EUCALL

The European Cluster of Advanced Laser Light Sources

Grant Agreement number: 654220

Work package 7 – Work package PUCCA

Deliverable D7.8

Wavefront analysis software package

SWaRP: Speckle Wavefront Reconstruction Package – User Manual

Lead Beneficiary: ESRF

Authors: Elena-Ruxandra Cojocaru, Sebastien Berujon, Eric Ziegler

Due date: 30 September 2018
Date of delivery: 25 September 2018

Project webpage: www.eucall.eu

Deliverable Type	
R = Report DEM = Demonstrator, pilot, prototype, plan designs DEC = Websites, patents filing, press & media actions, videos, etc. OTHER = Software, technical diagram, etc.	OTHER
Dissemination Level	
PU = Public, fully open, e.g. web CO = Confidential, restricted under conditions set out in Model Grant Agreement CI = Classified, information as referred to in Commission Decision 2001/844/EC	PU

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 Quick User Guide

SWaRP (Speckle Wavefront Reconstruction Package) is a numerical tool intended for wavefront recovery and reconstruction in both differential and absolute metrology modes, based on the X-ray Speckle Tracking (XST) principle. This pure Python software package consists of two main scripts (*detectorDistortion.py* and *waveFront.py*) and several libraries (*func.py* - containing the main functions of the code, *EdfFile.py* - dedicated to handling the EDF file format, *norm_xcorr.py* - computes subset displacements using cross-correlation in the real-space, *g2s.py* - performs *grad2surf* integration). Currently, the code is designed to be run directly from a terminal without a graphical user interface.

1.1 Installation

The SWaRP software package can be found at `https://gitlab.esrf.fr/cojocaru/swarp`, as of September 2018.

The README.md file gives installation instructions specific to the user's Operating System (OS).

1.2 Running the code

Here is an example on how to run the code:

```
#!/usr/bin/python

# call format: python waveFront.py image1 image2 file.ini
python waveFront.py ./testdata/image1.edf ./testdata/image2.edf ./
    testdata/test_wf.ini
# call format: python detectorDistortion.py file_directory file.ini
python detectorDistortion.py ./testdata/dd_files/ ./testdata/input_dd.
    ini
```

Listing 1.1: Run format and examples.

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

This example uses images in the EDF (ESRF Data format) provided in the software package as test data. The code can handle different file formats, as detailed in Sec. 4.2.

The *detectorDistortion.py* script requires the path to the directory containing the files to be processed and the path to the *ini file*, containing all other parameters. This script is used to compute the detector pixel size and two distortion correction maps.

The *waveFront.py* script needs two paths, for the pair of images and the path to the *ini file*, containing all other input parameters.

1.3 Input parameters

Tables 1.1 and 1.2 illustrate the content of the *ini files* needed for running a script. The correct input information has to be added to such a file by the user before calling the code. Wrong or absent input will either be replaced by a default value (when possible) or generate an error during execution. Details regarding accepted value ranges and default values for each user parameter are given in Tables 1.1 and 1.2.

The parameters can be divided in the following categories:

- Experimental values (e.g. photon energy, distance, pixel size).
- Paths to correction fields (dark field, flat field, distortion maps); prefix strings for these files in case the user wants to select only certain files from the directory.
- Prefix strings for the input image files (for *detectorDistortion.py*) and the file type.
- The path to the output directory.
- The metrology mode (select absolute or differential).
- Boolean parameters to enable or disable some options, e.g. choose if to undistort the input files.
- variables related to the cross-correlation process (e.g. the half-size of the correlation window, the grid resolution) and the reconstruction (e.g. choosing the integration method).

To better understand the purpose of some of these parameters, we recommend the user to read the following sections. These contain short explanations on how an XST experiment is organized, what data is acquired and how it is processed.

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

Table 1.1: Table describing all the input parameters that must be introduced by the user in the *ini file* before running *detectorDistortion.py*

Key	Description	Value type	Accecpted Values	Default	Comments
motor_step	Step for motors that move the detector on a N^2 grid	float	float > 0		1 (Frankot-Chellappa), 2 (grad2surf). See Sec. 4.1
fast_axis	Specify the direction of the fast motor in the nested mesh scan	int	1, 2		1 (horizontal), 2 (vertical). Needed. See Sec. 4.2
dir_out	Path to output directory	string	Absolute or relative path in OS format		Needed. Must have writing permissions. See Sec. 4.2
path_dark	Dark field file(s) detector 1	string	Absolute or relative path in OS format / 'none'	'none'	See Sec. 4.3 *
prefix_dark	Prefix dark field file(s)	string	Prefix string / 'none'	'none'	See Sec. 4.3
path_flat	Flat field file(s)	string	Absolute or relative path in OS format / 'none'	'none'	See Sec. 4.3 *
prefix_flat	Prefix flat field file(s)	string	Prefix string / 'none'	'none'	See Sec. 4.3
prefix_files	Prefix file(s)	string	Prefix string / 'none'	'none'	See Sec. 4.2 *
file_type	Image file type (suffix)	string	'edf' / 'tif' / 'tiff' / 'hdf5' / 'h5'		Needed. Case insensitive. See Sec. 4.2
ROI_mode	Region of interest mode	string	'auto' / 'full' / 'input'	'full'	Case insensitive. See Sec. 4.3

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

ROI_default	Region of interest	string	'int int int int' / 'none'	'none'	int > 0. See Sec. 4.3 *
under_sample	Undersampling step	int	int > 0	1	See Sec. 4.3
grid_resol	Correlation grid resolution for second pass	int	int > 0	1	See Sec. 4.4
corr_size	Half size of correlation window	int	int > 0	10	Correlation size for second (fine) pass. Corresponds to: corr_window = corr_size * 2 + 1. See Sec. 4.4
inter_corr_step	Step between distortion files to use as pair	int	int > 0	1	See Sec. 4.4
resol_first_pass	Correlation size for first (coarse) pass.	int	int > 0	35	See Sec. 4.3
half_width_fact	Factor between search area window and correlation window	float	float > 0	2	See Sec. 4.3
mode	Mode describing how the rigid translation is treated.	int	0 / 1 / 2	2	Rigid translation assumptions: 0 (no assumption), 1 (small variations), 2 (compute rigid translation). See Sec. 4.3
integ_meth	Integration method	int	1 / 2	1	1 (Frankot-Chellappa), 2 (grad2surf). See Sec. 4.5

* Case insensitive for 'none'.
** Should be omitted when using the differential metrology mode.

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

Table 1.2: Table describing all the input parameters that must be introduced by the user in the *ini file* before running *waveFront.py*

Key	Description	Value type	Accecpted Values	Default	Comments
energy	Photon energy	float	float > 0		Unit: keV. Needed. See Sec. 4.5
dist	Relevant distance	float	float > 0		Unit: mm. Needed. For absolute metrology, introduce distance between the two detectors. For differential metrology, introduce the distance between the detector and the speckle membrane or between the detector and the sample, whichever is shorter. See Sec. 4.5
pix_size1	Pixel size detector 1	float	float > 0	1	Unit: micrometer. See Sec. 4.5
pix_size2	Pixel size detector 2	float	float > 0	pix_size2	Unit: micrometer. See Sec. 4.5 **
diff_wf	Set differential mode	Boolean / string / int	True / False / 'true' / 'false' / 1 / 0	True	Case insensitive for strings. Will try to convert input into a True/ False Boolean value. See Sec. 4.4
undistort	Undistort images	Boolean / string / int	True / False / 'true' / 'false' / 1 / 0	False	Case insensitive for strings. Will try to convert input into a True/ False Boolean value. See Sec. 4.3
file_type	Image file type (suffix)	string	'edf' / 'tif' / 'tiff' / 'hdf5' / 'h5'		Needed. Case insensitive. See Sec. 4.2

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

dir_out	Path to output directory	string	Absolute or relative path in OS format		Needed. Must have writing permissions. See Sec. 4.2
path_dark1	Dark field file(s) detector 1	string	Absolute or relative path in OS format / 'none'	'none'	See Sec. 4.3 *
path_dark2	Dark field file(s) detector 2	string	Absolute or relative path in OS format / 'none'	'none'	See Sec. 4.3 *, **
prefix_dark1	Prefix dark field file(s) detector 1	string	Prefix string / 'none'	'none'	See Sec. 4.3 *
prefix_dark2	Prefix dark field file(s) detector 2	string	Prefix string / 'none'	'none'	See Sec. 4.3 *, **
path_flat1	Flat field file(s) detector 1	string	Absolute or relative path in OS format / 'none'	'none'	See Sec. 4.3 *
path_flat2	Flat field file(s) detector 2	string	Absolute or relative path in OS format / 'none'	'none'	See Sec. 4.3 *, **
prefix_flat1	Prefix flat field file(s) detector 1	string	Prefix string / 'none'	'none'	See Sec. 4.3 *
prefix_flat2	Prefix flat field file(s) detector 2	string	Prefix string / 'none'	'none'	See Sec. 4.3 *, **
horz_disto_map1	Horizontal distortion map detector 1	string	Absolute or relative path in OS format / 'none'	'none'	See Sec. 4.3 *
vert_disto_map1	Vertical distortion map detector 1	string	Absolute or relative path in OS format / 'none'	'none'	See Sec. 4.3 *

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

horz_disto_map2	Horizontal distortion map detector 2	string	Absolute or relative path in OS format / 'none'	'none'	See Sec. 4.3 *, **
vert_disto_map2	Vertical distortion map detector 2	string	Absolute or relative path in OS format / 'none'	'none'	See Sec. 4.3 *, **
ROI_mode	Region of interest mode	string	'auto' / 'full' / 'input'	'full'	Case insensitive. See Sec. 4.3
ROI_default1	Region of interest for first image (or batch)	string	'int int int int' / 'none'	'none'	int > 0. See Sec. 4.3 *
ROI_default2	Region of interest for second image (or batch)	string	'int int int int' / 'none'	'none'	int > 0. See Sec. 4.3 *, **
under_sample	Undersampling step	int	int > 0	1	See Sec. 4.3
image_order	Search for image 1 in image 2?	int	1//2	1	1 (search for image 1 in image 2), 2 (search for image 2 in image 1). See Sec. 4.4
grid_resol	Correlation grid resolution for second pass	int	int > 0	1	See Sec. 4.4
corr_size	Half size of correlation window	int	int > 0	10	Correlation size for second (fine) pass. Corresponds to: corr_window = corr_size * 2 + 1. See Sec. 4.4
resol_first_pass	Sampling grid step for the first (coarse) pass.	int	int > 0	35	See Sec. 4.4

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

half_width_fact	Factor between search area window and correlation window	float	float > 0	2	See Sec. 4.4
mode	Mode describing how the rigid translation is treated.	int	0 / 1 / 2	2	Rigid translation assumptions: 0 (no assumptions), 1 (small variations), 2 (compute rigid translation). See Sec. 4.4
integ_meth	Integration method	int	1 / 2	1	1 (Frankot-Chellappa), 2 (grad2surf). See Sec. 4.5
mask_half_size	Radius of wavefront mask	int	int > 0	0	When value is 0 no mask is applied. See Sec. 4.5

* Case insensitive for 'none'.
** Should be omitted when using the differential metrology mode.

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

2 X-ray Speckle Tracking principle

The X-ray Speckle Tracking (XST) principle relies on using near-field speckle [1] as markers of the trajectory of the X-rays. Near-field speckle represents a random intensity pattern produced by the interference between the transmitted beam and the waves scattered by the speckle membrane. In this near-field regime, the size and shape of the speckle suffer only small changes with distance. In brief, by considering two speckle images, a cross-correlation algorithm can identify and compute the displacement of small subsets of pixels from the first image in the second one. These displacements are computed with sub-pixel accuracy and eventually translate into the wavefront gradient information. Lastly, through bidimensional integration of the two orthogonal phase gradient maps, the wavefront is reconstructed.

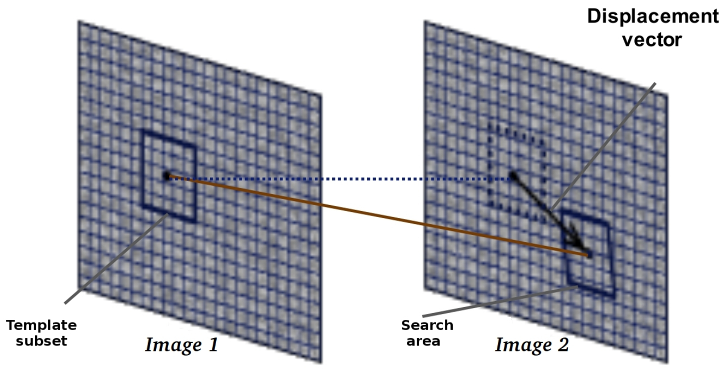

Figure 2.1: XST principle: visual representation of how the position of a template subset from one image is tracked in the second image using cross-correlation [2]

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

3 Experiment overview

Speckle-based wavefront metrology can be provided 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 or simply by a ”sample”. This mode allows for the individual characterization of optical elements by isolating their contribution on the X-ray beam phase, but also to operate phase contrast imaging by looking at the phase shift induced by a sample. For example, it was used at the ESRF for the individual characterization of each X-ray refractive lens composing an optical system [3] and for speckle tracking X-ray phase contrast imaging [4]. 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.

(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. If the beam is considered stable over time, the images can be taken sequentially by moving the detector along the beam propagation direction during the time interval between consecutive image recordings. However for the characterization of a pulsed beam, it is more demanding as it requires the use of two synchronized detectors plus a speckle membrane (see Fig. 3.1). In addition, the first detector optics must be semi-transparent and designed so that both detectors receive similar flux levels.

Figure 3.1 is a sketch of a proposed setup that allows to perform XST in both differential and absolute metrology mode, with the condition that the first detector must be semi-transparent. Such a setup enables all the modes explained above. The Concept Design Report for building a prototype of this instrument can be found here [5].

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

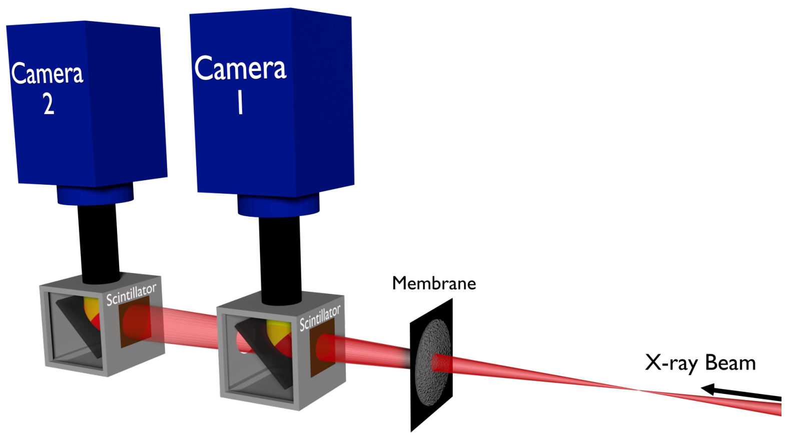

Figure 3.1: Sketch of experimental setup for absolute wavefront metrology

When the coherent beam passes through the low absorbing material (labelled membrane in the Fig. 3.1), a speckle pattern is generated. For the absolute metrology mode, an image of this speckle pattern is captured simultaneously, or sequentially in the case of a stable beam, by the two cameras, situated at different distances from the membrane. Employing the steps described in the following sections, the analysis of these two images allows the reconstruction of the wavefront of a pulse or of a series of pulses depending on the measuring setup (camera performance, X-ray beam configuration).

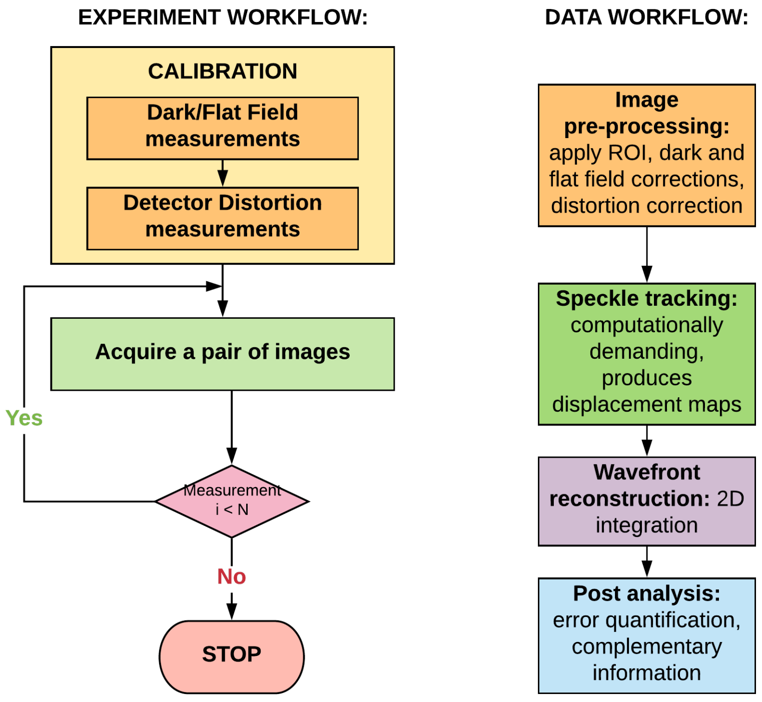
EXPERIMENT WORKFLOW:
CALIBRATION
Dark/Flat Field measurements
Detector Distortion measurements
Acquire a pair of images
Yes
Measurement i < N
No
STOP
DATA WORKFLOW:
Image pre-processing: apply ROI, dark and flat field corrections, distortion correction
Speckle tracking: computationally demanding, produces displacement maps
Wavefront reconstruction: 2D integration
Post analysis: error quantification, complementary information

Figure 3.2: Experiment and data workflows

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

An accurate speckle-based wavefront metrology experiment involves two stages:

CALIBRATION (should be done once at the beginning of the experiment or when the detector/optics system is modified):

1. Take dark field images for both detectors (take one image or take N images, compute the bidimensional median and use this for correction).
2. Take flat field images for both detectors (take one image or take N images, average them and use the average image for correction).
3. Record images while moving each detector over a grid of positions in the beam transverse plan, in order to then compute the detector distortion for each detector. N^2 images have to be collected for each detector to increase the accuracy. N is then the lateral size of the grid on which the detector is moved. This step can be skipped if no alterations have been made to the detector since the previous detector distortion measurements.

MAIN MEASUREMENTS (measure for each pulse or series of pulses depending on the measuring setup):

- When performing absolute pulse metrology: record simultaneous images with the two detectors for individual pulse characterization. This results in two images per acquisition for every measured pulse. The wavefront reconstruction is performed from every pair of images.
- When performing differential metrology or phase contrast imaging: take two images with the same detector, a *sample* image and a *reference* image. The reconstruction will represent the wavefront variations between the two.

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

4 Data workflow

4.1 Main scripts and flow-charts

The SWaRP package features two main scripts as of September 2018:

detectorDistortion.py:

- Needs N^2 images as input.
- Characterize vertical and horizontal distortion for (each) detector [6].
- Needs to be computed only once, at the beginning of the experiment, unless the camera or the optics are modified during the experiment; similarly, this step can be skipped if the detector distortion maps have been computed previously and the detector/optics were not modified in the meantime.
- Produces an accurate estimation of the detector pixel size, based on dividing the motor step size in millimeters (**motor_step**) by the average rigid translation in pixels.
- Produces two (horizontal and vertical) distortion maps as output; these can be then used as input for *waveFront.py* computations, to correct the distortion for each pair of processed images.

waveFront.py:

- Needs 2 images as input.
- Uses the XST method for individual pulse characterization and wavefront reconstruction.
- Can be used in both absolute and differential modes.
- Generates as output the wavefront reconstruction corresponding to the chosen metrology mode.

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

As one can notice from the workflow diagrams (Fig. 4.1 and 4.2), both scripts share a number of similar steps:

- file handling.
- image pre-processing.
- cross-correlation.
- integration.

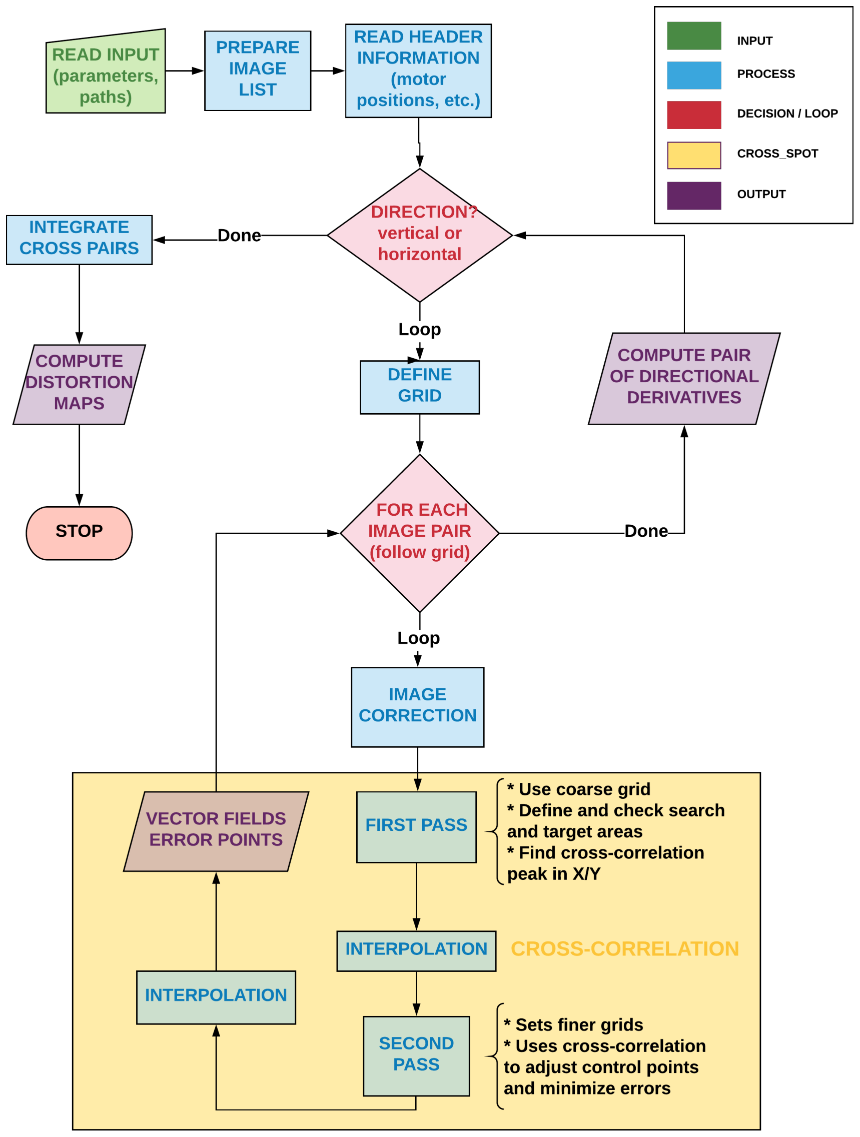
READ INPUT (parameters, paths)
PREPARE IMAGE LIST
READ HEADER INFORMATION (motor positions, etc.)
INPUT
PROCESS
DECISION / LOOP
CROSS_SPOT
OUTPUT
INTEGRATE CROSS PAIRS
Done
DIRECTION? vertical or horizontal
Loop
COMPUTE DISTORTION MAPS
DEFINE GRID
COMPUTE PAIR OF DIRECTIONAL DERIVATIVES
STOP
FOR EACH IMAGE PAIR (follow grid)
Done
Loop
IMAGE CORRECTION
VECTOR FIELDS ERROR POINTS
FIRST PASS
* Use coarse grid
* Define and check search and target areas
* Find cross-correlation peak in X/Y
INTERPOLATION
CROSS-CORRELATION
INTERPOLATION
SECOND PASS
* Sets finer grids
* Uses cross-correlation to adjust control points and minimize errors

Figure 4.1: Detector Distortion workflow diagram

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

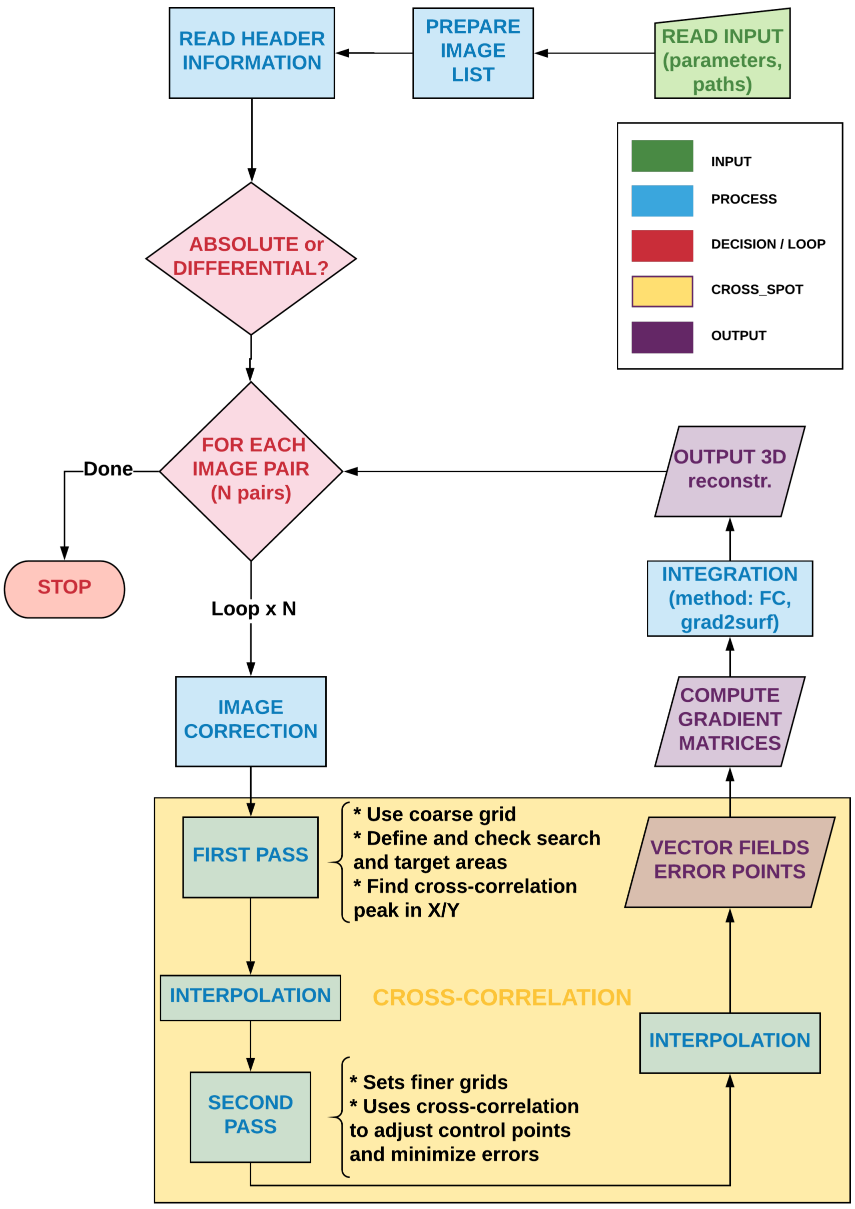
READ HEADER INFORMATION
PREPARE IMAGE LIST
READ INPUT (parameters, paths)
INPUT
PROCESS
DECISION / LOOP
CROSS_SPOT
OUTPUT
ABSOLUTE or DIFFERENTIAL?
FOR EACH IMAGE PAIR (N pairs)
Done
STOP
Loop x N
OUTPUT 3D reconstr.
INTEGRATION (method: FC, grad2surf)
COMPUTE GRADIENT MATRICES
IMAGE CORRECTION
FIRST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

Figure 4.2: Wavefront Reconstruction workflow diagram

4.2 File handling

The software package can currently handle three different file formats: EDF (ESRF Data Format), TIFF and HDF5. This is specified by the parameter **file_type**.

The path to the input files is specified when calling the scripts, as shown in Sec. 1.2. If needed, the user can set the parameter **prefix_files** to filter the files in the directory and use only the ones with names starting with that prefix. Also, when sorted alpha-numerically, the names of the files should follow the grid describing the nested movement of the detector. The user must

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

also specify which is the **fast_axis** for the nested grid movement of the two motors (1 - the horizontal axis, 2 - the vertical axis).

For the *detectorDistortion.py*, the path directory containing the set of N^2 files must be given when calling the script, whereas for the *waveFront.py* script two paths are needed, indicating the location of the pair of images to be processed.

In both cases, the user must also specify a path with writing permissions for the directory where all output files will be stored. This is done in parameter **dir_out** contained in the *ini* files.

4.3 Image pre-processing

For the files to be processed by the *detectorDistortion.py* script, the user can provide the paths to dark field (**path_dark**) and flat field files (**path_flat**). This paths can indicate the location of an individual file or a directory. In case it is a directory, the user can specify a prefix to filter the files in the directory (using the parameters **prefix_dark** and **prefix_flat**). Also, in case there are multiple files, the dark field correction will be obtained as the two-dimensional median of these files, while the flat field correction will be obtained as the two-dimensional average. Usually, dark and flat images generated from larger image stacks are better.

The flat and dark field correction can be inputed similarly for the *waveFront.py* script. One can also input paths for the detector distortion maps. When setting the parameter **undistort** to True, the correction will be performed. If it is set to False, the undistortion correction will be skipped.

The type of metrology is specified by the parameter **diff_wf**, where the value True indicated differential metrology, and the value False absolute metrology. For absolute metrology, the user can specify a dark field path (**path_dark1**, **path_dark2**), a flat field path (**path_flat1**, **path_flat2**) and distortion maps (**horz_disto_map1**, **vert_disto_map1**, **horz_disto_map2**, **vert_disto_map2**) for each detector, whereas for differential metrology the user can specify these once (**path_dark1**, **path_flat**, **horz_disto_map1**, **vert_disto _map1**).

Any path should be specified as a relative or absolute path in the format specific to the OS of the user. If the parameters related to the dark and flat field corrections are left with the default ”none” value, these corrections will simply not be performed.

Another important input parameter is the Region Of Interest (ROI) for each file to be processed. For *detectorDistortion.py* and for differential metrology, one must specify the **ROI_mode**, which can be ’auto’ (ROI is automatically calculated), ’full’ (the full image will be used) or ’input’. For the latter, the user must also introduce **ROI_default** (or **ROI_default1**), a string containing

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

four positive integer numbers separated by spaces. These correspond to '$y_{min}\ y_{max}\ x_{min}\ x_{max}$'. For absolute wavefront metrology in 'input' mode, the user must specify the region of interest for each image in the pair (**ROI_default1** or **ROI_default2**).

For the code to produce correct results, it is essential that the region of interest correspond to an area of the image that has fully been exposed to the X-rays. If the region of interest contains areas where there are no speckles, then the cross-correlation will fail to produce a peak for those areas and no accurate displacement will be computed.

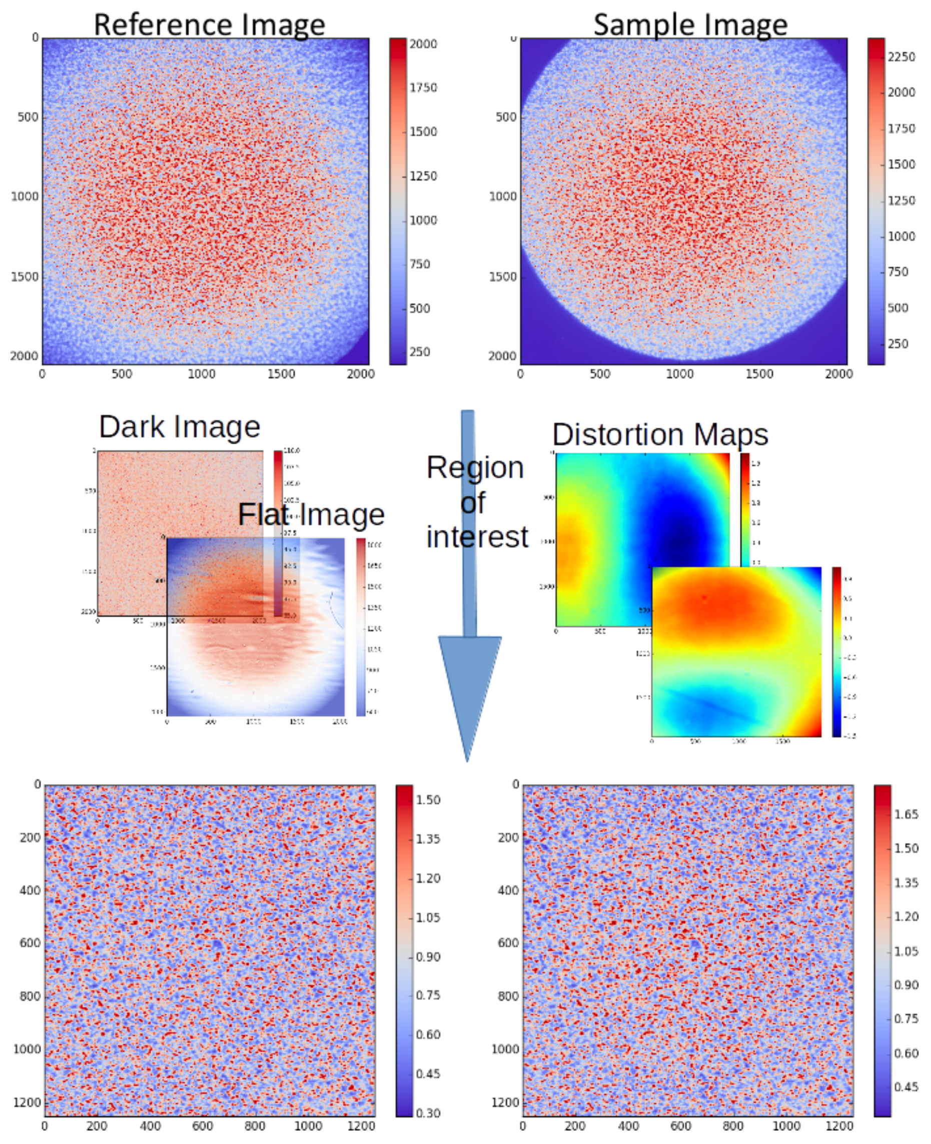

Figure 4.3: Example of a pair of images measured in differential metrology mode. A *reference* image and a *sample* image were taken before and after introducing an individual optical element the beam. The images are then preprocessed by applying the dark and flat field corrections, undistorting the images and applying the desired ROI. The resulting corrected images are shown on the bottom row.

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

4.4 Cross-correlation

A key component of the XST method is the normalized cross-correlation algorithm. To fulfil this function, we use the *TemplateMatching* algorithm implemented in C++, interfaced for Python from the OpenCV2 library [7]. This algorithm 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 returns a cross-correlation matrix, of size $(m-a+1) \cdot (n-b+1)$. This matrix contains float type numbers ranging 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 [8]):

$$R(x,y) = \frac{\sum_{x',y'}(T'(x',y') \cdot A'(x+x',y+y'))}{\sqrt{\sum_{x',y'}(T'(x',y'))^2 \cdot \sum_{x',y'}(A'(x+x',y+y'))^2}} \tag{4.1}$$

where:

$$\begin{aligned} T'(x',y') &= T(x',y') - \tfrac{1}{w \cdot h}\textstyle\sum_{x'',y''} T(x'',y'') \\ A'(x',y') &= A(x+x',x+y') - \tfrac{1}{w \cdot h}\textstyle\sum_{x'',y''} A(x+x'',x+y'') \end{aligned} \tag{4.2}$$

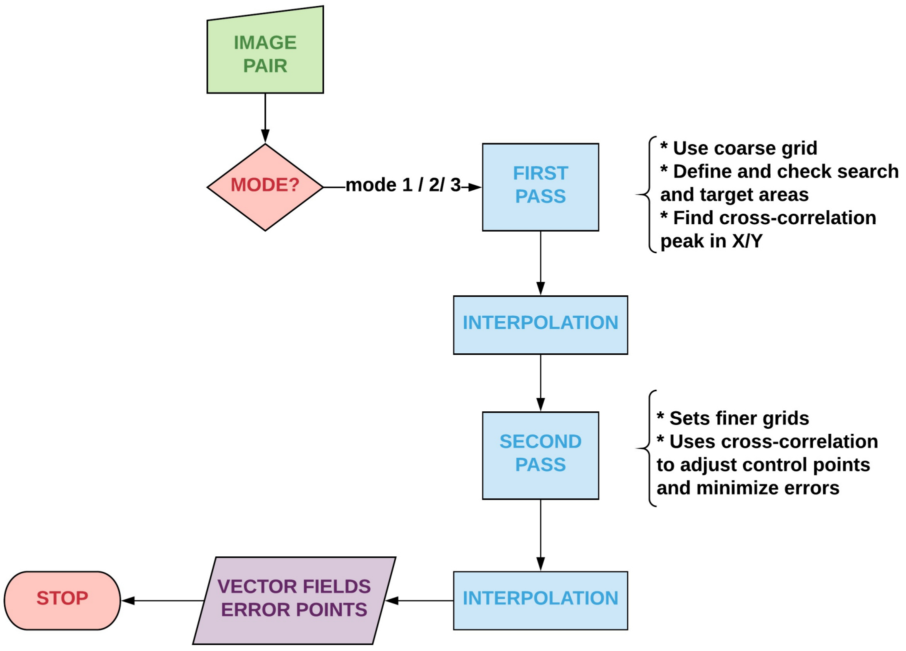
IMAGE PAIR
MODE?
mode 1 / 2/ 3
FIRST PASS
* Use coarse grid
* Define and check search and target areas
* Find cross-correlation peak in X/Y
INTERPOLATION
SECOND PASS
* Sets finer grids
* Uses cross-correlation to adjust control points and minimize errors
INTERPOLATION
VECTOR FIELDS ERROR POINTS
STOP

Figure 4.4: *cross_spot* subroutine workflow diagram

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

The correlation matrix returned by the template matching algorithm is 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 centred onto pixels of a grid of Image 1 are searched for in a corresponding search areas in Image 2. Figure 4.4 illustrates the flow diagram of the corresponding subroutine in our code, entitled *cross_spot*. This process involves three stages: a rigid translation calculation (done once for each image pair), then a *first pass* and a *second pass*.

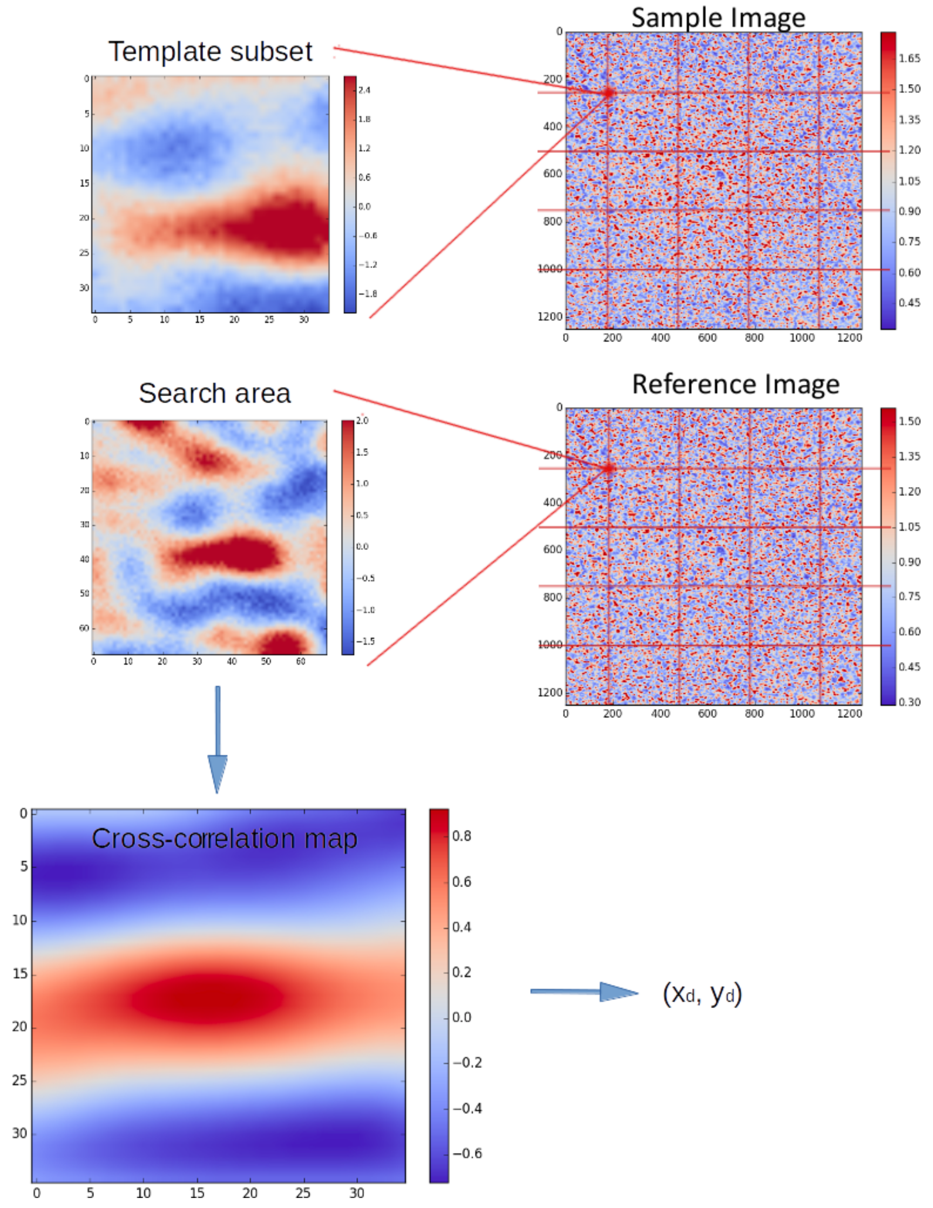

Figure 4.5: Visual representation showing how a grid is overimposed over each image and cross-correlation is performed by ”searching” for a template subset centred in each node in a corresponding search area in the other image

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

In the *first pass*, a coarse grid is applied to the image, centring a template subset in each pixel of this grid in Image 1 with a corresponding larger search area in Image 2. An example of this is illustrated in Fig. 4.5. The parameter **resol_first_pass** sets the resolution of this coarse grid. 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 an internal parameter of the code, called *mode*. This specifies whether to compute and consider the rigid translation of the speckle pattern, depending on which type of metrology is performed. 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 mesh grid. The parameter **grid_resol** sets the resolution of the finer grid. This second pass consists of a loop where parameters, such as the template subset size, are optimized. The user sets an initial size for the template window through the parameter **corr_size**. This value is actually a half-size, the actual size of the template window is calculated as **corr_size** $* 2+1$.

The cross-correlation parameters affect the computational speed and the accuracy of the results. A sampling mesh with a coarser step can speed-up the code, but might also reduce the accuracy of the computed wavefront. Similarly, the size of the template window must be sufficiently large as to obtain accurate results, but the larger the template window, the slower the computation will be and the lower the final resolution. 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 maximum values, until the errors in each control point are satisfactorily minimized. Another user parameter is **half_width_fact**, specifying the factor between the size of the template and that of the search area. Usually, the default value 2 for this factor is sufficient, but if displacements are very large, it might help to increase this value slightly, although that will incidentally increase the computational time. Finally, the obtained displacement maps are again interpolated for all the pixels in the image to recover the original numerical sampling resolution (different from the optical sampling resolution).

4.5 Integration

As previously explained, this method is sensitive to the gradient of the wavefront (see Eq. 4.3), and a integration stage is necessary to obtain the reconstructed wavefront.

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

$$\alpha = -\frac{\lambda}{2\pi}\frac{\partial \phi}{\partial x} = -\frac{\partial W}{\partial x} \tag{4.3}$$

where: α is the angular displacement, λ is the wave-length of the X-rays, ϕ the phase and W the wavefront.

In *waveFront.py* script, once the two gradient maps have been computed, a three-dimensional reconstruction of the wavefront is obtained by integrating these two gradient maps. The user can choose between two integration methods set by the parameter **integ_meth**. The value 1 specifies to use an implementation of the Frankot-Challappa algorithm [9] based on the Fast Fourier Transform (FFT), whereas the value 2 corresponds to the *grad2surf* algorithm [10, 11].

Next, the user can input a positive integer value for the parameter **mask_half_size** that specifies the radius of a circular mask. This mask can be used to isolate a certain region of the three-dimensional reconstruction, which can prove useful for post-analysis. An example of a three-dimensional reconstruction is given in Fig. 4.6.

For this final reconstruction to be quantitatively correct, one must also take into account the distance (parameter **dist**) and the pixel size (**pix_size1** and **pix_size2**). For the absolute mode, one must specify the distance in millimeters between the two detectors and the pixel size in micrometers for each detector. The pixel size can be obtained by previously running the detector distortion measurement procedure for measurements from each detector. For the differential metrology mode, one must specify the distance in millimeters between the detector and the speckle membrane or between the sample and the detector, whichever is the shortest, in addition to the the pixel size of the detector (**pix_size1**).

The **energy** of the X-rays is eventually used to convert the wavefront information to phase information.

The *detectorDistortion.py* script also has an integration step, to obtain the distortion maps from the pairs of directional derivatives, the same parameter **integ_meth** is used here.

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

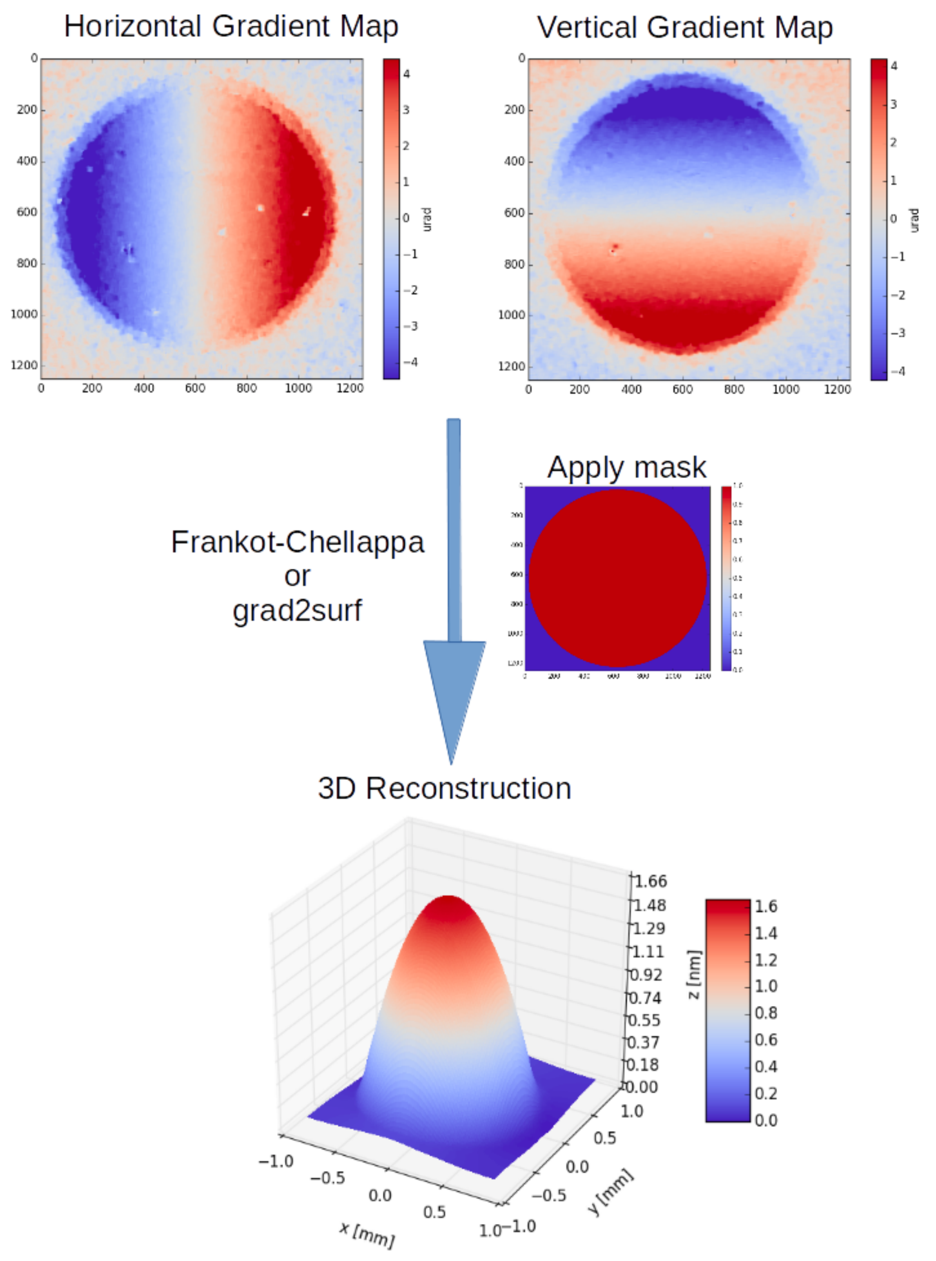

Figure 4.6: After having obtained the horizontal and vertical gradient maps, these are integrated to obtain the three-dimensional reconstruction and a circular mask is also applied. This mask allows to isolate exclusively the contribution of the individual optical element inserted in the beam. See text for more details.

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

Bibliography

[1] R. Cerbino et al., Nature Physics **4**, 238–243 (2008) 10.1038/nphys837.

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

[3] T. Roth et al., *X–ray Refractive Lenses Metrology. Experiment at BM05, ESRF*, Apr. 2017.

[4] H. Wang et al., Scientific Reports **5**, 8762 (2015) 10.1038/srep08762.

[5] R. Cojocaru et al., *Deliverable D7.7: CDR to build a wavefront sensor for [8-25] keV*, Sept. 2018, 10.5281/zenodo.1434896.

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

[7] *OpenCV-Python: Template Matching*, http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_template_matching/py_template_matching.html.

[8] *OpenCV: Template Matching*, https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template_matching/template_matching.html.

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

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

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

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