\*\*Introduction\*\*

SenAOReFoc is a complete closed-loop sensorbased adaptive optics (AO) and remote focusing control software that works with a deformable mirror (DM) and a Shack-Hartmann wavefront sensor (SHWS). When a large stroke DM is used, such as an Alpao-69 or Alpao-97, both the closed-loop AO correction and remote focusing functionalities can be exploited. If the DM exhibits insufficient stroke for remote focusing, the remote focusing unit can be ignored, leaving the closed-loop AO correction functionalities fully functional without additional modifications to the software. The graphic user interface (GUI) exhibits modular widget arrangements, and interactive commands are displayed in the message box for user guidance.

SenAOReFoc is developed under PySide2 with the GUI built using Qt Designer. Communication of events during processes are then handled using Qt's signals and slots mechanism. Common events between different threads, such as displaying an image or error message, reuse the same slots.

Closed-loop AO correction can be performed using both the zonal method, which updates DM control voltages in terms of the raw slope values; and the modal method, which updates DM control voltages in terms of orthogonal Zernike polynomials. There are four sub-modes tagged to each of the two methods: 1) standard closed-loop AO correction; 2) closed-loop AO correction with consideration of obscured search blocks; 3) closed-loop AO correction ignoring defocus; 4) closed-loop AO correction with consideration of obscured search blocks and ignoring defocus.

Remote focusing can be performed by scanning the focus axially with a pre-determined axial range, step increment and step number, or by manually adjusting a toggle bar on the GUI for random access remote focusing. The former also incorporates options of whether or not to perform closed-loop AO correction at each remote focusing depth.

The software has been validated on a reflectance confocal microscope. However, it should also be fully compatible with fluorescence microscope systems designed in a closed-loop configuration.

\*\*Statement of need\*\*

The performance of optical microscopes degrades significantly in the presence of optical aberrations that develop due to misalignment of the optical system and inhomogeneity in refractive index of the imaging sample. This results in compromised image resolution and contrast. Adaptive optics (AO) is a powerful technique that corrects for optical aberrations using reconfigurable adaptive devices, such as deformable mirrors (DM) and spatial light modulators (SLM), to restore the image quality [1].

Remote focusing is another technique recently introduced for axial refocusing while avoiding slow movements of the sample stage and objective lens during real-time applications and simultaneously compensating for system aberrations introduced by beam divergence at different depths [2].

Both AO and remote focusing are widely used in the microscopy community. However, despite software being openly available for AO modelling and analysis [3], as well as for general implementation of AO [4], there is not an existing open-source software that combines both AO and remote focusing within the single package. As a result, time and effort has to be spent reimplementing existing techniques for different systems.

SenAOReFoc aims to fill this gap and has been designed for open development to work with different hardware devices, which would greatly benefit the microscopy community. Closed-loop sensorbased AO solutions, which enable fast AO correction, are provided with multiple sub-modes to address various practical requirements. Remote focusing can also be performed under different sub-modes for additional flexibility.

\*\*Scripts included in this folder and general descriptions\*\*

* app.py - Sets up GUI and event handlers, displays the GUI, instantiates devices such as DM and SHWS
* AO\_slopes.py - Performs closed-loop AO correction via direct slopes (zonal correction)
* AO\_zernikes.py - Performs closed-loop AO correction via Zernikes (modal correction)
* calibration\_RF.py - Performs remote focusing calibration using closed-loop AO correction
* calibration\_zern.py - Generates DM control matrix
* calibration.py - Performs DM calibration
* centroid\_acquisition.py - Methods for centroiding of SH spots
* centroiding.py - Performs system aberration calibration
* config.py - Sets configuration file path
* config.yaml - Configuration file incorporating all system and static parameters **(please change and add as needed according to the specific system)**
* conversion.py - Performs slope - Zernike conversion
* data\_collection.py - Performs automated characterisation of microscope AO performance
* HDF5\_dset.py - Methods for generating and appending data to HDF5 datasets
* image\_acquisition.py - Methods for image acquisition
* log.py - Method to set up event logs
* mirror.py - Performs DM initialisation
* SB\_geometry.py - Performs initialisation of search block geometry
* SB\_position.py - Performs repositioning of search blocks
* sensor.py - Performs SHWS initialisation
* SH\_acquisition.py - Performs SHWS image acquisition
* zernike.py - Generates Zernike polynomials and derivatives

\*\*Installation instructions\*\*

You will need the following open-source software installed to function and modify SenAOReFoc:

* Python 3.6 +: <https://www.python.org/downloads/>
* Git: <https://www.python.org/downloads/>
* A source-code editor (such as Visual Studio Code): <https://code.visualstudio.com/>

You will need open-source HDFView software installed to directly view data stored in the HDF5 file:

* HDFView: <https://portal.hdfgroup.org/display/support/Download+HDFView>

Once these are installed, in the command prompt clone the source-code from the Git repository:

git clone -b cleaned\_v1 <https://github.com/jiahecui/SensorbasedAO.git>

Then set up a virtual environment (optional but recommended):

python –m venv venv

And activate the virtual environment:

venv\Scripts\activate

Then install all package dependencies:

pip install –r requirements.txt

These include:

* numpy
* PySide2==5.13.2
* qimage2ndarray
* Click
* scipy==1.5.4
* h5py==2.9.0
* pyyaml

And install the project in editable mode:

pip install –e .

Finally, SenAOReFoc can be run in **standard mode** in the availability of hardware devices (DM and SHWS:

python sensorbasedAO/app.py

As well as **debug mode** in the absence of hardware devices, which then sets up dummy devices for the purpose of testing software functionality:

python sensorbasedAO/app.py -d

Detailed descriptions of how to perform functionality tests in **debug mode** without connected hardware can be found in the ‘Software functionality tests’ section.

After exiting the software, to also exit the virtual environment, in the command prompt run:

deactivate

\*\*Getting started\*\*

Before running SenAOReFoc on the microscope system in **standard mode** or modifying the software scripts, please first read the ‘**Notes for development**’ section at the bottom of this document and follow instructions accordingly. Then, check the ‘config.yaml’ file which incorporates all system and static parameters required to correctly function the software. Modifications should be made according to the specific system and new parameters can be added as needed. All parameters have been commented. The current example works with an Alpao-69 DM and custom SHWS with a Ximea CMOS camera. In particular, parameters that need to be modified for basic closed-loop AO correction include:

camera:

  SN: "XXXXXXXX"

  exposure: 40000  # us

  frame\_rate: 20.0  # Hz

  sensor\_width: 2048  # pixels Ximea - 2048

  sensor\_height: 2048  # pixels Ximea - 2048

  sensor\_diam: 11.26  # mm Ximea - 11.26

  bin\_factor: 2  # Ximea - 2

  pixel\_size: 5.5  # um Ximea - 5.5

DM:

  SN: "XXXXXXXX"

  exercise: 0  # Flag for whether to exercise DM upon initialisation

  vol\_min: -0.1  # V Negative voltage to apply on DM during calibration

  vol\_max: 0.1  # V Positive voltage to apply on DM during calibration

   vol\_bias: 0  # Neutral voltage of DM actuators

  settling\_time: 0.001  # DM membrane settling time

DM0:

  actuator\_num: 69  # Alpao

  pitch: 1500  # um

  aperture: 10.5  # mm

relay:

  mirror\_odd: 0  # Flag for whether there is an odd number of mirrors in between DM and lenslet

  relay\_odd: 0  # Flag for whether there is an odd number of relays in between DM and lenslet

lenslet:

  lenslet\_pitch: 150  # um

  lenslet\_focal\_length: 5200  # um

search\_block:

  pupil\_diam\_0: 2.2  # mm Diameter of beam incident on SHWS

sys\_calib:

  sys\_calib\_mode: 1  # Mode flag for system aberration calibration method, 0 – load previous system aberration calibration profile, 1 – perform new system aberration correction, 2 – no system aberration correction

AO:

  zern\_gen: 0  # Flag for generating zernike modes on DM in AO\_zernikes.py and AO\_slopes.py, 0 - off, 1 - iterative generation

  loop\_max\_gen: 15  # Maximum number of loops for closed-loop control during generation of zernike modes

  recon\_coeff\_num: 69  # Number of zernike modes to use during wavefront reconstruction

  control\_coeff\_num: 20  # Number of zernike modes to control during AO correction  # 3rd - 14, 4th - 20, 5th - 27, 6th - 35, 7th - 44, 8th - 54, 9th - 65, 10th - 77

Note that the ‘DM exercise’ option may need to be set to 1 for electromagnetic mirrors that observe substantial thermal effects [5], which will start a process of sending random voltages within [-0.5, 0.5] to all DM actuators after pressing [Initialise SB]. After above parameters have been modified, the procedures below can be performed for basic closed-loop AO correction. Note that for a microscope using reflectance contrast, this should be performed while scanning over a small region on scattering samples to avoid specular reflections that lead to the double-pass effect [6], which causes errors to the aberration measurements. For fluorescence microscopes, a fluorescent bead can be used as the sample with a static beam.

**DM calibration:**

On the GUI press [Initialise SB] -> [Position SB] (follow instructions in message box to reposition search block area) -> [Calibrate-S] -> [S-Z Conv] -> [Calibrate-Z]

**System aberration calibration:**

To perform system aberration calibration anew, make sure the corresponding mode flag is left as default in ‘config.yaml’.

sys\_calib:

  sys\_calib\_mode: 1  # Mode flag for system aberration calibration method, 0 – load previous system aberration calibration profile, 1 – perform new system aberration correction, 2 – no system aberration correction

Then on the GUI press [Calibrate-Sys]. This will run closed-loop AO correction (zonal control) until the Strehl ratio is above the given tolerance factor (default Maréchal criterion 0.81), save final DM control voltages (DM system flat file) to the HDF5 file, and halt with the DM system flat file applied such that separate imaging can be performed.

To load a previous DM system flat file, change the corresponding mode flag in ‘config.yaml’.

sys\_calib:

  sys\_calib\_mode: 0  # Mode flag for system aberration calibration method, 0 – load previous system aberration calibration profile, 1 – perform new system aberration correction, 2 – no system aberration correction

Then on the GUI press [Calibrate-Sys]. This will automatically load the DM system flat file from last calibration session and halt with the DM system flat file applied.

To ignore system aberration correction for all subsequent closed-loop AO correction processes, change the corresponding mode flag in ‘config.yaml’.

sys\_calib:

  sys\_calib\_mode: 2  # Mode flag for system aberration calibration method, 0 – load previous system aberration calibration profile, 1 – perform new system aberration correction, 2 – no system aberration correction

Then on the GUI press [Calibrate-Sys]. This will save the current centroid coordinates as the reference centroid coordinates for convergence during closed-loop AO correction.

**Closed-loop AO correction:**

Closed-loop AO correction can be performed without first generating Zernike modes to the DM by setting the corresponding mode flag to its default value in ‘config.yaml’.

AO:

  zern\_gen: 0  # Flag for generating zernike modes on DM in AO\_zernikes.py and AO\_slopes.py, 0 - off, 1 - iterative generation

Then, closed-loop AO correction can be performed using 2 different approaches, each with 4 different sub-modes, as explained in the functionality documentation below.

Alternatively, for characterisation purposes, Zernike modes can be first applied to the DM before performing closed-loop AO correction by changing the corresponding mode flag in ‘config.yaml’.

AO:

  zern\_gen: 1  # Flag for generating zernike modes on DM in AO\_zernikes.py and AO\_slopes.py, 0 - off, 1 - iterative generation

\*\*Functionality documentation\*\*

SenAOReFoc consists of 5 main units, the SHWS initialisation and DM calibration unit, the Zernike aberration input unit, the AO control and data collection unit, the miscellaneous control unit, and the remote focusing unit. There are also 2 auxiliary units, a message box which informs the user of required inputs and the current task progress, and a software termination module.

\*SHWS initialisation and DM calibration unit\*

[Initialise SB] determines the search block (SB) geometry given user informed parameters of the lenslet pitch (the actual diameter of one lenslet), the dimension and pixel size of the camera sensor, as well as the incident pupil diameter. This button also exercises the DM membrane by generating a user specified number of random patterns on the DM.

[Position SB] repositions the SB region on the camera sensor such that it is concentric with the incident beam. This can be achieved in two ways depending on whether or not the DM requires recalibration. User input is needed at this stage by entering 'y' or 'n' on the keyboard. By entering 'y', the user can reposition the SB region by entering '↑', '↓', '←', '→' arrows on the keyboard, which will shift the entire SB region by one pixel at a time and display it at its new location. When the central SH spot(s) is centred within its corresponding SB, by pressing 'Enter' on the keyboard, the user can confirm the new SB position and trigger the process of automatically calculating new SB reference centroid coordinates. Upon entering 'n', SB and DM calibration related parameters from the last session will be automatically loaded from the HDF5 file.

[Calibrate-Sys] performs system aberration calibration in three modes. Mode 0 automatically loads the DM control voltages that correct for system aberrations (DM system flat file) from last calibration session onto the DM. Mode 1 runs closed-loop AO correction (zonal control) until the Strehl ratio is above the given tolerance factor and saves final DM control voltages (DM system flat file) to the HDF5 file. Mode 2 is designed to ignore system aberration correction for small stroke DMs by saving current centroid coordinates as reference for convergence during closed-loop AO correction.

[Calibrate-S] performs DM calibration to retrieve the control matrix (CM) for direct slopes (zonal) control. The positive and negative bias control voltages provided by the user are sequentially applied to each DM actuator to retrieve the 'x' and 'y' slope values, which are then used to calculate the DM influence function (IF) matrix and slope CM by computing its pseudo-inverse.

[S-Z Conv] calculates the conversion matrix between raw slope values and Zernike coefficients. This can be used to calculate the Strehl ratio for evaluation of the image quality during zonal control, or to further acquire the DM IF matrix and CM for modal control.

[Calibrate-Z] retrieves the CM for modal control by using results acquired in [Calibrate-S] and [S-Z Conv]. Before performing pseudo-inverse of the DM IF matrix, the results after singular value decomposition are first evaluated to identify a cut-off value for small singular values. This is usually selected as the value after which a significant drop is seen. System modes below this value are removed during pseudo-inverse for acquisition of a stable Zernike CM.

\*Zernike aberration input unit\*

The Zernike aberration input unit is mainly used for thorough characterisation of the system AO performance by specifying certain amounts of Zernike modes to generate on the DM membrane. Zernike modes are ordered according to the OSA/ANSI standard and piston is removed such that mode 1 starts with tip. [Zernike mode spin box] allows the user to specify a single mode number, the amplitude of which is given in [Zernike value spin box] in the unit of microns. Mode combinations can be specified in [Zernike array edit box] by providing an array of coefficients in the correct mode order. Irrelevant modes can be set as zero and only modes up to the maximum mode number controlled during AO correction (user-defined) can be generated.

\*AO control and data collection unit\*

This is the core software unit responsible for closed-loop AO control and automated data collection. Both Zernike (modal) and direct slopes (zonal) control can be performed in 4 different modes, which will be discussed below.

[Zernike AO 1] performs basic closed-loop modal AO control by updating DM control voltages with the Zernike CM obtained in [Calibrate-Z]. Sub-mode 1 does not generate Zernike aberrations modes before closed-loop AO correction. Sub-mode 2 generates aberrations specified in the Zernike aberration input unit on the DM membrane before performing closed-loop AO correction. The latter sub-mode is especially useful for simple characterisation of the system AO performance.

[Zernike AO 2] is a modified version of [Zernike AO 1] that takes into account missing or ill-formed spots (obscured SBs) detected by the SHWS due to arbitrarily shaped pupils [7-9].

[Zernike AO 3] is a modified version of [Zernike AO 1] that suppresses the correction of Zernike defocus such that refocusing of the focal spot is minimised.

[Slope AO 1] performs basic closed-loop zonal AO control by updating DM control voltages with the direct slope CM obtained in [Calibrate-S]. The 2 sub-modes described in [Zernike AO 1] also apply.

[Slope AO 2] is a modified version of [Slope AO 1] fulfilling the same purpose as [Zernike AO 2].

[Slope AO 3] is a modified version of [Slope AO 1] fulfilling the same purpose as [Zernike AO 3].

[Zernike Full] and [Slope Full] perform closed-loop AO control taking into account both obscured SBs and suppression of Zernike defocus at the same time.

[Data Collection] is designed to perform automated AO performance characterisations of the microscope system, though any data collection process that would benefit from automation can be written beneath the hood. Each operation mode is identified by a unique mode flag in 'config.yaml', which should be determined before software initialisation. Example execution of automated AO performance characterisations on a real microscope system can be found in the ‘Example usage’ section below.

\*Miscellaneous control unit\*

Independent image acquisition can be performed in 3 modes. [Live Acq] continuously acquires images and displays them on the GUI at the user-defined frame rate and exposure time until terminated by clicking the button a second time. [Burst Acq] acquires a user-defined number of frames in burst mode before automatic termination. [Single Acq] acquires only one image.

[Reset DM] resets the actuators to their neutral position. [Camera exposure time spin box] controls the exposure time of the SHWS given in units of microseconds. [Maximum loop no. spin box] determines the maximum number of iterations before termination of the AO control loop.

\*Remote focusing unit\*

The remote focusing unit controls all parameters and procedures relevant to the calibration and execution of remote focusing using a DM.

[CALIBRATE] triggers the calibration of DM control voltages for remote focusing. Calibration is performed by displacing a planar sample in incremental steps along the optical axis and compensating for the displacement using closed-loop AO correction. This allows system aberration at each remote focusing depth to be corrected for as well. For details of the remote focusing calibration process, please refer to [10, 11]. The software performs this as a 2-step process that proceeds along the negative direction before the positive. The direction of calibration should be determined within the configuration file prior to software initialisation. Other important parameters include the number of and axial distance between incremental steps.

RF\_calib:

  calib\_step\_num: 11  # Number of steps used in one direction during calibration of remote focusing

  calib\_step\_incre: 10  # um Axial increment between each step during calibration of remote focusing

  calib\_direct: 0  # Direction of calibration, 0-negative, 1-positive

Messages are displayed in the message box to inform users of the current calibration direction and step number, as well as to provide users with options to 1) confirm that the sample has been moved to the next calibration step; and 2) save or discard calibrated voltages and exit the thread. Commands can then be entered via the keyboard. A typical example of the interactive command message is: *Move sample to positive position. Press [y] to confirm. Press [s] to save and exit. Press [d] to discard and exit.* The software automatically saves calibrated voltages to the HDF5 file after reaching the maximum user-defined step number or does so on request at an arbitrary step. After interpolation of DM control voltages acquired at each calibration step, all remote focusing voltages can then be loaded upon software initialisation.

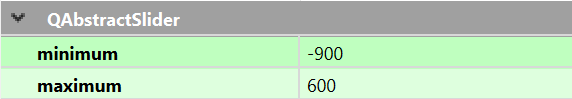
[Scan Focus] controls whether remote focusing is to be performed at only one depth or at multiple depths. When unticked, only the first spin box on the left-hand-side panel is accessible while others are disabled. When ticked, the otherwise is true.

[MOVE] performs remote focusing at only one depth by first retrieving DM control voltages corresponding to that specified in [Focus Depth (microns)], and then applying those voltages on the DM to move the focus relative to the natural focal plane. Negative values move towards -z-axis and positive ones +z-axis.

[SCAN] performs remote focusing sequentially at multiple depths by applying different sets of voltages to the DM membrane and scanning the focus depth-wise. [Step Increment (microns)] controls the displacement interval between consecutive remote focusing planes, [Step Number] the number of planes to be accessed, [Start Depth (microns)] the depth at which to start the remote focusing process relative to the natural focal plane, and [Depth Pause Time (s)] the time for which the focal spot remains stable at each remote focusing plane. Entering a negative value for [Step Increment (microns)] permits remote focusing in the -z-direction.

[AO Correction Type] allows the user to choose whether or not to apply AO correction at each remote focusing plane and which type to apply. Four options are available from the drop box: [Zernike AO 3], [Zernike Full], [Slope AO 3], and [Slope Full], the features of which have been discussed above. All options are applicable to the remote focusing process behind both [MOVE] and [SCAN], default is set to [None].

[Remote focusing toggle bar] permits random access of different remote focusing planes during imaging by simply dragging the bar to a desired depth. Default is set to the natural focal plane and max/min limits are provided by the user according to remote focusing calibration results by changing the minimum and maximum value of the ‘RFslider’ widget in Qt Designer. The values are calculated as [depth (um) / step increment (um)].



\*\*Example usage\*\*

Three examples are given for automated AO performance characterisations of a real reflectance confocal microscope described in [10, 11].

**Example 1**: The dynamic range of the SHWS for the first *N* Zernike modes can be characterised by generating and correcting for them in closed-loop using mode 0/1 of [Data Collection]. Parameters in ‘config.yaml’ under ‘AO’ and ‘data\_collect’ can be set as follows.

AO:

control\_coeff\_num: 20  # Number of zernike modes to control during AO correction

data\_collect:

  data\_collect\_mode: 1  # Mode flag for Data Collection button, detailed function descriptions in app.py

  loop\_max\_gen: 15  # Maximum number of loops during closed-loop generation of zernike modes in mode 0/1/2/3

  incre\_num: 15  # Number of zernike mode amplitudes to generate in mode 0/1

  incre\_amp: 0.02  # Increment amplitude between zernike modes in mode 0/1

  run\_num: 5  # Number of times to run mode 0/1/2/3

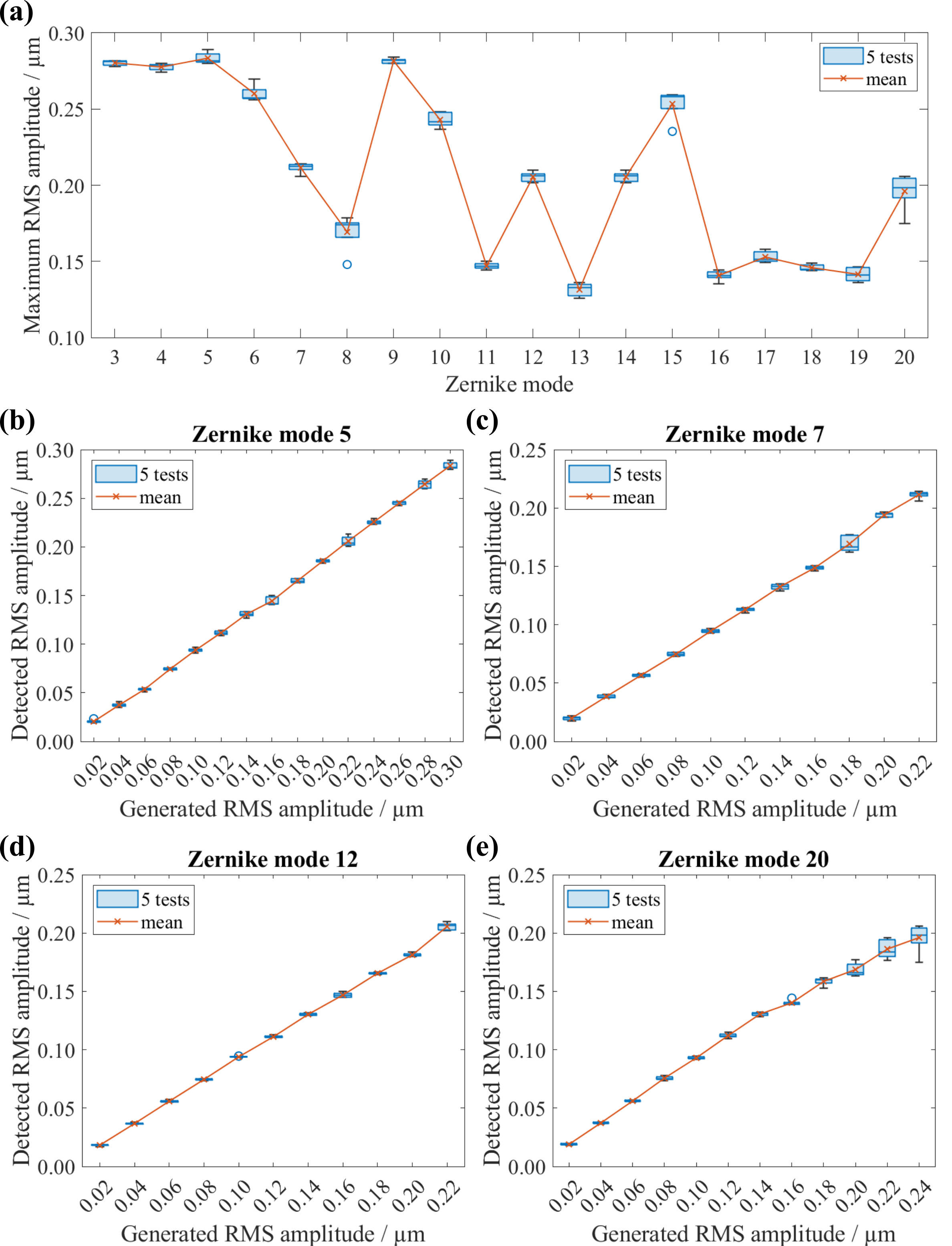


Figure 1. Characterisation results of system AO correction performance. (a) Dynamic range of the SHWS for Zernike modes 3∼20 (excl. tip/tilt). (b)-(e) Generated and detected RMS amplitudes of odd/even Zernike modes (b) 5, (c) 7, (d) 12, and (e) 20, in increments of 0.02 μm. 5 tests were performed for each measurement.

**Example 2**: The degree of Zernike mode coupling upon detection at the SHWS can be characterised by individually generating the same amount of each mode on the DM and constructing a heatmap of the detected Zernike coefficients using mode 0/1 of [Data Collection]. Parameters in ‘config.yaml’ under ‘data\_collect’ can be set as follows.

data\_collect:

  data\_collect\_mode: 1  # Mode flag for Data Collection button, detailed function descriptions in app.py

  loop\_max\_gen: 15  # Maximum number of loops during closed-loop generation of zernike modes in mode 0/1/2/3

  incre\_num: 1  # Number of zernike mode amplitudes to generate in mode 0/1

  incre\_amp: 0.1  # Increment amplitude between zernike modes in mode 0/1

  run\_num: 5  # Number of times to run mode 0/1/2/3

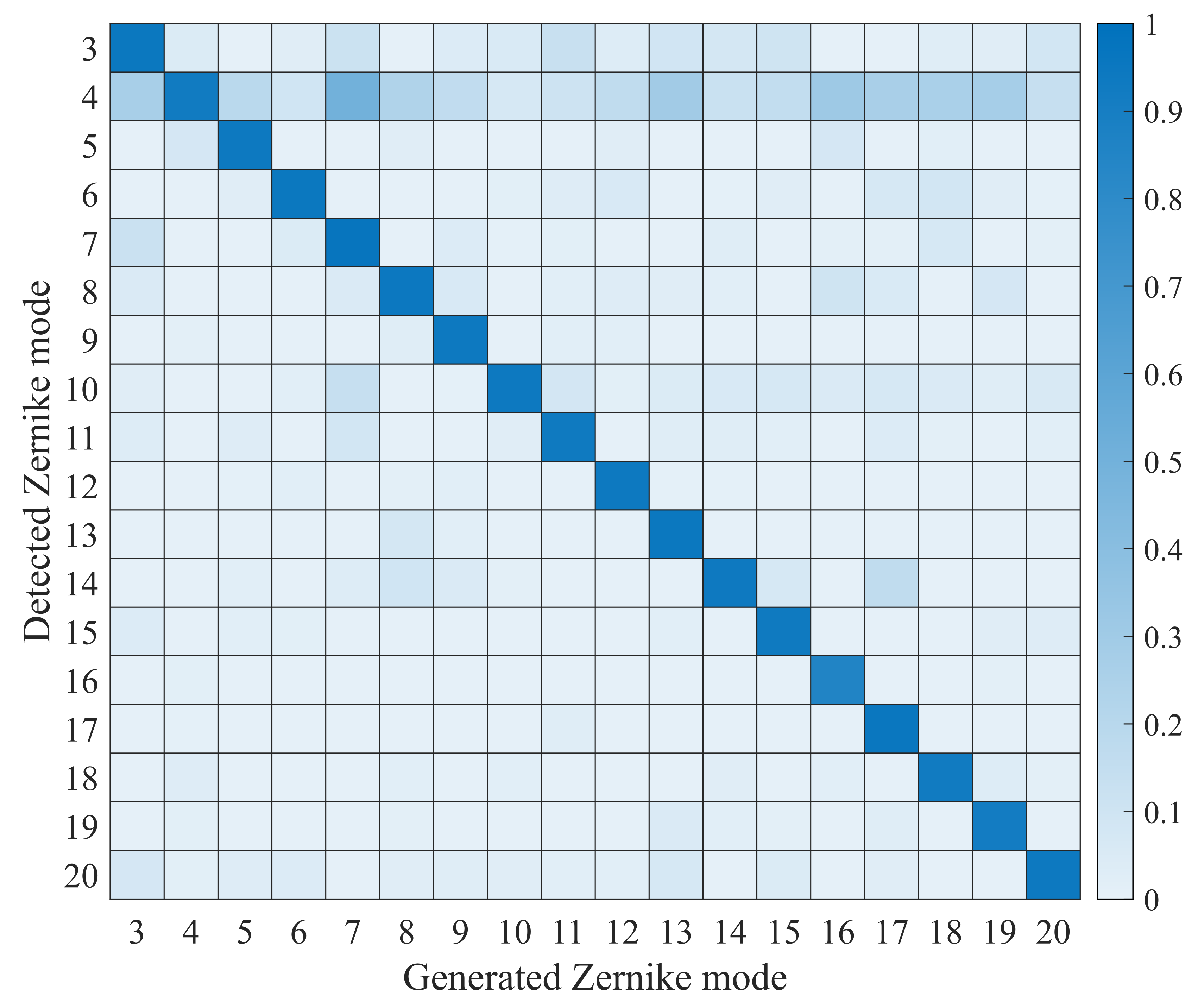


Figure 2. Heatmap of correlation coefficients between detected and generated mode values for 0.1 μm of Zernike modes 3∼20 (excl. tip/tilt).

**Example 3**: To ensure the system could correct for multiple Zernike modes with good stability and minimal mode coupling, different combinations of odd and even Zernike modes can be generated and corrected for in closed-loop using mode 2/3 of [Data Collection]. Parameters in ‘config.yaml’ under ‘data\_collect’ can be set as follows.

data\_collect:

  data\_collect\_mode: 3  # Mode flag for Data Collection button, detailed function descriptions in app.py

  loop\_max\_gen: 15  # Maximum number of loops during closed-loop generation of zernike modes in mode 0/1/2/3

  run\_num: 5  # Number of times to run mode 0/1/2/3

And [Zernike array edit box] can be set to [0 0 0 0 **0.1** 0 **0.1** 0 0 0 0 **0.1** 0 0 0 0 0 0 0 **0.1**].

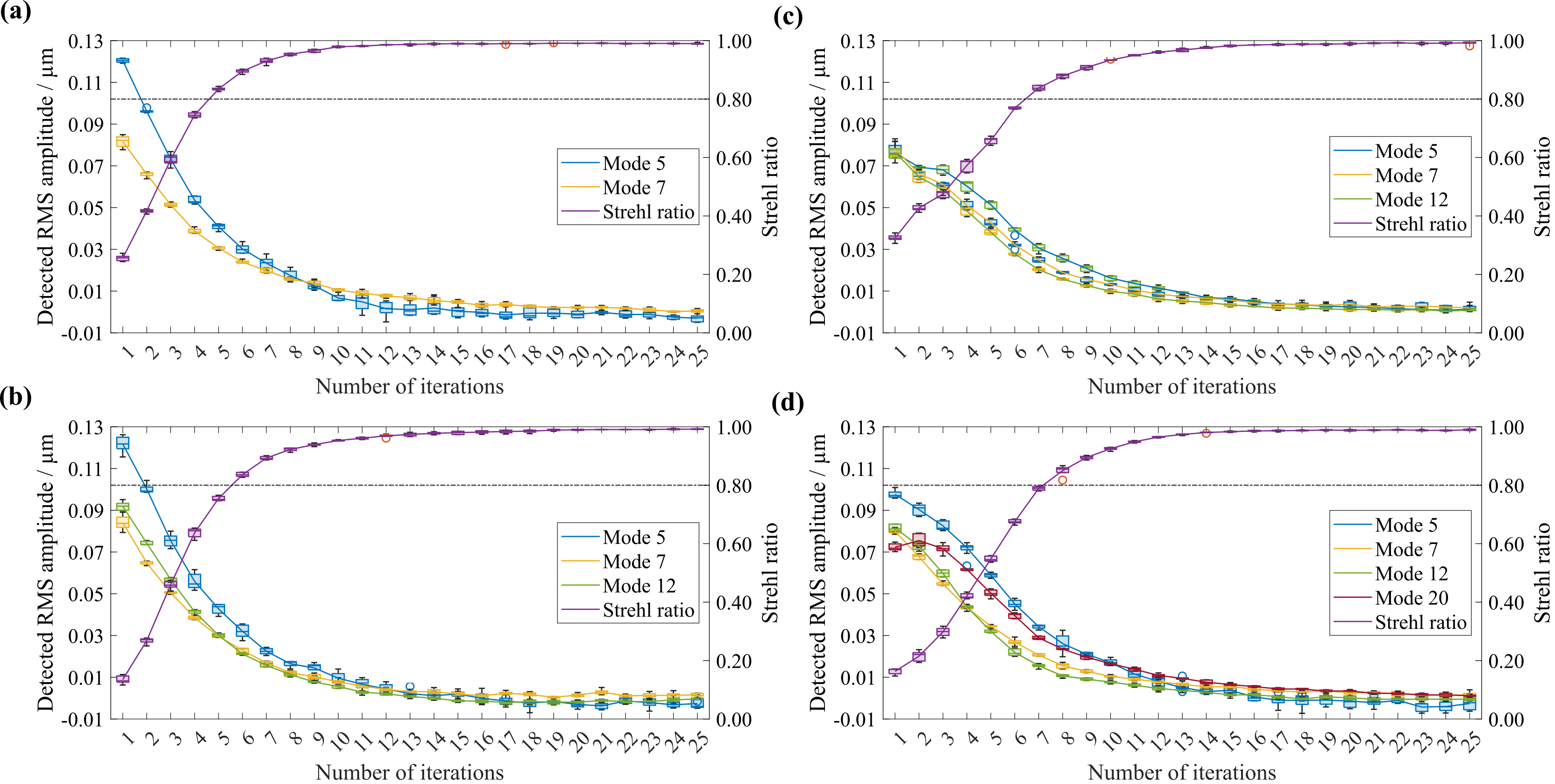


Figure 3. Detected amplitudes of generated odd and even Zernike mode combinations and Strehl ratio calculated using first 69 Zernike modes (excl. tip/tilt) at each iteration of closed-loop AO correction. 5 tests were performed for each measurement.

\*\*Software functionality tests\*\*

To perform functionality tests without hardware, leave all parameters in ‘config.yaml’ as default, run the software in **debug mode**, and perform the following tests in sequence:

* [Initialise SB]: A search block region should appear in the ImageViewer with 14 search blocks across the diameter. Message box: “**Search block geometry initialised. Number of search blocks within the pupil is: 148.**”
* [Position SB]: A white dot should appear at the centre of all the search blocks. Message box: “**Need to calibrate DM? [y/n]**” -> Click on the command prompt and press ‘y’ on the keyboard. Message box: “**Reposition search blocks using keyboard. Press arrow keys to centre S-H spots in search blocks. Press Enter to finish.**” -> Press '↑', '↓', '←', '→' keys on the keyboard to reposition search blocks and press ‘Enter’ to confirm. Message box: “**Search block position confirmed.**”
* [Calibrate-S]: Message box: “**Dummy DM actuator coordinates loaded.**”
* [S-Z Conv]: Message box: “**Exiting slope - Zernike conversion process.**”
* [Calibrate-Z]: Message box: “**Exiting Zernike control matrix calibration process.**”
* [Reset DM]: Message box: “**DM reset success.**”
* Adjust the remote focusing toggle bar: Message box: “**Focus position: X.X um.**”

\*\*Notes for development\*\*

1. The current software assumes using an electromagnetic DM, where the actuator displacement is linearly proportional to the applied control voltage [12]. In this case, the device can be driven linearly in both the negative and positive directions by applying a normalised control voltage of , where is the maximum control voltage. However, if an electrostatic DM is to be used, the actuator displacement would be proportional to the square of the applied control voltage. In such a case, in order to drive the device linearly, the normalised control voltage should be given as .
2. The SHWS used in the current example is custom built and uses a Ximea camera as the sensor. The interfacing of a different SHWS or different camera requires minor modifications of the software from two aspects: sensor instantiation and image acquisition. The former can be modified from ‘sensor.py’, and a new class should be added for the new sensor in a similar way as the given example in ‘sensor.py’. The latter can be modified from ‘image\_acquisition.py’, and a new image instance should be created according to the specific sensor API.
3. The DM used in the current example is an Alpao-69 DM. Similar as above, the interfacing of a different DM requires modifications of the software from two aspects: mirror instantiation and DM control commands. The former can be modified from ‘mirror.py’, and a new class should be added for the new DM in a similar way as the given example in ‘mirror.py’. The latter should use new commands to replace where applicable:

self.mirror.Send(voltages)

self.devices['mirror'].Send(voltages)

self.mirror.Reset()

self.devices['mirror'].Reset()

\*\*References\*\*

1. M. J. Booth, “Adaptive optical microscopy: the ongoing quest for a perfect image,” *Light Sci Appl* 3, e165 (2014). <https://doi.org/10.1038/lsa.2014.46>
2. N. Ji, J. Freeman, and S. Smith, “Technologies for imaging neural activity in large volumes,” *Nat Neurosci* 19, 1154–1164 (2016). <https://doi.org/10.1038/nn.4358>
3. M. J. Townson, O. J. D. Farley, G. Orban de Xivry, J. Osborn, and A. P. Reeves, “AOtools: a Python package for adaptive optics modelling and analysis,” *Opt. Express* 27, 31316-31329 (2019). <https://doi.org/10.1364/OE.27.031316>
4. N. Hall, J. Titlow, M. J. Booth, and I. M. Dobbie, “Microscope-AOtools: a generalised adaptive optics implementation,” *Opt. Express* 28, 28987-29003 (2020). <https://doi.org/10.1364/OE.401117>
5. U. Bitenc, “Software compensation method for achieving high stability of Alpao deformable mirrors,” *Opt. Express* 25, 4368-4381 (2017). <https://doi.org/10.1364/OE.25.004368>
6. P. Artal, S. Marcos, R. Navarro, and D. R. Williams, “Odd aberrations and double-pass measurements of retinal image quality,” *J. Opt. Soc. Am. A* 12, 195-201 (1995). <https://doi.org/10.1364/JOSAA.12.000195>
7. B. Dong and M. J. Booth, “Wavefront control in adaptive microscopy using Shack-Hartmann sensors with arbitrarily shaped pupils,” *Opt. Express* 26, 1655-1669 (2018). <https://doi.org/10.1364/OE.26.001655>
8. J. Ye, W. Wang, Z. Gao, Z. Liu, S. Wang, P. Benítez, J. C. Miñano, and Q. Yuan, “Modal wavefront estimation from its slopes by numerical orthogonal transformation method over general shaped aperture,” *Opt. Express* 23, 26208-26220 (2015). <https://doi.org/10.1364/OE.23.026208>
9. J. Cui, B. Dong, and M. J. Booth, “Shack-Hartmann sensing with arbitrarily shaped pupil (1.0),” Zenodo (2020). <https://doi.org/10.5281/zenodo.3885508>
10. J. Cui, R. Turcotte, K. Hampson, N. J. Emptage, and M. J. Booth, “Remote-Focussing for Volumetric Imaging in a Contactless and Label-Free Neurosurgical Microscope,” in Biophotonics Congress 2021, C. Boudoux, K. Maitland, C. Hendon, M. Wojtkowski, K. Quinn, M. Schanne-Klein, N. Durr, D. Elson, F. Cichos, L. Oddershede, V. Emiliani, O. Maragò, S. Nic Chormaic, N. Pégard, S. Gibbs, S. Vinogradov, M. Niedre, K. Samkoe, A. Devor, D. Peterka, P. Blinder, and E. Buckley, eds., OSA Technical Digest (Optical Society of America, 2021), paper DTh2A.2 (2021). <https://www.osapublishing.org/abstract.cfm?uri=BODA-2021-DTh2A.2>
11. J. Cui, R. Turcotte, N. J. Emptage, and M. J. Booth, “Extended range and aberration-free autofocusing via remote focusing and sequence-dependent learning,” *Opt. Express* 29, 36660-36674 (2021). <https://doi.org/10.1364/OE.442025>
12. U. Bitenc, N. A. Bharmal, T. J. Morris, and R. M. Myers, “Assessing the stability of an ALPAO deformable mirror for feed-forward operation,” *Opt. Express* 22, 12438-12451 (2014). <https://doi.org/10.1364/OE.22.012438>