OOOS: A hardware-independent SLR control system

Daniel Hampf[*], Fabian Sproll, Thomas Hasenohr

German Aerospace Center, Institute of Technical Physics, Pfaffenwaldring 38-40, 70569 Stuttgart, Germany

Abstract

German Aerospace Center develops and tests novel technologies for both traditional SLR and space debris laser ranging. Currently, it operates an engineering ILRS station at Stuttgart Uhlandshöhe with a fibre-based transmitter. Research is focused on high repetition ranging (> 10 kHz) and the use of near-infrared lasers. Furthermore, a new container-based station called STAR-C has been set up near Stuttgart. It contains a coudé path and is developed mainly for space debris laser ranging.

To reduce the overhead, both stations are operated by the same software, called OOOS (orbital objects observation software). It is designed with the goal of utmost flexibility to be used in many different scenarios. A three layer approach with loose coupling between different modules ensures that any part of the software can easily be replaced with a new module if the need arises. The modules can be implemented in different programming languages, and run on different PCs in the same local network. Only the lowest level contains hardware-specific code, and new hardware can be included easily by adding one single third-layer module.

The software is designed for a high degree of automatization, which will be implemented step by step and tested on various hardware systems. The long term goal is a robotic and partly autonomous operation of both SLR stations.

Scope and use cases

OOOS is designed to be a comprehensive software suite that takes over all tasks related to the operation of an SLR station. The main components are:

- Experiment preparation (Scheduling)
- Experiment control (Tracking and ranging)
- Experiment evaluation (Data validation and reduction)
- Infrastructure control (Dome control, power supply, etc.)

The two main design goals are flexibility and automatization. Flexibility ensures easy integration of new hardware, but also of new ideas and procedures, e.g. a new search pattern or a new residual filter algorithm. Automatization is important to minimise the chance of human errors

[*] daniel.hampf@dlr.de

(e.g. for laser safety), minimise user bias (e.g. in data analysis) and to reduce the workload on scientists running the system. Flexibility and automatization can be contradictory to some extent, and every design decision must reflect a good balance between the two.

Currently, OOOS is used in four DLR observation systems (see also Fig. 3): The experimental ILRS station on Stuttgart Uhlandshöhe (Hampf, et al., 2016), the new space-debris laser ranging system STAR-C (Humbert, et al., 2017) (Hasenohr, et al., 2017), a small mobile passive-optical observation system and an autonomous staring system.

Flexibility through modular design

To achieve the desired flexibility, the software is divided into various modules, with every module taking care of one specifically designed task. The structure of the modules closely mirrors the hardware of the ranging system, and the typical procedures of the measurement. The coupling between the modules is as loose as possible and uses a universal interface scheme to facilitate the integration of new modules. Each module has exactly the information needed for its task, and performs this task with as little interaction with other modules as possible. This design also improves the possibilities for thorough tests on module level.

The design is structured into three layers, as shown in figure 1. The graphical user interface is merely used for displaying information to the user, and registering user input. These inputs are relayed to the daemons on the second layer, which perform the actual work. The tracking daemon, for example, controls the mount, records the telescope images and, in case of visible objects, performs a closed-loop tracking. However, it only has an abstract notion of a ‘telescope mount’ and a ‘camera’. The actual hardware connection is implemented on the third layer. For each type of hardware, an interface with a certain set of functions and attributes is defined. Each specific hardware class implements this interface, which makes substituting e.g. one event timer for another a very straightforward process. Simulation modules for all hardware interfaces enable the user to run the software without any connected hardware for testing or training.

Streamlined user interface

Users usually want to be able to operate software without referring to any manual. Therefore, the user interface should be as intuitive as possible. In OOOS, this is realised by a consistent structuring of functions (e.g. planning, tracking, ranging, reviewing data). The main displays contain only the most important user information and very little controls (Fig. 2). More specific information and settings are available through dialogs that are sorted along the same main pattern. All configuration settings (for GUI, daemons and hardware interfaces) are accessible through the same dialog window. Throughout all parts of the user interface, the same kind of displays (e.g. LEDs showing green, yellow or red) and the same operation logic are used.

Experimental features and outputs are not included in the GUI prevent its cluttering, but rather implemented in separate scripts.

Implementation

The software is implemented mostly in python. Some high-performance functions are written in C and integrated in python as libraries. The code can be run on Windows, Mac OS or Linux, depending on the hardware drivers. As the communication between the daemons is done via TCP-IP, each daemon may run on another PC in a local network. Usually, we run most of the software on Windows 7 PCs, however some daemons are occasionally run on a Linux machine to connect to hardware that comes only with Linux drivers.

To increase performance, all time consuming tasks are executed in separate processes. On a multi-core PC, this increases the speed significantly. Combined with efficient algorithms, this allows us to keep a responsive and stable system even at laser repetition (and return) rates of 100 kHz and above. Nevertheless, the software contains no hard real time requirements. Even if the operating system paused the execution of all OOOS code during a measurement for a second or two, this would not result in any data loss.

Automatization

Every software used to control hardware in an experiment usually provides automatization. If, for example, some function stores the recorded event timer data on a specific location on hard disk without user interaction, this may well be called automatization. Thus, automatization is not a question of yes or no, but rather of degree. While some automatization is indispensable, developing a fully autonomous system such as at Mt Stromlo (Moore, 2017) may not be reasonable in all cases (Ricklefs, 2017). For OOOS, however, a very high degree of automatization is planned (and partly implemented already), because some of the systems currently under development are envisaged to run fully autonomously in the long term.

To automate a specific task hitherto performed by the user, a three-step procedure is followed:

1. The relevant hardware is modified to be fully controllable via PC. If the task is, for example, the adjustment of a mirror, the mirror stage will be outfitted with motors and connected to the PC. The function is then integrated into OOOS via a hardware interface.
2. The user behaviour concerning this task is closely monitored. What information is relevant for the user to decide when and how to perform the task? How is the user performing the task? How does the user evaluate the result, when does he consider the job done? The extensive logging feature of OOOS that records every user input proved to be a valuable tool for this.

3. Step by step, the user actions are implemented as software functions. At first, this may reduce the workload on the user (e.g. only click ‘adjust mirrors’ rather than moving the mirror motors via buttons). Eventually, all user interaction becomes unnecessary.

However, most high-level automation features in OOOS can easily and quickly be overridden by the user to react to unexpected events or – very important in an experimental SLR system – test new ideas and procedures.

For fully autonomous, unmanned operation, unforeseen events and errors are the paramount challenge. To even consider unmanned operation, two features are essential:

1. In case of an error the system is put into a safe state (e.g. dome roof is closed)
2. Some user / maintainer is notified via email, text message etc.

The latter is easily implemented. In OOOS, we use a web server that continuously receives status updates from the stations. If these are suspicious, or no longer coming in at all (e.g. after a complete breakdown of the power supply), the web server sends out an alert.

The former is much more tricky, because it depends on hardware reliability. Redundancy is a must, and inherently fail safe systems make things easier (e.g. a shutter held open by an electromagnet, which closes in case of power failure). A person on site, even if somewhat unfamiliar with the system, might help to achieve a safe state quickly after a problem.

This is however more a question of procedures rather than of software development. Legal issues might further complicate unmanned operations. Therefore, OOOS is designed to be capable of any degree of automation desired by the user, but its full potential will only be used in some of the systems.

Conclusion

The decision to develop a new control software for our laser ranging system has proven to be worth the effort. Interfacing new hardware has become a quick and painless process. Using the same software on multiple systems accelerates the development, as every improvement needed and implemented somewhere can instantly be used by all systems. The encapsulation of tasks according to the actual technical structure of the system has improved the testing capabilities and thus the stability significantly. So far, the software is used on two DLR laser ranging systems, a passive optical tracking telescope and a staring system. An interconnection between these systems will be realized in the future through the OOOS web interface, allowing experiments with hand-overs, bi-static tracking and ranging, stare-and-(laser-)chase and more. While the software is so far only used internally, it may be made available, completely or in parts, for external project partners on request.

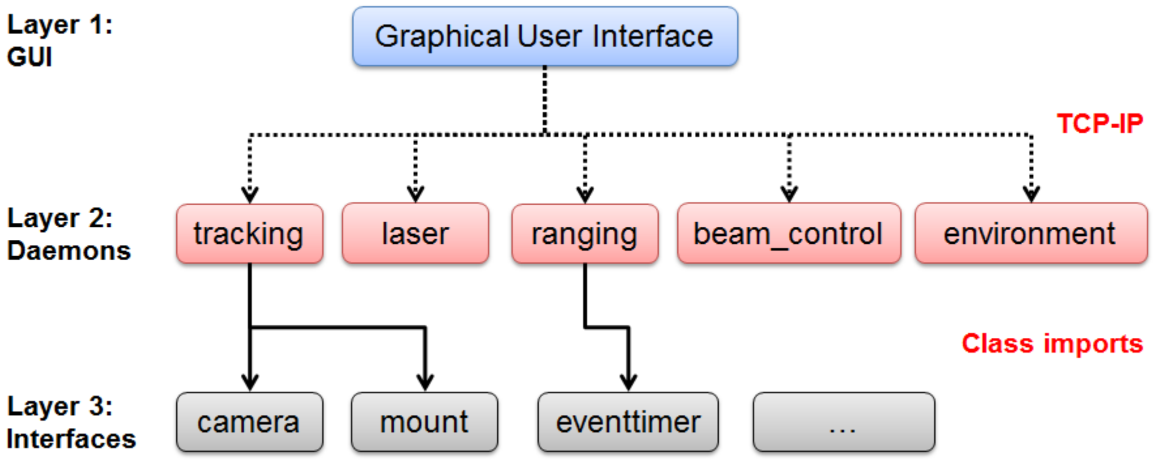
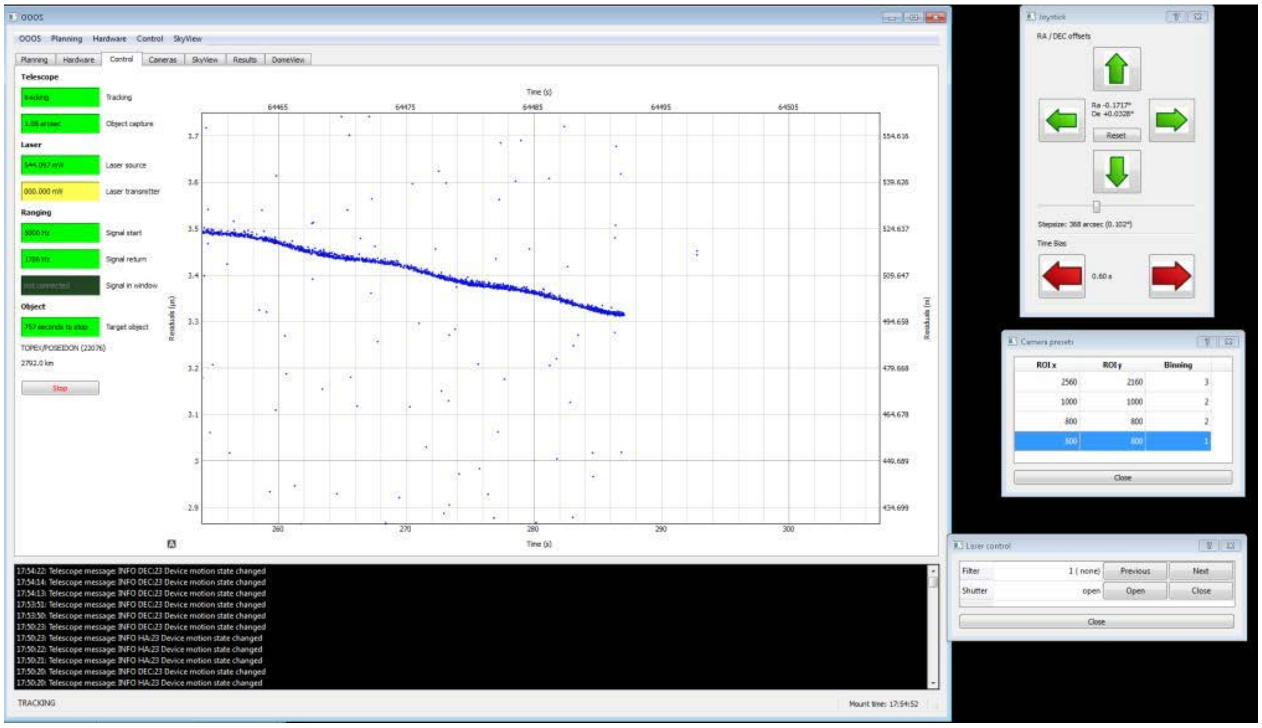


Figure 1: Three layer software layout with GUI, daemons and hardware interfaces.



Telescope

tracking	Tracking
1.08 arcsec	Object capture

Laser

544.097 mW	Laser source
000.000 mW	Laser transmitter

Ranging

5000 Hz	Signal start
1786 Hz	Signal return
not connected	Signal in window

Object

757 seconds to stop	Target object

TOPEX/POSEIDON (22076)
2792.0 km

ROI x	ROI y	Binning
2560	2160	3
1000	1000	2
800	800	2
800	800	1

```
17:54:22: Telescope message: INFO DEC:23 Device motion state changed
17:54:14: Telescope message: INFO DEC:23 Device motion state changed
17:54:13: Telescope message: INFO DEC:23 Device motion state changed
17:53:51: Telescope message: INFO DEC:23 Device motion state changed
17:53:50: Telescope message: INFO DEC:23 Device motion state changed
17:50:23: Telescope message: INFO DEC:23 Device motion state changed
17:50:23: Telescope message: INFO HA:23 Device motion state changed
17:50:22: Telescope message: INFO HA:23 Device motion state changed
17:50:21: Telescope message: INFO HA:23 Device motion state changed
17:50:20: Telescope message: INFO DEC:23 Device motion state changed
17:50:20: Telescope message: INFO HA:23 Device motion state changed
```

TRACKING

Mount time: 17:54:52

Figure 2: Graphical User Interface during a ranging experiment to the TOPEX satellite.

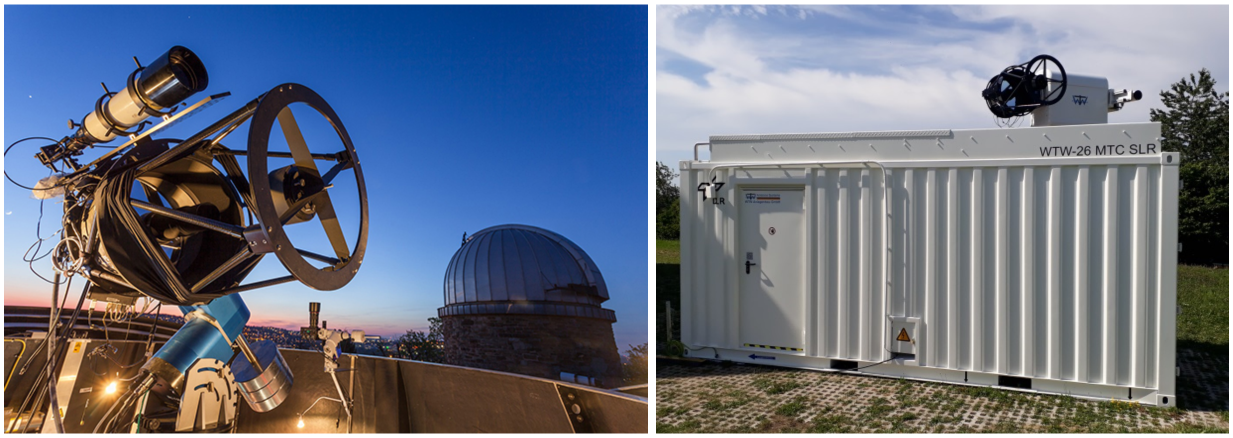


Figure 3: The two main DLR systems using OOOS: The Uhlandshöhe Research Observatory (left) and the space debris laser ranging system STAR-C (right)

References

Hampf, D. et al., 2016. First successful satellite laser ranging with a fibre-based transmitter. *Advances in Space Research*, Volume 58(4), pp. 498-504.

Hasenohr, T., Humbert, L., Hampf, D. & Riede, W., 2017. *STAR-C: Towards a transportable Laser Ranging Station.* Adelaide, IAC Proc..

Humbert, L., Hasenohr, T., Hampf, D. & Riede, W., 2017. *Design and commissioning of a transportable laser ranging station STAR-C.* Maui, AMOS Conf..

Moore, C., 2017. *Mt. Stromlo experience gained with automation.* Riga, ILRS Technical Workshop.

Ricklefs, R., 2017. *When Does Automation Make Sense?.* Riga, ILRS Technical Workshop.