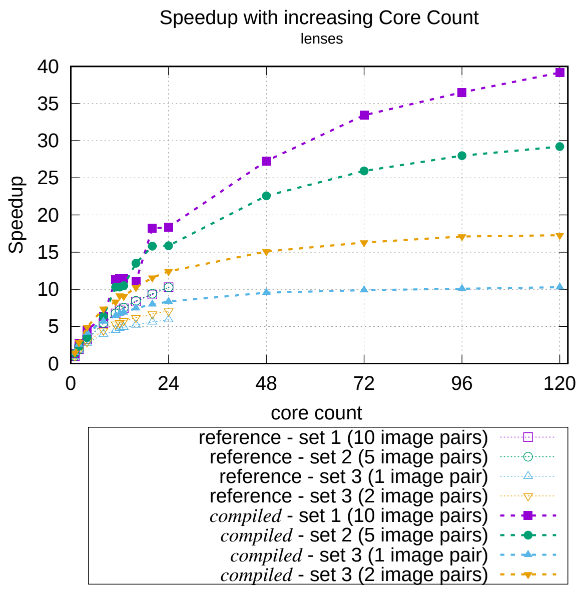


Figure 6.1: Speedup of optimized implementation relative to the increase in the number of processing cores used

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220.

Figure 6.1 shows a significant speed-up of the final MPI implementation when compared to the Python implementation. Tests performed using data sets containing CRL measurements in differenttial mode showed a maximum acceleration factor of 40 for Set 1.

During the course of this study, several more trivial optimization solutions have been implemented, ultimately showing that further potential in terms of optimization can be achieved. For example, node-internal communication could be done with intra-communicators or shared optimized memory to reduce the copying effort of the data. Although many optimization solutions have already been implemented in this work and a high speed-up has been achieved, more could be gained. It would be possible to accelerate the gradient integration and also form data blocks based on the most frequently called functions, processing them in such a way that the overhead of the function calls decreases. Further processing of these data blocks by means of numpy, numexpr or numba could be attempted. The CUDA and OpenCL interfaces for numba could also be used here, implementing a General Purpose Computing on Graphics Processing Units (GPGPUs) outsource.

Another point to improve would be load balance. This would mean grouping the image pairs into packages and editing them one after the other so that a continuous operation could be achieved. Other optimizations of the algorithm are still conceivable, for example in the template matching part and the initial calibration.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220.

7 Conclusions and Outlook

With six months left until the official release date of the code, the main modules of the software package have been written and tested. As expected, they are performing well. The following months will be dedicated to continuing testing the Python code on other data sets, to creating a user interface that unifies the different modules and helps the user easily introduce the needed input parameters, and to writing a user manual for the software package.

After the release date at the end of September 2018, we will foresee more functionalities that could be added to the code, especially in terms of expanding its use beyond the XST method.

As shown in Chap. 5 and 6, through the work of our colleagues from WP5: UFDAC, much improvement could be made in terms of optimization, providing additional resources in terms of time and manpower would be made available. Workflow-wise the Python interface is preferable as it helps making the code more readable especially in view of an open-source release. However, for reaching the level of optimization that one might desire for intensive and recurrent use under experimental conditions, a full GPU-based solution written in C/C++ would be the best alternative. The current work done by J. Schenke helped improve the overall performance of the code and solved some bottlenecks. Even so, better results could be obtained in terms of processing times by changing to a full high performance solution.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220.

Bibliography

[1] H. Winnick, Particle Accelerator Conference, IEEE (1997).

[2] P. Mercere et al., Proc.SPIE **5921**, 10 (2005) 10.1117/12.622799.

[3] B. Flöter et al., New Journal of Physics **12**, 083015 (2010).

[4] M. Idir et al., Nuclear Instruments and Methods in Physics Research A **616**, 162–171 (2010) 10.1016/j.nima.2009.10.168.

[5] S. Rutishauser et al., Nature Communications **3**, 947, 947 (2012) 10.1038/ncomms1950.

[6] S. Berujon et al., Journal of Synchrotron Radiation **22**, 886–894 (2015) 10.1107/S1600577515005433.

[7] T. Roth et al., *X-ray Refractive Lenses Metrology. Experiment at BM05, ESRF* (Apr. 2017).

[8] S. Berujon et al., Phys. Rev. Applied **5**, 044014 (2016) 10.1103/PhysRevApplied.5.044014.

[9] S. Bérujon et al., Phys. Rev. Lett. **108**, 158102 (2012) 10.1103/PhysRevLett.108.158102.

[10] *OpenCV-Python: Template Matching*, http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_template_matching/py_template_matching.html, Accessed: March 13th, 2018.

[11] *OpenCV: Template Matching*, https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template_matching/template_matching.html, Accessed: March 13th, 2018.

[12] R. T. Frankot et al., IEEE Trans. Pattern Anal. Mach. Intell. **10**, 439–451 (1988) 10.1109/34.3909.

[13] P. O'Leary et al., in IEEE Indian Conference on Computer Vision, Graphics and Image Processing (2008).

[14] P. O'Leary et al., IEEE T. Instrumentation and Measurement **61**, 1237–1251 (2012).

[15] C. Jordan, *pyGrad2Surf*, https://github.com/cjordan/pyGrad2Surf, Accessed: March 13th, 2018.

[16] J. Schenke, "Parallelisierung des Wellenfrontrekonstruktionsalgorithmus auf Multicore-Prozessoren", bachelor thesis (Technische Universität Dresden, Mar. 2018).

[17] O. Grisel et al., *Embarrassingly parallel for loops*, https://github.com/joblib/joblib/blob/master/doc/parallel.rst#old-multiprocessing-backend,

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220.

Accessed: February 24th, 2018.

[18] J. Schenke, *CPU Utilization*, https://github.com/ComputationalRadiationPhysics/Wavefront-Sensor/blob/f2fc5c2e5f8b4ef0e9b3ca3a4e770db67f230588/doc/cpu_util.md", (Private), 2018.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220.