JATIS Journal of Astronomical Telescopes, Instruments, and Systems

RESEARCH PAPER

SHINS: the SHARK-NIR instrument control software

Davide Ricci,[a,*] Fulvio Laudisio,[a] Alessandro Lorenzetto,[a] Marco De Pascale,[b] Andrea Baruffolo,[a] Daniele Vassallo,[a] Domenico Barbato,[a] Maria Bergomi,[a] Florian Briegel,[c] Elena Carolo,[a] Simone Di Filippo,[a] Marco Dima,[a] Valentina D'Orazi,[a] Tânia Sofia Gomes Machado,[a] Davide Greggio,[a] Luca Marafatto,[a] Dino Mesa,[a] Lars Mohr,[c] Gabriele Rodeghiero,[a] Kalyan Kumar Radhakrishnan Santhakumari,[a] Gabriele Umbriaco,[d] Valentina Viotto,[a] and Jacopo Farinato[a]

[a]INAF – Osservatorio Astronomico di Padova, Padua, Italy
[b]Leibniz Supercomputing Center of the BAdW, München, Germany
[c]Max-Planck-Institut für Astronomie, Heidelberg, Germany
[d]Alma Mater Studiorum Università di Bologna, Dipartimento di Fisica e Astronomia "Augusto Righi", Bologna, Italy

ABSTRACT. SHARK-NIR is a compact instrument for coronagraphic imaging, direct imaging, and coronagraphic spectroscopy in the near-infrared wavelengths (0.96–1.7 μm) mounted at the left bent Gregorian focus of the Large Binocular Telescope (LBT). Taking advantage of the telescope's adaptive optics system, it provides high-contrast imaging with coronagraphic and spectroscopic capabilities and is focused on the direct imaging of exoplanets and circumstellar discs. We present SHINS, the SHARK-NIR instrument control software, mainly realized with the TwiceAsNice framework from MPIA, Heidelberg, and the Internet Communications Engine (ICE) framework using the C++ programming language. We describe how we implemented the software components controlling several instrument subsystems, through the adaptation of already tested libraries from other instruments at LBT, such as LINC-NIRVANA. The scientific detector comes with its own readout electronic and control software interfaced with our software through Instrument-Neutral Distributed Interface (INDI). We describe the C++ core software Observation Control Software, responsible for dispatching commands to the subsystems, also implementing a software solution to avoid a potential collision among motorized components, fully transparent to final users. It exposes an ICE interface and can be controlled by clients developed in different languages. Observation, calibration, and maintenance procedures are implemented by means of template scripts, written in python language, controlling Observation Control Software through its ICE interface. These templates and their parameters are configured using "ESO-style," XML observation blocks (OBs) prepared by observers, or in general SHARK-NIR users. The high-level control is carried out by REST HTTP APIs implemented in a python back-end, also acting as a web server for the several browser-based front-end graphical user interfaces that allow the OBs to edit and sequence, as well as individual device movement and monitoring. Finally, we present the first scientific results obtained by SHARK-NIR using coronagraphic mode.

 [DOI: 10.1117/1.JATIS.11.2.025005]

Keywords: SHARK; Instrument Control Software; software; coronagraphy; spectroscopy; imaging; astronomy; HTTP API.

*Address all correspondence to Davide Ricci, davide.ricci@inaf.it

Paper 25016G received Jan. 21, 2025; revised Apr. 8, 2025; accepted Apr. 17, 2025; published May 15, 2025.

1 Introduction

SHARK-NIR is the near-infrared arm of SHARK, an instrument proposed for the Large Binocular Telescope (LBT) in the framework of the "2014 Call for Proposals for Instruments Upgrade and New Instruments" and composed of two channels (the second twin being called SHARK-VIS[1]), installed one for each LBT side.[2–5] This compact instrument is mounted at the left bent Gregorian focus, and exploits the Single conjugated adaptive Optics Upgrade for LBT (SOUL),[6] operating in a wavelength range between 0.96 and 1.7 μm and focused on high contrast imaging. It provides coronagraphic imaging, direct imaging, and coronagraphic spectroscopic capabilities. In this introduction, we describe its opto-mechanical components following the path of the light through the instrument (see a schematic view in Fig. 1). The description is structured to reflect the hardware and software control that will be expanded in the following sections.

1.1 Motion Devices

The light beam coming from the tertiary mirror of LBT (see the top part of Fig. 1) is split by a PI LS-180 deployable dichroic (inbeam_dep). The visible wavelength proceeds to SOUL; the remaining part is reflected toward an "entrance" PI M-232 tip-tilt mirror (inbeam_tt) until it reaches the Maxon GP16 instrument's shutter (shutter). Then, the beam passes through the coronagraphic system provided by a set of PI M116 DGH apodizers, coronagraphic mask, and Lyot stop (apo, coro, and lyot), each set being mounted on a separate motorized wheel. These wheels

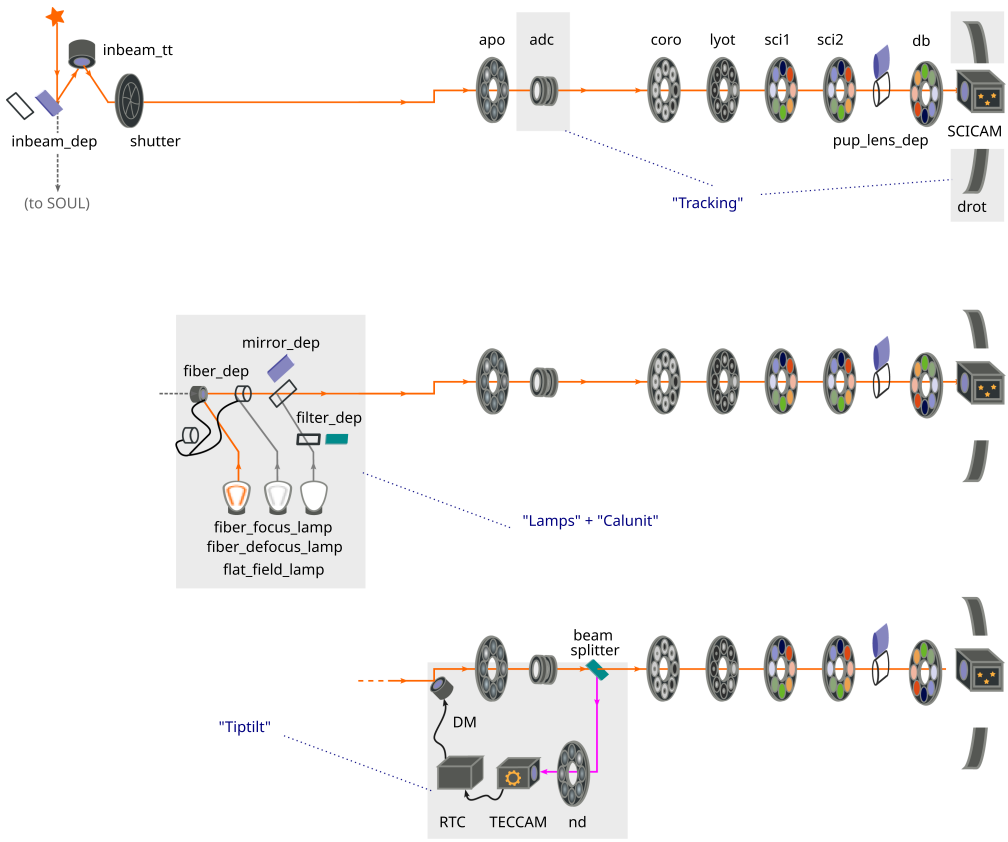

Fig. 1 Schematic view of SHARK-NIR, replicated three times following the three principal "light paths." The top part shows the path of the light from a celestial source through the science detector and highlights the tracking devices. The middle part shows the the path from an internal calibration lamp through a fiber deployer, which are two of the components of the calibration unit. The bottom part is focused on the internal adaptive optics loop system of the instrument. See Sec. 1 for details.

also account for ancillary devices to carry out optical quality calibration procedures, slits for spectroscopy, and corresponding dispersing elements. This allows coronagraphic long slit spectroscopy at the low and middle resolution, also involving a separate STANDA 8MT30-50DCE pupil lens deployer pup_lens_dep. Afterward, a set of PI M-232 scientific wide, narrow-band, neutral density, and dual-band filters are available on three separate wheels (sci1, sci2, and db) allowing several filters combinations before reaching the Teledyne HAWAII H-2RG Scientific Camera (SCICAM).

1.2 Tracking Devices

Field derotation and atmospheric dispersion correction (ADC) are ensured by tracking devices: a Fulling Motor 57SH76-4AM (drot) and two combined STANDA 8MPR16-1 (adc). The whole instrument is installed onto a large bearing, allowing its rotation around an axis passing through the center of the field of view, following an arbitrary trajectory. The derotation trajectory is obtained from the Telescope Control Software (TCS) and allows to follow and compensate for the rotation of the sky. The ADC, consisting of two counter-rotating prisms, is placed between the apodizer and the coronographic wheel. This correction is applied at a slower rate and allows the compensation of the atmospheric dispersion.

1.3 Lamps and Calunit Devices

The instrument is equipped with a calibration unit (see the middle panel of Fig. 1): one flat field lamp and two fiber lamps (one to provide an in-focus source and one for a defocus source) used for calibration (flat_field_lamp, fiber_focus_lamp, fiber_defocus_lamp), and disposed in the optical path using a PI M403.2DG deployer (fiber_dep) and two PI M112-1.DG deployers (mirror_dep and filter_dep).

1.4 Tip–tilt Devices

The instrument is also equipped with an internal wavefront control system (see the bottom part of Fig. 1): a beam splitter between the ADC doublet and the coronagraphic wheel directs a fraction of the beam through a PI M116 DGH neutral density filter wheel (nd), and then to a FirstLight CRED-2 Technical Camera (TECCAM), which acts as a wavefront sensor for real-time computer (RTC) provided by Microgate, calculating the real-time controls for a ALPAO 97-15 deformable mirror (DM). This system ensures fast tip-tilt correction up to 2 kHz, non-common path aberrations (NCPA) correction, and fine point spread function (PSF) positioning behind the coronagraph.

1.5 SCICAM

The scientific detector is an Hawaii 2RG 2048×2048 px from Teledyne with 18 μm pixel size, with a 18×18 in. field of view and a pixel scale of 14.5 milliarcseconds per pixel. The detector operates under cryogenic temperature and is therefore contained inside a cryostat. The temperature is controlled by Lake Shore 336.

The main scientific goal of SHARK-NIR is the imaging (direct, coronagraphic) and long-slit spectroscopy of exoplanets, their detection, and characterization. Other science objectives include brown dwarfs, protoplanetary discs, stellar jets, quasi-stellar objects, and active galactic nuclei. SHARK-NIR started its scientific operations in 2024, after a wider assembly, integration, and test phase[7–13] and the commissioning, soon obtaining promising scientific results. These scientific results[14,15] are the contribution to the characterization of an eccentric giant planet detected through radial velocity around the star HD 57625, and a deep imaging study of three accelerating stars: HIP11696, HIP47110, and HIP36277.

In this paper, we present in detail SHINS, the SHark INstrument control Software, expanding and structuring the improvements described in a dedicated series of SPIE proceedings.[16–19]

2 Control Electronics

In this section, we present the hardware responsible for controlling the optomechanical parts, the lamps, the wavefront control system, and the cameras of the instrument. See Table 1 for hardware details and the gray squared boxes in Fig. 2.

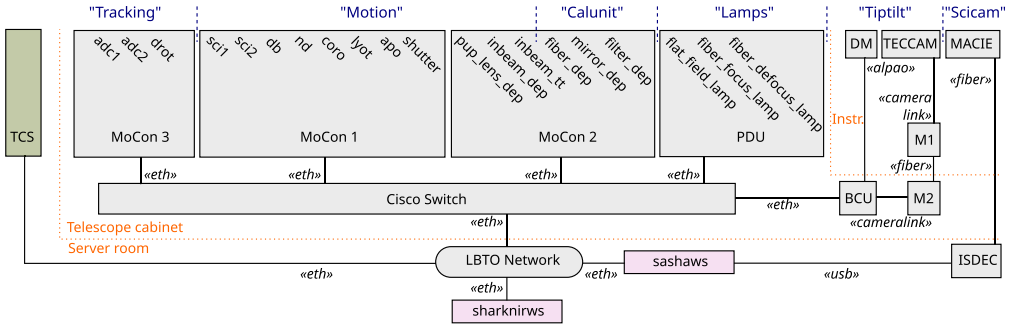

Fig. 2 How the SHARK-NIR devices (gray boxes), workstations (pink boxes), and TCS (olive green box), described in Sec. 1, are controlled by Control Electronics and interfaced in the control network to the instrument workstation. The location of the hardware is also specified in clear between dotted lines. See Secs. 2 and 3 for details.

2.1 MoCon Boards

All motorized functions (motion devices, Calunit devices, and tracking devices) are electronically controlled by a group of three MoCon boards, custom-made and provided by the Max-Planck-Institut für Astronomie (MPIA) in Heidelberg, which are the same type motion control boards already employed by the LINC-NIRVANA[20] instrument at LBT.

The three MoCon boards can be operated from outside using a telnet connection and a list of predefined commands. They can mount modules to drive dc motors in closed loop or stepper motors in open loop and can read feedback from incremental encoders and absolute encoders. Each board can control up to eight modules, each with up to three sensors, two "end of stroke" and a home sensor. The sensors can be "normally open" switches, "normally closed," or level switches. The firmware of the MoCon automatically reads the incremental encoders and corrects the power using a proportional-integral-derivative (PID) controller. The first board is dedicated to the motion device wheels and the shutter: the second is dedicated to the motion device "entrance" tip–tilt mirror and deployers, as well as the Calunit devices deployers; the third and last is dedicated to the tracking device derotator and ADCs.

The instrument has seven filter wheels, six linear stages, one tip–tilt actuator; moreover, there is the ADC with two motorized prisms and the derotator (see Table 1 for details). The wheels have one reference sensor used as home position while the linear stages have two end-of-stroke switches, whereas the ADC prisms are mounted on circular stages, with stepper motors equipped with a reference sensor. The shutter is the only linear stage built from scratch using a DC motor connected to an encoder and to a threaded rod through a reduction gearbox. Two induction sensors act as the end of strokes. It is planned to be refurbished with a commercial shutter.

The derotator is slightly more complex, using a custom bearing connected in parallel, to minimize backlash, to two high-torque stepper motors. Two induction sensors act as the end of stroke, and an absolute encoder with 2^{16} positions is connected to the bearing by means of a zero backlash gearbox. Each line of the absolute encoder corresponds to $\approx 20''$. A special procedure to minimize backlash is performed at the beginning of the observations. This procedure required customization of one of the MoCon boards that allowed moving the two stepper motors, controlled by a single module, independently in opposite directions to ensure a tight fit between the motor gears and the main bearing.

2.2 Lamps, Adaptive Optics, and Scientific Camera

Lamps are switched on and off by an off-the-shelf power distribution unit (PDU) produced by the CyberPower company. We also plugged into the same PDU: the three MoCon boards, the scientific camera electronics, the DM electronics the RTC, the TECCAM and its related components, and the Lake Shore controller, so that it was possible to set up a shutdown/startup script for maintenance purposes, as well as power cycle individual components as troubleshooting procedure.

Table 1 SHARK-NIR device descriptions and related hardware. Calibration lamps are switched on and off through a CyberPower power distribution unit. See Secs. 1 and 2 for details.

Device	Description	Motor/control
Motion devices		
inbeam_dep	Deployable dichroic	PI LS-180
inbeam_tt	Entrance tip-tilt mirror	PI M-232
shutter	Instrument shutter	Maxon GP16
apo	Apodizer wheel	PI M-116 DGH
coro	Coronagraphic mask wheel	PI M-116 DGH
lyot	Lyot stop and grism wheel	PI M-116 DGH
pup_lens_dep	Pupil lens deployer	STANDA 8MT30-50DCE
sci1, sci2, db	Scientific filter wheels	PI M116 DGH
Tracking devices		
drot	Field derotator	Fulling Motor 57SH76-4AM
adc	ADC prisms	STANDA 8MPR16-1
Lamps and Calunit devices		
flat_field_lamp	Flat field calibration lamp	(CyberPower PDU)
fiber_focus_lamp	Fiber lamp for in-focus calib.	(CyberPower PDU)
fiber_defocus_lamp	Fiber lamp for defocused calib.	(CyberPower PDU)
fiber_dep,	Optical path deployer	PI M-403.2DG
mirror_dep,	Optical path deployer	PI M-112-1.DG
filter_dep	Optical path deployer	PI M-112-1.DG
Tiptilt devices		
nd	Neutral density filter wheel	PI M-116 DGH
TECCAM	Wavefront sensing camera	FirstLight C-RED2
RTC	RTC basic control unit	Microgate
DM	DM for tip–tilt correction	ALPAO 97-15
SCICAM	Scientific detector	Teledyne HAWAII H-2RG

The internal wavefront control system of the tip–tilt devices includes the DM, the TECCAM, and the RTC. The DM is a 97-actuator DM from ALPAO which comes with its own control electronics. The FirstLight CRED-2 TECCAM, which has been set to operate at a temperature of −40°, runs at a frame rate of up to 400 Hz in full frame and up to 2 kHz depending on the subframe's dimension. The choice of the subframe depends on whether the camera is used for tip–tilt correction or in "patrol" mode, and it is detailed in Sec. 8.2. The RTC, called basic control unit (BCU) and developed by Microgate, has a custom motherboard with a camera link port, that receives the frames from the TECCAM. The BCU motherboard also has a Molex connector compliant with the DM control electronics. A dedicated firmware detects the light centroid variation in the frames from the technical camera and applies the proper correction to the deformable mirror.

Finally, the SCICAM is controlled by a multi-purpose control and interface electronics by Markury Scientific Inc. connected to the detector workstation (sashaws) which runs the control software and the Instrument-Neutral Distributed Interface (INDI) server.

3 Control Network

In this section, we describe how the control electronics are embedded in the control network of the instrument (see Fig. 2 for details).

The three MoCon boards, which are located in a cabinet close to the instrument, are Ethernet-connected to the SHARK-NIR Cisco Switch, which connects to the LBTO (LBT Observatory) internal network via standard Ethernet LAN. The PDU controlling the lamps is also connected to the same switch. In the same cabinet are placed the control electronics running the RTC: the communication between the DM and the BCU takes place on ALPAO dedicated cable, whereas the communication between the TECCAM and the BCU is sent via CameraLink and mediated by a media converter (M1 and M2 in Fig. 2) via optical fiber. Then, the BCU communicates via Ethernet through the switch.

The scientific camera is controlled by a dedicated component, run by a separate workstation, and developed by the Steward Observatory in Tucson. Direct communication between camera electronics (Teledyne SIDECAR) and its workstation is realized using a dedicated field programmable gate array (FPGA)-based interface, called ISDEC, via optical fiber on the camera side and via USB on the workstation side (see Fig. 2). This workstation is called sashaws, and it is placed in the LBTO server room.

Then, the control software of SHARK-NIR is managed by an instrument workstation located in the same server room. Control information is transmitted through the LBTO network node, via standard Ethernet LAN, to the controllers connected to the Switch and the scientific camera workstation.

Figure 2 also shows the TCS workstation, which is not part of SHARK-NIR, but it is used as a resource by the instrument control software, as described in Sec. 4.

SHARK-NIR network devices belong to a dedicated VLAN. From the LBTO control network, it is possible to directly connect to the instrument devices, whereas from outside the control network, a VPN access is needed. Most SHARK-NIR network devices are protected by telnet login (Mocon boards) and HTTP BasicAuth credentials (PDU, CISCO switch, and other web interfaces), which is enough, once in the local network, to avoid mistakes while operating other instruments.

4 Subsystems Control Software

LBT allows a high degree of flexibility to the instrument teams developing the architecture of the control software for their instruments, although they do express a preference for web-based user interfaces. Given the experience of our team with ESO VLT control software, we organized the architecture of SHINS in a way similar to that of ESO VLT instruments: dedicated Subsystems Control Software refer to a central component, called Observation Control Software, which is in charge of coordinating them and dispatching commands. On top of this component, observation, calibration, and maintenance procedures are implemented by means of template scripts, which are called by a sequencer through a set of parameters.

We now describe in detail the control software of the several subsystems. Figure 3 shows a schematic view of the control software.

`motion_ctrl` service controls motion devices: As we used MoCon boards to control motors, we also decided to adopt its high-level control framework, also developed by MPIA for the LINC-NIRVANA instrument, called TwiceAsNice (TaN) in the form of a set of C++ libraries.[21] A common feature of all the sub-packages developed using the TaN framework is that each sub-package is mapped to a TaN service available through the TaN server process, each service controlling a subset of SHARK-NIR devices. The adoption and customization of TaN libraries represent most of the code reuse of SHINS.

The service loads the configuration for each motor, opens the telnet communication with the MoCon boards, and keeps track of the status of the motor, the position, the speed, the status of the switches, and the errors. In the configuration for each motor can be specified the ranges of the motor (software limits), the named positions, the different units to control the device, and the rules to convert these units.

`motion_ctrl` is then the service dedicated to motion devices, the devices used by the instrument during science operations.

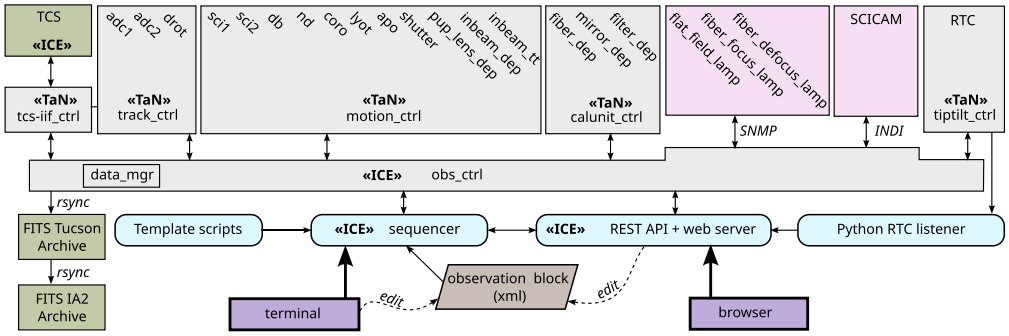

Fig. 3 SHARK-NIR software stack. See Secs. 4–8, for details. In the top part, gray boxes refer to C++-based control developed over TaN and ICE frameworks, whereas pink boxes are for non-C++ code interfaces, and their external communication protocols are shown out of the boxes. TCS, FITS Tucson Archive, and FITS IA2 Archive, shown in olive green boxes on the left, refer to external interfaces. Rounded boxes refer to Python code, some of which use ICE to connect to the central C++ component `obs_ctrl`. Configuration files for templates are also shown in the slanted box. Finally, bold purple boxes show two different ways to control SHINS: via terminal commands, manually editing custom templates setup for OB creation and then launching the sequencer, or via the web browser. The browser also allows the XML observation block editing and the control of individual components, as well as the monitoring of the RTC diagnostic data via a dedicated Python listener.

`calunit_ctrl` is the service controlling the motors used during calibration operations, i.e., the Calunit devices.

`track_ctrl` is the service controlling the derotator and the two ADCs, i.e., the tracking devices. However, in this case, there is the need to calculate the position information based on the telescope's position at a given time. Tracking functions follow the telescope movement and, thus, are active during observation and controlled by the `track_ctrl` component. SHINS implements two tracking functions: field derotation and atmospheric dispersion correction.

The derotator is one of the most critical devices because it has to remain as much in sync with the telescope as possible; for this reason, the trajectory is queried and uploaded to the derotator every second. The derotator service addresses the peculiarity of the chosen setup. The bearing compensates for the apparent field of view rotation due to the Alt-Az nature of the telescope. Each of the eight MoCon channels can follow an arbitrary trajectory profile loaded in a shared memory on the board, so once the derotator service obtains from the telescope interface the derotation profile (that depends on the current preset), it converts it into a series of cubic segments. These segments are sent to the MoCon firmware to approximate an arbitrary profile, keeping track of the position by reading the absolute encoder.

In contrast, the correction for atmospheric dispersion, performed by the ADC, in SHARK-NIR is less critical than the derotator, and it does not need knowledge of the telescope trajectory. The ADC position is updated with a time scale of few seconds and is a function of pressure, temperature, and zenith angle.

For all these reasons, derotator trajectory and ADC position are a function of the telescope position, velocity, and acceleration. The LINC-NIRVANA control software team also expanded for SHARK-NIR the TaN service used as an interface to the TCS, which was already developed for LINC-NIRVANA. This component is called `tcs_ctrl`, and it is described here below.

`tcs_ctrl` is the service interfacing with, and controlling the TCS. The Large Binocular Telescope provides a set of API for instruments control software to interface to TCS, called instrument interface (IIF). Such APIs are declared in the specification language for ICE (SLICE) and implemented in C++. The IIF APIs are accessible to SHINS thanks to a TaN service, `tcs_ctrl`, developed and tested by MPIA for LINC-NIRVANA.

`tiptilt_ctrl` is the service controlling the internal wavefront control system of SHARK-NIR. The RTC is responsible for updating the tiptilt of the DM in real time, so to reduce telescope vibrations. Moreover, to correct NCPA, it is possible to upload a static shape to the DM in the form of a vector of coefficient of Zernike polynomials, kept during the tip–tilt loop. Moreover, it is also used for fine-positioning of the scientific object behind the coronagraphic

mask. All of the communication happens through the BCU, and there is no direct communication with DM electronics or TECCAM. The private company that produced the BCU, Microgate, provided control software made by a set of Matlab functions. These functions have been ported to C++ and implemented in the `tiptilt_ctrl` service logic so that it is possible to write directly in the BCU memory the required quantities and send them from the workstation to the BCU with User Datagram Protocol (UDP) packets using the Microgate Protocol. At a lower level, we reused the TaN libraries of ARGOS,[22] the binocular laser–guided ground layer adaptive optics at LBT, to communicate with the BCU, which prepares the UDP packets and sends them to BCU.

5 Observation Control Software

The TaN framework is built on top of Internet Communication Engine (ICE), an object-oriented remote procedure call framework.[23] For this reason, we developed in C++ the central component of the software, called `obs_ctrl` or Observation Control Software, on top of ICE.

Services run in separate threads, allowing parallel operations (for example the instrument setup while the telescope is moving). Moreover, The ICE framework allows exposing both synchronous and asynchronous versions of all class methods of `obs_ctrl`. This feature is highly used for high-level control (see Sec. 6), ensuring non-blocking operations.

`obs_ctrl` connects at startup to all other services and to the INDI server and controls all the operations of SHARK-NIR. The simultaneous communication with the subsystem services, along with its direct interaction with the scientific detector allows `obs_ctrl` to control and execute all requested operations.

This component stores the state of each SHINS device in a dedicated object, which is updated by querying the PDU [via Simple Network Management Protocol (SNMP)], and the services, at a rate of 1 Hz, whereas INDI pub/sub protocol provides the update of the state of the SCICAM based on events. So, it is possible for client components to retrieve the instrument status without querying individual subsystems and to monitor state changes if the devices are operated externally. This system provides an instant status for SHINS, whereas an external alert system, based on EPICS (https://pyepics.github.io/pyepics/overview.html), is fed by cron jobs with telemetry about the temperature and pressure of the cryogenic system and is monitored by LBTO.

Apart from coordinating and interfacing with the service subsystems described in Sec. 4, the Observation Control Software implements embedded controls and checks that are discussed in the following Secs. 5.1–5.4.

5.1 SCICAM and PDU Control

The scientific camera control software is developed directly in `obs_ctrl` using the Instrument Neutral Distributed Interface pub-sub protocol, already used at LBTO to control the LBTI scientific camera, LMIRCam.[24]

Lamps are switched on and off by controlling their PDU through an SNMP. Two separate `obs_ctrl` methods provide the setting and getting the status of the lamps. Given the simplicity of the device, no separate service has been set up to control these components.

5.2 Anti-Collision Software

During the assembly, integration and test (AIT) phase, a potential collision issue between two motorized components emerged, namely, the coronagraphic motor wheel and the fiber deployer. We developed a software solution at the `obs_ctrl` level, that is also available while using other software such as engineering panels. The anti-collision system is based on three protection layers:

1. The first level of protection consists in the fact that the wheel before making any movement checks the position of the fiber deployer, and if this is below a certain threshold, it raises an exception.
2. The second level of protection consists of a couple of functions that intervene during the setup phase of the instrument. The setup is classified according to wheel and deployer movements and is divided into nine scenarios. Of these nine scenarios, five do not need

any special attention, whereas for the last four, the software splits the setup into two or three consecutive steps where the coronagraphic wheel never moves when the fiber is in the cutoff region, minimizing execution times. Eventually, the fiber is moved to the safe region and then returned to the initial position at the end of the wheel setup.

3. The third level of protection is based on a set of callbacks installed on the services controlling the motors. Whenever the coronagraphic slit wheel starts any movement, the framework calls back the Observation Software which in turn launches a new thread that controls continuously the calibration fiber deployer position. If it is beyond the interdiction threshold, it commands an abort the movement and move the fiber deployer into the safe area. This third level of protection intervenes whenever a movement of the coronagraphic wheel is commanded by any client of the services provided that SHINS is up and running, thus protecting the instrument also from accidental errors of an operator working with the engineering graphical user interfaces (GUIs).

The anti-collision system is driven by the Observation Control Software and relies on the position of the calibration fiber deployer; for this reason, at the startup of the Observation Software, such deployer is always homed.

5.3 NCPA Rotation

Each SHARK-NIR observation estimates and corrects the NCPA. Due to the structure of the instrument, it is necessary to distinguish between two types of NCPA: internal-static NCPA and external-rotating NCPA.

All SHARK-NIR devices excepted to inbeam_dep and inbeam_tt are mounted on a rotating bearing, to allow field-stabilized observations. Therefore, they are integral to the field rotation. From such path is retrieved the internal-static NCPA. On the other hand, the portion of the path that ranges from the telescope primary mirror to the instrument shutter is not involved in the bearing rotation and it does not follow the field rotation. These are called the external-rotating NCPA and result in an aberration pattern that depends on the derotator angle.

These two complementary effects are isolated, modeled, and corrected.

Internal-static NCPAs, which are integral with the rotation and are computed with the internal light source, are loaded on the DM as a static shape, retrieved after a focal plane wavefront sensing technique called phase diversity.[25] By applying a static shape, these NCPAs are corrected.

External-rotating NCPAs are corrected by uploading other modes on the DM, on top of the static shape. These modes are retrieved by an optimization procedure based on a "trial and error" approach, performed on every observation's target, and saved as a different shape in the form of a vector of positions. This shape is software-rotated every second based on the position of the bearing (i.e., of the whole instrument) and suddenly sent to the BCU through a `tiptilt_ctrl` command, which adds it to the calculated tip–tilt DM shape it creates internally, to compensate for the derotation effect and fine-optimize the optical quality.

The whole DM shape is then maintained as a superposition of internal-static and external-rotating NCPA. The RTC architecture reserves the first two modes for a fast tip–tilt loop; therefore, once the DM shape is defined, it is maintained even when the internal tip–tilt loop is operating.

In the current "early science" phase, internal-static and external-rotating NCPAs are calculated before each observation to have a consistent statistics about the conditions of temperature and telescope position in which they are modeled. Based on this information, it is foreseen to set up the best strategy to minimize the time loss due to this optical quality calibration procedure.

5.4 Data Manager

The data manager, called data_mgr, is a class of `obs_ctrl` responsible for scheduling the acquisitions of the SCICAM, retrieving the data, and creating FITS file, using cfitsio, reporting on the FITS header all the information of the instrument status. For what concerns header merging, the FITS file for the observation produced by the scientific camera is received by `obs_ctrl` through the INDI interface. This is implemented by listening to the INDI TCP port, receiving the file as a "blob" of raw bytes and writing them to disk; in this process, it is not

possible to add new FITS keywords to the header; thus, a second blob is created by `obs_ctrl`, with no data and with a header populated by all the required FITS keywords. The data_mgr component receives this last file and the FITS file produced by the scientific camera. Moreover, this component merges the scientific camera header in the full header produced by `obs_ctrl` as well as copying the data, using cfitsio library APIs. Data are then copied to the FITS Archive system in Tucson via rsync through a Network File System remotely mounted directory and from there to the Italian Center for Astronomical Archive (IA2) archive in Trieste,[26] Italy. Dedicated cron jobs on the instrument workstation sharknirws and on the scientific detector workstation sashaws provide a rotation of the files to avoid disk saturation, over a timespan period of 1 year.

6 Sequencer, Templates, and Observation Blocks

SHARK-NIR is operated using observing blocks (OBs), the same way this is done with instruments hosted in modern astronomical observatories such as the VLT.

The ICE Framework allows to expose as Python modules the functions of `obs_ctrl`. Then, on a higher layer, we developed a sequencer (seq), a Phyton-based application that acts as a client to `obs_ctrl` and allows the execution of the OBs.

OBs are logical blocks, stored as XML files; the content specifies the sequence for automated operations, called templates, and the parameters to be passed to them. Each template implements an observation, calibration, or maintenance procedure. It is composed of a reference file, that includes a short description of the template and all the parameters that are fixed during the execution of the template; a signature file, describing all the parameters the user can specify; and the template script itself, which is the actual implementation of the procedure (see rounded boxes in Fig. 3).

Templates scripts are implemented as Python scripts accepting five input arguments in the form of a Python dictionary, an unordered list of labels (representing the parameters) and values. Each dictionary specifies the setups as follows: instrument setup, detector setup, telescope setup, real-time tip/tilt subsystem setup, and observation/program info setup.

When an OB is loaded, seq process operates as follows: reads the specified XML files and extract an ordered list of templates to execute; for each template, and the corresponding list of parameters is then checked against its scheme. The resulting Python dictionaries are merged with the ones containing the fixed parameters to obtain a complete setup. Finally, the OB is executed.

In general, templates start from an initial instrument setup, followed by one or more science or technical images, which at each loop or phase of the script performs checks, moves the instrument to a different setup, or alternates a set of operations. Error handling during the execution of the template is managed at the template and sequencer levels by separate Python decorators.

At a lower level, if an error occurs, such as the reaching of a limit switch, or an unavailability of a service, the error is propagated to the sequencer and shown to the final user.

At a higher level, the status is checked at any relevant operation. As templates are complex operations involving the whole instrument and the telescope, the template is aborted if an error occurs and no recovery or error clearance is foreseen in case of failure: a talking error message is displayed in the template log so that the operator can verify it, and relaunch the OB.

During execution, seq calls the list of Python scripts in the order specified by the OB, passing the corresponding dictionary objects as a parameter. Figure 4 shows an example of OB calling a single template script.

7 REST API

OBs can be in principle prepared with any text editor, whereas the sequencer can be executed as a standalone script in the instrument workstation to launch them.

To ease the OBs preparation and execution and, in general, to easily use SHARK-NIR, we first developed a set of Representational State Transfer Application Programming Interface (REST API)[27] using the common HTTP verbs (get to retrieve a resource, put to modify it, post

```
<?xml version="1.0"?>
<ObservationBlock       xsi:schemaLocation="ObservationBlock.xsd"
    OBID="TestID"       ProgramID="TestProgID"
    PIName="El Condor"  PIID="TestPIID">

    <Template>
    <TPLID>TestForJatis</TPLID>
    <TPLName>Test for JATIS</TPLName>

    <InstrumentSetup>
      <SCI_FILT_W1>HOLE</SCI_FILT_W1>
      <SCI_FILT_W2>BB_H</SCI_FILT_W2>
    </InstrumentSetup>

    <DetectorSetup>
      <DIT>1</DIT>
      <NDIT>1</NDIT>
    </DetectorSetup>

    </Template>

    <Template>
    ... an additional template in the OB ...
    </Template>

</ObservationBlock>
```

```
from shinsapp import SHINSApp
from util import exception
from util.shinsLogger import logger

class TestForJatis(SHINSApp):

    @exception.handler_for_run
    def run(self, ins_s=None, det_s=None, tel_s=None, rtc_s=None):
        logger.info(__name__)
        obsController = self.obsController

        # Async instrument and detector setup
        async_ins = obsController.begin_setupInstrument(ins_s)
        async_det = obsController.begin_sashaSetup(det_s)

        # Waiting Instrument and detector setup
        self.status = obsController.end_setupInstrument(async_ins)
        self.status = obsController.end_sashaSetup(async_det)

        logger.info("Instrument and detector setup... completed")

        # Sync exposure
        self.status, self.filenames = obsController.sashaExpose()
        logger.info("FITS files are "+str(self.filenames))
```

Fig. 4 Examples of OB in XML format (top) and setting parameters for the corresponding template script in Python language (bottom).

to create a new resource or start an operation, and delete to delete a resource or stop an operation). The Python package Flask RestX was used to build the API to provide automatic documentation generated by the Swagger module,[28] a set of tools that simplify API development and documentation for users, teams, and enterprises.

With these APIs, it is possible to create, edit, delete, and launch OBs; check the value of the fixed parameters of a given template script; monitor the execution of the OBs; directly interact with every single motor device; know its state and position; interact with the RTC; and control the scientific camera.

8 Web-Based Graphical User Interfaces

On top of the REST APIs, we developed a Flask web server and several GUI panels using the Bootstrap five framework.

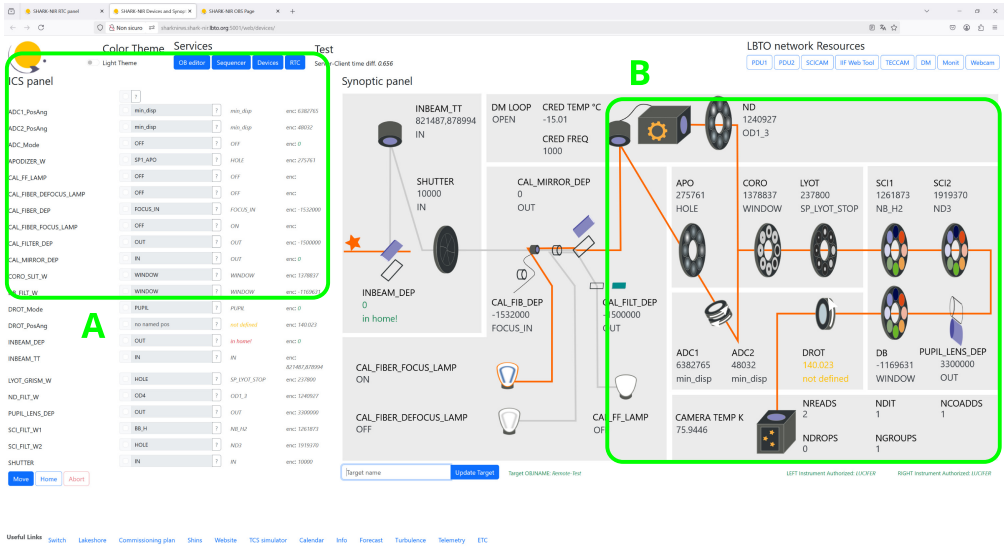

Fig. 5 Devices and synoptic panel overview. See "A" (Fig. 8) and "B" (Fig. 9) for details.

Each panel is a web page. It is possible to open panels in separate browser tabs, arrange them in separate browser windows, or a combination of both, as well as resize them with a reasonable responsiveness.

Web pages use the REST API to trigger commands and the Socket.IO websocket library for the constant communication of the SHINS status: when the web app is operative, a client connection to `obs_ctrl` is created and periodically requests for status information regarding the whole instrument.

A server class that stores status information in its attributes. A single process asks the status of the instrument to `obs_ctrl` and updates the class attributes. Web clients periodically receive the values of these attributes using the websocket connection.

This "REST for control + websocket for status" approach has been successfully tested[29,30] in an 80-cm size telescope, and best practices have been applied to SHARK-NIR.

In Sec. 8.1–8.4, we present the devices and synoptic panel of the instrument (Fig. 5), providing quick look and control of individual motors and lamps, and the RTC engineering panel (Fig. 6), allowing quick look and deep control of the TECCAM, the DM, and the RTC low-level functions. Then, after describing a separate, unreleased "wizard" tool for the preparation of the OBs, we present the general-purpose, "user" observation panel (Fig. 7), which was developed following the suggestions of the scientific team to provide the SHARK-NIR operator with a minimum but high-level set of functions and information. This panel also embeds a local version of the wizard, so that OBs can also be easily generated on the fly while operating at the telescope.

A typical flow of scientific observation of SHARK-NIR is then the following: the observations are prepared using the Wizard OB editor, then the folder with the OBs is uploaded on the instrument workstation. At the telescope, the user is expected to operate on the observation panel to load and execute the OBs, keeping an eye on the synoptic panel if necessary, and opening the RTC panel only for engineering purposes or debug.

8.1 Devices and Synoptic Panel

Although the TaN framework provides a low-level, engineering GUI developed in Qt, for each one of the `motion_ctrl`, `calunit_ctrl`, and `track_ctrl` services, we decided instead to develop a "global" instrument control panel for SHARK-NIR, which we call device and synoptic panel (see Fig. 5). We chose to go on with the APIs approach, exposing HTTP methods to get and manage the status of devices. This means that, even if this is mainly an engineering tool, these controls pass through `obs_ctrl` instead of operating at the subsystem level.

The left part of the panel (see a zoomed view in Fig. 8), labeled as "devices panel," is used to control and monitor the status of individual motors. Bulk operations on multiple motors can be performed by selecting the corresponding checkboxes before clicking on the "move" button.

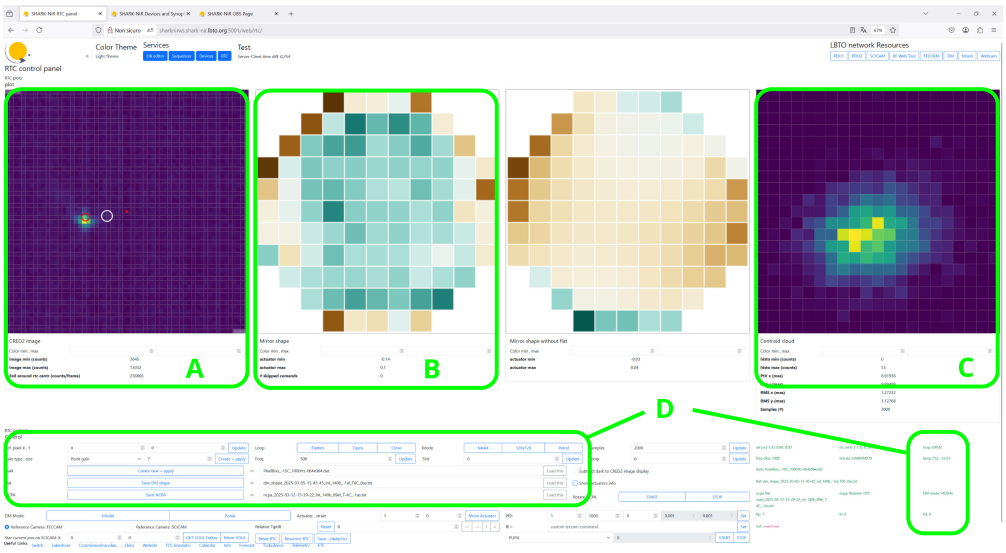

Fig. 6 RTC engineering panel overview. See "A" (Fig. 10), "B" (Fig. 11), "C" (Fig. 12), and "D" (Fig. 13) for details.

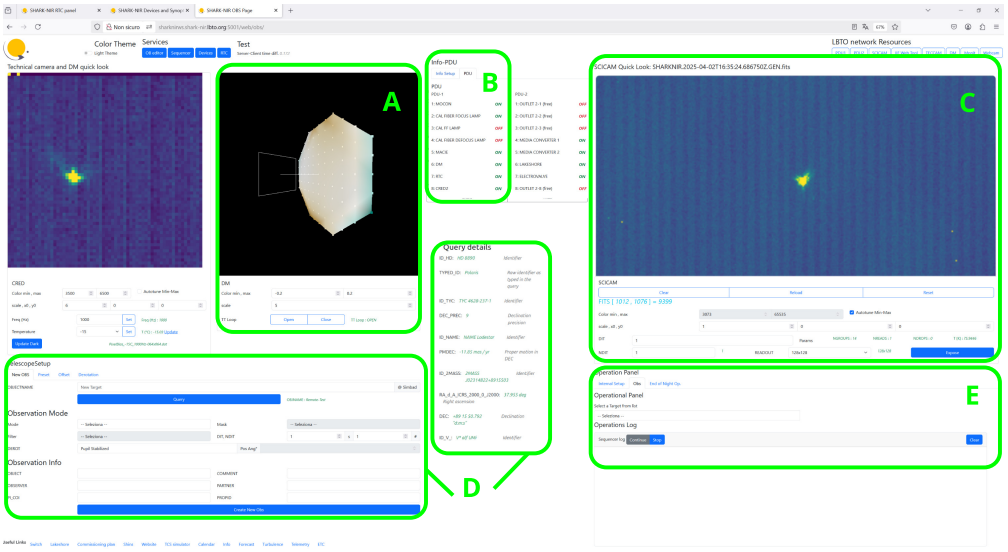

Fig. 7 Observation panel overview. Details in "A" (Fig. 15), "B" (Fig. 16), "C" (Fig. 17), "D" (Fig. 18), and "E" (Fig. 19).

An additional checkbox on top of the list allows selecting/deselecting all devices with one click. Named position and position in a unit of encoder steps are also shown. They are shown in green if in the home position, red if in error (for example, when in contact with the limit switches), yellow if moving or undefined, and otherwise in gray.

The right part of the panel (see a zoomed view in Fig. 9), labeled as "synoptic panel," shows a graphical representation of the instrument's hardware devices, similar to that shown in Fig. 1. This helps the operator to have a quick view of the setup. The synoptic panel does not reflect the real spatial displacement of the components; instead, it shows the devices as they are crossed by the optical path. The optical path changes its color based on the setup to enhance whether the light source is coming from a specific calibration lamp or from a source in the sky: gray if not traveled by the light, otherwise light orange, and makes this path easier to understand. Readout parameters of the SCICAM are also shown.

Moreover, the synoptic panel provides a base level of animation for some devices; for instance, the shutter image can be opened or closed, and the deployers' position changes based on the setup, and the lamps can be on or off. Text fields show the motors' position (expressed in

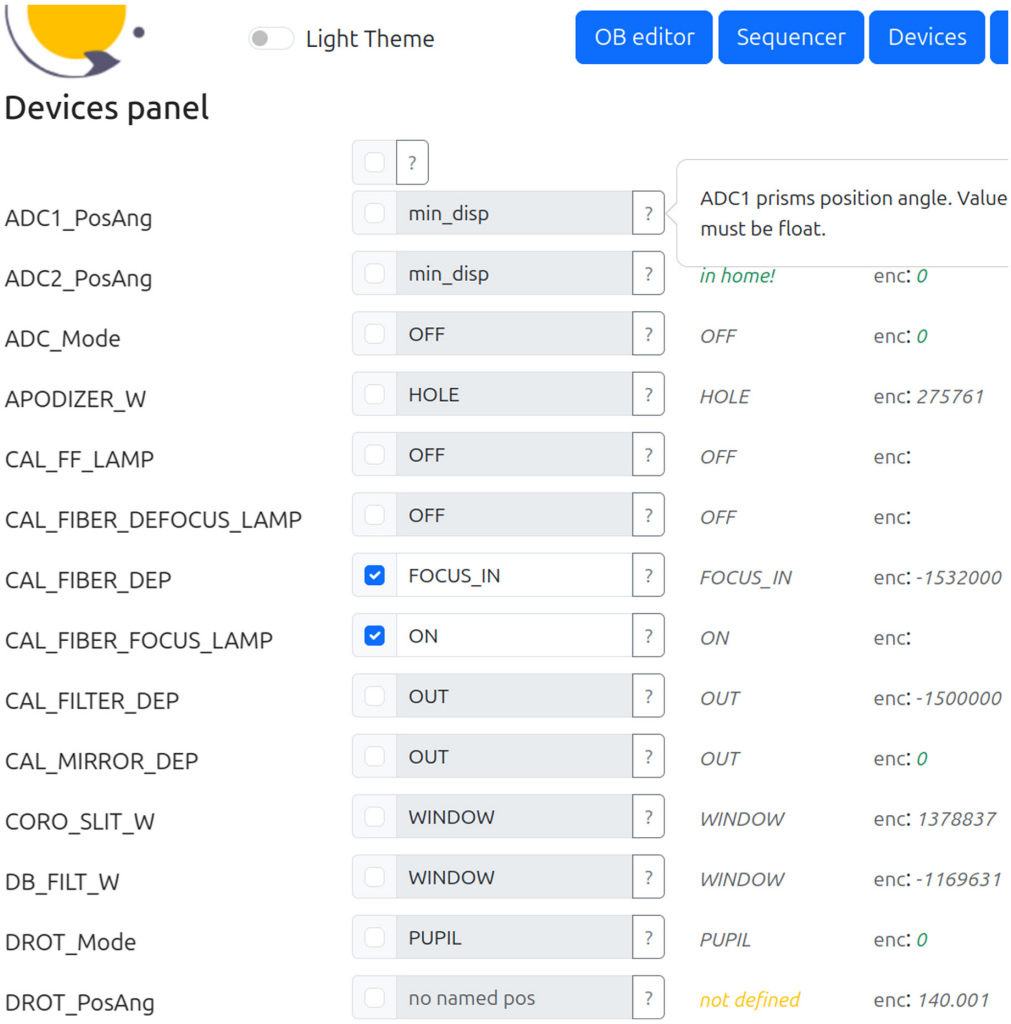

Fig. 8 Zoom on the devices panel (see Fig. 5). Two devices are selected to be moved together. The first ADC shows the help tooltip.

units of encoder steps) and their named position, if applicable, following the same color schema of the device section.

8.2 RTC Panel

The RTC panel (see Fig. 6) allows monitoring and fine control of the wavefront control system of the instrument, and it was specifically designed to support the operations during commissioning. The RTC panel was initially designed for quick look and simple operations only, but as the assembly, integration, commissioning, and early science activities proceeded, more and more additions became necessary for testing, shaping it to a comprehensive engineering panel for the fine-tuning and the debug of the SHARK-NIR Adaptive Optics subsystem. The panel is divided into two main areas: the plot area on top and the control area on bottom.

The plot area is based on a websocket communication established directly between the upper Python-based level and the RTC. The communication is separately managed by the socketio python (server-side) and js (client-side) package, which implements a robust websocket system, periodically updating an internal attribute with the diagnostic information received by the RTC (reconstruction of the image on the TECCAM, measured centroid, and DM shape) by a separate python thread, called "Python RTC listener" in Fig. 3, which was adapted from an existing Python script provided by Microgate. This allows us to reduce the request load on `obs_ctrl`.

Figure 6 shows four plots, each one representing fundamental diagnostic information.

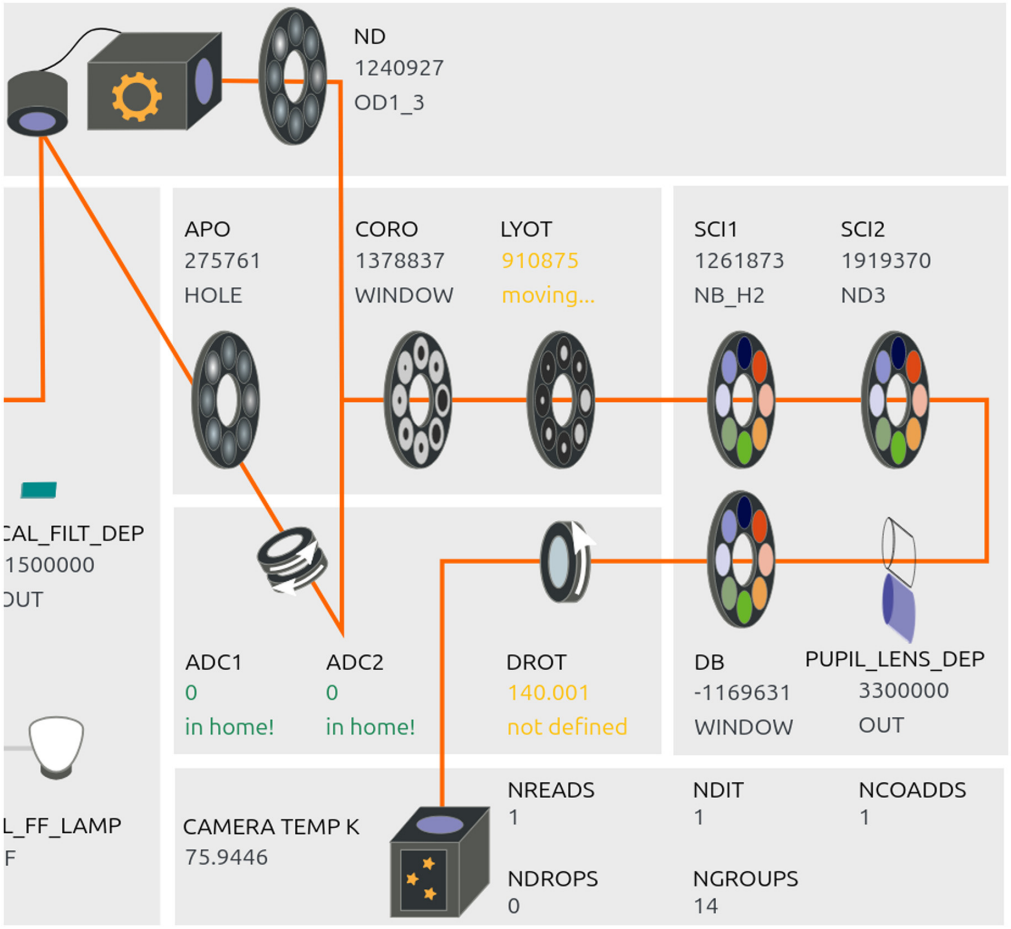

Fig. 9 Zoom on the synoptic view (see Fig. 5). The orange line shows the optical path receiving light.

8.2.1 *C-RED2 image*

A TECCAM plot (see Fig. 10), a representation of the active region of the technical camera, lets the instrument operator always check the position of the PSF and monitor the tracking status of the loop. This plot is also shown as a white circle the point corresponding to the center of the coronagraphic mask. Moreover, a red circle shows the point that the RTC calculates as the PSF centroid. Finally, a red dot shows the point where the RTC moves the PSF while operating in a closed loop. In Fig. 10, the loop is open, and the PSF is still not placed behind the mask.

8.2.2 *Mirror shape*

A graphical visualization of the stroke value of the 97 actuators of the DM (see Fig. 11) is drawn as a set of squares in their corresponding position behind the mirror. The level of stroke of each actuator is represented using a color scale that ranges from the min to max value of the current configuration.

8.2.3 *Mirror shape without flat*

A second reproduction of the actuator, similar to the previous one, but without taking into account the internal-static NCPA (see Sec. 5.3), that is called "flat" in the RTC panel. This third plot is mainly used for the definition and optimization of the external-rotating NCPA that are called simply "NCPA" in the panel.

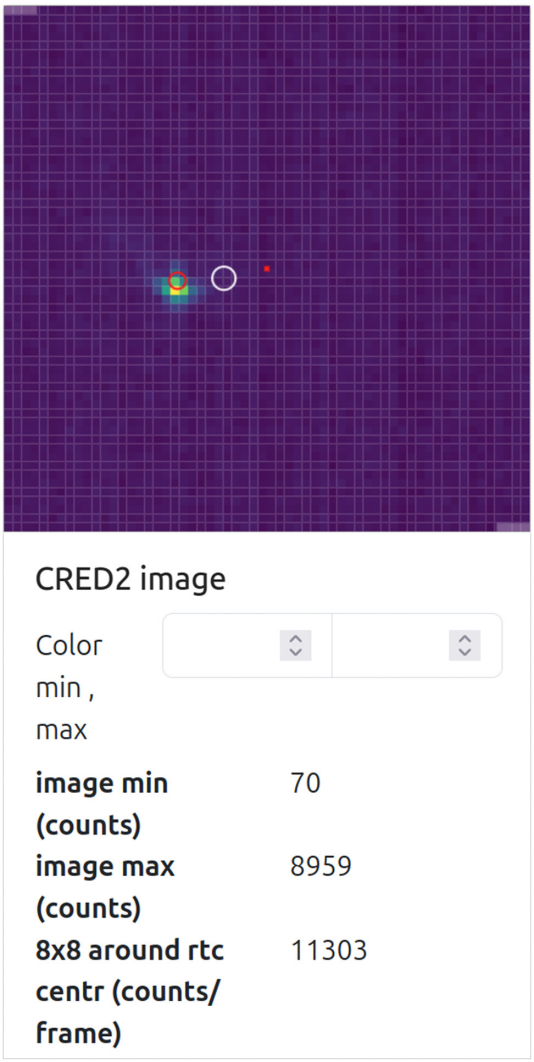

Fig. 10 Zoom on the CRED image (see Fig. 6).

A statistical representation of the centroid position in the last N TECCAM images (see Fig. 12), helping to monitor internal loop performances and residual jitter.

Below each plot, a custom color range can be selected, and the values are updated on the plots at every refresh of the information. Basic statistical information is also shown and continuously updated.

The control area, on the other hand, allows to sending commands to `tiptilt_ctrl` through `obs_ctrl` via REST API to the RTC. Figure 13 shows a selection of the main control operations, including setting TECCAM parameters (as frequency and integration time), defining the gain mask (a sub-area on the TECCAM to optimize the computation of the PSF centroid for close loop operations), and loading the dark background of the TECCAM and the NCPA.

The RTC calculates the PSF centroid on a 64-px subaperture of the TECCAM by default. This region is used while operating the DM in a closed loop. Moreover, for test purposes, the TECCAM can also operate full-frame as a "patrol camera," without the possibility to operate for tip-tilt correction. TECCAM mode buttons allow you to switch between tip–tilt correction mode and patrol camera mode.

Other commands allow to open/close the internal fast tip–tilt loop, set the reference position for the PSF in closed loop, and command single relative tip–tilt commands to the DM. Moreover, the control panel also allows to change the internal interaction matrix of the RTC, being able to switch between modal mode to zonal mode, with the latter that allows to singly command the actuators, and to change the PID of the DM.

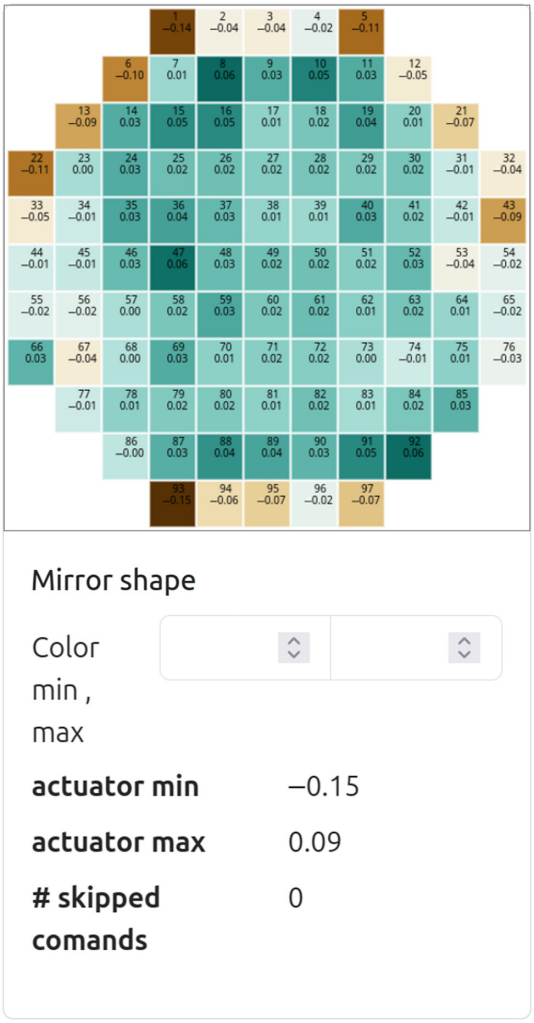

Fig. 11 Zoom on the mirror shape (see Fig. 6).

The right side of the control area (see an example on the right of Fig. 13, in green) collects all the status information, collected by websocket from `obs_ctrl` and updated every 0.5 s.

8.3 Wizard Editor

At the end of the commissioning phase, we brainstormed all the technologic and scientific team to condensate the typical use of SHARK-NIR into an "operation flow." This lead to the creation of two tools: the first, presented in this section, is an external "wizard" editor (see Fig. 14), allowing astronomers to prepare in advance, and in separate servers with respect to the instrument workstation (i.e., currently in an external server accessible by the SHARK-NIR team only), a complete set of OBs based on the instrument mode and a minimal set of information, including of course the scientific target.

First, the target coordinates can be retrieved by a preset query that searches the object on the Simbad[31] catalog. A further software development will also handle non-sidereal targets, which are currently handled directly at LBT by the telescope operator.

The query automatically populates the target coordinates. The user completes this section by adding the additional LBT pointing flags, including the use of secondary mirror's adaptive optics, the telescope mode (i.e., which LBT side is leading the observation), and deotator mode, as well as the flags controlling the binocular mode.

Then, the user fills the rest of the form, choosing the SHARK-NIR instrument mode (coronagraphy, direct imaging, or long-slit spectroscopy). Depending on the chosen mode, then a set of coronagraphic masks, no masks, or a set of spectroscopy masks are available. Depending on

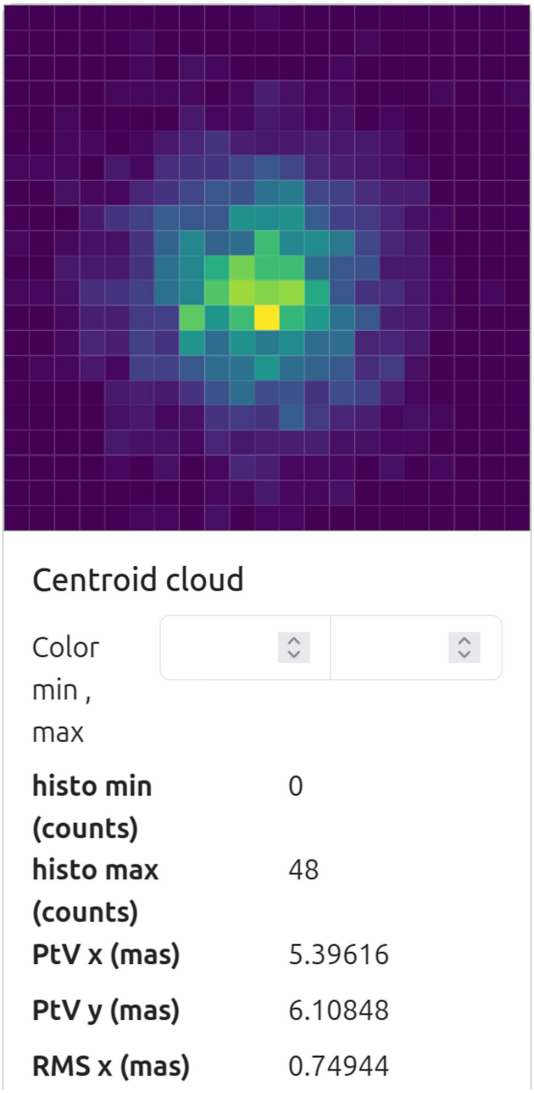

Fig. 12 Zoom on centroid cloud (see Fig. 6).

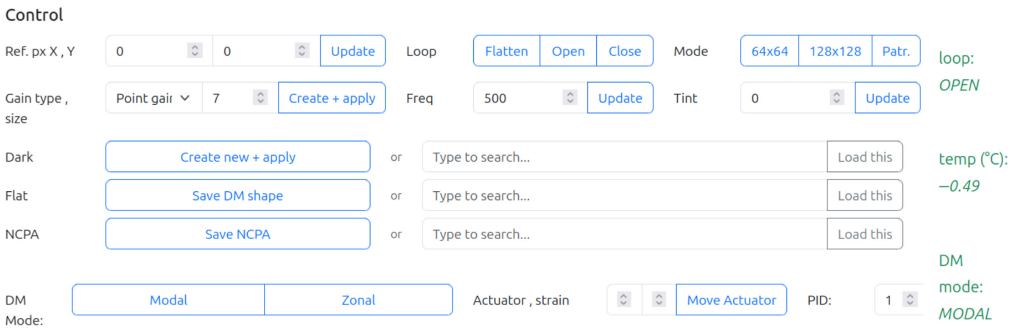

Fig. 13 Zoom on the control area (left) and on the status information (right, green) of the RTC panel (see Fig. 6).

both the observation mode and the mask, available combination of filters or no filters are available. The last information is the derotation type, which can be pupil-stabilized (no instrument derotation), a field-stabilized mode that maximizes the derotation angle, and a field-stabilized mode at a custom position angle. Finally, typical observation information (such as proposal ID and name of the principal investigator) is filled by the astronomer. By clicking on the button "save," a complete set of observation blocks based on the previous information are created and

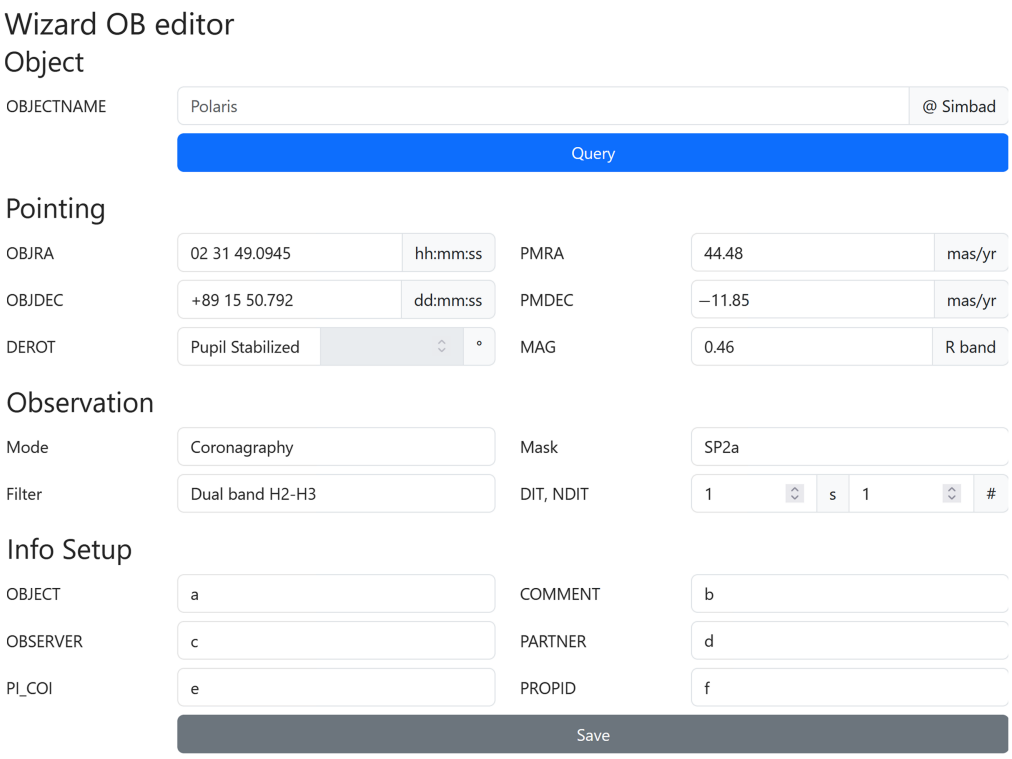

Fig. 14 Wizard OB editor, currently for internal use of the SHARK-NIR team. It produces a folder with a complete set of OBs in XML format, including the acquisition procedure, calibrations relative to the chosen configuration, and observation sequence.

saved on a new directory. These OBs include the target acquisition sequence, the alignment procedures corresponding to the chosen combination of instrument mode and mask, the calibration OBs, and of course the scientific observation sequence.

Currently, the wizard is for the internal use of the SHARK-NIR team, and the directories with the OBs are uploaded on sharknirws manually. It is foreseen to make this tool publicly available.

8.4 Observation Panel

The second tool developed after the brainstorming, presented in this section, is the SHARK-NIR "observation panel," containing minimal information to control and monitor the instrument at the telescope, and to operate basic troubleshooting. This step reduced the degrees of freedom given by the engineering panels and by the parameters of the template scripts. This allowed developing a high-level interface mainly for scientific use. The panel also embeds a slightly modified version of the "wizard" editor, to create on-the-fly an observation directly from the panel, if necessary, while operating at LBT.

We describe in detail this panel as an effort to make this paper useful also as a short version of the user manual of the instrument.

The top part of the panel (Fig. 7) shows the following.

8.4.1 *Technical camera quick look*

A WebGL image, and its respective color range and pan-zoom controls, in a way similar to what provided by the RTC panel. It allows tuning the temperature, the frequency, and updating the dark, which is subtracted to improve the PSF centroid signal-to-noise ratio.

8.4.2 *DM quick look*

For a WebGL 3D image and its respective color range, see Fig. 15. In this case, scale controls are used to enhance the DM stroke. The user can open and close the tip–tilt loop as a quick debug feature.

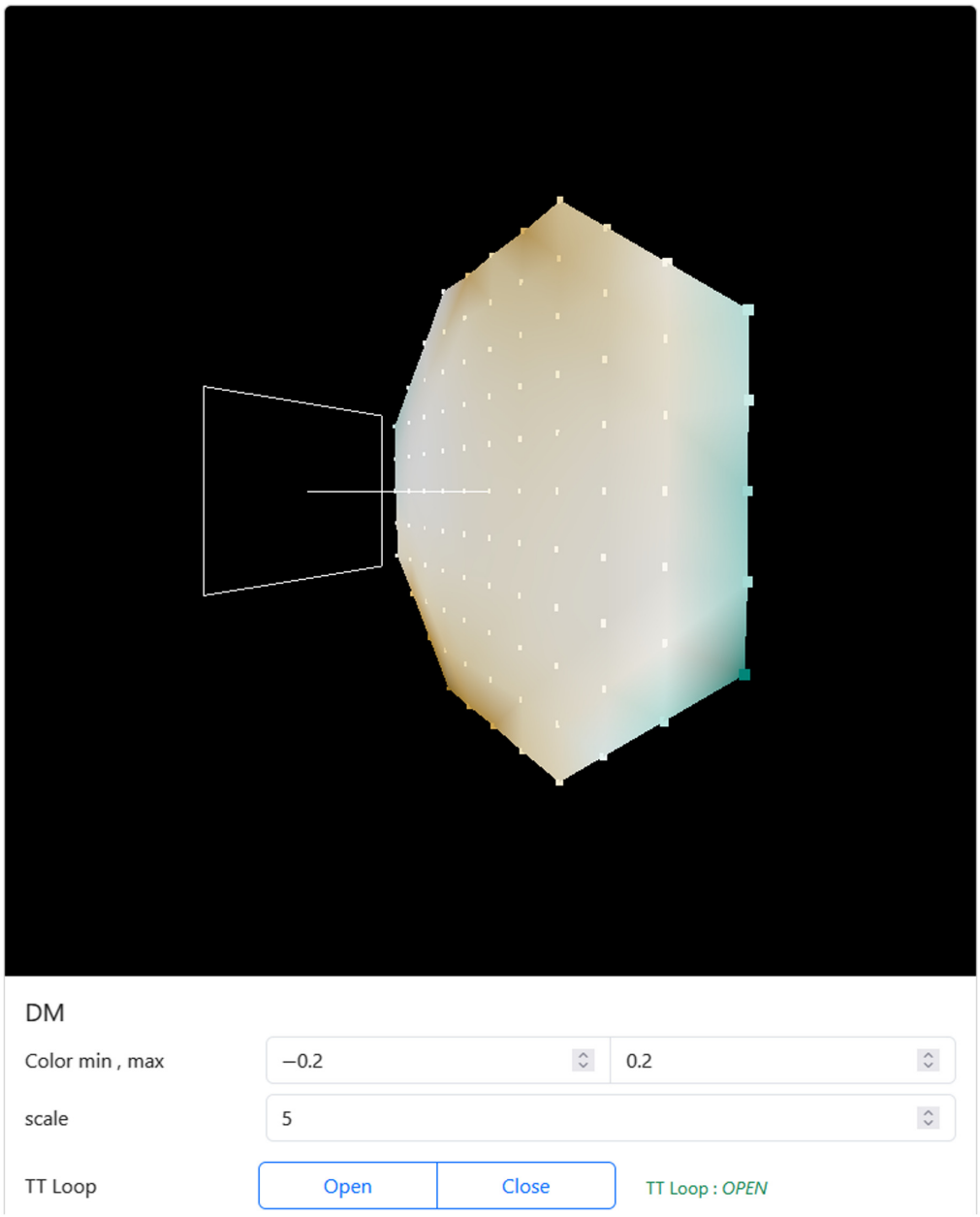

Fig. 15 Zoom on the 3D DM quick look section in the observation panel (see Fig. 7), with commands to open and close the tip–tilt loop.

8.4.3 *Info-PDU*

It provides two tabs (see Fig. 16). The first shows the complimentary setup information; the second allows bare monitoring of the state of the PDU outlets that control the lamp devices described in Sec. 2, as well as other outlets powering the instrument.

8.4.4 *SCICAM quick look*

It shows the latest image taken with the scientific camera, and its respective color range and pan-zoom controls (see Fig. 17). It also allows exposing test snapshot images by specifying the detector integration time, the number of sub-integrations, and the readout region from a pre-defined list of regions of interest. Pan and zoom on the images are also available by mouse drag and wheel events. As in the other panels, the status information in green is retrieved via websocket from `obs_ctrl`.

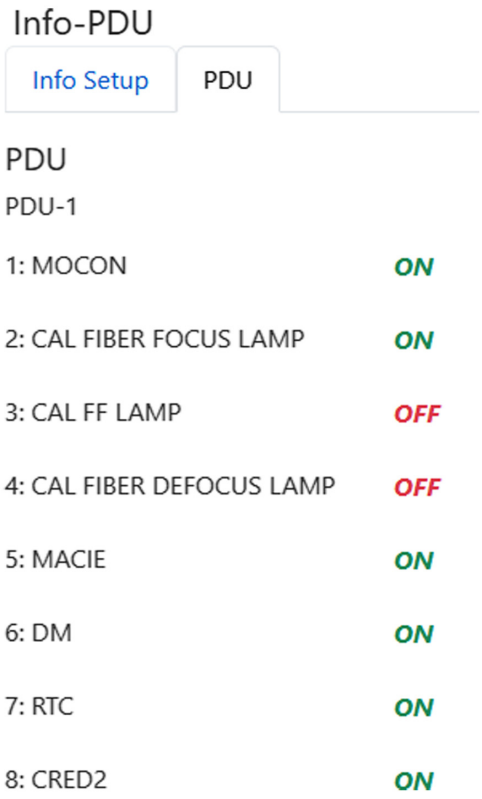

Fig. 16 Zoom of the central tabs in the observation panel (see Fig. 7). Info setup (in background), allowing to edit the complimentary OB information, and read-only PDU information (in foreground).

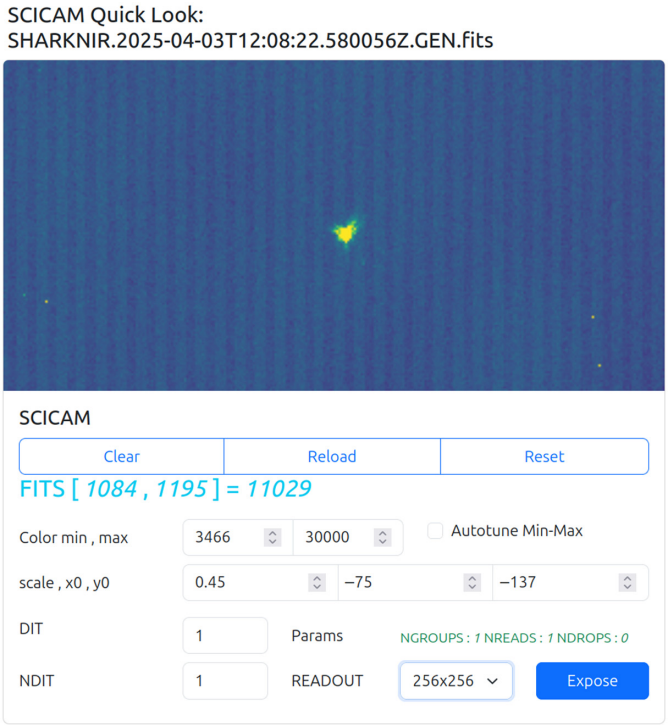

Fig. 17 Zoom on the SCICAM control section of the observation panel (see Fig. 7).

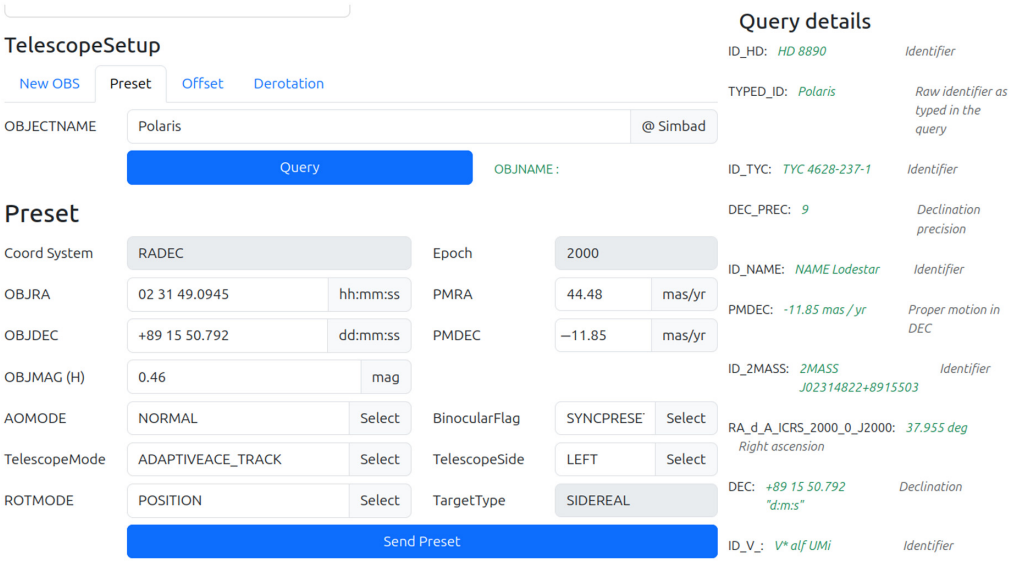

Fig. 18 Zoom on the lower tabs section of the observation panel (see Fig. 7). On the right, the details of the queried object, retrieved from Simbad.

As in the RTC engineering panel, this top part of the observation panel is dedicated to monitor and quick tests. The bottom part of the panel shows the following.

8.4.5 *Local wizard and telescope control tabs*

Figure 18 background "new OBS" tab hides a local replica of the wizard, that is used to create on-the-fly a set of OBs, if necessary. It also populates the foreground "preset" tab that allows modifying the coordinates and the telescope observation mode. This tab is also used independently to communicate with the telescope, if necessary. The send preset button allows to actually point the telescope. Other tabs allow fine-tuning the telescope offset and or change the SHARK-NIR derotation mode.

8.4.6 *Operation section*

Is the top-level control of SHARK-NIR. The user retrieves from a dropdown menu the OB directories previously generated by the wizard OB editor and uploaded on the instrument workstation or generated on the fly from the local wizard (2025-03-18_Polaris_DI in Fig. 19). The OBs are loaded in a numbered, button-based sequencer, so that the operator can run them one by one, or repeat a given step changing common parameters if necessary. A tooltip shows the template help. Figure 19 also shows the "internal setup" and "end of night op." tabs, containing OBs with leading and trailing common operations.

Operations logs are shown in a box that reads the instrument log files and updates it each 0.5 s.

If there is a need to access less common parameters for peculiar tests or conditions, then a separate panel called "OB editor," that it is not described here, but is available by clicking the "edit" button of each operation step. Developed for the AIT phase is thought of as a simplified version of the ESO P2 system and also shows the reference setup in disabled boxes.

9 Scientific Results

In its early science phase, SHARK-NIR obtained its first scientific high-contrast image results in the H band, using the Gaussian coronagraphic mask, which is one of the installed coronagraphs together with shaped pupil masks and a four-quadrant phase mask.[32] Observations, obtained in pupil-tabilized mode, have been combined with observations with LBTI/LMIRCam in L', all involved instruments operating in parallel at LBT.

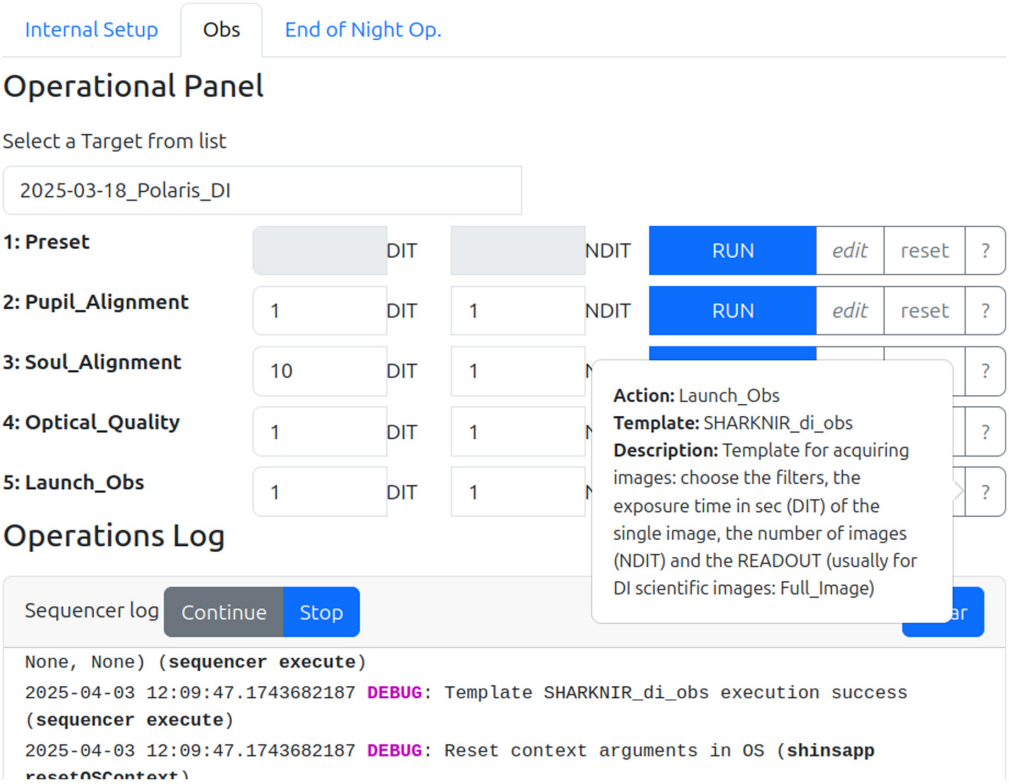

Fig. 19 Zoom on the operational tabs of the observation panel (see Fig. 7) and the operations log. See Sec. 8.4 for details.

The first result[14] is the contribution to the characterization of an eccentric giant planet detected around the star HD 57625. LBT observations have been combined with available SOPHIE radial velocities and Hipparcos-Gaia Proper Motion Anomaly (PMa) measurements, finding the following parameters: a $8.43^{+1.1}_{-0.91} M_{\rm Jup}$ planetary companion mass, an orbital separation of $5.70^{+0.14}_{-0.13}$au and an eccentricity of $0.52^{+0.04}_{-0.03}$.

The second result[15] is a deep imaging study of three accelerating stars: HIP11696, for which the analysis suggests the presence of a companion with a mass between 4 and $16 M_{\rm Jup}$. at a 2.5- to 28-au separation; HIP47110, for which the study finds a 2 to $10 M_{\rm Jup}$ companion at 3 to 30 au; and finally, HIP36277, suggesting second candidate companions in addition to the first, detected by SHARK-NIR observations (see Fig. 20) and already confirmed by Gaia.

10 Conclusion

We presented SHINS, the Instrument Control Software of SHARK-NIR, a new instrument at LBT for high contrast imaging.

SHINS is principally based on the TaN framework to control most of its subsystems while exploiting SNMP and INDI protocols to control the scientific camera and the calibration lamps, respectively. Its central component, the Observation Control Software, is developed on the ICE framework, all being based on C++. The central components also implement safety procedures and NCPA correction management, which is one of the most critical tasks of the wavefront control system of the instrument.

High-level components are developed in Python, and include a sequencer for template scripts called by XML OBs. Even if the sequencer can be called by terminal, SHINS users can operate thanks to a set of web-based GUIs for observation and engineering purposes that make use of REST API for control and websockets for monitoring. WebGL is used for quick look of scientific and technical images, as well as to monitor the shape of the internal DM.

Users can easily plan their operations with a "wizard" tool that helps the configuration of OBs for all steps of the observation.

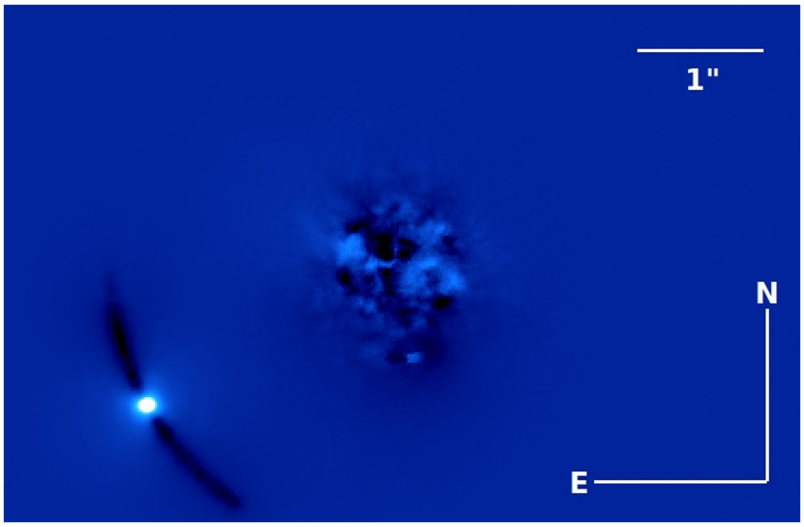

Fig. 20 Final image of HIP36277 observed on February 21, 2024, with SHARK-NIR using the Gaussian coronagraph and the broad H-band filter (BB_H). The total rotation of the field of view was 33.6 deg, whereas the total exposure time on the target was 5660.5 s with single detector integration times (DITs) of 61.5 s. A bright stellar companion is visible south-east from the star, whereas a fainter candidate companion is visible at a shorter separation (0.625 in.) from the star.

Early science tests demonstrated that SHINS allows to operate SHARK-NIR exploiting its full potential, allows significant results in the exoplanetary field, and is ready for regular operations.

Disclosures

The authors declare there are no financial interests, commercial affiliations, or other potential conflicts of interest that have influenced the objectivity of this research or the writing of this paper.

Code and Data Availability

Data sharing is not applicable to this article.

Acknowlegments

We thank Tom Herbst from MPIA-Heidelberg, and the LINC-NIRVANA team, for sharing part of their instrument control SW to operate the motorized axis of SHARK-NIR. We also express our appreciation to NASA and Marcia Rieke, the principal investigator of JWST/NIRCam, for granting us the opportunity to utilize one of the NIRCam spare detectors as the primary detector for the SHARK-NIR scientific camera. Observations have benefited from the use of ALTA Center (alta.arcetri.inaf.it) forecasts performed with the Astro-Meso-Nh model. Initialization data of the ALTA automatic forecast system come from the General Circulation Model (HRES) of the European Centre for Medium Range Weather Forecasts. DR acknowledges the funding from "MINI-GRANTS (2023) di RSN5" (Grant No. C93C23008400001). The LBT is an international collaboration among institutions in the United States, Italy, and Germany. The LBT Corporation partners are as follows: the University of Arizona on behalf of the Arizona University system; Istituto Nazionale di Astrofisica, Italy; LBT Beteiligungsgesellschaft, Germany, representing the Max Planck Society, the Astrophysical Institute Potsdam, and Heidelberg University; the Ohio State University; and the Research Corporation, on behalf of The University of Notre Dame, University of Minnesota and University of Virginia.

References

1. F. Pedichini et al., "The V-SHARK high contrast imager at LBT," *Proc. SPIE* **9908**, 990832 (2016).
2. V. Viotto et al., "SHARK-NIR system design analysis overview," *Proc. SPIE* **9911**, 991127 (2016).
3. J. Farinato et al., "SHARK-NIR: from K-band to a key instrument, a status update," *Proc. SPIE* **9909**, 990931 (2016).

4. J. Farinato et al., "The NIR arm of SHARK: system for coronagraphy with high-order adaptive optics from R to K bands," *Int. J. Astrobiol.* **14**, 365–373 (2015).
5. E. Carolo et al., "SHARK-NIR coronagraphic simulations: performance dependence on the Strehl ratio," *Proc. SPIE* **10701**, 107012B (2018).
6. E. Pinna et al., "SOUL: the single conjugated adaptive optics upgrade for LBT," *Proc. SPIE* **9909**, 99093V (2016).
7. P. Cerpelloni et al., "Unravelling the performance of the SHARK-NIR Four Quadrant Phase Mask in a controlled environment," *Proc. SPIE* **13096**, 130963T (2024).
8. D. Barbato et al., "SHARK-NIR commissioning and early science runs," *Proc. SPIE* **13096**, 130961W (2024).
9. M. Bergomi et al., "SHARK-NIR: from design to installation, ready to dive into first light," *Proc. SPIE* **12187**, 1218709 (2022).
10. D. Vassallo et al., "Laboratory demonstration of focal plane wavefront sensing using phase diversity: a way to tackle the problem of NCPA in SHARK-NIR: Part II: new characterization tests and alternative wavefront sensing strategies," *Proc. SPIE* **12185**, 121858I (2022).
11. G. Umbriaco et al., "Deformable lens for testing the performance of focal plane wavefront sensing using phase diversity," *Proc. SPIE* **12185**, 121856W (2022).
12. J. Farinato et al., "SHARK-NIR, ready to "swim" in the LBT Northern Hemisphere "ocean"," *Proc. SPIE* **12185**, 1218522 (2022).
13. L. Marafatto et al., "SHARK-NIR on its way to LBT," *Proc. SPIE* **12184**, 121843V (2022).
14. D. Barbato et al., "A multi-technique detection of an eccentric giant planet around the accelerating star HD 57625," *Astron. Astrophys.* **693**, A81 (2025).
15. D. Mesa et al., "Deep imaging of three accelerating stars using SHARK-NIR and LMIRCam at LBT," *Mon. Not. R. Astron. Soc.* **536**, 1455–1466 (2025).
16. A. Lorenzetto et al., "SHARK-NIR instrument control software: new features and tools approaching science runs," *Proc. SPIE* **13101**, 131013S (2024).
17. D. Ricci et al., "Improvements to SHINS, the SHARK-NIR instrument software, during the AIT phase," *Proc. SPIE* **12189**, 1218920 (2022).
18. M. De Pascale et al., "SHARK-NIR: implementation of the instrument control software SHINS," *Proc. SPIE* **11452**, 114521T (2020).
19. M. De Pascale et al., "Design of SHINS: the SHARK-NIR instrument control software," *Proc. SPIE* **10707**, 107071M (2018).
20. T. M. Herbst et al., "LINC-NIRVANA: the Fizeau interferometer for the Large Binocular Telescope," *Proc. SPIE* **7013**, 701326 (2008).
21. J. Berwein et al., "An SOA developer framework for astronomical instrument control software," *Proc. SPIE* **7019**, 70191T (2008).
22. S. Rabien et al., "ARGOS at the LBT. Binocular laser guided ground-layer adaptive optics," *Astron. Astrophys.* **621**, A4 (2019).
23. ZeroC company, "Complete solution for distributed systems," 2025, https://zeroc.com/ice.
24. J. M. Leisenring et al., "On-sky operations and performance of LMIRcam at the Large Binocular Telescope," *Proc. SPIE* **8446**, 84464F (2012).
25. D. Vassallo et al., "Validating the phase diversity approach for sensing NCPA in SHARK-NIR, the second-generation high-contrast imager for the Large Binocular Telescope," *Proc. SPIE* **10705**, 1070516 (2018).
26. Italian center for Astronomical Archive, "Home page," 2025, https://www.ia2.inaf.it/
27. R. T. Fielding and R. N. Taylor, "Architectural styles and the design of network-based software architectures," PhD thesis, University Of California, Irvine (2000).
28. SmartBear software, API Development forEveryone", 2025, https://swagger.io/
29. D. Ricci et al., "Easy remote observations using web interfaces: controlling an Italian telescope from Japan, and more," *Proc. SPIE* **13098**, 130980T (2024).
30. D. Ricci et al., "Toward the remotization and robotization of the OARPAF Telescope," *Proc. SPIE* **12186**, 121860P (2022).
31. CDS (Strasbourg), "SIMBAD Astronomical Database," 2025, http://simbad.u-strasbg.fr/simbad/.
32. D. Rouan et al., "The four-quadrant phase-mask coronagraph. I. Principle," *Publ. Astron. Soc. Pac.* **112**, 1479–1486 (2000).

Davide Ricci is a technologist at the INAF - Osservatorio astronomico di Padova. He is involved in Instrument Control Software of ESO and LBT projects, as well as in the development of small observatories for science and outreach purposes. His interests span from web user interfaces for telescopes, to extrasolar planets observations.

Biographies of the other authors are not available.