Number	8
Publication Year	2020
Acceptance in OA@INAF	2020-02-27T14:59:50Z
Title	Out-Of-Focus Holography Tool for the Sardinia Radio Telescope
Authors	BUFFA, Franco; PINNA, Andrea; POPPI, Sergio; MURGIA, MATTEO; SERRA, Giampaolo
Affiliation of first author	O.A. Cagliari
Handle	http://hdl.handle.net/20.500.12386/23075; http://dx.doi.org/10.20371/INAF/TechRep/8

Out-Of-Focus Holography Tool for the Sardinia Radio Telescope

F. Buffa[1], A. Pinna[2], S. Poppi[1], M. Murgia[1] and G. Serra[3]

[1]INAF - Osservatorio Astronomico di Cagliari
[2]CRS4 - Centro di Ricerca, Sviluppo e Studi Superiori in Sardegna
[3]ASI - Cagliari

Authors

author	email
Franco Buffa	franco.buffa@inaf.it
Andrea Pinna	andrea.pinna@crs4.it
Sergio Poppi	sergio.poppi@inaf.it
Matteo Murgia	matteo.murgia@inaf.it
Giampaolo Serra	giampaolo.serra@asi.it

Acronyms

AS	Active Surface
FPA	Focal Plane Array
IF	Intermediate Frequency
LUT	Lookup Table
OOF	Out-Of-Focus Holography
SNR	Signal to Noise Ratio
SR	Spatial Resolution
SRT	Sardinia Radio Telescope

Contents

1 Introduction

In the last few years, we have successfully commissioned and tested a new primary focus holographic system for the Sardinia Radio Telescope (SRT) [1]. Based on the phase-coherent microwave holography technique, the system has allowed to measure and correct the SRT primary mirror deformations with high spatial resolution (SR) and accuracy at two antenna elevation angles (30 and 44 deg), where the geosynchronous satellites are available at the telescope latitude.

Recently, we decided to extend the analysis to the entire elevation range by implementing the out-of-focus holography (OOF), a phase-retrieval technique, proposed by B. Nikolic et al., which is routinely used in many single-dish facilities [2, 3, 4, 5]. In this approach, the telescope aperture field phase is retrieved by measuring three antenna far-field maps (one in-focus and the other two got by symmetrically defocusing the sub-reflector) in a standard configuration for radio astronomy observations and, then, implementing a fitting algorithm based on Zernike polynomials. In our case, we preferred the configuration of the K-band cryogenic receiver, a 7-beam focal plane array (FPA) hosted in the SRT Gregorian focus position, operating in the frequency range 18-26.5 GHz, the telescope maximum frequency and therefore the frequency range demanding the best antenna performances, for the moment. Then we choose to perform the holography test This allows us to consider both the astronomical sources, so extending widely the elevation angular range, and geostationary satellites, but only after ensuring the response linearity of the radio frequency chain.

The original OOF algorithm, developed by Nikolic et al., was recently ported in Python by T. Cassanelli. The Cassanelli's software, `pyoof`, is distributed on GitHub under the following terms:

The original `pyoof` code was deeply revised and adapted to the SRT case. The major changes concern the effect of the extra path, due to the defocus, resulting in the field phase on the antenna aperture plane. The SRT shaped reflectors cannot be described in terms of an analytic function, but they can only be defined as discretized tabulated points. Thus, the phase can be only evaluated by ray-tracing the extra path due to the subreflector displacement Δz long the telescope axis (z-axis) [5]. As the ray-tracing implementation is extremely time-consuming, the defocus contribution must be pre-calculated and parametrized by a bi-variate polynomial fitting where ρ, the radial distance from the antenna axis, and Δz are the variables.

This document represents a short guide to the `pyoof` installation and describes the changes the authors made in the software package to process a SRT OOF data set. Please, refer to the Cassanelli's `pyoof` manual for all the parts not regarding the changes here listed.

The `pyoof` installation procedure and the differences between the two versions are described in Section 2 and 3 respectively. Section 4 deals with the input parameters required by `pyoof` and Section 5 shows how to run the `pyoof` code and which arguments are needed. Moreover, in Section 6, the results of a typical processing of an OOF data set measured with the SRT at 22.23 GHz are discussed and, then, compared to those we got by close range photogrammetric measurement. Finally, summary and conclusions are reported in Section 7.

2 Installing `pyoof`

As specified in the `pyoof` reference manual, the software needs Python 3.7 and the following libraries:

- `setuptools`
- `NumPy`
- `SciPy`
- `Astropy`
- `pytest`
- `matplotlib`
- `PyYAML`
- `pip`

All these libraries must be correctly installed before starting the `pyoof` installation. In the Appendix A we report a warning related to the matplotlib library.

The following procedure, based on a Debian-like distribution, is valid, a part of a few details, for a generic Linux distribution.

2.1 Installing `anaconda`

The best way to guarantee a correct handling of dependencies is to install `anaconda` (Python 3.7). The `anaconda (https://repo.anaconda.com/archive/Anaconda3-2019.03-Linux-x86_64.sh)` installation script, corresponding to the last available stable release, is available at the `anaconda` website together with much more details. To install `anaconda` type the following command in a bash terminal:

```
bash ${PATH_TO_INSTALLATION_SCRIPT}/Anaconda3-2019.03-Linux-x86_64.sh
```

2.2 Installing LaTeX

LaTeX dependencies are not specified in the official `pyoof` documentation, but LaTeX is required by plotting procedures to represent properly math symbols and special characters. The LaTeX package installation command is:

```
sudo apt-get install texlive-full
```

2.3 Installing `pyoof` (official version)

Skip this subsection if you intend to install the SRT version. The best way to install `pyoof` is to type the following command:

```
pip install pyoof
```

As an alternative, you could prefer to install the last version (hence not necessarily a stable version) of `pyoof`. The first step is to clone the GitHub repository:

```
git clone https://github.com/tcassanelli/pyoof.git
```

The second step is the installation:

```
python setup.py install
```

2.4 Cloning the pyoof repository (SRT version)

Before installing `pyoof` for SRT you have to copy the *bundle* file[1] to your PC, for example at this path:

```
${BUNDLE_DIRECTORY}/pyoof-srt.bundle
```

Before cloning we recommend you to verify the *bundle* integrity:

```
git bundle verify ${BUNDLE_DIRECTORY}/pyoof-srt.bundle
```

Once the bundle integrity has been checked, let's move to the `pyoof` code folder, for example:

```
cd ${CODE_DIRECTORY}
```

Now, we can proceed with the `master` branch cloning:

```
git clone -b master ${BUNDLE_DIRECTORY}/pyoof-srt.bundle
```

The SRT `pyoof` code (and the entire repository) will be copied on:

```
${CODE_DIRECTORY}/pyoof-srt
```

2.5 Installing pyoof (SRT version)

From the code folder, we can install `pyoof` by typing:

```
cd ${CODE_DIRECTORY}/pyoof-srt
pip install .
```

Now `pyoof` can be "globally"recognized and run by a script or from a python terminal by typing `import pyoof`.

3 Differences between pyoof and pyoof for SRT

3.1 aperture/aperture.py

The new method `compute_deformation` in `aperture.py` transforms the aperture phase (in radians) to surface deformations (in millimeters), ready to fill the main reflector Lookup Table (LUT) after changing the sign. Moreover, the methods `radiation_pattern` and `aperture` go through the following changes:

- the new argument `opd`, i.e. the optical path difference is now calculated outside the fitting algorithm
- the argument `d_z`, the defocus, is not required anymore.

[1] Send a request to the authors to get the *bundle* file.

3.2 aux_functions.py

The following new methods are now inside `aux_functions.py`:

- `get_run_config`, loads the parameter configuration
- `init_output_dir`, creates the output folder
- `init_logger`, logger, inizializes the log file and standard output
- `extract_real_data_srt`, loads the map files (SRT format)
- `extract_synthetic_data_srt`, loads the synthetic map files (GRASP format)
- `extract_data_srt`, create the `.fits` file, required by `pyoof`, from the map files data
- `precompute_srt_delta_opd`, calculate the optical path for the defocus of shaped mirror

Finally, method `store_data_csv` goes through a few minor changes regarding the antenna deformation data output.

3.3 fit_beam.py

A new parameter loader is now available for methods `compute_deformation` (module `aperture`) and `precompute_srt_delta_opd` (module `aux_functions`) in `fit_beam.py`.

Regarding the methods `residual_true` and `residual` the new `pyoof` for SRT differs in:

- the new argument `opd`, related to the variable having the same name, now evaluated in the method `fit_beam` by a call to `precompute_srt_delta_opd`
- the argument `d_z` deleted and no longer used.

The method `fit_beam` was deeply revised adding new calls to the logger (see next sections) regarding with:

- the extra path due to the defocusing evaluated for shaped reflector antennas (`opd`)
- the new method `compute_deformation` for the deformation analysis (in millimeters)

3.4 plot_routines.py

`plot_routines.py` has now the following new features:

- `aperture` module calls the `compute_deformation` method
- the definition of the `plot_error_map` method and the related call in `plot_fit_path` method is now available for the plot of the deformation map

3.5 telgeometry/telgeometry.py

The `telgeometry.py` script contains now the following new methods:

- `precompute_srt_delta_opd` evaluates the extra path due to the defocusing as an alternative to `opd_srt` (deprecated)
- `block_srt` evaluates the SRT aperture blockage due to the quadripode and the subreflector
- `block_srt_wo_legs` evaluates the SRT aperture blockage due to the subreflector only
- `block_srt_wo_legs_and_sr` no blockage is considered

3.6 sample_input_files

The folder sample_input_files contains some demo files:

- run_config.yaml, is a configuration file used for processing a standard measured SRT OOF data set
- synthetic_run_config.yaml, is a configuration file used for processing a simulated SRT OOF data set calculated by GRASP-TICRA
- opt_vars.yaml, configuration file used for the setup of the optimization variables
- 20190426-101052-S0000-MAPPA_IN.txt, 20190426-110938-S0000-MAPPA_OUT1.txt and 20190426-112403-S0000-MAPPA_OUT2.txt, are three files of a typical SRT OOF measured data set, available in the real_data folder
- ffmap_in.grd, ffmap_-out.grd and ffmap_+out.grd, are three files of a typical SRT OOF simulated data set, available in the synthetic_data folder

4 Input parameters required by pyoof

The file run_config.yaml contains the configuration parameters used by pyoof. The extension .yaml refers to the YAML markup language, for which a Python library is available. As a general rule, the parameter type (str, float, int and bool) must be explicitly specified. A second .yaml file (the name is defined by the user) sets the optimization variables by means of the optimization_variables parameter. The parameters are grouped in the following five sets:

- params
  - radiotelescope, the radio telescope name
  - radius, the primary mirror radius (m)
  - focus_primary_reflector, primary mirror focal length (m)
  - total_focus, effective total focal length of the telescope (m)
  - frequency, the signal frequency (Hz)
  - delta_z, the subreflector defocus long the telescope axis (m), a negative value moves the subreflector towards the main reflector, a positive one in the opposite direction
  - residual_opd is a flag used to select the optical path correction required by shaped profiles. It is True for SRT case (bool)
- fit
  - optimization_variables, a user defined .yaml file contains the initial values and the limits of the range of the fitting variables. In addition, the user can select a sub-set of variables to be excluded in the optimization algorithm
  - optimization_method selects one of the three least square optimization methods listed here below: Trust Region Reflective (trf), Levenberg-Marquardt (lm) and Dogleg (dogbox)
  - fit_previous, if True the code uses the fitting results of the k-1th Zernike polynomial order as start values for the kth one (bool)
  - max_order, maximum Zernike polynomial order at which the fit stops
  - pixel_resolution, map resolution (pixel)
  - box_factor, is used as re-sampling factor for the FFT2 (int)
- input
  - real_data, if True, input is a SRT data set, otherwise the input is a SRT simulated data set (bool)

  - `input_dir`, directory containing the three input files (data set consisting of three OOF far-field maps)
  - `oof_minus`, $\delta z < 0$ defocused far-field file name (string)
  - `in_focus`, $\delta z = 0$ in focus far-field file name (string)
  - `oof_plus`, $\delta z > 0$ defocused far-field file name (string)

- `output`
  - `output_dir`, directory containing the result files
  - `overwrite_dir`, if `True`, the new output folder `output_dir` overwrites the previous one (bool)
  - `plot_figures`, set `True` to save output plots (bool)

- `info`
  - `author`, the user name (string)
  - `label`, data set label used in plots (string)
  - `observation_date`, observation date (string)
  - `comment`, a note describing the run (string)

The two files,

`sample_input_files/run_config.yaml` and
`sample_input_files/synthetic_run_config.yaml`

available in the SRT `pyoof` suite, are a good hint to make your own parameter file. Here below an example:

```
params:
  radiotelescope: !!str SRT
  radius: !!float 32.004
  focus_primary_reflector: !!float 21.0236
  total_focus: !!float 149.76
  frequency: !!float 22.23E+9
  delta_z: !!float 0.027
  residual_opd: !!bool True
fit:
  optimization_variables: !!str /home/franco/oac/pyoof_data/srt_data/opt_vars.yaml
  optimization_method: !!str trf
  fit_previous: !!bool True
  max_order: !!int 5
  pixel_resolution: !!int 256
  box_factor: !!int 5
input:
  real_data: !!bool True
  input_dir: !!str /home/franco/oac/pyoof_data/srt_data
  oof_minus: !!str 20190426-110938-S0000-MAPPA_OUT1.txt
  in_focus: !!str 20190426-101052-S0000-MAPPA_IN.txt
  oof_plus: !!str 20190426-112403-S0000-MAPPA_OUT2.txt
output:
  output_dir: !!str /home/franco/oac/pyoof_data/srt_output/run_20190426
  overwrite_dir: !!bool True
  plot_figures: !!bool True
info:
  author: !!str Franco Buffa
  label: !!str run_20190426
  observation_date: !!str 2019-04-26
  comment: !!str test
```

4.1 Fitting variables

The variables of the optimization problem are included into the .yaml file and specified by optimization_variables (see fit parameter group). As an example, see sample_input_files/opt_vars.yaml. The parameters belong to the following five groups:

- params_bound_max, specifying the variable upper limit (not used by lm, Levenberg-Marquardt method)
- params_bound_min, specifying the variable lower limit (not used by lm, Levenberg-Marquardt method)
- params_excluded, the index of the variables the fitting algorithm does not execute
- params_fixed, variables with a assigned valued and, thus, left out of the fitting algorithm
- params_init, the starting values for the variables of the fitting algorithm

Each group consists of a list of values, corresponding to the variables to be optimized, a part of those variables the user prefers to leave out of the fitting algorithm. Those variables and the corresponding indexes are:

0. i_amp
1. c_dB or sigma_dB
2. x0
3. y0
4. K(0,0)
5. K(1,-1)
6. K(1,1)
7. K(2,-2)
8. ...

The first four variables are related to the feed illumination, i.e., how the feed illuminates the subreflector in term of the taper amplitude and angle and to the offset coordinates x0 and y0. The remaining (n+1)*(n+2)/2 variables are the Zernike coefficients, where n is the Zernike polynomial maximum order (see max_order in the fitting parameter description).

4.2 SRT data format

The telescope data (real_data = True) must be recorded in three input files (see oof_minus, in_focus and oof_plus). Each file contains three columns u, v and the power of measured signal which are indexed to create a m*m matrix:

```
u_1   v_1   P_1,1
u_1   v_2   P_1,2
u_1   v_3   P_1,3
...
u_m   v_m-2 P_m,m-2
u_m   v_m-1 P_m,m-1
u_m   v_m   P_m,m
```

An example of the SRT data format is available in the sample_input_files/real_data folder.

4.3 GRASP data format

The SRT `pyoof` version accepts as input the GRASP-TICRA simulated far-fields (`real_data = False`). The data format is:

- the first five rows contain:
  - software version
  - header
  - source field name
  - frequency label
  - frequency (GHz)
- the sixth row is a separator (`++++`)
- the rows from the seventh to the ninth are not used
- the tenth row contains the limits of the `u-v` map
- the eleventh row specifies the `n_u` and `n_v` dimensions of the `u-v` map
- The last rows contains the real and imaginary parts of the: co-polar (columns #1 and #2) and cross-polar (columns #3 and #4) component of the far-field radiation pattern.

In the `sample_input_files/synthetic_data` folder three input files for `pyoof` (`oof_minus`, `in_focus` and `oof_plus`) in the GRASP data format are availlable. Here below an example:

```
VERSION: TICRA-EM-FIELD-V0.1
HEADER: Field data in grid
SOURCE_FIELD_NAME: PO_pry
FREQUENCY_NAME: Freq
FREQUENCY: 26.0000000000000 GHz,
++++
   1
   1   3   2   1
   0   0
 -0.1407433000E-02 -0.1407433000E-02 0.1407433000E-02 0.1407433000E-02
 101 101   0
  0.9831180383E+02 -0.4940033911E+02 0.2608371766E+01 0.1359483561E+03
  0.6469147786E+02 -0.8420159469E+02 -0.1094768568E+02 0.1610393322E+03
  0.3732886444E+02 -0.9947082980E+02 -0.2765264654E+02 0.1757128440E+03
  0.1803181342E+02 -0.9645946364E+02 -0.4563895629E+02 0.1784351487E+03
  0.7576312405E+01 -0.7785241600E+02 -0.6312211725E+02 0.1688318711E+03
  0.5727063861E+01 -0.4726605123E+02 -0.7858145489E+02 0.1477138354E+03
  0.1131813515E+02 -0.8723677397E+01 -0.9089037964E+02 0.1169699899E+03
  0.2237939198E+02 0.3381507043E+02 -0.9938599100E+02 0.7934413693E+02
  ...
```

5 Executing pyoof

The user can easily run `pyoof` by means of the script `run_pyoof.py` needing the argument `run_config.yaml` only. This argument is a file containing (see Section 4):

- the antenna geometry
- the optimization method
- the input file names

- the output setups
- some ancillary information needed for running the code

So, the user can type the following command to run `pyoof`:

```
python run_pyoof.py run_config.yaml
```

The reader can read the structure of the `run_pyoof.py` here below:

```python
#!/usr/bin/env python

import sys
import pyoof
from pyoof import aperture, telgeometry

def main():

    # Read configuration file
    config_file = sys.argv[1]
    config = pyoof.get_run_config(config_file)

    # Initialize output directory
    pyoof.init_output_dir(config)

    # Initialize logger
    logger = pyoof.init_logger(config)
    logger.info('Starting "pyoof for SRT"...')

    # Read data from input files
    metadata, observation_data = pyoof.extract_data_srt(config, logger)

    # Telescope definition
    telescope = [telgeometry.block_srt_wo_legs,  # Blockage distribution
                 telgeometry.opd_srt,             # OPD function
                 config['params']['radius'],      # Primary dish radius
                 config['params']['radiotelescope']] # Telescope name

    # Aperture function
    aperture_function = aperture.illum_gauss

    # Fit beam
    pyoof.fit_beam(data_info=metadata,
                   data_obs=observation_data,
                   method=config['fit']['optimization_method'],
                   order_max=config['fit']['max_order'],
                   illum_func=aperture_function,
                   telescope=telescope,
                   resolution=config['fit']['pixel_resolution'],
                   box_factor=config['fit']['box_factor'],
                   fit_previous=config['fit']['fit_previous'],
                   make_plots=config['output']['plot_figures'],
                   config_params_file=config['fit']['optimization_variables'],
                   config=config,
                   logger=logger)

if __name__ == '__main__':

    main()
```

6 pyoof outputs

In this section the results of the processing of an OOF dataset measured at SRT in April 2019 are reported. The SRT was set to observe the radio source W3(OH), a water maser emitting a narrow band (about 500 KHz) signal around 22.23 GHz, by means of the K-band FPA receiver. In this experiment only the central feed of the FPA was used. A telescope observing schedule based on a standard on-the-fly azimuth scan was implemented to acquire three OOF maps, each one 0.2°*0.2°extent (about 49 scans) and $\sim$ 15 minutes long (about 18 s per scan). Two of the three OOF maps were acquired by setting, in the telescope schedule, an on-axis displacement δz equal to -2λ and 2λ, with $\lambda = 13.48$ mm.

The frequency of the receiver local oscillator was chosen equal to 21588 MHz in order to frequency down-convert the signal spectrum in a suitable position of the SRT intermediate frequency (IF) base band (100-2100 MHz). During each map acquisition, a 1250 MHz-band of the IF signal was digitalized and measured every 30 ms by the Sardara backend [6], setting the maximum number of the frequency channels (16384) with a resulting frequency resolution equal to 76 kHz.

The experiment was performed at 65°elevation, keeping the main reflector AS parked. It means that it was not correcting for gravitational deformations, as our goal was to characterize such effects. For this reason the in-focus far-field maps (see Figure 1) appear to be affected by coma and other aberrations. The parameters of the measurement session are summarized here following:

– `source`: W3(OH)

– `active surface status`: parked

– `subreflector status`: ON

– `frequency`: 22.23 GHz (only the feed #0 was considered)

– `Half Power Beam Width`: 0.0147°(SRT beam size @ 22.23 GHz)

– `elevation`: $\sim 65°$

– `scanning method`: 49 OTF azimuthal scans ($\Delta(elevation) \simeq 0.0042°$, each map took $\sim$ 15 minutes)

– `map size`: 0.2°*0.2°($\sim$ 14 beams)

– `map #1`: 20190426-101052-S0000-MAPPA_IN.txt ($\delta z = 0$)

– `map #2`: 20190426-110938-S0000-MAPPA_OUT1.txt ($\delta z = -2\lambda$)

– `map #3`: 20190426-112403-S0000-MAPPA_OUT2.txt ($\delta z = +2\lambda$)

– `backend`: Sardara (SArdinia Roach2-based Digital Architecture for Radio Astronomy), channel resolution 76 kHz, selected channels: 7096, 7097 for LHP and RHP (FITS indexes: 7096, 7097, 23480 and 23481)

– `integration time`: 30 ms

– `SNR`: 21 dB

The FITS files generated by the backend where preliminarily processed and transformed in the `u-v` plane. After the `pyoof` run, the output files are saved in the folder `output_dir` and described here below. The first one is a `.log` file containing all the code messages and outcomes (see Subsection 6.1). Then, a `.fits` file, containing the three measured far-field maps interpolated in the `u-v` plane, is created. This preliminary step is needed for the fitting algorithm. During the run, the Zernike order `k` is progressively increased until `n`, the maximum order defined by `max_order`. At the end of each step the following files are recorded:

- `pyoof_info.yml`, summary of the run
- `u_data.csv`
- `v_data.csv`

- beam_data.csv

At each optimization step and for each Zernike order, the following files are recorded:

- res_nk.csv, fitting residuals
- jac_nk.csv, Jacobian matrix
- grad_nk.csv, gradient
- phase_nk.csv, aperture plane phase (rad)
- error_nk.csv, antenna deformations map (mm)
- cov_nk.csv, variance-covariance matrix
- corr_nk.csv, correlation matrix
- fitpar_nk.csv, the optimized parameters (compared with the k-1 results)

The following plots (pdf files) are generated, for each order, at the end of each optimization step (plots folder):

- fitbeam_nk.pdf
- fitphase_nk.pdf
- error_map_nk.pdf
- residual_nk.pdf
- cov_nk.pdf
- corr_nk.pdf

In addition, pyoof creates the file obsbeam.pdf, representing the three (one in focus and two out-of-focus) measured radiation patterns, and the fitbeam_nk.pdf file containing, for each k, the three radiation patterns resulting from the fitting procedure (see Subsection 6.2).

6.1 .log file example

```
2019-07-03 19:28:51,417 : INFO : Starting "pyoof for SRT"...
2019-07-03 19:28:51,417 : INFO : Reading data from file "/home/oac/pyoof_data/srt_data/20190426-
                                 110938-S0000-MAPPA_OUT1.txt"...
2019-07-03 19:28:51,466 : INFO : Reading data from file "/home/oac/pyoof_data/srt_data/20190426-
                                 101052-S0000-MAPPA_IN.txt"...
2019-07-03 19:28:51,538 : INFO : Reading data from file "/home/oac/pyoof_data/srt_data/20190426-
                                 112403-S0000-MAPPA_OUT2.txt"...
2019-07-03 19:28:51,596 : INFO : Writing data to
                                 /home/oac/pyoof_data/srt_output/run_20190426/run_20190426.fits...
2019-07-03 19:28:51,616 : INFO : Done!
2019-07-03 19:28:51,629 : INFO : Reading data and configuration parameters...
2019-07-03 19:28:51,633 : INFO : Done!
2019-07-03 19:28:51,633 : DEBUG : Maximum order to be fitted: 5
2019-07-03 19:28:51,633 : DEBUG : Telescope name: SRT
2019-07-03 19:28:51,633 : DEBUG : Description label: run_20190426
2019-07-03 19:28:51,633 : DEBUG : Frequency: 22230000000.0 Hz
2019-07-03 19:28:51,633 : DEBUG : Wavelength: 0.0135 m
2019-07-03 19:28:51,634 : DEBUG : d_z (out-of-focus): [-0.027 0. 0.027] m
2019-07-03 19:28:51,634 : DEBUG : Illumination to be fitted: gauss
2019-07-03 19:28:51,634 : INFO : Precomputing optical path difference...
```

```
2019-07-03 19:28:56,884 : INFO : Done!
2019-07-03 19:28:56,884 : INFO : ---------------------------------------------------
2019-07-03 19:28:56,884 : INFO : Fitting power pattern for order 1...
2019-07-03 19:28:56,884 : INFO : Initial params: default
2019-07-03 19:28:56,885 : INFO : Number of parameters to fit: 3
2019-07-03 19:28:56,885 : INFO : Starting optimization...
2019-07-03 19:28:56,885 : INFO : ---------------------------------------------------
2019-07-03 19:29:00,065 : INFO : ---------------------------------------------------
2019-07-03 19:29:00,066 : INFO : Optimization done!
2019-07-03 19:29:00,071 : INFO : ---------------------------------------------------
2019-07-03 19:29:00,074 : INFO : Parameter   Initial value      Fitted value
2019-07-03 19:29:00,074 : INFO : i_amp      0.100000000000    1.000000000000
2019-07-03 19:29:00,074 : INFO : sigma_dB -15.000000000000  -12.891700060557
2019-07-03 19:29:00,074 : INFO : x_0        0.000000000000    0.000000000000
2019-07-03 19:29:00,074 : INFO : y_0        0.000000000000    0.000000000000
2019-07-03 19:29:00,074 : INFO : K(0, 0)    0.000000000000    0.000000000000
2019-07-03 19:29:00,074 : INFO : K(1, -1)   0.100000000000    0.922999569541
2019-07-03 19:29:00,074 : INFO : K(1, 1)    0.100000000000    0.095877194462
2019-07-03 19:29:00,074 : INFO : ---------------------------------------------------
2019-07-03 19:29:00,236 : INFO : Saving data...
2019-07-03 19:29:01,261 : INFO : Done!
2019-07-03 19:29:01,261 : INFO : Making plots...
2019-07-03 19:29:10,458 : INFO : Done!
2019-07-03 19:29:10,458 : INFO : ---------------------------------------------------
2019-07-03 19:29:10,458 : INFO : Fitting power pattern for order 2...
2019-07-03 19:29:10,460 : INFO : Initial params: n=1 fit
2019-07-03 19:29:10,461 : INFO : Number of parameters to fit: 6
2019-07-03 19:29:10,461 : INFO : Starting optimization...
2019-07-03 19:29:10,461 : INFO : ---------------------------------------------------
2019-07-03 19:29:14,857 : INFO : ---------------------------------------------------
2019-07-03 19:29:14,857 : INFO : Optimization done!
2019-07-03 19:29:14,860 : INFO : ---------------------------------------------------
2019-07-03 19:29:14,861 : INFO : Parameter   Initial value      Fitted value
2019-07-03 19:29:14,862 : INFO : i_amp      1.000000000000    1.000000000000
2019-07-03 19:29:14,862 : INFO : sigma_dB -12.891700060557  -12.734169372400
2019-07-03 19:29:14,862 : INFO : x_0        0.000000000000    0.000000000000
2019-07-03 19:29:14,862 : INFO : y_0        0.000000000000    0.000000000000
2019-07-03 19:29:14,862 : INFO : K(0, 0)    0.000000000000    0.000000000000
2019-07-03 19:29:14,862 : INFO : K(1, -1)   0.922999569541    0.910474820243
2019-07-03 19:29:14,862 : INFO : K(1, 1)    0.095877194462    0.095188193908
2019-07-03 19:29:14,862 : INFO : K(2, -2)   0.100000000000   -0.033010939019
2019-07-03 19:29:14,862 : INFO : K(2, 0)    0.100000000000   -0.346763748595
2019-07-03 19:29:14,862 : INFO : K(2, 2)    0.100000000000    0.143045326587
2019-07-03 19:29:14,862 : INFO : ---------------------------------------------------
2019-07-03 19:29:15,066 : INFO : Saving data...
2019-07-03 19:29:16,853 : INFO : Done!
2019-07-03 19:29:16,853 : INFO : Making plots...
2019-07-03 19:29:22,075 : INFO : Done!
2019-07-03 19:29:22,076 : INFO : ---------------------------------------------------
2019-07-03 19:29:22,076 : INFO : Fitting power pattern for order 3...
2019-07-03 19:29:22,078 : INFO : Initial params: n=2 fit
2019-07-03 19:29:22,078 : INFO : Number of parameters to fit: 10
2019-07-03 19:29:22,078 : INFO : Starting optimization...
2019-07-03 19:29:22,078 : INFO : ---------------------------------------------------
2019-07-03 19:29:45,450 : INFO : ---------------------------------------------------
2019-07-03 19:29:45,451 : INFO : Optimization done!
2019-07-03 19:29:45,457 : INFO : ---------------------------------------------------
2019-07-03 19:29:45,459 : INFO : Parameter   Initial value      Fitted value
2019-07-03 19:29:45,459 : INFO : i_amp      1.000000000000    1.000000000000
```

```
2019-07-03 19:29:45,459 : INFO : sigma_dB -12.734169372400   -8.000000000000
2019-07-03 19:29:45,459 : INFO : x_0        0.000000000000    0.000000000000
2019-07-03 19:29:45,459 : INFO : y_0        0.000000000000    0.000000000000
2019-07-03 19:29:45,459 : INFO : K(0, 0)    0.000000000000    0.000000000000
2019-07-03 19:29:45,459 : INFO : K(1, -1)   0.910474820243   -0.147880231388
2019-07-03 19:29:45,459 : INFO : K(1, 1)    0.095188193908    0.059931299004
2019-07-03 19:29:45,459 : INFO : K(2, -2)  -0.033010939019   -0.053736371116
2019-07-03 19:29:45,459 : INFO : K(2, 0)   -0.346763748595   -0.273329817694
2019-07-03 19:29:45,460 : INFO : K(2, 2)    0.143045326587    0.007743326417
2019-07-03 19:29:45,460 : INFO : K(3, -3)   0.100000000000   -0.269432566742
2019-07-03 19:29:45,460 : INFO : K(3, -1)   0.100000000000   -0.544374590823
2019-07-03 19:29:45,460 : INFO : K(3, 1)    0.100000000000   -0.025120736826
2019-07-03 19:29:45,460 : INFO : K(3, 3)    0.100000000000    0.023723012099
2019-07-03 19:29:45,460 : INFO : --------------------------------------------------
2019-07-03 19:29:46,025 : INFO : Saving data...
2019-07-03 19:29:47,890 : INFO : Done!
2019-07-03 19:29:47,890 : INFO : Making plots...
2019-07-03 19:29:52,976 : INFO : Done!
2019-07-03 19:29:52,976 : INFO : --------------------------------------------------
2019-07-03 19:29:52,976 : INFO : Fitting power pattern for order 4...
2019-07-03 19:29:52,978 : INFO : Initial params: n=3 fit
2019-07-03 19:29:52,978 : INFO : Number of parameters to fit: 15
2019-07-03 19:29:52,979 : INFO : Starting optimization...
2019-07-03 19:29:52,979 : INFO : --------------------------------------------------
2019-07-03 19:30:53,627 : INFO : --------------------------------------------------
2019-07-03 19:30:53,627 : INFO : Optimization done!
2019-07-03 19:30:53,630 : INFO : --------------------------------------------------
2019-07-03 19:30:53,632 : INFO : Parameter   Initial value      Fitted value
2019-07-03 19:30:53,632 : INFO : i_amp      1.000000000000    1.000000000000
2019-07-03 19:30:53,632 : INFO : sigma_dB  -8.000000000000   -8.000000000000
2019-07-03 19:30:53,632 : INFO : x_0        0.000000000000    0.000000000000
2019-07-03 19:30:53,632 : INFO : y_0        0.000000000000    0.000000000000
2019-07-03 19:30:53,632 : INFO : K(0, 0)    0.000000000000    0.000000000000
2019-07-03 19:30:53,633 : INFO : K(1, -1)  -0.147880231388   -0.094318101151
2019-07-03 19:30:53,633 : INFO : K(1, 1)    0.059931299004    0.075395889058
2019-07-03 19:30:53,633 : INFO : K(2, -2)  -0.053736371116   -0.019151246487
2019-07-03 19:30:53,633 : INFO : K(2, 0)   -0.273329817694   -0.221473222855
2019-07-03 19:30:53,633 : INFO : K(2, 2)    0.007743326417   -0.164260111193
2019-07-03 19:30:53,633 : INFO : K(3, -3)  -0.269432566742   -0.313137504177
2019-07-03 19:30:53,633 : INFO : K(3, -1)  -0.544374590823   -0.457379741463
2019-07-03 19:30:53,633 : INFO : K(3, 1)   -0.025120736826   -0.017064859998
2019-07-03 19:30:53,633 : INFO : K(3, 3)    0.023723012099    0.001203225552
2019-07-03 19:30:53,634 : INFO : K(4, -4)   0.100000000000    0.031697794331
2019-07-03 19:30:53,634 : INFO : K(4, -2)   0.100000000000    0.019946487372
2019-07-03 19:30:53,634 : INFO : K(4, 0)    0.100000000000    0.025817184413
2019-07-03 19:30:53,634 : INFO : K(4, 2)    0.100000000000   -0.119010214791
2019-07-03 19:30:53,634 : INFO : K(4, 4)    0.100000000000    0.003815338995
2019-07-03 19:30:53,634 : INFO : --------------------------------------------------
2019-07-03 19:30:54,583 : INFO : Saving data...
2019-07-03 19:30:56,578 : INFO : Done!
2019-07-03 19:30:56,578 : INFO : Making plots...
2019-07-03 19:31:02,926 : INFO : Done!
2019-07-03 19:31:02,926 : INFO : --------------------------------------------------
2019-07-03 19:31:02,926 : INFO : Fitting power pattern for order 5...
2019-07-03 19:31:02,928 : INFO : Initial params: n=4 fit
2019-07-03 19:31:02,929 : INFO : Number of parameters to fit: 21
2019-07-03 19:31:02,929 : INFO : Starting optimization...
2019-07-03 19:31:02,929 : INFO : --------------------------------------------------
2019-07-03 19:34:19,156 : INFO : --------------------------------------------------
```

```
2019-07-03 19:34:19,156 : INFO : Optimization done!
2019-07-03 19:34:19,159 : INFO : ----------------------------------------------------
2019-07-03 19:34:19,161 : INFO : Parameter   Initial value      Fitted value
2019-07-03 19:34:19,162 : INFO : i_amp      1.000000000000    1.000000000000
2019-07-03 19:34:19,162 : INFO : sigma_dB  -8.000000000000   -8.000000000000
2019-07-03 19:34:19,162 : INFO : x_0        0.000000000000    0.000000000000
2019-07-03 19:34:19,162 : INFO : y_0        0.000000000000    0.000000000000
2019-07-03 19:34:19,162 : INFO : K(0, 0)    0.000000000000    0.000000000000
2019-07-03 19:34:19,162 : INFO : K(1, -1)  -0.094318101151    0.033278121701
2019-07-03 19:34:19,162 : INFO : K(1, 1)    0.075395889058    0.078214901187
2019-07-03 19:34:19,162 : INFO : K(2, -2)  -0.019151246487   -0.023313693326
2019-07-03 19:34:19,162 : INFO : K(2, 0)   -0.221473222855   -0.160415295253
2019-07-03 19:34:19,162 : INFO : K(2, 2)   -0.164260111193   -0.031109303254
2019-07-03 19:34:19,162 : INFO : K(3, -3)  -0.313137504177   -0.147363356332
2019-07-03 19:34:19,163 : INFO : K(3, -1)  -0.457379741463   -0.251348670793
2019-07-03 19:34:19,163 : INFO : K(3, 1)   -0.017064859998   -0.015447174484
2019-07-03 19:34:19,163 : INFO : K(3, 3)    0.001203225552    0.022017642643
2019-07-03 19:34:19,163 : INFO : K(4, -4)   0.031697794331    0.020252888888
2019-07-03 19:34:19,163 : INFO : K(4, -2)   0.019946487372    0.014508683366
2019-07-03 19:34:19,163 : INFO : K(4, 0)    0.025817184413    0.088208795895
2019-07-03 19:34:19,163 : INFO : K(4, 2)   -0.119010214791   -0.009767299100
2019-07-03 19:34:19,163 : INFO : K(4, 4)    0.003815338995    0.137765087427
2019-07-03 19:34:19,163 : INFO : K(5, -5)   0.100000000000    0.238588554915
2019-07-03 19:34:19,163 : INFO : K(5, -3)   0.100000000000    0.075955442996
2019-07-03 19:34:19,163 : INFO : K(5, -1)   0.100000000000    0.160119657661
2019-07-03 19:34:19,163 : INFO : K(5, 1)    0.100000000000   -0.003686858775
2019-07-03 19:34:19,164 : INFO : K(5, 3)    0.100000000000   -0.005213239175
2019-07-03 19:34:19,164 : INFO : K(5, 5)    0.100000000000    0.017083694019
2019-07-03 19:34:19,164 : INFO : ----------------------------------------------------
2019-07-03 19:34:21,035 : INFO : Saving data...
2019-07-03 19:34:23,224 : INFO : Done!
2019-07-03 19:34:23,224 : INFO : Making plots...
2019-07-03 19:34:32,246 : INFO : Done!
2019-07-03 19:34:32,246 : INFO : ----------------------------------------------------
2019-07-03 19:34:32,247 : INFO : Power pattern fit completed after 5.68 minutes!
```

6.2 `obsbeam.pdf & fitbeam_n5.pdf`

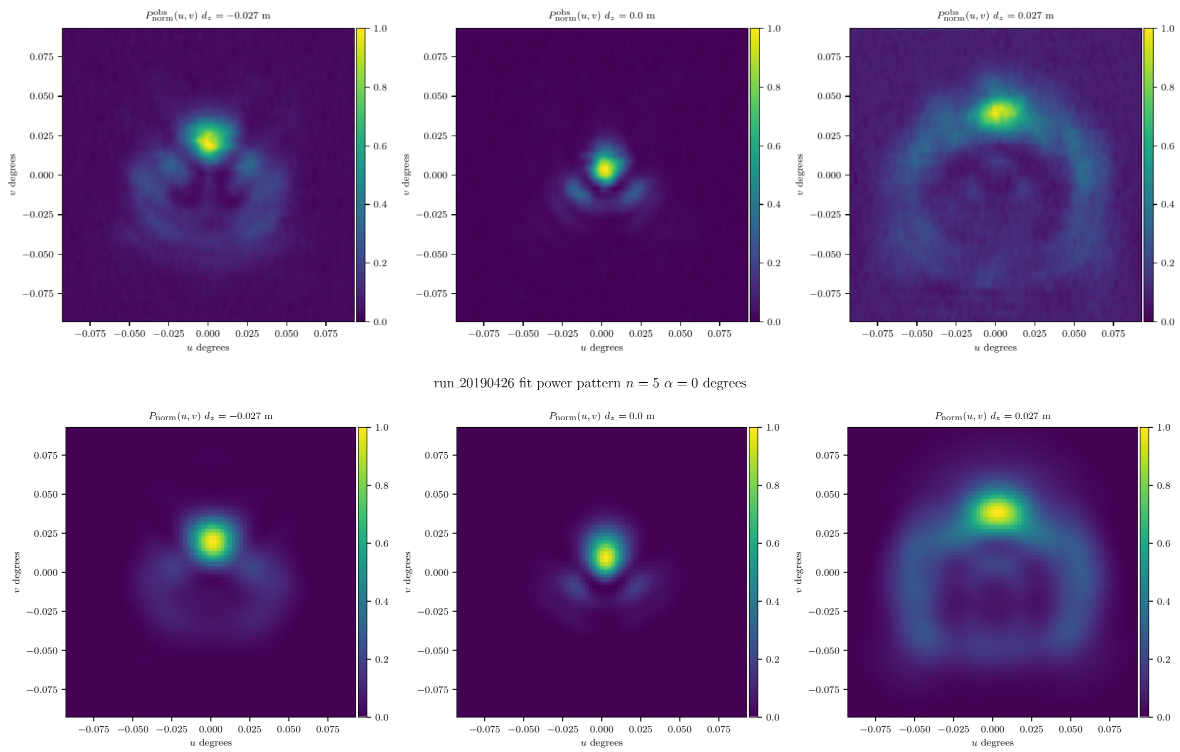

Figure 1: Observed (top) and fitted (bottom) pattern maps, from left to right: $\delta z = -2\lambda$, $\delta z = 0$ and $\delta z = +2\lambda$ (n=5).

Three SRT OOF far-field pattern maps measured in April 2019 and saved in the obsbeam.pdf file are here compared to the related pattern maps resulting from the `tlr` fitting method stopped at the order `n`=5, see respectively first and second row in Figure 1. It is worth to noting that a good agreement (see Subsection 6.3) turns out to be between the measured and fitted maps (compare columns in Figure 1), especially, in depicting the in-focus pattern aberrations (mainly coma). Such aberrations are due to the fact that the AS was in parking mode. Of course, a slight misalignment between the axes of the main and the secondary reflectors may cause, in theory, further aberrations.

Another consequence of the AS parking is the poor level (about 21 dB) of the signal to noise ratio (SNR) reached during the observation. However, this is not a critical issue for this preliminary OOF experiment at SRT. In fact, this experiment was thought to allow a comparison between the deformation map resulting from the OOF measurements and the deformation map deriving from a photogrammetry campaign performed in 2012 (see Subsection 6.5). Of course, for the next experiments requiring a greater SNR (better than 30 dB), different approaches are recommended such as observing astronomical calibration sources and increasing the integration time. Further discussions about the optimization of the measurement set-up goes over the purposes of this internal report. Therefore, they will be faced in future works.

6.3 `residual_n5.pdf`

The three residual maps resulting from the OOF fitting stopped at the order `n`=5 are shown in Figure 2. The low value of the residuals shown in the maps gives a measure of the good agreement between the observed and the fitted map.

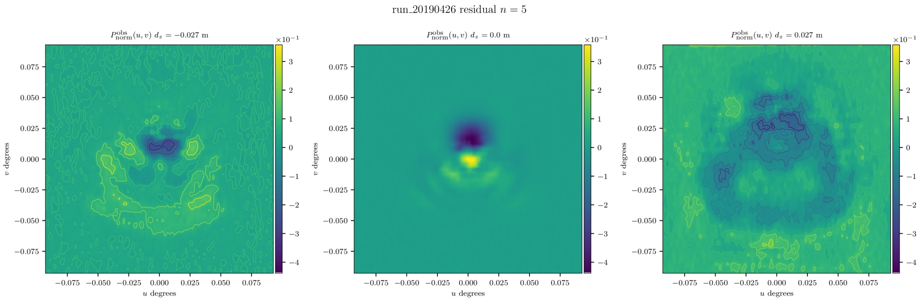

Figure 2: Fitting residuals in the u-v map, from left to right: $\delta z = -2\lambda$, $\delta z = 0$ and $\delta z = +2\lambda$ (n=5).

6.4 fitphase_n5.pdf

Figure 3 shows the map of the aperture field phase calculated by the OOF algorithm. It depicts the large scale deformations due mainly to the optical misalignment and the surface deformations of the main and secondary reflector, with respect to an ideal constant phase plane on the telescope aperture.

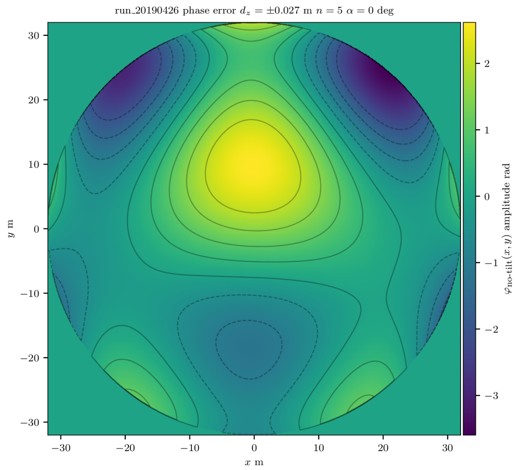

Figure 3: Phase map [rad] at elevation equal to 65°(n=5).

6.5 error_map_n5.pdf

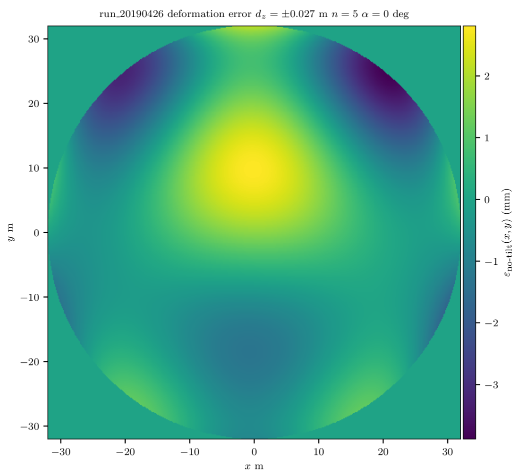

Figure 4: OOF deformation map [mm] at elevation equal to 65°(n=5).

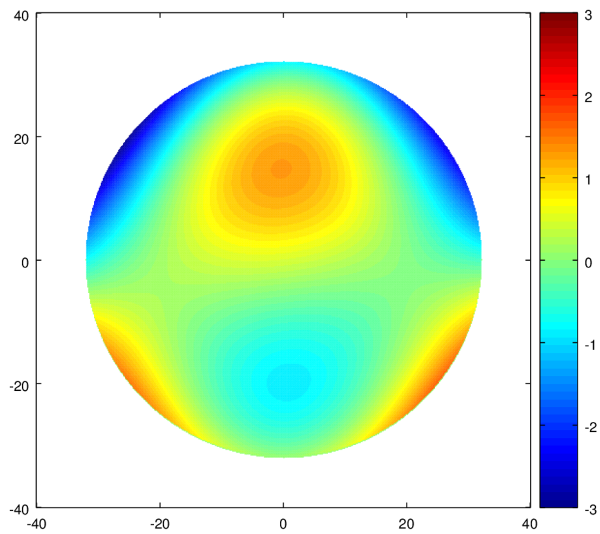

Figure 5: Close range photogrammetric map [mm] at elevation equal to 60°. A smoothing with a high order Zernike polynomial fitting was applied.

In this subsection a comparison between the map of the surface deformations measured at 65°by OOF method (Figure 4) and the map of the surface deformations measured by the close range photogrammetry [7] at 60°(Figure 5) is shown. In both cases the SRT AS was set in parking mode. Such a comparison has to take into account the intrinsic differences between the two methods. First of all, the OOF measured the sum of the deformations and the misalignment of the SRT main and secondary reflector surfaces. Instead, the photogrammetry measured only the SRT main reflector surface deformations. OOF is an inverse method, whose SR is relatively poor (in our case we estimate SR $\simeq$ 14 m); conversely, the photogrammetry is a direct method whose accuracy is imposed by the camera *internal* and *external* parameters and whose SR depends, ultimately, by the target arrangement (SR $\simeq$ 1.7 m in the 2012 measurement). That said, the two maps, taken at close elevation angles, look pretty similar. The in-focus OOF map shows similar large scale surface deformations, having almost the same amount and position, thus consistent with the action of gravitational loads on the SRT main reflector.

7 Summary and conclusions

In this document we have described how we have changed `pyoof` with the aim of processing simulated and measured SRT OOF datasets.

Moreover, we have analyzed a SRT OOF dataset measured in April 2019, discussing the results and comparing them with those we got by means of close range photogrammetry in 2012. This comparison has shown that the version of `pyoof` adapted to the SRT case produces results consistent with the "real" gravitational large scale deformations, in spite of the poor SNR dataset we measured pointing a water maser.

New experiments will be soon scheduled at the SRT addressed to improve the measure SNR, chosing, for instance, an astronomical calibrator emitting a broad band signal and increasing the integration time.

Finally, we hope to be able soon to test `pyoof` with a full multi-beam dataset. A multi beam observation would make the OOF maps acquisition significantly faster and would allow to measure even the large scale deformations due to thermal gradient and correct for them within a reasonable time before a scientific observation.

A Appendix

A.1 matplotlib

While `pyoof` is generating the plots, `matplotlib`, fails in finding the `serif` font-set specified by `pyoof/data/pyoof.mplstyle`:

```
~/anaconda3/lib/python3.7/site-packages/matplotlib/font_manager.py:1241:
UserWarning: findfont: Font family ['serif'] not found. Falling back to DejaVu Sans.
(prop.get_family(), self.defaultFamily[fontext]))
```

The warning may be ignored as the `serif` is automatically replaced by the `DejaVu Sans` font-set.

References

[1] Buffa F., Serra G. and Poppi S., *HoP User Guide*, OAC - Internal Report, N. 71, 2018.

[2] Nikolic B., Hills R. E. and Richer J. S., *Measurement of antenna surface from in- and out-of-focus beam maps using astronomical sources*, A&A, 465, 679-683, 2007.

[3] Nikolic B., Prestage R. M., Balser D. S., Chandler C. J. and Hills R. E., *Out-of-focus holography at the Green Bank Telescope*, A&A, 465, 679-683, 2007.

[4] Bach U., *Out of focus holography at Effelsberg*, 12th European VLBI Network Symposium and Users Meeting, 7-10 October 2014, Cagliari, Italy.

[5] Jian D., Weiye Z., Jinqing W., Qinghui L. and Zhiqiang S., *Correcting Gravitational Deformation at the Tianma Radio Telescope*, IEEE Transactions on Antennas and Propagation, 66, 4, 2018.

[6] Melis A., Concu R., Trois A., Possenti A. et al., *SArdinia Roach2-based Digital Architecture for Radio Astronomy (SARDARA)*, Journal of Astronomical Instrumentation, 07, 01, 2018.

[7] Süss M., Koch D. and Paluszek H., *The Sardinia Radio Telescope (SRT) optical alignment*, Proc. SPIE 8444, Ground-based and Airborne Telescopes IV, 84442G, 2012.