RTS2 overview and real-life processing of the images

Petr Kubánek

IAA CSIC Granada, GACE Valencia

Credits

M. Jelínek, A. Castro-Tirado (IAA-CSIC) R.Cunniffe (Cork)
J.French, G.Mellady, B.McBreen, L.Hanlon (UCD, Dublin)
M.Nekola, F.Munz, J.Štrobl, R.Hudec, M. Kúcka (AsU AV ČR, Ondřejov)
A.de Ugarte (ESO, ex. IAA), S. Vítek (FEL ČVUT, ex. IAA)
M.Prouza (FU AV ČR), J. Frank, I. Kotov, P. O'Connel (BNL, LSST)
M. Wildi (Basel), L. Gilwood (Libnova), V. Reglero, B. Sanchéz (GACE)

... and others (which were forgoten) ...

RTS2 - history

- Remote Telescope System, 2nd Version
- There was of course RTS (1)
  - .. RIP (1999-2002)
  - Python, without database, ..
  - Worked (57 seconds for some GRB)
- C++ (originally pure C)
- Put to public SourceForge Subversion this year
- ~ 80k lines of code (and growing)
- Open source from beginning

RTS2 - goals

- apt-get install rts2
- Configure it (in graphical wizard)
- Test it
- Connect it to network
- Run it
- Do science
- Call ***rts2-make-paper {journal}*** few times a year, have a cup of your preferred drink and enjoy live

RTS2 - primary goals

- GRBs (Gamma Ray Bursts)
- For that we need fully autonomous system
  - Few visible during year on a single site
  - Need really fast (seconds) reaction to triggers to do interesting science
- On first look solved
  - .. but more detailed look show it is not true
    - Missing transient detection
    - Calibrations
    - ...

RTS2 - secondary goals

- Effective system for control of a fully autonomous observatory
- Full scale solution for observatory automation, including:
  - Calibrations
  - Scheduling
  - Full image and data processing
    - Light curves
    - Transients
    - ....

RTS2 - so far

- BART (1999-)
- SuperBART (2007-)
- FRAM (2006-)
- Watcher (2006-)
- BOOTES (200?2-)
  - 1A, 1B, IR, 2,
- BOOTES all sky (2007-)
- LSST testing lab (2007-)
- Markus observatory (Switzerland) (2007-)

RTS2 - future

- University of Columbia lunar brightness telescope
- 1.23 CAHA
- 65cm @ Ondrejov (close to BARTs)
- New Zealand (2009?)
- Reunion Island (under negotiations)
- India (this fall?)
- Russia (next year?)
- others?

RTS2 structure

- Common library parts
  - Astrocalc done by libnova (.sf.net)
- Central daemon
- Devices daemons
  - CCDs, mounts, domes,..
- Services
  - executor, selector, imgproc, grbd, auger, ..
- All connected by TCP/IP
  - ASCII (text) protocol

RTS2 scripting

- Describes how RTS2 observe targets
- Own scripting language, described in man rts2.script
- Simple commands for exposures, filter changes,..
  - F 0 E 10 F 1 E 20
- Loops
  - F 1 for 10 { E 10 filterpos+=1 }
- And more...

RTS2 scripts

- Designed to be easy to code
- System tries to solve synchronization
  - Do not expose while filter wheel is moving,..
- The question is if that is what we wanted
  - Does users wants easy scripting, which will require complex RTS2 processing, and which will sometimes not work (and will be very hard to fix)?
  - Or they are looking for scripts which will allow them to control observing sequence, at costs that they must handle synchronization?

More complex scripts?

- Instead of
  - F 0 E 10
- You will need to write
  - F 0 wait_idle E 10
- Instead of
  - F 0 for 10 { E 10 filter+=1 }
- You will need to write
  - F0 wait_idle for 10 { E 10 not_exposing filter+= 1 }

XML-RPC approach

- Presented by Subaru team at SPIE 2008

```
req = XMLRPC.request (filter=10)
req2 = XMLRPC.request (dither=10)
req.wait ()
req2.wait ()
```

- So they build XML-RPC script with synchronization points (wait for commands completion)
- To go this way, that is the question..

Image acquisition with RTS2

- Following apply to images acquired in autonomous mode
- Image processing from command line is possible, but not supported by RTS2 (if you will know what to do, you will be able to handle it..)
- Images are what we get for science, yet image processing on them is not an easy think

image acquisition and processing

- Done in executor or image processor (imgp)
- Phases
  - FITS creation
  - FITS population
  - Image processing
  - Observation processing

FITS creation

- Empty FITS file is created
- Path is created using substitutions
  - See man rts2.ini for details which strings are allowed
  - Only % strings works in FITS file creation
  - you cannot use $<fits key>$ at this point
- FITS file is created when camera start exposure
  - Change of state from IDLE to EXPOSING triggers image creation

FITS population

- Keywords and values from different components present in the system must find their way to FITS file
- Rts2Values are optimized for writing to FITS file
  - Description (FITS comment)
  - Flag write (and when to write)
    - Exposition start, end,..
  - Important keywords less then 8 characters

Image processing

- So far done:
  - Some dark & flat processing (Martin)
  - Astrometry using
    - RTOpera2 (whatever called)
    - astrometry.net
  - → feedback to telescope (corrections)
  - After that, light curve extraction is beyond my current knowledge
  - Everything called from /etc/rts2/img_process script

Observation processing

- Idea is call observation processing script after all images are acquired
  - and were processed by individual image processing script
- Currently script gets only observation ID
  - But I agree it should be given access to list of images, ..
  - The problem is that with current model it is not an easy thing to do
    - → I need to change that

Current path model

- Subject to change! (it is now in rts2.ini)
- Queue, archive, trash
- I know I cannot live with it any more..
  - .. and need your input how to change it
- This is overview how it works now
  - To start discussion how it can work better

Current path model - example

- Image base is /images
- Epoch is 1 (or 001)
- Image comes from camera C0
- Image is for target 01234
- Exposure started on 26th June 2008 at 20:45:45.123 UT

Image live cycle

- Image is created in que_path
  - /images/001/que/C0/20080626204545-123-RA.fits
- Image is processed by image processor, is good (have on-line astrometry)
  - /images/001/archive/01234/C0/object/ 20080626204545-123-RA.fits
- Image does not have astrometry
  - /images/001/trash/01234/C0/ 20080626204545-123-RA.fits

RTS2 (image) database

- PostgreSQL
- Include image coordinates
  - → possible to search for images which contains object of interest
  - Virtual Observatory extension
- Should we aim at creation of a generic tool
  - Which will include possibility to store any FITS keyword from headers
- And what about user access?
  - Web, GUI, command line, XML-RPC, VO,...?

Disadvantages of current model

- Images are not grouped by observations
- Currently it is not clear from image location if image is raw, has dark frame or flat field subtracted, ..
- It is very hard to construct image path from database entry
  - It is possible, but it can be easier if location of images will not change between trash and good (archive) images

Ideal path model

- Two users
  - Computer science / operative
    - Needs separated images by observations
    - Needs access to data by nights, months,.., so he/she can quickly move part of data to different data storage
  - Astronomer / scientists
    - Needs access to all (calibrated) images of given target
    - Sorted by filter,..

Ideal path model

- Use computer science model for data storage
  - Something like /images/<year>/<month>/<night>/<obsid> /camera_hhmmss.sss.fits
- And provide tools to transfer that to astronomer wish model
  - rts2-image with strings for substitutions to move files
  - Recipes for image calibrations and processing
  - Recipes for data extraction

Recipes for image processing

- Give me all images from given object
  - Calibrated, raw
  - With object no closer then n arcmin to image edge
- Build structure with directories for filters,..
- Extract light curve for given object
  - Aperture or PSF photometry
  - with calibration stars taken from the field
  - or with instrument calibration from calibration runs

Problem with ideal path model..

- I need user input
  - That is one of the reasons why I called this meeting
- I am sure that this is not a work for single developer / astronomer
  - That is why we need to learn how to collaborate and share our work

RTS2 - problems

- Complexity (→ not for a single developer)
- Documentation (→ for a single developer)
- Time lost on solving operational issues
  - New telescopes, cameras, problems in night runs
  - Currently about 70-90% of my time, fluctuates, but usually do not drop bellow 30%
- Range of issues
  - Hardware, database, XML-RPC
  - Synchronization
  - Image processing

Fears?

- RTS2 have ~80k lines of code
- Developed for 8 years → 10k lines / year
- It is still not what I want
- Rule of thumb:
  - Good coder can design, write, debug and document 100 lines / day
  - I can do that (100 work days / year on average)
  - .. but I know that is not enough ..
- Thinking telescope has ~ 200k lines
  - Expect to reach more then 400k lines

RTS2 - development ideas I

- Rts2Image library extension – afternoon discussion
- XML-RPC used as interface between hardware and executor
  - So executor / observatory control can be written in Python,..
  - Executor then can use Python / any other language scripts for observation control
    - Scripts will become observations blocks, if you like that term

RTS2 - development ideas II

- GUI (Graphical User Interface)
  - PyGTK, XML-RPC – please come to see example during coffee break
- Web interface
  - Again with XML-RPC, Web 2.0, Google Web Toolkit
- Scheduling
  - Genetics algorithms, please ask for details
  - My project for finishing first part of the PhD.

RTS2 - development ideas III

- Faster image transfer
  - When possible, use shared memory
- Binary protocol
  - Faster then ASCII, UDP possible
- Networking component
  - My PhD. thesis topic
  - To control, monitor and use everything
  - Network scheduling
  - Strong monitoring and problem solving support