Additional Software Required for Robotic Telescopes

Petr Kubánek (1,2), Stanislav Vítek (2), John French (3), Alberto Castro–Tirado (2), Martin Jelínek (2) and Martin Nekola (4)

(1) GACE, Universidad de Valencia, Spain (2) IAA-CSIC Granada, Spain (3) University College Dublin, Dublin, Ireland (4) Astronomický ústav Akademie věd České republiky, Fričova, Ondřejov, Czech Republic

We have spent a considerable amount of time developing, installing and monitoring an open–source package for observatory control, primarily aimed at fast GRB follow–up observations. We have integrated some image processing steps inside our observatory control system and have obtained and published interesting results. But we know we currently miss a lot of interesting events.

There are many desired features missing from our software. As we cannot find any commercially available solutions for these, we develop our own solutions. This is of course more difficult then using an available one. The list includes pure Java astronomical calculation library, a web interface for robotic telescope (RT), a global RT planner and scheduler with a display of the instrument's status and the observations. All items are presented in this poster.

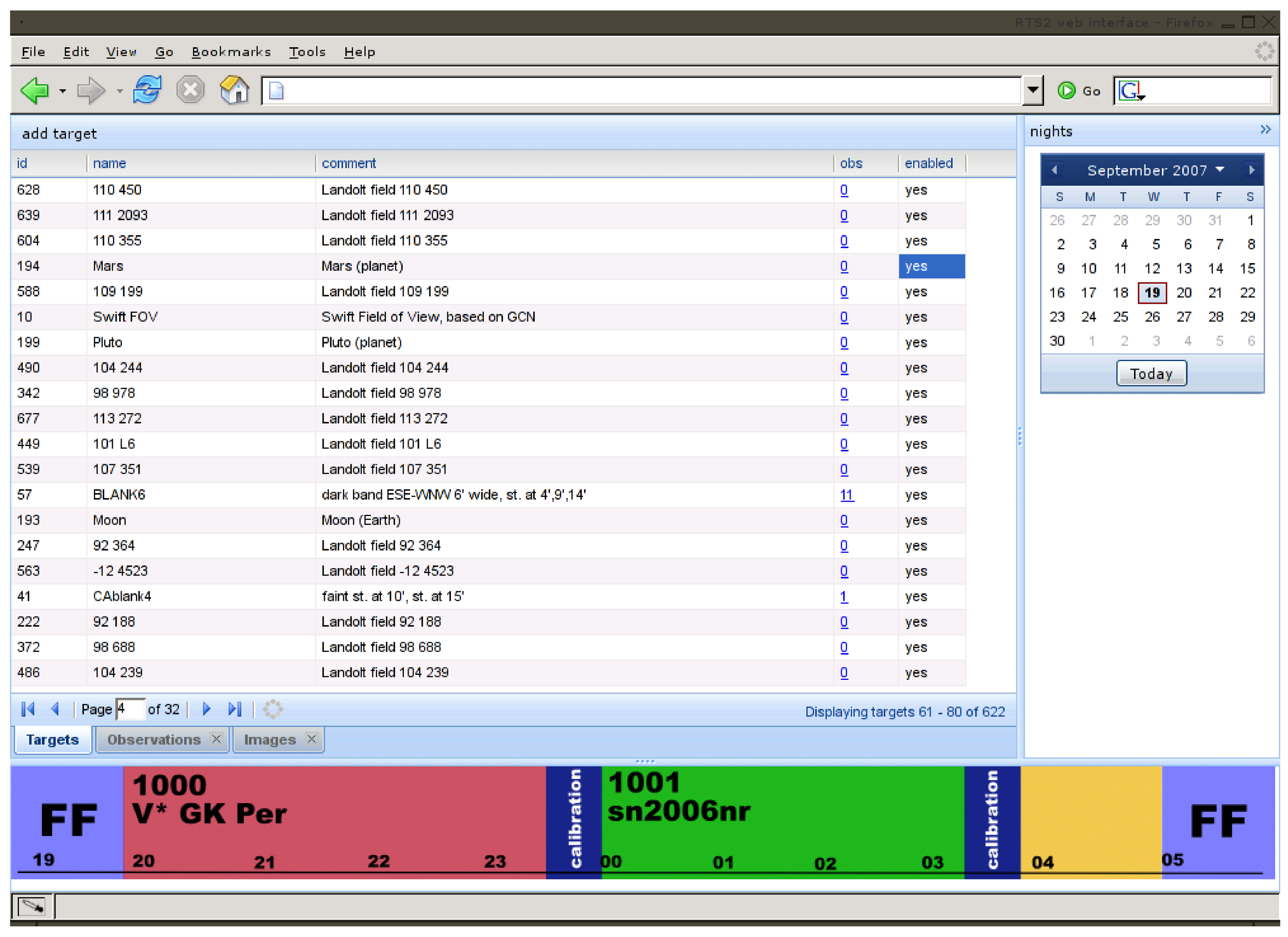


Figure 2: Web Interface design

1. Introduction

Fully robotic telescopes are carrying out a growing number of tasks. The majority of them are focused on Gamma-ray Burst (GRB) follow-ups. This is a field where their speed and precision provide significant advantages over human controlled observervations. The amount of data is significantly reduced by satellite trigger informations.

Over the past number of years, we have established a collaboration of astronomers interested in using robotic telescopes. The collaboration uses the RTS2 (Remote Telescope Software) package, a robotic observatory manager, to control their systems. RTS2 is designed as a modular observatory manager. Parts of the system can be added or removed at any time without affecting observatory operation.

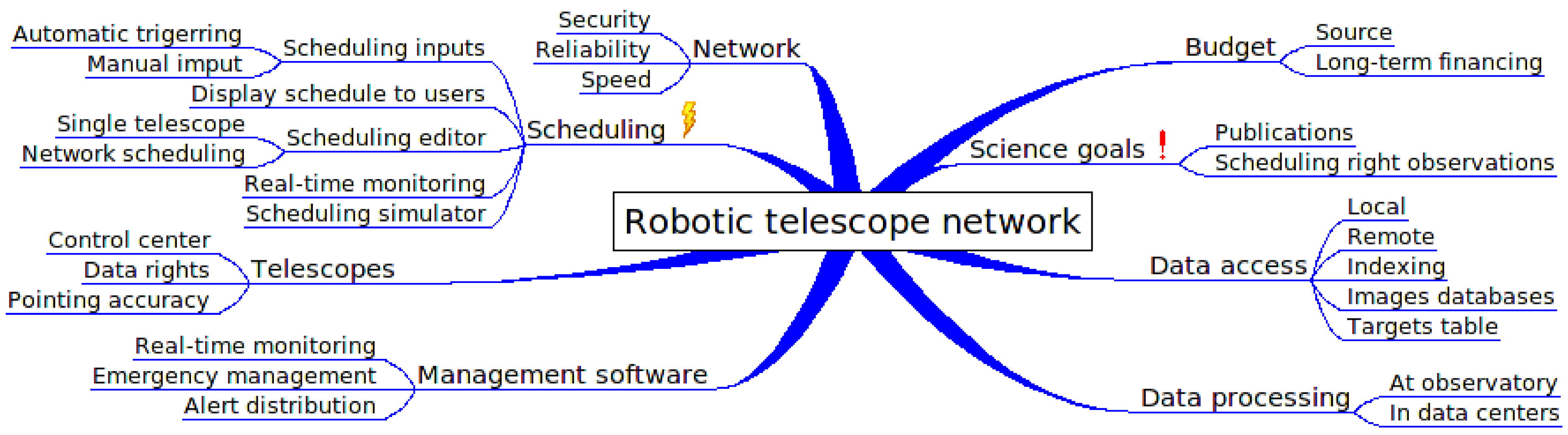


Figure 1: Telescope network topics

The first goal established during the development of RTS2 was to properly obtain and save CCD images. Subsequent features added to the system allowed for observation management, GRB follow-up observations, image processing, human interaction and monitoring of the system performance, and presentation of results. The progress and results achieved are described in [K+06].

One of the bottlenecks in RTS2 development is the lack of a central observatory network manager, and thus a central scheduling and management ability. This problem is described and possible solutions are discussed in [Kub08], and outlined in figure 1.

The second bottleneck is the interface that enables the user to interact with the system. We lack a proper GUI (Graphical User Interface), which will enable end users to enter, manage, check and access the observations.

Although we have a scheduler, we are not completely satisfied with its results, and we are looking for a proper scheduler which will allow for improved scheduling of observation runs.

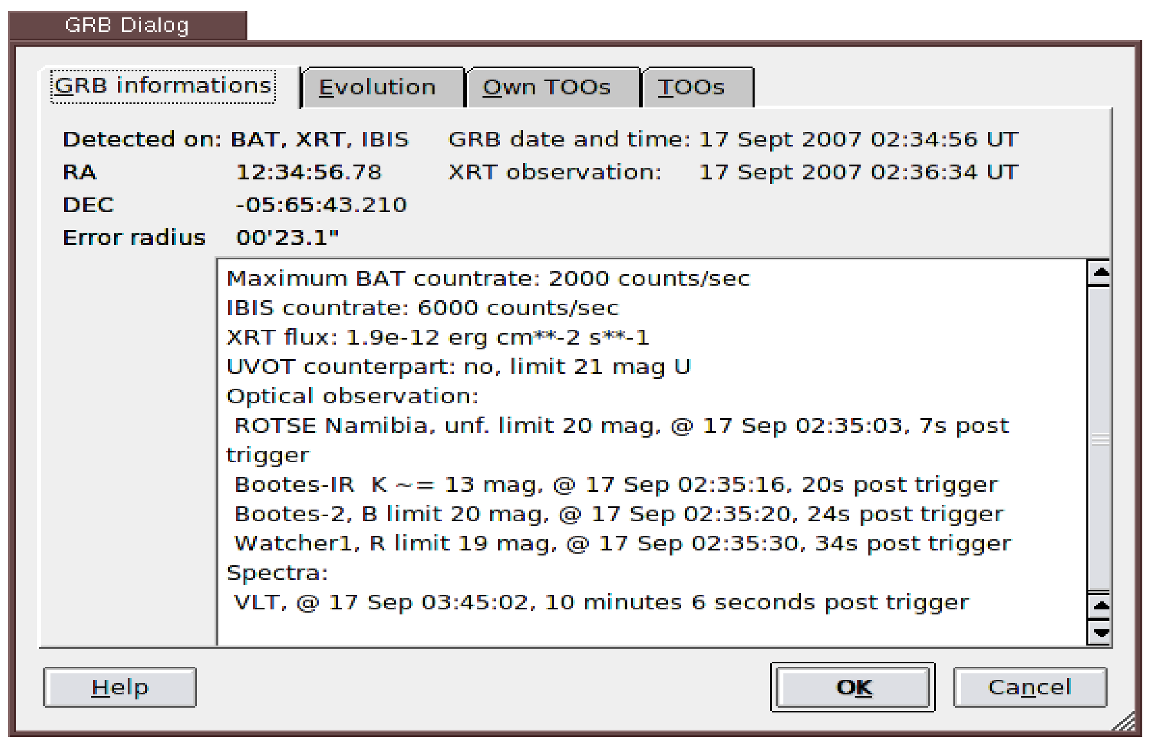


Figure 3: Sample GRB information dialog

4. Web Interface for RT

The RTS2 web interface is based on the thin client idea. Data is taken from a PostgreSQL database used by RTS2. Display is provided by Java servlets running on an Apache Tomcat server and a couple of Java custom classes. Graphical user interface uses Java-Script client-side framework Ext. The graphical user interface uses the Java-Script client-side framework Ext. The communication between server and client side is asynchronous, using XML as the message format (AJAX).

The web client can also offer limited control over the devices and servers controlled by RTS2. A SOAP server, part of RTS2, is used to pass commands from the servlets.

5. Image processing library

Currently, image processing in RTS2 is performed by calling a shell script. The script consists of calls to various image processing routines, utilising either independent binaries or routines in IRAF, Octave or other image processing environments.

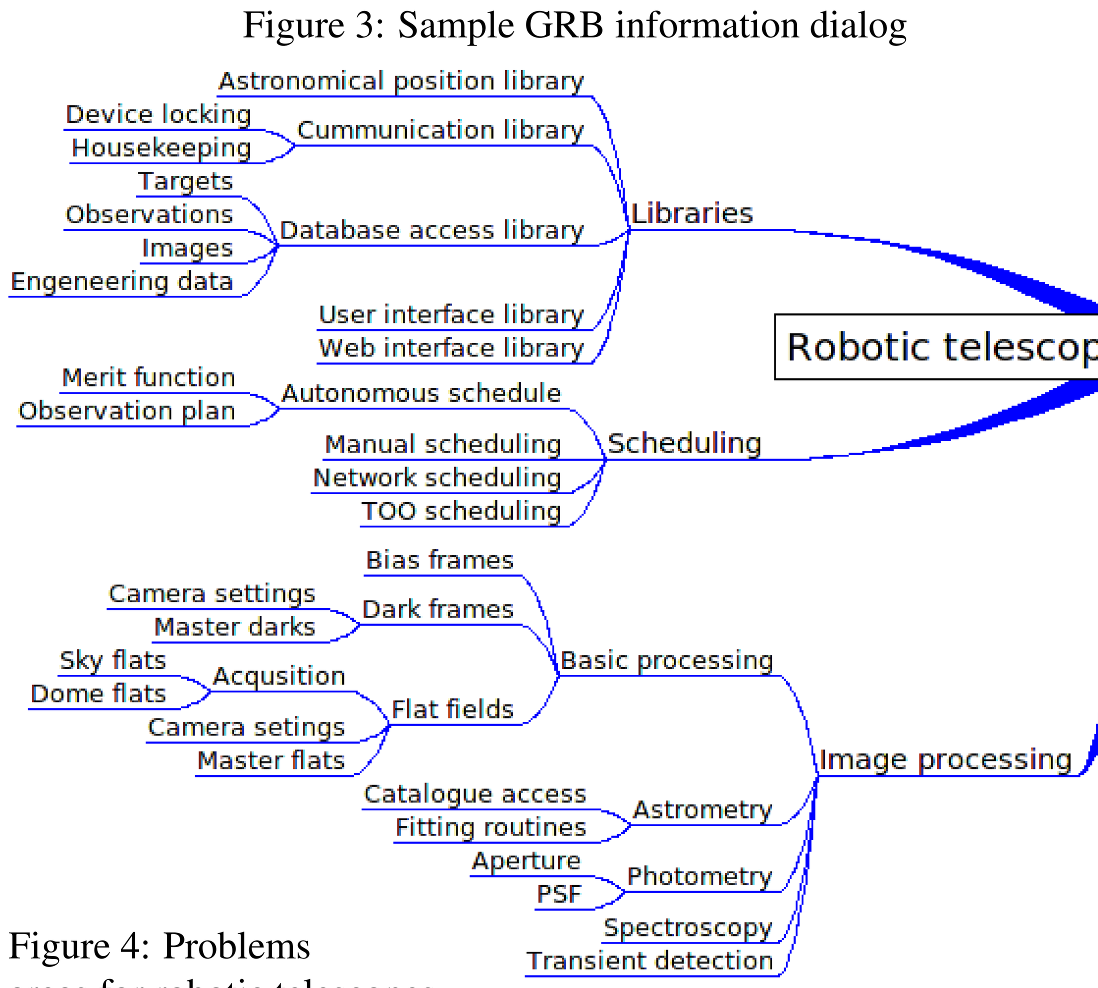
While that approach works, it adds a significant processing load on the system when the program is initialised. As the script subroutines are independent programs, each must open a FITS file where the image is stored, call parsing routines, parse the FITS file, do some computation and store the resulting image. Even with the improved performance of modern operating systems, achieved by caching data in memory as much as possible to avoid accessing data from hard drives (or other slow I/O channel) unless absolutely necessary, significant computational costs associated with the parsing of image meta–data remain.

2. Robotic telescope software

Robotic telescope software is a specialised product. For large projects, software is included in expected costs of the telescope, and is usually developed as a fully customised system using existing components. Some automation is included in every large project. But due to costs, requirements, and expected complexity, only a few large project telescopes are designed as fully autonomous, operating without any human control, and intelligently following targets which appears during the night.

In agreement with the 90/10 rule of software development, basic remote telescope operation, which simply allows an observer to remotely carry out observations, comprises 90% of the software's functionality while requiring just 10% of the development effort. Fully autonomous observation, which completely replaces the human observer in the control loop and makes autonomous decisions based on obtained images, is the last 10% of desired observatory functionality, but due to the significant complexity of the task it can require 90% of development effort to implement. This estimate is supported by evidence in [S+07].

We aim to develop software for all astronomers, amateur or professional, observers of GRBs or solar system bodies. Creating a fully featured observatory manager, independent of device and operation system selection, will allow the robotic telescope community to grow and contribute significant results over the coming years. If such a software base is not available, newcomers to the field will waste time and effort on "reinventing the wheel", instead of adding new functionalities to an existing system.



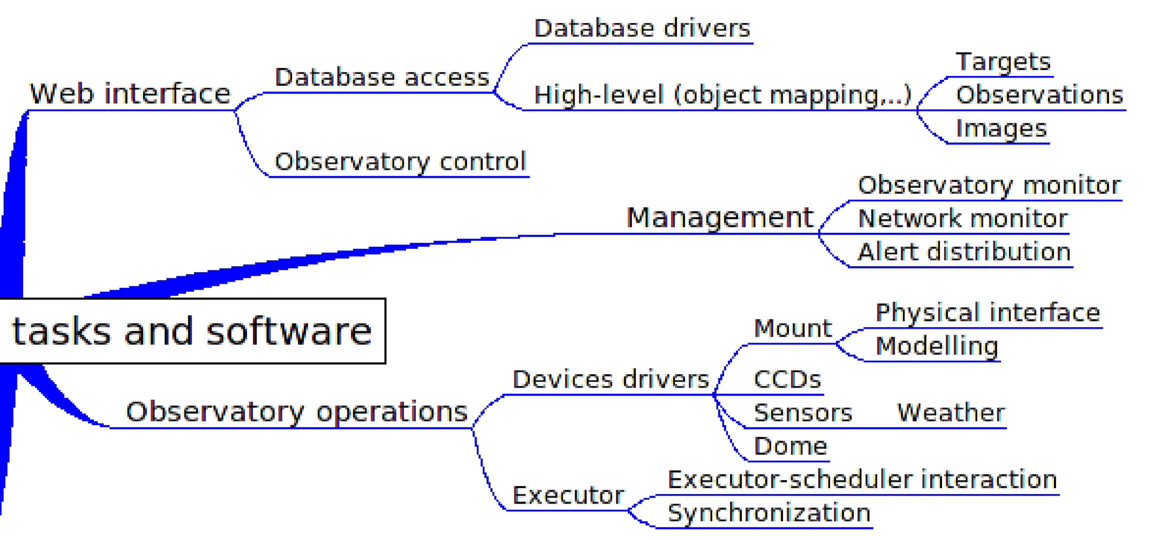


Figure 4: Problems areas for robotic telescopes

Of course these computational overheads are negligible and remain unnoticed when the system is acquiring new images at the rate of one every few seconds. But when the images are acquired at the rate of several frames per second, these delays become significant.

We would like to see the development of an open–source, well documented image processing library, which will provide all the functionality currently provided by IRAF (and other similar packages), but allows this functionality to be easily accessed from either C/C++ or some other higher level language (e.g. Python).

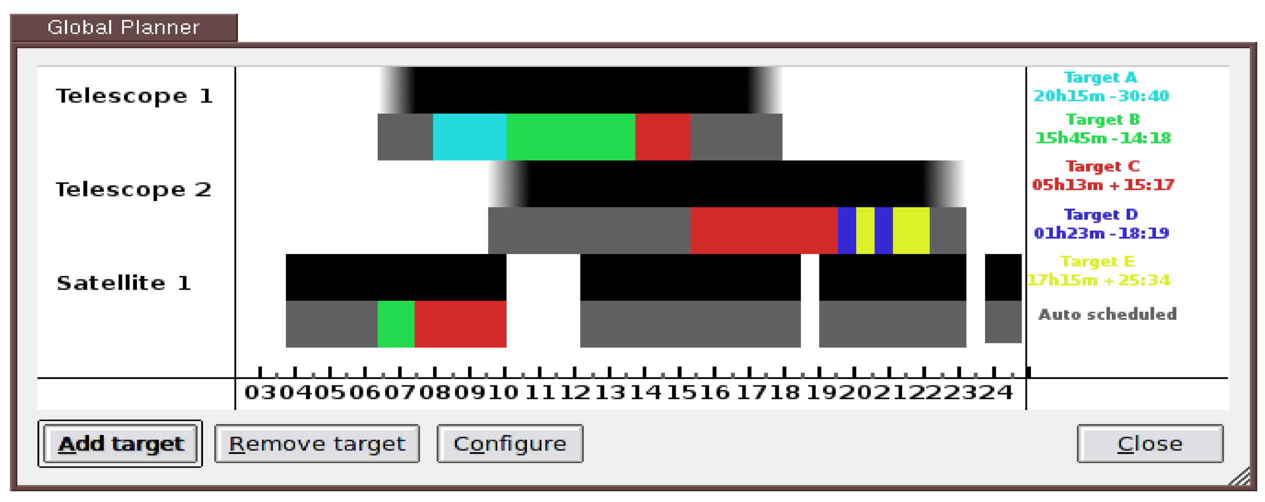


Figure 5: Network observations scheduling dialog

3. Global RT planner and scheduler

GRBs and other target–of–opportunity (TOO) observations have an important common aspect – it is essential to minimise any delay in commencing observations. In the case of GRB observations, every minute can make a difference between obtaining data with a detection of an optical transient, or only an upper limit. The strategy observers usually follow is to obtain rapid response data with small robotic telescopes, and once an optical counterpart has been identified, trigger their TOO observing programmes on larger telescopes.

When a GRB observer is making a decision which instruments can be used to further study the GRB in question, he should be presented with:

the most up–to–date information about GRB, including its position and brightness in various bands, with a prediction of its future brightness. An estimation of the redshift is highly desirable.

the on–going observations of the event, so he/she can choose the best possible strategy for ongoing observations.

TOO triggering details, e.g. how to trigger an observation, whom to contact, how many triggers are left. In an ideal world, this information would be shared among other GRB observers, but the highly competitive nature of the GRB field of research precludes easy access to such information.

This data must be presented in a quick, efficient, and clear manner. Thus, the development of our own GUI, or sophisticated AJAX elements, appears to be a potentially good solution. Drafts of the information dialog boxes are presented in figures 3 and 5.

6. Java astronomical calculation library

RTS2 uses Libnova, an open source celestial mechanics, astrometry and astrodynamics library. Libnova provides RTS2 with all its calculus machinery. Precise calculations of lunar and planetary positions require thousands of lines of source code. As we are developing our new WWW user interface as a Java servlets application, we would like to have a pure Java solution to compute various target properties, and display target information on the web page.

Libnova can provide us with that information, but as it is written in C and is not object–oriented, it will be very difficult to use it in Java. We believe that converting the C Libnova library to a Java library by using a Java Native Interface (JNI) will result in a pseudo object–oriented library, which we think is not worth the effort.

Ideally we would like to have the precision and stability of Libnova coded natively in an object-oriented manner in Java. According to our estimates, this task would require one to two months of design, and about the same for both development and testing, and so can be completed in approximately six months.

Acknowledgements

PK would like to acknowledge a support from the Spanish Ministry of Education and Science via grant BES-2006-11506.

References

[K+06] P. Kubánek et al. RTS2: a powerful robotic observatory manager. In *Advanced Software and Control for Astronomy. Proceedings of the SPIE, Volume 6274, pp. 62741V (2006).*, July 2006.

[Kub08] P. Kubánek. RTS2 - Lessons learned from a widely distributed telescope network. 2008. Submitted to *Astronomische Nachrichten*.

[S+07] K. G. Strassmeier et al. Telescope and instrument robotization at Dome C. *Astronomische Nachrichten*, 328:451, July 2007.