A NEW COMMUNICATION INTERFACE FOR THE EUROPEAN SOUTHERN OBSERVATORY (ESO)'s VERY LARGE TELESCOPE TECHNICAL DETECTOR CONTROL SYSTEM USING ARAVIS, AN OPEN-SOURCE LIBRARY FOR GenICam CAMERAS

K. F. Mulholland*, Observatory Sciences Ltd., Kirkcaldy, U.K.
J. Knudstrup, F. Pellegrin, European Southern Observatory, Garching, Germany

Abstract

The European Southern Observatory's [1] Very Large Telescope (VLT) [2] Core Control System (CCS) [3] provides support for high-performance industrial cameras with its Technical Detector Control System (TDCS). Until now, TDCS has used a communication interface based on an API from Allied Vision Technologies (AVT) [4], which only supports cameras made by AVT. As part of the VLT 2019 release, a new communication interface has been developed for TDCS using Aravis [5], the open-source library for GenICam [6] cameras.

Aravis has been independently developed to provide support for cameras from any vendor, although this is not guaranteed. It reads the GenICam interface of a GigE Vision camera to enable control. It also has capabilities for USB3Vision cameras.

With this new communication interface, support for other manufacturers is now possible. It has been tested with cameras from AVT and Basler [7], and further tests using a CameraLink camera with a GigE Vision adapter are planned. This paper will discuss the capabilities of Aravis, considerations in the design of the communication interface, and lessons learnt from the implementation.

INTRODUCTION

The VLT [2] is one of the most advanced facilities for ground-based astronomy in the world. It is composed of four 8.1-m Unit Telescopes (UTs) and four 1.8-m Auxilliary Telescopes (ATs). The VLT CCS [3] is a software framework providing general operational tools and facilities for communication, logging, an online database, and an alarm system. It also includes a Real-Time Display (RTD) system, interfaces for instrument control, and toolboxes to build applications, amongst other functionality.

Although the VLT first started operations in the late 1990's, it is still constantly evolving to incorporate new techniques and technologies. For example, operating high-performance industrial cameras that transmit high-speed video over a Gigabit Ethernet (GigE) network, with the GigE Vision standard, has been achieved with the Technical Detector Control System (TDCS).

Cameras that implement the GigE Vision standard use the Generic Interface for Cameras (GenICam) [6]. This aims to provide a generic interface for cameras, irrespective of the connection they use. For example, as well as GigE Vision, GenICam is also used by USB3 Vision and Camera Link cameras. The GenICam standard mandates that the camera parameters are stored in a self-describing XML file, which maps the features to the camera's registers. An application API can then easily read and set parameters in the XML file, and GenICam is able to retrieve or set the matching values from the camera's registers.

Up until now, TDCS was only able to communicate with cameras from Allied Vision Technologies (AVT) [4]. It has used the PvAPI SDK, which is no longer supported. While changing the communication interface to be based on the new Vimba SDK, also made by AVT, was considered, an alternative, Aravis, was chosen because Aravis is:

- open source
- actively maintained
- vendor independent

TDCS

TDCS is designed to be compliant with the overall VLT CCS. It allows the controlling and configuration of GigE Vision cameras, streaming the camera data to a real-time display, the application of customised data processing recipes, and storage of data in FITS files. It implements a state machine which is also generally compatible with the VLT's CCS architecture.

TDCS is implemented in C++ and is multithreaded. Several threads are used in order to maximise performance. There is one permanent thread, which is used to handle commands. Other threads perform specific tasks such as acquiring data from the camera, or process the data. The overall architecture of TDCS is shown in Fig. 1.

TDCS currently uses a communication interface based on AVT's PvAPI SDK to communicate with GigE Vision cameras. Messages are sent to TDCS via the VLT CCS messaging system. These messages might be to change the state of TDCS, to start or stop acquisition, or to change parameters, etc. TDCS parses commands to determine what to forward to the camera, and "translates" the commands into the camera's API calls. Similarly, it receives frames from the camera and is able to *publish* them to e.g. the Real Time Display (RTD) module of VLT CCS, convert and store them into a FITS file format, and/or apply customisable *recipes* to perform user-defined data processing.

TDCS is currently used by ESPRESSO [8], and will be used by four other new instruments currently under construc-

* kfm@observatorysciences.co.uk

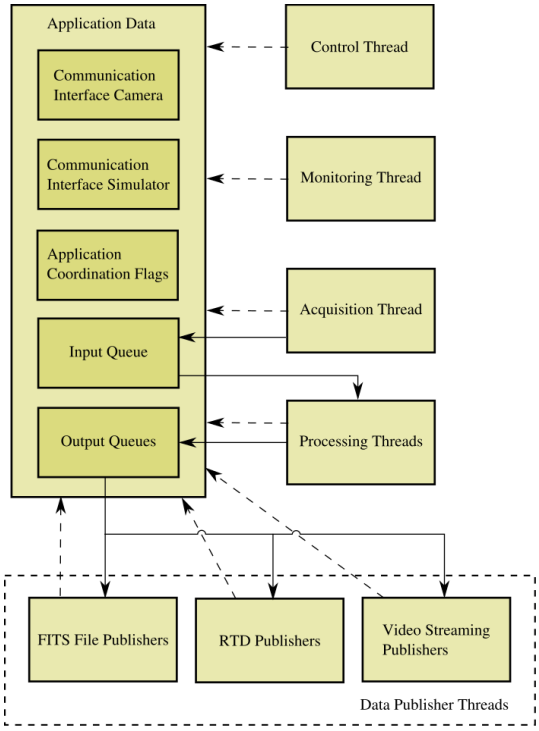

Figure 1: The internal architecture of TDCS. Solid lines represent data transfer, and dashed lines represent communication.

tion: NIRPS [9], SoXS [10], MOONS [11], and ERIS [12]. In addition, METIS [13] will use a product derived from TDCS on the Extremely Large Telescope.

ESPRESSO is capable of using light from up to four of the VLT's UTs. The light from each UT is monitored by two technical cameras, each of which are connected to a tip-tilt mount, to automatically apply stabilisation. One further technical camera is used as an exposure meter.

ARAVIS

Aravis [5] is an open-source library for GenICam-based cameras. It is based on glib/gobject [14, 15] and implemented in C. It is actively maintained and has an active community. It is released under the LGPL v2+.

Aravis implements the GigE Vision and USB3 protocols used by high-performance industrial cameras. As well as the library, Aravis includes some basic tools, a basic ethernet camera simulator, and a simple video viewer. One of the tools which may be particularly useful is `arv-tool-<version>`. This is a command-line tool which allows viewing and setting of camera parameters.

IMPLEMENTATION

The Aravis-based interface is implemented as a C++ class, which inherits from the TDCS base communication interface class. The interface is responsible for implementing virtual functions defined in its parent class.

The design of the new communication interface is the same as the previous PvAPI-based interface and is shown in Fig. 2. A separate thread for receiving frames is used to maximise performance and maintain responsiveness while waiting for data to be acquired. Frames are also checked for errors in this thread, and a struct is used to store the frame along with pertinent information. This is then passed back to the communication interface class in the Acquisition Thread.

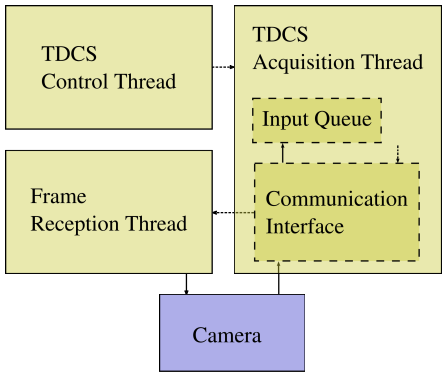

Figure 2: How the communication interface receives frames from the camera (solid lines), and passes commands and configuration changes to the camera (dashed lines).

The camera is queried for its parameters upon achieving a successful connection. In the XML file describing the camera parameters, there is a special node named "Root", from which every other node can be found. Each node can be queried to test if it is a category or a feature, so the nodes are iterated through, and for each category node, its children are then also iterated. A map is used to record parameter information along with a string key, for each feature node (Fig. 3).

Vendor	Parameter name map	
AVT	Generic name	Vendor-specific name
	GenName1	AVTName1
	GenName2	AVTName2
Basler	Generic name	Vendor-specific name
	GenName1	BaslerName1
	GenName2	BaslerName2

Figure 3: A representation of the C++ maps used to store variable parameter names for different vendors. Adding more vendors is simply a case of adding another entry into the top-level map.

As discussed above, moving from AVT's PvAPI SDK to Aravis allows to use not only AVT cameras, but will also allow the use of cameras from other manufacturers in the future. Currently, the communication interface has been tested with cameras from Allied Vision and a Basler acA1300-30gm camera. While the GenICam Standard mandates a

common *format* for camera parameters, it does not necessarily mandate e.g. the parameter *names*. Therefore, the communication interface uses C++ maps for the parameters which are known to be variable, dependant on manufacturer. A top-level map uses the vendor name as the *key*, and another map as the *value*. The bottom-level maps use the same, generic, names as *keys* and the vendor-specific names as the *values*. As support for more manufacturers is required, these maps can easily be expanded.

BENCHMARKING

The new, Aravis-based, communication interface has been benchmarked in order to compare it to the old, PvAPI-based, communication interface. Four tests were run for each communication interface. The tests were written in Python, and involved sending messages to the TDCS control process, via the CCS messaging system, to change state and perform commands. The time taken for each test was measured using the Linux system command *time* [16].

The tests performed the following:

1. Change state to Off, then Standby, then Online; start and then stop acquisition; change state to Standby, then Off

2. Change state to Off, then Standby, then Online; start acquisition, publish a frame, and then stop acquisition; change state to Standby, then Off

3. Change state to Off, then Standby, then Online; start acquisition, acquire six frames, and then stop acquisition; change state to Standby, then Off

4. Change state to Off, then Standby, then Online; start acquisition, and acquire a frame; send a command to change the exposure time to 0.2 s; acquire another frame, and then stop acquisition; change state to Standby, then Off

where the state Off means that there is no active connection with the camera and no threads for publishing or processing data are running; the state Standby means that there is an active connection with the camera but only the control thread is running, and the state Online means that there is an active connection with the camera and all TDCS threads are running.

As seen in Fig. 4, the new communication interface ran each test around 1.5 s faster than the old communication interface. It is therefore likely that there will not be a negative impact on performance of the overall TDCS by changing to the new Aravis-based communication interface.

CONCLUSION AND FURTHER WORK

A new communication interface has been implemented for the VLT's TDCS, which controls technical detectors over a Gigabit Ethernet connection. The new communication interface is based on Aravis, an open-source, glib/gobject-based

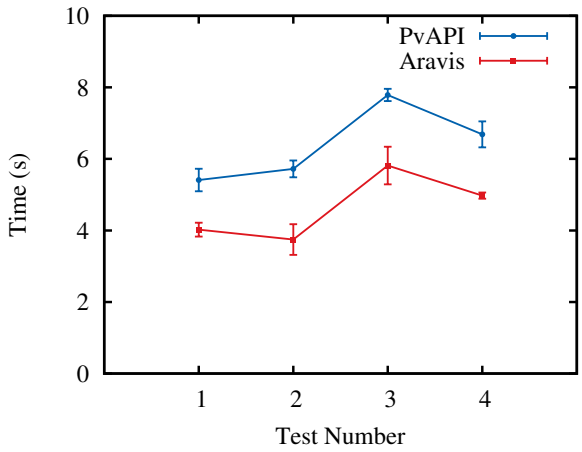

Figure 4: Benchmarking results, showing the average time taken for four different tests to be run, in seconds, for the old, PvAPI-based, communication interface (solid blue circles), compared to the new, Aravis-based, communication interface (solid red squares). The standard deviation of each measurement is also shown.

library for communication with GigE Vision and USB3 Vision cameras. It has been tested with cameras manufactured by Allied Vision Technologies and Basler. Benchmarking results indicate that the new communication interface should not negatively impact performance.

The new communication interface is being released with the 2019 version of the VLT software. It is planned to add support for more camera manufacturers, and to explore if a CameraLink camera (perhaps using a GigE converter) could be supported. New versions of Aravis have already been released since development began, and upgrading to a newer version of Aravis will occur for VLT's 2020 release.

Testing will occur at the VLT in October 2019, after which ESPRESSO will be migrated to use the new communication interface.

REFERENCES

[1] European Southern Observatory, https://www.eso.org

[2] Very Large Telescope, https://www.eso.org/public/unitedkingdom/teles-instr/paranal-observatory/vlt/

[3] Central Control Software, https://www.eso.org/projects/vlt/sw-dev/wwwdoc/FEB2007/VLT-MAN-ESO-17210-0619/Output/Introduction.html

[4] Allied Vision Technologies, https://www.alliedvision.com/en/

[5] The Aravis Project, https://wiki.gnome.org/Projects/Aravis

[6] The Generic Interface for Cameras, EMVA, https://www.emva.org/standards-technology/genicam/

[7] Basler, https://www.baslerweb.com/en/

[8] F. Pepe *et al.*, "ESPRESSO – An Echelle SPectrograph for Rocky Exoplanets Search and Stable Spectroscopics Observations", *The Messenger*, ESO, vol. 153, pp. 6-16, Sep. 2013. https://ui.adsabs.harvard.edu/abs/2013Msngr.153....6P

[9] https://www.eso.org/sci/facilities/develop/instruments/NIRPS.html

[10] https://www.eso.org/sci/facilities/develop/instruments/SoXS.html

[11] https://www.eso.org/sci/facilities/develop/instruments/MOONS.html

[12] https://www.eso.org/sci/facilities/develop/instruments/eris.html

[13] https://www.eso.org/public/unitedkingdom/teles-instr/elt/elt-instr/metis/

[14] GLib, https://wiki.gnome.org/Projects/GLib

[15] GObject, https://developer.gnome.org/gobject/stable/

[16] *time* man page, http://man7.org/linux/man-pages/man1/time.1.html