Insert here your thesis' task.

Czech Technical University in Prague
Faculty of Information Technology
Department of Software Engineering

Bachelor's thesis

Ondřejov Southern Sky CCD Photometry Survey - Image Server

Daria Mikhailova

Supervisor: Škoda Petr RNDr., CSc.

15th May 2013

Acknowledgements

I would like to thank my supervisor Mr. Škoda for his guidance and great support throughout the whole project, Mr. Demleitner for his patience, help and valuable advice, Mr. Fernique for his prompt replies and Jiří Nádvorník for his great cooperation on this project. The last but not the least I want to thank my mother who was immensly supportive and very excited about this project.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46(6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity. However, all persons that makes use of the above license shall be obliged to grant a license at least in the same scope as defined above with respect to each and every work that is created (wholly or in part) based on the Work, by modifying the Work, by combining the Work with another work, by including the Work in a collection of works or by adapting the Work (including translation), and at the same time make available the source code of such work at least in a way and scope that are comparable to the way and scope in which the source code of the Work is made available.

In Prague on 15th May 2013 ....

Citation of this thesis

Mikhailova, Daria. *Ondřejov Southern Sky CCD Photometry Survey - Image Server*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2013.

Abstract

The goal of the thesis is to provide astronomical community or any other interested person a way to study and analyze the archive of Czech remote observations with the DK154 telescope in Chile. Furthermore this thesis fully explains the process of publishing images to IVOA compatible server, discusses what formats could be used to store the images and introduces different tools for image exploration.

Keywords IVOA, SIAP, GAVO DaCHS, astronomy, image publishing

Abstrakt

Cílem této bakalářské práce je poskytnout astronomické komunitě nebo jakýmkoliv jiným osobam přístup k Českému archivu vzdáleného pozorování pomocí DK154 telescopu v Chile. Dále táto práce poskytuje kompletní návod pro zpracování a publikování snímku do IVOA kompatibilního serveru, popisuje jaké formáty jsou použitelené pro uložení astronomických dat a zavádí rúzné nástroje pro zpracování snímku z pozorování.

Klíčová slova IVOA, SIAP, GAVO DaCHS, astronomie, publikování pozorování

Contents

List of Figures

Introduction

Astronomical world is not only about stars and constellations but it is also about immense amounts of data. Due to new technologies amount of data grows rapidly. Everyone wants to have their data well organized and indexed so it is easily accessible, but almost everyone does it in a different way. Astronomers started realizing, that it is very complicated to search through and analyze data in different formats, stored on various servers, and processed with unique techniques. Once this fact has been acknowledged, idea of the International Virtual Observatory Alliance was born.

The International Virtual Observatory Alliance is an astronomical community which tries to unify standards for keeping, processing, storing and communicating data. It makes it much easier for an astronomer to examine data. My supervisor, one of active members of IVOA, Petr Škoda propagates IVOA standards to the astronomical community. Result of this thesis is creation of a compatible IVOA service, explanation of IVOA standards and full description of how astronomers could effectively use these tools for their observation. IVOA's tools are very powerful and increase work effectivity.

The project on which I will demonstrate IVOA capabilities is a CCD photometry survey project; its goal is to measure time variations of stars' brightness (so called light curve). Aim of my thesis is to provide this information in a database to a community working on this project so they are able to examine and control the stellar data in reduction phase. Work on this project is divided in two parts, where another student Jiří Nádvorník implements a catalogue server of photometric measurements of variable star. His and mine thesis together will allow a full advanced exploitation of archive of Czech remote observations with the DK154 telescope in Chile.

The reason why I chose this topic is that the result will be useful to science, which, nowadays, people do not appreciate enough. The topic seemed interesting and new to me and therefore it was a pleasure to get a chance to participate in this project.

Chapter 1

Concepts involved

1.1 About IVOA

The International Virtual Observatory Alliance (IVOA) is an organization which goal is to unify standards and tools for working with astronomical data and provide a better environment for conducting astronomical research. Amount of data which is produced by astronomical observations daily is huge. Modern technologies development is so fast that there have already been constructed astronomical detectors which can produce terabytes of data per night.

Figure 1.1: IVOA logo

As stated at [4], the need for the development of the IVOA was driven by two key factors. Firstly, the explosion in the size of astronomical data sets delivered by new large facilities. The processing and storage capabilities necessary for astronomers to analyze and explore these data sets greatly exceed the capabilities of the types of desktop systems astronomers currently have available to them. Secondly, there is a great scientific gold mine going unexplored and underexploited because large data sets in astronomy are unconnected. If large surveys and catalogues could be joined into a uniform and inter-operating "digital universe", entire new areas of astronomical research would become feasible.

1.2 IVOA Protocols

1.2.1 Simple Image Access Protocol

Simple Image Access Protocol (SIAP) is one of the standards introduced by the IVOA. It is a protocol designed for retrieving image data from VO servers and astronomical repositories through a uniform interface. It is an easy to implement simple standard which is based on HTTP GET queries. SIAP is designed as a service providing images on demand, which are searched for according to the given coordinates of position and size specified in the query.

In order to build a SIAP compliant service, it has to support Image Retrieval and Image Query web methods.

- *Image Retrieval method* allows to retrieve desired image by its access reference, it is a getImage web method. Access reference is a simple URL, no special rules are applied to its structure.
- *Image Query method* allows users to inquire about images on a certain position. Also user can specify other optional parameters such as observation time or intersection in order to get a more precise result. Service in turn returns results in a form of a VOTable, which is a table listing of all images which meet the query with access reference to each (explained in depth in chapter 1.3.1). The image request is sent via HTTP GET method to a URL[1], which is composed of two parts:
    1. A base server URL, Example:

       `http://<server-address>/<path>?`
    2. A list of ampersand-delimited parameters for GET request, Example:

[1] Unified Resource Locator

```
<name>=<value>&<name>=<value>
```

```
http://vos2.asu.cas.cz/dk154_rawdata/q/sia/siap.xml?POS=
14.5848316773,-72.5666431241&SIZE=0.23&band=V&FORMAT=image/
fits&img_type=LIGHT
```

Figure 1.2: Siap query

The main parameters which are obligatory for a SIAP service are listed below. Service may also support other parameters. The first three parameters are mandatory, the rest is optional, but it is important for this thesis.

- *POS* (mandatory) is coordinates of the requested target, defined as the right ascension and declination in the equatorial coordinate system. Declination is an angle between diurnal circles, which are parallel to equator, and an equator which is its natural zero. Zero for right ascension is in vernal equinox and the angle is measured using hour circles (perpendicular to equator), where 15 degrees are equivalent to 1 hour. Basically celestial right ascension is the same as the terrestrial longitude and declination is an equivalent to terrestrial latitude. Values are decimal numbers delimited by comma.
  Example: POS = 13.15833,-72.80027

- *SIZE* (mandatory) is angular size of the region of interest (ROI), Value is written in decimal degrees, if only one value is specified it is applied to both axes of coordinates, otherwise first value is width of right ascension axis and the other value is declination axis width.

- *FORMAT* (mandatory) is a parameter which describes image format and has to be in a form of a MIME-type [2]. Also has to support "ALL" which would mean images in all format are available from the server. Examples: image/fits, image/jpeg, image/png.

- *BAND* (optional) is a parameter for a bandpass filter which is used for the image. Bandpass filter is a filter which transmits frequencies in a certain range and weakens frequencies below and above it. It is expressed as a wavelength range. In this thesis we work with Johnson UBVRI system [3] implemented by Bessel such as for example Bessel V

[2]Multipurpose Internet Mail Extensions specification for recognition of content type

[3]It has five different passbands which stretch from the blue end of the visible spectrum to beyond the red end

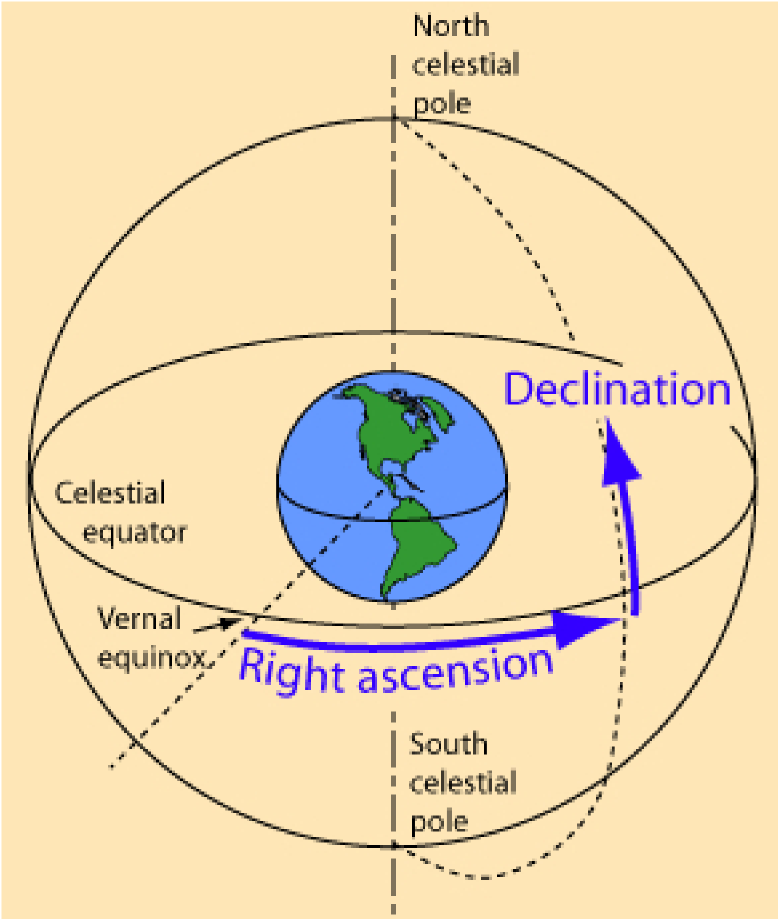

Figure 1.3: Right ascension and declination

or Bessel R. For more information please see [12]. For our own specific purposes we also use Narrow band H-alpha.

- *IMG_TYPE* (optional) is a type of an image. For more detail about different image types see chapter 1.4.
- *OBS_DATE* (optional) is an observation date of the image. User can specify the date when images were taken. It is in ISO 8601 format.

1.2.2 Table Access Protocol

TAP is a flexible protocol for accessing tabular data from relational database management systems. It necessarily supports Astronomical Data Query Language (ADQL) and optionally Parameterized Query Language (PQL) to form queries. ADQL is a mutation of a Structured Query Language (SQL) with certain operators for performing astronomical queries, for instance cone search. Response to the query is a table of data – VOTable, its support is mandatory. Table Access Protocol supports several types of queries: data queries, metadata queries and Virtual Observatory Support Interface (VOSI) queries.

1.2.3 Simple Cone Search Protocol

SCS is a web query protocol which returns all the sky objects located in the cone which size is specified in the query. I will not specify any more details concerning this protocol since I will not work with it in my thesis. However I added it in the list of involved concepts because it is one of the requirements for our Image server, to be SCS compatible.

1.3 Data formats

1.3.1 VOTable

The VOTable format is an XML standard for the interchange of data represented as a set of tables. VOTable is an unordered set of rows, each of a uniform structure, as specified in the table description (the table metadata). Each row in a table is a sequence of table cells, and each of these contains either a primitive data type, or an array of such primitives. VOTable is derived from the Astrores format, itself modeled on the FITS Table format; VOTable was designed to be close to the FITS Binary Table format.

VOTable is designed as a flexible storage and exchange format for tabular data, with particular emphasis on astronomical tables. Interoperability is encouraged through the use of standards (XML).[10]

Main element of the VOTable is **VOTABLE** which contains descriptions and definitions for the parameters and columns of a table. They are followed by the **DATA** tag values. **DATA** values are the actual requested data. VOTable can be used as data storage, transport format, or solely for storing metadata (just the structure of a table).

1.3.2 FITS

Flexible Image Transport System was formerly developed for interchange of data in astronomical community. Nowadays it is a very popular format for archiving, analyzing and exchanging data between observatories. The format is endorsed by NASA and the Astronomical Union. Ability to store metadata and other scientific information is what makes FITS special. FITS file consists of HDU segments each 2880 bytes long. HDU unit is comprised of ASCII text header and the binary data. First HDU header is considered a primary header. Primary header consists of various metadata, defining information concerning an image, such as for instance TELESCOPE type or FILTER used. Binary data represent an image which could be followed by extensions, other HDUs. Example of FITS primary header is in Appendix D .

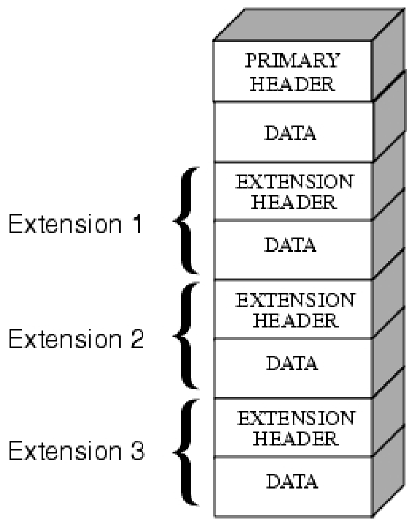

Figure 1.4: FITS file structure

1.4 Image types

1.4.1 BIAS

The technique for processing BIAS images is called debiasing. BIAS images are images which demonstrate the so-called electronic noise. This noise is sometimes referred to as instrumental signature. BIAS images are taken when the telescope's shutter is closed and at the shortest possible exposure time. These images are taken throughout the observation night in order to collect as much information about the noise as possible. Then they are merged into a master BIAS which is later subtracted from the actual sky observations to receive clear images with a minimum level of electronic noise.

1.4.2 FLAT

The technique is called flat fielding and it is used to calibrate relative sensitivity of individual detector pixels and as well to improve digital image quality. The main concern is that when observing the sky not the whole detector is equally illuminated. Some areas of detector could be shadowed due to dust or other outer influence on optical surfaces. Each pixel has a different sensitivity. Those artifacts distort the image and light level is not correct. FLAT images are taken in order to compensate those effects. FLAT image is an exposure of a homogeneously illuminated area. It could be a uniformly illuminated screen or sky opposite the sun short time after the sunset(the so called sky flats). In order to keep calibration level low many FLAT pictures should be taken.

1.4.3 DARK

If detector is not properly cooled then some electrons might end up in the picture even without the need for an activating photon. Those electrons are called dark current. It is like a BIAS picture, the shutter is also closed but the exposure time is long. DARK pictures are combined into a master DARK which is then substracted from the image. They remove dark current from pixels. However after substraction there is dark current noize that is left therefore it is better to take as many DARK images as possible to eliminate the noise. Dark current itself and its noize can be removed by cooling down the detector. Since BIAS is already included in the DARK you do not do debiasing when you work with DARK images.

1.4.4 LIGHT

Light images are the actual exposure of the Celestial field. Some times this image type is referred to as Science.

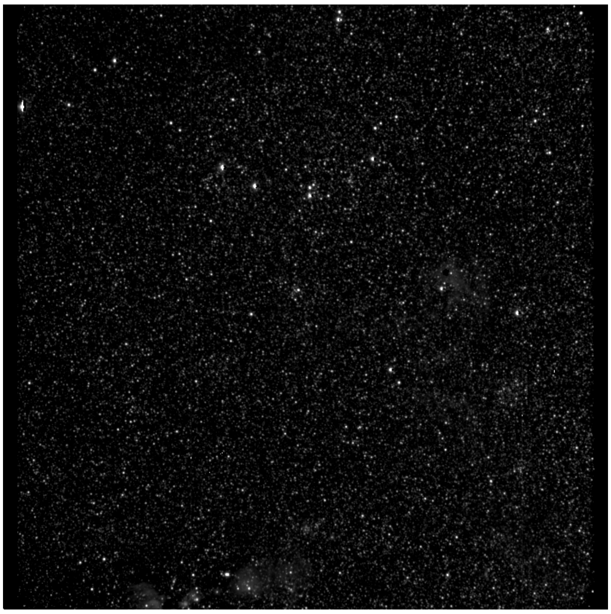

Figure 1.5: LIGHT image

1.5 Image Reduction process

Images which are received from the CCD detector are raw images in FITS data format. There is no astrometric information on the pictures, just an approximate guess which direction the telescope is pointed. Those images undergo a process of reduction. During reduction instrumental influence is removed, such as dust, fiber or anything else which happened to be on the lens during the observation - debiasing and flat-fielding is applied. Reduction is an automatic process. In Ondřejov Astronomical Institute it is done by a program called Munipack by Dr. Filip Hroch [9]. Reduction result is a clear image with astrometric information and calibration — position of all the sky objects on it. Please see example of astrometrical reduction in FITS header in Appendix D. I will not further explain the reduction process in detail in this thesis since it is not part of the topic.

1.6 Ondřejov Astronomical Institute

Astronomical Institute of the Academy of Science of the Czech Republic is one of the oldest institutions of its kind. It has a long rich history and is the direct successor of the Observatory o the Jesuit College from 1722. Part of the institute is located in Prague and another one is in Ondřejov.

The famous 2-meter telescope is located in Ondřejov.

The project I am participating in is initiative of the Stellar Physics Department of Astronomical Institute and it is unofficially called Ondřejov Southern Photometry Survey (OSPS). The remote observations are held with the DK154 telescope in Chile.

This survey tries to provide uniform access, compliant with IVOA standards, to observations collected by the consortium of Czech astronomers sharing the part of the observing time of the Danish 1.5 meter telescope at La Silla in Chile in remote observing mode.

1.7 VO Registry

Registry is a set of VO records and resources presented for public access. GAVO DaCHS contains a publishing registry therefore it is pretty easy to do it. Publishing process for GAVO DaCHS can be explored at [6].

Chapter 2

Analysis and design

2.1 Requirements

2.1.1 Functional Requirements

2.1.1.1 Database

Database server has to be PostgreSQL. It supports such modules as *q3c* and *PgSphere* which work with spherical geometry and searches in the circular regions on the sphere which is required by IVOA standards and thus essential for the server. The database will hold formation about various sky objects, their position, names, bandpass filters, observation time and a lot of other important metadata.

2.1.1.2 Flexible data ingesting

Data ingestion has to be very flexible in order to provide an option of mapping data to database fields. Some database fields have to hold computed values or are influenced by other columns. Therefore we needed something which will allow calculations, conversions, pattern recognitions and usage of various programming functions in the database ingesting process.

2.1.1.3 Meta-data processing

Ingesting has to be able to extract metadata from FITS file's header. It should know how to work with them and how to effectively parse them.

Also every column has to have meta attributes like name, UCD[4], utype[5] and other according to the VOTable format.

2.1.1.4 Data importing and updating

Data importing has to be a simple console script. The script should be able to update the database from the filesystem when new files arrive. It will be periodically run so the information is always up to date.

2.1.1.5 Operating environment

Operating environment has to be UNIX based, preferably Linux. We have chosen Debian. Debian is a very stable operating system, based on UNIX. Apart from supporting free software and being overwhelmingly satisfied with Linux environments, Debian was chosen because it is an operating system which runs on Virtual Observatory servers which were provided to me by my supervisor to run the Image server on.

2.1.1.6 SIAP server

The server has to be a compliant Simple Image Access server. It should support all the requirements for a SIAP service, mainly understanding of SIAP queries for retrieving multiple row results, response in a form of a VOTable and individual image retrieving.

2.1.1.7 Compatibility with IVOA standards

Table Access Protocol has to be supported for table accessing. Another protocol which we needed is Simple Cone Search. Therefore the server has to support circular searches or searches in a circle. I don't work with this protocol in my thesis however it is an important part of the project as a whole and it is a part of Jiří Nádvorník's thesis.

2.1.1.8 Vizualization requirements

It was required to create previews for already existing FITS images. The previews have to keep the identification of astronomical objects and be of a small size. Original FITS images produced by DK154 telescope are approximately 16MB, so the required size was about 200 - 400 Kb. Images

[4]Unified Content Descriptor - used in astronomical databases for easier recognition of data content

[5]Unified Type - used in astronomical databases for easier recognition of data types

have to be compressed since they only serve for previewing purposes and not for direct measurements or work.

2.1.1.9 Authentication

Secured access to images in FITS format is necessary since they are used in a currently running project. Images which are produced in an unfinished project of a consortium of several organizations have to be kept under restricted access even some time after completion of the project.

2.1.2 Non functional requirements

2.1.2.1 Integration with IVOA previewing tools

Essential requirement is communication with previewing tools such as Aladin. Images stored on the server have to be correctly previewed in this utility. Astrometrical information must be the same as in their equivalent FITS pictures.

2.2 Feasibilty study

Further I will describe feasibility study I had to take in order to analyze the task. I will introduce different options I was considering such as already existing astronomical tools and my solution. I will also talk about image formats for previews.

2.2.1 Astronomical servers

Image publishing is a procedure used by astronomers daily. There have been many different astronomical utilities developed purely for storing and communicating data.

Saada

- Author: Laurent Michel
- State: frequently updated
- Technologies: Java, Apache Tomcat, various relational databases

Saada is an astronomical database generator written in Java. It has easy installation and pleasant graphical interface. It is actively developed and it supports different services such as Spectra, Images, source and TAP. Saada uses its own query laguage SaadaQL. It is compliant with many IVOA protocols, for instance it is compliant with Simple Image Access Protocol, Simple Spectra Access Protocol, Table Access Protocol and Simple Cone Search. Database ingesting is simple and is done automatically. Special field mapping is done by configuring import routine.

VODance

- Author: Marco Molinaro from Centro Italiano Archivi Astronomici

- State: new, under development

- Technologies: Django, XML, Java, MySQL, Oracle database management systems

There is no ingesting mechanism, no need to export data, user just has to link his database to the server and fill in meta data manually. Creates service out of a table or view. It has already been used for publishing SIAP and SCS services.

GAVO DaCHS

- Author: Markus Demleitner from Astronomisches Rechen-Institut of Heidelberg University

- State: actively developed

- Technologies: PostrgreSQL, Python, XML

This server is very well written and has a full set of useful features. It is compatible with IVOA standards. Installation might seem complicated at first but there is very good documentation which explains manual installation process step by step [7]. The server is also available in a Debian package. Author is very flexible and communicative so any problems which may arise during working with the server are solved immediately by contacting him. The server is being constantly updated and new features are being added. Data ingestion is configurable and the user can predefine the mapping of fields and implement any needed calculations and modifications before the data is loaded to the database. Provides embargo mechanism, which restricts access to images fot the certain period of time.

2.2.2 My solution

One of the options is to implement my own solution. However it would be very complicated and time consuming. I believe that it would be just a reproduction of one of the above mentioned toolkits with minor differences. Therefore we agreed upon choosing an already existing solution from the ones presented in the feasibility study.

2.2.3 Image formats and techniques

Firstly I had to choose which format to use for the previews. Here are two formats I was considering.

JPEG

- Name: Joint Photographic Experts Group
- Popular for: small size of images, format for digital cameras and web
- Compression: Lossy

JPEG is a widely used format for storing images. It uses a lossy type of compression, as described by Independent JPEG Group [2], this means that an image which was compressed looses some of its quality during encoding and is not the same as the original. JPEG compression uses DCT[6]. Compression reduces high frequency information which an eye will not easily notice e.g. tiny details. Our eye is also less sensitive to mild color changes than to brightness changes. The level of compression depends on the image data, for instance digital photograph could be reduced from 20:1 to 25:1 without noticable decrease in quality, but with a tremendous size change. Compressed images save space on your disk and are quickly transmitted and loaded on web. User can manage compression level with a *Q factor*, otherwise called *quality setting*, in ranges from 1 to 100. If quality level of compression is not lower than 80%-90% human eye won't notice small changes in quality. 100% quality JPEG image which is demonstrated on figure 2.1 is almost the same as the original however it is much smaller. Figure 2.2 shows JPEG with 10% quality of compression, where DCT artifacts are very noticable and image is distorted.

JPEG was considered one of the options because of the compressoin which doesn't allow any photometric measurements and any other machine analysis.

[6] Discrete consine transformation algorithm

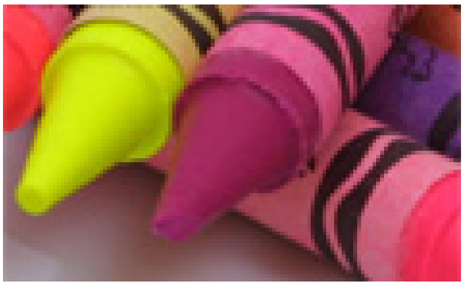

Figure 2.1: JPEG 100% quality

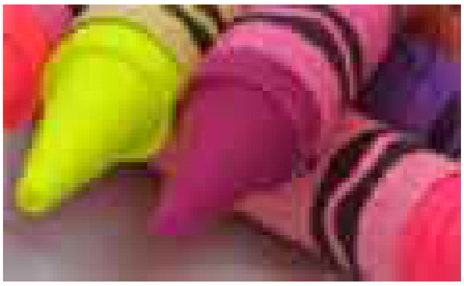

Figure 2.2: JPEG 10% quality

PNG

- Name: Portable Network Graphics
- Popular for: Bitmap graphics, good quality
- Compression: Loseless

PNG can manipualte with a wide color range, e.g. 16bit grayscale images or truecolor images with up to 48 bit per pixel. PNG uses loseless compression, which means that once image is decompressed it is perfectly identical to the original. PNG uses 2 step compression, where firstly it filters the image and then deflates it. Filtering is a prediction method, which predicts values for each pixel before compression, taking in consideration neighbouring pixels. There are various filtering methods. Filtering helps the compression to be more efficient and make image size smaller. Then the Deflate loseless data compression mechanism is applied, it is using combination of LZ77 and hufman coding. The algorithm is based on the principle of a sliding window. Data in the image repeats itself, image colors have only limited range. Once the algorithm runs into a repeated piece of information it simply puts a so-called pointer to the previous occurence of this data piece. The pointer is compressed using Huffman-encoding. It consists of length of repeated data and length of the distance back to the same information. Sliding window means that data stream is used as a dictionary to encode data with window sliding there and back once position is updated. (for more info please see [13])
Following figure 2.3 shows an example of a png image with a nice color range. No matter how many times the image will be compressed it will preserve its quality. Little interesting fact about PNG, according to the standard it supposed to be pronounced as 'ping' however it is never spelled that way. There has been a lot of debates and discussions concerning this rather peculiar issue.

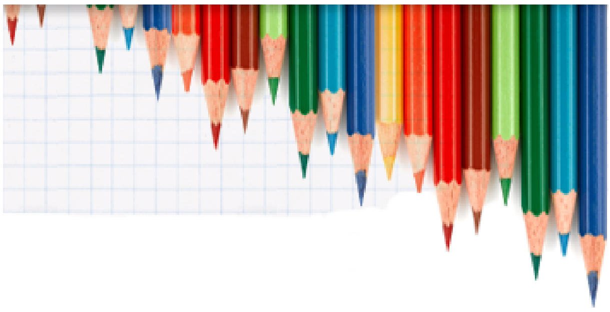

Figure 2.3: PNG image

PNG vs. JPEG
JPEG is not used for images which will be edited over and over since it looses more and more data with each saving and the final image can be totally destroyed. PNG works very well for this purposes bacause of its loseless compression.

PNG images are large therefore they will not be useful when it comes to photography, where JPEG fits perfectly. JPEG was created for storing high quality 24-bit[7] photographic images.

When working with images which contain text or large solid color areas PNG would suit better because JPEG will produce noticable artifacts on these types of images.

2.3 Analysis

2.3.1 Server structure

The figure 2.4 shows proposed architecture for our Image server. SIAP server receives request from a previewing client in a form of a SIAP query, then it processes the query and sends back a VOTable. While processing the request server communicates with database and retrieves information about images to form a reply. Database is populated with the help of a database ingestor. There is a *cron* which periodically runs the database updating job.

In the meantime we are only provided with one server and the load is not supposed to be too big, because only people who work on the project will use it. Therefore there is no need to consider options which handle big data requests loads. However it is always better to plan such things in advance. In case of providing access to our service to a bigger amount of

[7] 16 000 000 colors

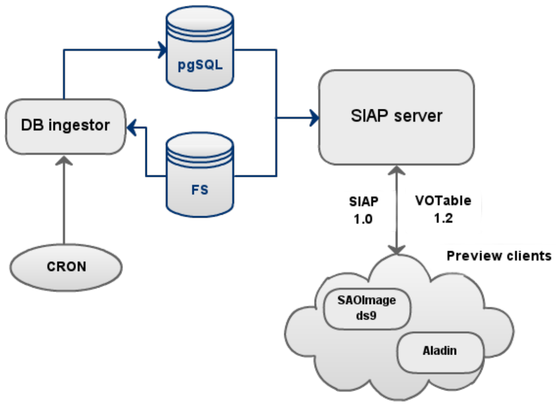

Figure 2.4: SIAP server architecture

users it will be necessary to obtain another server and to set a proxy. Proxy will balance the server load and prevent it from overloading with requests.

2.3.2 Discussion on the chosen server

After a thorough consideration I decided not to implement my own solution since it would be very time consuming and I also noticed that there is already some very well made publishing infrastructures which I described in the feasibility study.

Firstly I was studying Saada which seemed to me as a good solution. Its installation process is very easy, it is just a click through thing. Everything is wrapped up in a nice graphical user interface so even people who are not skilled in IT field would be able to use it with no problems. Saada can work with any database system which is a great advantage since, using it, one does not have to switch from their favorite database system. However after taking a closer look I realized that it has some drawbacks and can not be used for our purposes. To start with, the documentation is quite confusing and is not corresponding to the latest release of Saada. Biggest problem is the lack of flexibility when ingesting data. Saada does provide certain mechanism for mapping fields but it is not as flexible as we would like it to be. We would like to have an ability to do complicated computation and changes before insertion of data.

GAVO DaCHS seems to be a good choice. It is a very flexible server. At first it might seem a bit complicated, since it requires understanding of programming and mark-up languages. Documentation is very detailed and covers almost all the problems one can come across. In case there is something that is not mentioned I advice to write to the author, who promptly responses to questions. DaCHS is constantly supported by Mr. Demleitner. Database ingesting allows to preprocess data before populating the database. I will describe data ingesting process in chapter 3.2. DaCHs has a nice web interface which is also configurable. You can define different fields for filtering database data and you can define the filtering process itself in the database ingestor. It means that you are able to define your own search patterns. It is very handy since sometimes there are special requests which would otherwise demand changes inside the server. It also supports all the necessary IVOA standards such as SIAP, TAP and SCS.

VODance was not even considered due to its architecture; It s not what we were looking for.

2.3.3 Discussion on the chosen format

FITS image format is very well known and there is a lot of various tools supporting its conversion to other image formats. DaCHS already has fits to jpeg conversion binary. It is written in C using libjpeg library by Mr. Demleitner and it is primarily used for generating previews in the database. Output is a compressed jpeg file with heavy DCT artifacts. The reason for artifacts being so noticable is that as I already mentioned before JPEG is better not be used for images with solid color areas. All images we are processing are solid black with small sparkles of stars. Therefore we decided to use another format which does not use lossy compression - PNG. Loseless compression maintains image quality and after several graphical improvements information is still visible, however, astrometrical measurements are impossible.

At first I was experementing with ImageMagick library which supports various programming languages and console commands. Documentation is very good and it seems like ImageMagick supports all the various filter types for image processing. For more information about ImageMagick please see [1].

FITS images produced by DK154 are very dark so one cannot tell stars and the sky apart. That's why I was looking for something to improve contrast balance and increase white color intensity. After some time experimenting with ImageMagick I came across a nice utility purely designed for our purposes. It called fitspng routine by Dr. Filip Hroch from Masaryk University in Brno. It is a very neat program which automatically adjusts color levels in order to bring stars to the front. It was specifically designed to process observation images.

Chapter 3

Implementation

This chapter describes GAVO DaCHS, database, ingesting of data and implementation of visualization tools, which was an optional part of my thesis.

3.1 GAVO DaCHS

GAVO DaCHS is a good solid publishing infrastructure, it is written in Python. It separates controller logic from protocol definitions and that allows extending it as much as needed. DaCHS easily processes FITS files, VOTable and other VO-compatible files with the help of IVOA protocols. It's installation I will describe in Appendix B. What I particularly liked about DaCHS is that it is built in a way that a user can configure almost anything he needs when ingesting data and sending response to data requests without interfering into server's architecture.

3.2 Database and data ingesting

3.2.1 Database

PostgreSQL was chosen as a database system for this project. GAVO DaCHS works with this database and there is a well written documentation on setting this db up on the server. Another reason for that as I already mentioned earlier is the set of plugins which support various astronomical, trigonometrical functions - *PgSphere*, *q3c*.

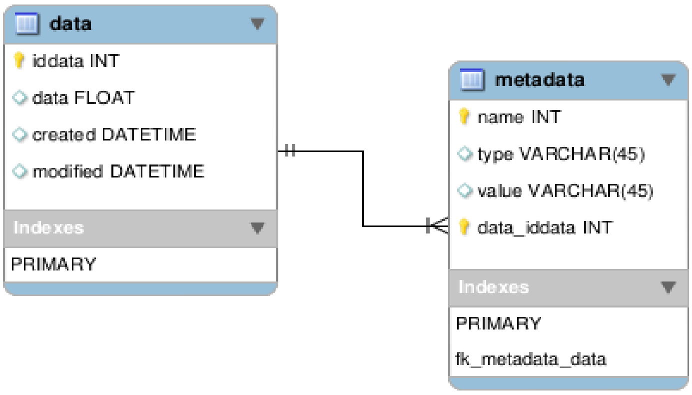

Figure 3.1: Simple database schema

3.2.2 Data ingestor

The ingestor is a console script for processing FITS files and populating the database with its metadata and data.

Features

- Ingestor creates an SQL create script.
- It is possible to modify data input before populating the database. Define calculations and conversions for each table and then do the mapping which is a Python dictionary with lambda functions.
- Ingestor extracts metadata from FITS files. It effectively parses data using different grammars for each particular case.
- There is an update function. If ingestor is run with a -u parameter then database is only updated and no files are overwritten.
- It is easy to remove the table with the help of a DROP function.
- Automatic table generation with SIAP metadata. These tables are generated according to the SIAP protocol which is defined in DaCHS.
- Programming of database response to input query parameters.

To perform database ingesting in GAVO DaCHs I wrote a resource descriptor. Resource descriptor is an XML document with nested Python code blocks and PQL, ADQL expressions. For purposes of this thesis I created two tables:

- *dk154_rawdata* - Table for raw images, which come directly from telescope with no modification.
- *dk154_reduced* - Table for images which has undergone reduction process. More infromation about reduction process please see Chapter Image Reduction process

All the examples below are from *dk154_rawdata* resource descriptor, which can be found in Appendix C. The whole ingesting process could be separated into four parts: define target database and tables, parse input data, map data and start the ingestion.

Define target db and tables
Start with defining the name of your database and some meta information about it. In our case it is *dk154_rawdata*.

```
<resource schema="dk154_rawdata">
    <meta name="title">DK154 Chile telescope</meta>
    <meta name="subject">Stars: Variable</meta>
```

Resource *tag* wraps the whole resource descriptor. *Meta* tags are used for adding meta information about your service which will be later displayed on the GAVO DaCHS web interface when accessing the database.
Then define the table you want to create. All columns of the table must be defined. In case of writting resource decriptor for some of the VO standard services there is a set of functions which predefine mandatory columns. For instance there are functions for SIAP and SSAP. It is important to define columns according to the standard.

```
<table id="raw_images" adql="True" onDisk="True" mixin="//siap#pgs">
    <mixin
      facilityName="'ASU CAS - NBI'"
      collectionName="'ASU CAS - NBI/DFOSC_FASU'"
      calibLevel="1"
    >//obscore#publishSIAP</mixin>
    <column name="exposure" tablehead="Exp. time" unit="s"
      ucd="time.duration;obs.exposure"/>
    <column name="telescope" type="text"
      ucd="meta.id;instr.tel"
      description="Telescope used for the observation"
      verbLevel="25"/>
```

Tag *table* defines the table you are creating, setting *adql* to *True* allows the table to be queried with AQDL. Metadata tables can be easily recognized since they have got *onDisk* attribute set to *true*. It means that they are stored in a database, not in memory. *Mixin* attribute is set to *siap#pgs* which is a simple support for SIAP queries. In the example above I defined two columns and set their UCDs accordingly. Correct ucds[8] can be found at [5], which is a tool for finding appropriate UCD names for your data. *Mixin* tag ensures certain functionality for the table. *Mixins* create columns and rows for already defined protocols and standards.

```
<data id="import_image_content" auto="false">
  <sources recurse="True" pattern="data/*.fits"/>
  <fitsProdGrammar qnd="True">
    <rowfilter procDef="//products#define">
      <bind key="owner">"beusers"</bind>
      <bind key="embargo">'2018-12-31'</bind>
      <bind key="table">"dk154_rawdata.raw_images"</bind>
    </rowfilter>

    <rowfilter name="addPNG">
      <code>
        yield row
        row["prodtblAccref"] = row["prodtblAccref"].replace("/data/", "/dataPNG/", 1)
        row["prodtblPath"] = row["prodtblAccref"].replace("/data/", "/dataPNG/", 1)
        row["prodtblEmbargo"] = '2010-12-31'
        row["prodtblPath"] = row["prodtblPath"][:-5]+".png"
        row["prodtblAccref"] = row["prodtblAccref"][:-5]+".png"
        row["prodtblMime"] = "image/png"
        yield row
      </code>
    </rowfilter>

  </fitsProdGrammar>
```

Data tag is for defining source and the process of working with data. We define source in *sources* tag, saying that folder should be read recursively. *Rowfilter* tag is a row generator coming from grammar in our case *fitsProdGrammar*, which returns FITS header metadata as a set of dictionaries. *ProcDef* parameter in rowfilter tag invokes *products#define* Python function with the help of which we set embargo, restricted access to our files. Another rowfilter *AddPNG* generates fake PNG rows for us. These rows are exact copies of rows for the equivalent FITS files, we just change access reference (prodtblAccref), path (prodtblPath) and mime to *image/png*. We generate theses rows because our png preview images are always the same when it comes to astrometry as their equivaent FITS images.

Parse input data

You can convert values, write your own functions or define search patterns. Resource descriptor also lets you define variables, which comes in handy.

[8]Unified Content Descriptor - the semantic description of a variable based on the controlled dictionary.

You would ask why it is so important to process input data before the ingestion. In some cases it is crucial. For instance almost all FITS files have *dateObs* (observation date) set in a wrong format. IVOA reccommends to use *ISO 8601* as a time format however FITS files have *YYYY-MM-DD* and time in seconds after midnight. Other example is filters. In our project we are using certain filters which are in Johnson system but are implemented by Bessel. Those filters are not listed in DaCHS natively. This means that other details such as bandpass range and bandpass reference value will not be set automatically since DaCHS will not recognize our filters. It is important for our project to have information about bandpass in the database because it is the way how to tell apart images of different types. Therefore I defined a Python dictionary where "R" filter from FITS would correspond to "Bessel R". Then DaCHS recongnizes "R" filter and can further process it as you will see below. I also constructed image names which we write in the database from the input data. As you can see it is necessary to modify input data in these cases. See Example:

```
<make table="raw_images">

  <rowmaker id="makeRawFrame">
      <var name="dateObs">dateTimeToMJD(parseTimestamp(@DATE_OBS.split(".")[0]))</var>
      <var name="obs_collection">"%s/%s"%(vars["ORIGIN"],vars["INSTRUME"])</var>

    <var key="canonicalFilterName">
        {
        "R": "Bessel R",
        "I": "Bessel I",
        "U": "Bessel U",
        "B": "Bessel B",
        "V": "Bessel V",
        }.get(vars.get("filt"))

      </var>
```

Tags *make* and *rowmaker* wrap the whole Parse and Map data part of the resource descriptor. Tag *var* defines rawmaker variables. Example is wrapped in a *rawmaker* tag which defines mapping between grammars input and finished rows. Rowmakers usually consist of variables and procedures. Metadata from FITS file can be accessed by its name as an array index, e.g. *vars["ORIGIN"].*
Example of calling a built in SIAP processing function:

```
<apply procDef="//siap#setMeta">
  <bind key="title">"OSPS %s %s %s"% (vars["DATE_OBS"], vars["filt"], @OBJECT)</bind>
  <bind key="instrument">@INSTRUME</bind>
  <bind key="dateObs">@dateObs</bind>
  <bind key="bandpassId">vars["FILTB"]</bind>
  <bind key="bandpassLo">vars["bandpassLo"]</bind>
```

```
  <bind key="bandpassHi">vars["bandpassHi"]</bind>
  <bind key="bandpassRefval">vars["bandpassRefval"]</bind>
</apply>
```

Tag *apply* inserts a code fragment to manipulate the result row. *Bind* tag passes input arguments for the *setMeta* function, defined by the *key* parameter.

Map data

Thirdly, you map parsed input data to database columns with the help of a *map* function.

```
<map key="bandpassId" source="filt"/>
<map key="telescope" source="TELESCOP"/>
```

Previously defined variable *'filt'* is mapped to *bandpassId* column in the database.

Data ingestion

Lastly you define your service and ingest data.

```
<service id="sia" allowed="siap.xml,form">
  <meta name="title">SIAP on Ondrejov DK154 observations</meta>
  <meta name="shortName">dk154_rawdata raw_images</meta>
  <meta name="sia.type">Pointed</meta>
  <siapCore id="sia" queriedTable="raw_images">
```

Once service is defined you save your resource descriptor as *q.rd*, which is standard name for resource descriptor in DaCHS. Now you can start ingestion. There are several different parameters ingestion has:

- *gavo imp q.rd* - for simple ingestion of files
- *gavo imp -c q.rd* - ignores corrupted files, prints their names on the output in terminal.
- *gavo imp -u q.rd* - ingests data without overwritting already existing data.
- *gavo imp -m q.rd* - updates metadata only, which is useful if you just need to change column description.

After doing preparation for database ingesting which I described above I can start configuring the way our SIAP server is requested. Our first concern was defining All parameter for filters and image types. Even though it is not in the standard we decided to use it since we know that it is going to be in the next SIAP version.

Resource descriptor has *condDesc*, which is an input definition. By writting a *condDesc* you define a parameter in SIAP request for both browser search and an option for data filtering in the DaCHS web interface. Example of All parameter for filters:

```
<condDesc id="bandCond">
  <inputKey name="BAND" type="text" description="Wavelength (range)
          of interest (or symbolic bandpass names)" unit="m"
            std="True" utype="ssa:DataId.Bandpass">
  <values default="" id="base_BAND_values">
          <option title="ALL">ALL</option>
          <option title="B">B</option>
          <option title="I">I</option>
          <option title="R">R</option>
          <option title="U">U</option>
          <option title="V">V</option>
          <option title="Ha">Ha</option>
  </values>
  </inputKey>
  <phraseMaker>
    <code>
        key = inputKeys[0].name
        lit = inPars.get(key, None)
        if lit is None:
          return

        if lit=="ALL":
          yield "bandpassId IN (select distinct bandpassId from dk154_rawdata.raw_images)"
        else:
          yield pql.PQLPar.fromLiteral(lit, key).getSQL("bandpassId", outPars)
    </code>
  </phraseMaker>
</condDesc>
```

You can find whole resource descriptor attached to this thesis in Appendix C.

3.3 Visualization tools implementation

One of the tasks in my thesis is to create observation images previews. I had to create previews due to the following reason. It is not allowed to provide FITS images, which are used in an unfinished project of consortium of several organizations, to public. Because of that all the FITS images from our server are password protected, so only people who are currently working on a project have access to them. Password protection is set on the DaCHs server when previewing the database and also when trying to access images using Aladin tool.

3.3.1 Conversion solution

Fitspng
As written by the author Mr. Filip Hroch at [8] Fitspng uses global *tone mapping* technique to convert images of a wide dynamical range (which can be practically infinite) to limited range of modern displaying devices. The usage of the *tone mapping* is crucial to get a visually satisfactory picture results.

The tone mapping is generally described by following transformation:

$$PNGvalue = f0 \times Func[(FITSvalue - black) \times sensitivity] + zero \quad (3.1)$$

- *Func* is a selected function (linear, asinh, ..)
- *f0*, *zero* are profile scaling parameters (*f0* is defaulted to 1 and *zero* to 0)
- *FITS values* are prescaled by parameters *black* and *sensitivity* to an appropriate domain of *Func*.
- The *black* sets a level of FITS values corresponding to the black on output.
- The *sensitivity* parameter sets an artificial (fake) sensitivity of detector.

Both parameters *black* and *sensitivity* can be left unspecified. In this case, ones are estimated by the included machine algorithm with help of statistical parameters: median and mean of absolute deviations (mad). Both parameters are describing a histogram of data: median marks center of histogram and mad scales width of histogram. In the case of gray image, just a *tone mapping* and the *gamma correction* is applied to output images.[8]

Gamma correction is an ability to change the way computers interpret color values. Different computers interpret colors differently. Something that looks good on PC might not look good on Mac. Gamma is an approximate value for the brightness of the image.

Figures 3.2 and 3.3 show histogram of a FITS file before modifications and histogram of a png file produced by fitspng routine. It is noticable that before the modification the amount of pixels in the darkest region of histogram (left part) is at its highest possible level while there is no signs of life in the other parts of midtones(middle) and highlights(right side). Second histogram shows better color spreading, where there are some very light parts (right side) and some middletones shown by a small pyramide.

Figure 3.2: FITS histogram

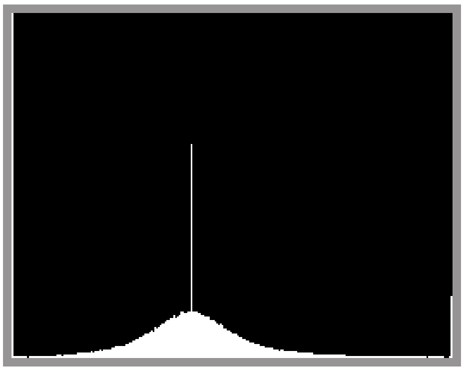

Figure 3.3: PNG histogram

FITStoPNG.py

For the above mentioned purposes I created a Python script FITStoPNG.py which recursively goes through folders with our observation images and where i call fitspng to convert images from fits to png.
Requirements:

- ImageMagick library, which is inbuilt in almost every Linux based operation system
- Python 2.5+
- Pyfits
- Fitspng

It is run everytime when new data arrive. Before ingesting the data fits images will be converted to png. It is run from terminal either for:

- raw images - *python FITStoPNG.py raw*
- reduced images - *python FITStoPNG.py reduced*

During conversion images are scaled. Fitspng is written in C therefore the conversion and scaling is very fast. Important part is including WCS[9] header, which should be resized accordingly. Resizing includes recalculation of the following header metadata: NAXIS1, NAXIS1, CRPIX1, CRPIX2

[9]World Coordinate System

and the following if they are present: CDELT1, CDELT2, CD1_1, CD2_1, CD1_2, CD2_2. For WCS header resizing I am using DaCHS function called *ShrinkWCSHeader* which could be found at *gavoSourceFolder/gavo/utils/-fitstools.py* of your GAVO DaCHS installation. WCS information is added to the comment segment of the file. To do so I use ImageMagick function *comment*. At first adding WCS to the comment field seemed a little tricky since there are three type of text segments in PNG:

- *tEXt Textual data* contains simple textual data. Encoded as ISO/IEC 8859-1 (Latin-1) character set.
- *zTXt Compressed textual data* holds textual data, more useful for storing large data as it takes advantage of compression. Encoding used is Latin-1.
- *iTXt International textual data* holds textual data, encoded in UTF-8 instead of Latin-1.

Aladin, our main graphical testing tool (described in detail in chapter 5.1) didn't recognize compressed image comment *zTXt* segment to which *comment* function saved WCS but it only read from *tEXt*. I contacted Mr. Pierre Fernique who kindly helped me with this problem and added this extra feature to Aladin. And now Aladin reads astrometry from both *tEXt* and *zTXt* text segments of PNG.
Image produced by my script includes astrometrical information and can be used for illustrational purposes in previewing tools.
You can find the whole script on the CD attached to this thesis.

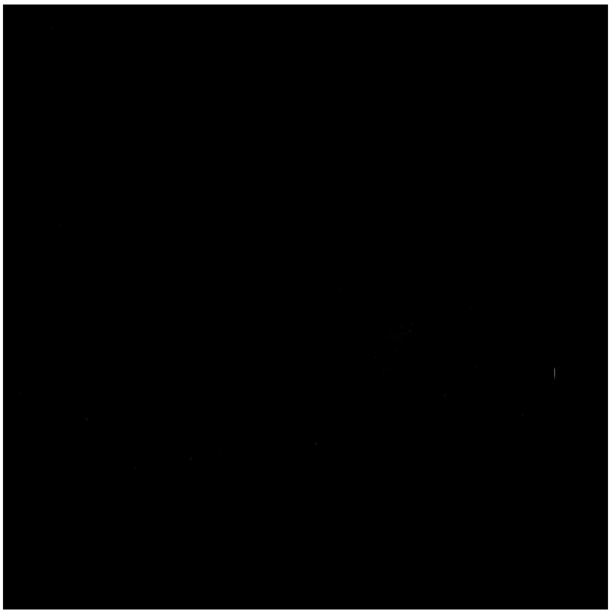

Figure 3.4: Fits file before fitspng

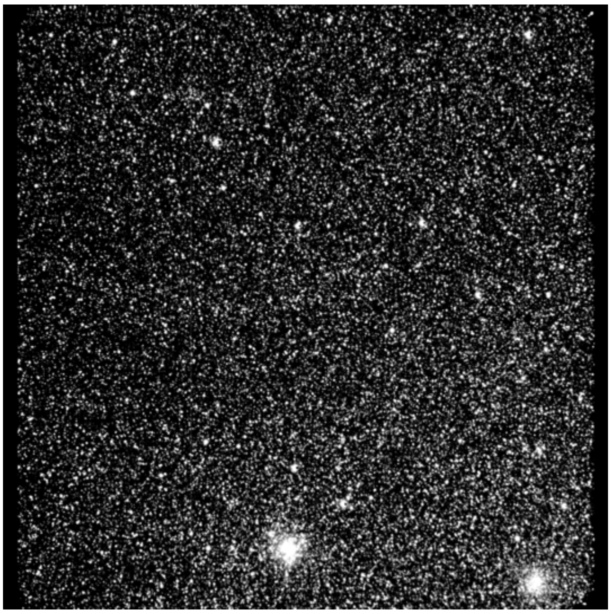

Figure 3.5: Fitspng output

Chapter **4**

Testing

All the test scenarios were tested by users, astronomers from Ondřejov observatory. All test results are positive. No bugs were found and everything works according to the test cases. Therefore I can confirm that all test were successful and our server works according to our requirements. We also tested how our server integrates with astronomical previewing tool Aladin. For Aladin testing I produced a special file called glurecord. It is described in detail in chapter 5.1.2 and an example is in Appendix E. In test scenarios we assume that such file exists and test user has access to it.

4.1 User test scenarios

Test users are free to use both *dk154_rawdata* and *dk15_reduced* tables for the following test cases. I used examples from *dk154_rawdata* since it is a bigger table with more results.

4.1.1 Execute a SIAP query

Prerequisites: SIAP query example:
`http://vos2.asu.cas.cz/dk154_rawdata/q/sia/siap.xml?par&par`

1. Open any type of internet browser.
2. Type in SIAP query with parameters:
   - Position (POS) of the target is 14.5848316773 -72.5666431241
   - Bandpass (BAND) is V

- SIZE is 0.23
- FORMAT is image/png
- Image type (img_type) is LIGHT

3. Receive response in a form of a VOTable with many results pointing to various observation images. One of the results:

```
<TR>
<TD>0.002</TD>
<TD>LIGHT</TD>
<TD>
http://vos2.asu.cas.cz/getproduct/dk154_rawdata/data/20121026/NGC330field01_000003.png
</TD>
<TD>image/png</TD>
<TD>17608320</TD>
<TD>14.199985872609499</TD>
<TD>-72.453305621746907</TD>
<TD>OSPS 2012-10-26T23:58:07.028 V NGC330field01</TD>
<TD>DFOSC_FASU</TD>
<TD>56226.998692129702</TD>
<TD>2</TD>
<TD>2148 2048</TD>
<TD>0.00011 0.00011</TD>
<TD>ICRS</TD>
<TD>2000.0</TD>
<TD>TAN</TD>
<TD>14.200200000000001 -72.453400000000002</TD>
<TD>1074.0 1024.0</TD>
<TD>
-0.000106094 0.00034567199999999999 -0.000106094 3.2562599999999999e-05
</TD>
<TD>V</TD>
<TD>m</TD>
<TD>5.4000000000000002e-07</TD>
<TD>6.7000000000000004e-07</TD>
<TD>4.8500000000000002e-07</TD>
<TD/>
<TD>
Polygon ICRS 14.5911239476 -72.5666709012
             14.5929286337 -72.3413923273
             13.8140855668 -72.3392940315
             13.8025368396 -72.5645463521
</TD>
</TR>
```

4.1.2 Download a preview observation image

1. Open VOTable downloaded in the previous test.
2. Copy one of the links listed in table row answers. Image has to be in PNG format.
3. Paste the link to the browser. Hit enter.
4. Image will start downloading.

4.1.3 Download FITS observation image

Prerequisites: login details for DaCHS

1. Download VOTable just as described in Test case 1 but instead of FORMAT = image/png put image/fits.

2. Copy one of the links listed in table row answers. Image has to be in FITS format.

3. Paste the link to the browser.

4. Dialog window for authentication will appear. Log in with details which you received before the test. Hit enter.

5. Image will start downloading.

4.1.4 Search for objects in DaCHS web interface

1. Open an internet browser

2. Go to `http://vos2.asu.cas.cz/dk154_rawdata/q/sia/form`

3. You see an html form for searching SIAP on Ondřejov DK154 observations - Observations captured by ASU CAS facility by DK154 telescope.

4. Fill in html form with the following details:

   - Position (POS) of the target is 14.5848316773 -72.5666431241
   - Bandpass (BAND) is V
   - SIZE is 0.23
   - FORMAT is image/png
   - Image type (img_type) is LIGHT

5. Press search or hit enter.

6. You will see a list of results for your query.

7. Check whether results satisfy your query and whether positions of images are within requested search parameters.

4.1.5 Results of a SIAP request

1. Open an internet browser
2. Go to `http://vos2.asu.cas.cz/dk154_rawdata/q/sia/form`
3. You see an html form for searching SIAP on Ondřejov DK154 observations - Observations captured by ASU CAS facility by DK154 telescope.
4. Fill in html form with the follwoing details:
   - Position (POS) of the target is 77.3621568609 -14.8515549821
   - Bandpass (BAND) is All
   - SIZE is 0.23
   - FORMAT is image/jpeg
   - Image type (img_type) is LIGHT
5. Press search or hit enter.
6. You will see a list of results for your query.
7. Check whether there are different (All) results for BAND. Check whether results satisfy your query and whether positions of images are within requested search parameters.
8. Fill in html form again and set BAND to any filter you like. Set Image type to ALL.
9. Check whether All various image types are dislayed. Check whether results satisfy your query.

4.1.6 Open observation image in Aladin

Prerequisites: glufile for SIAP server

1. Open Aladin
2. Select Open Local file in File menu.
3. Choose glufile which is prepared for this test.
4. Choose Open from File menu.

5. Choose our server DK154 from right column of servers, then choose Raw or Reduced table.
6. Fill in the following information on the form which appeared:
   - Position (POS) of the target is 'ngc 330'
   - Bandpass (BAND) is ALL
   - SIZE is 0.23
   - FORMAT is image/png
   - Image type (img_type) is LIGHT (if you chose Raw service)
7. Choose any image from the list and open it
8. Image successfuly opened. Astrometrical information is shown. No measurement can be made.

Chapter 5

Usage

In this chapter I am going to describe practical usage of this thesis. I will explain how Aladin is used to retrieve astronomical images, how a GLU record is created in order to represent your service in Aladin.

5.1 Aladin

5.1.1 Intro to Aladin

Aladin is a software tool for previewing astronomical images, created by Mr. Pierre Fernique, Thomas Bosch, Francoius Bonnarel and Anais Oberto from Centre de Donnees Astronomiques de Strasbourg. Aladin is a sky atlas and anyone who is willing to provide their observation collection to public can join this atlas. Aladin has a lot of different functionalities, for instance user can adjust image color and contrast, add astrometrical information manually in case it is missing or preview various sky observations through Aladin's Server selector. Aladin provides access to different observation images of various sky surveys.

5.1.2 Create a GLU record

In order to join sky atlas one either writes a glu record, which is a temporary issue, or publishes the collection to registry. For more about registry see Chapter 1.7. GLU record does not have to be loaded to Aladin everytime, it could be embeded.

GLU record basically allows user to search your observation images. It defines parameters which are sent in request, in a form of a url to your server. How your server responses is your thing and you have to configure it yourself. In our case it is done with the help of *conDesc* in resource descriptor (see chapter 3.2.2), where we defined SIAP responses to input parameters.

Throughout this section I will refer to GLU record example in Appendix E.

Firstly you define some metadata as you can see on lines 1 to 5. On line 6 you write your server access url, where the requests will be sent, and define arguments as parameters. In our case it is a SIAP service therefore there is *siap.xml* in the query. Then on lines 8 to 19 you define parameter descriptions and their data types according to your database. Lines 20 to 35 are definition of options that a user can choose from as you see on the figure 5.1. On line 36 you define the data type of results returned to the user. Next lines are optional. Line 40 is the name which will be shown in Aladin for your service.

5.1.3 How to use GLU record

Once you have written your own GLU record it is time to test it. Open Aladin and click on Open Local file in File menu. Choose your GLU record and hit enter. Go to File menu again and chose Open. A window of Server selector will open, as you can see on figure 5.1. In the left column find your

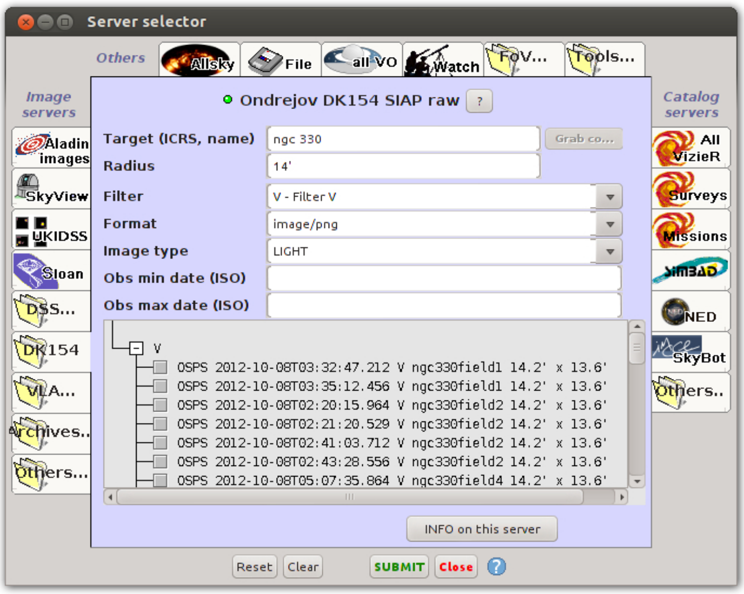

Figure 5.1: Aladin server selector

service. It will be called the same as you defined it in GLU record. In our case the name is DK154 as you can see on line 40. Click on the service and a request form will appear just as on figure 5.1. Once you filled in request data and sent it to your server you will receive a list or results. Check one or more of the results and press Submit button. Results are loaded to Aladin and you can view them. See figure 5.2.

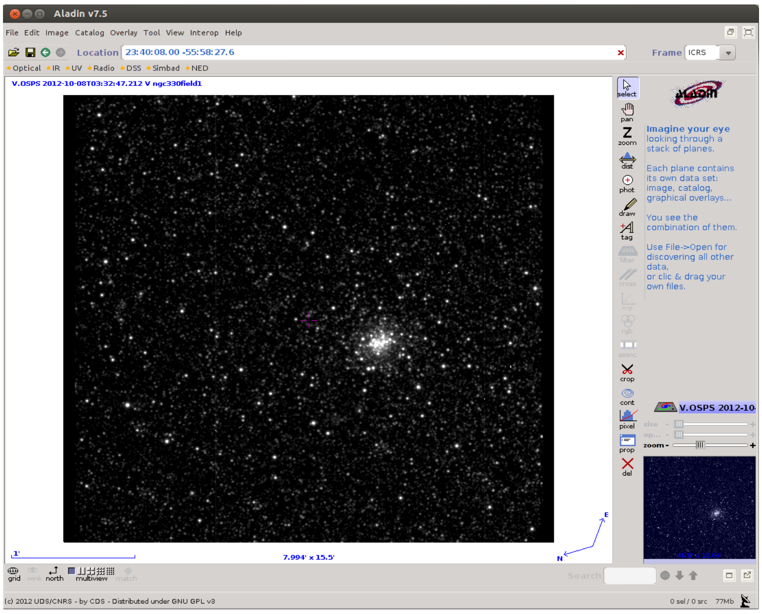

Figure 5.2: Aladin

In order to change color range, make image brighter or darker, hold *Shift* and right mouse button and move cursor horizontally. For a whole list of Aladin features see [11].

5.2 DaCHS web interface

Our image database is also accessible through GAVO DaCHS web interface.

- information about *dk154_rawdata* table is found on `http://vos2.asu.cas.cz/dk154_rawdata/q/sia/info`
- information about *dk154_reduced* table is found on `http://vos2.asu.cas.cz/dk154_reduced/q/sia/info`

By clicking HTML form you will be redirected to an html form for SIAP requesting as on figure 5.3. All fields in the form are generated by resource descriptor that we previously wrote. That means that you can add new fieds by adding more *condDesc* to your resource descriptor. Some fields are generated automatically based on the service that you defined.

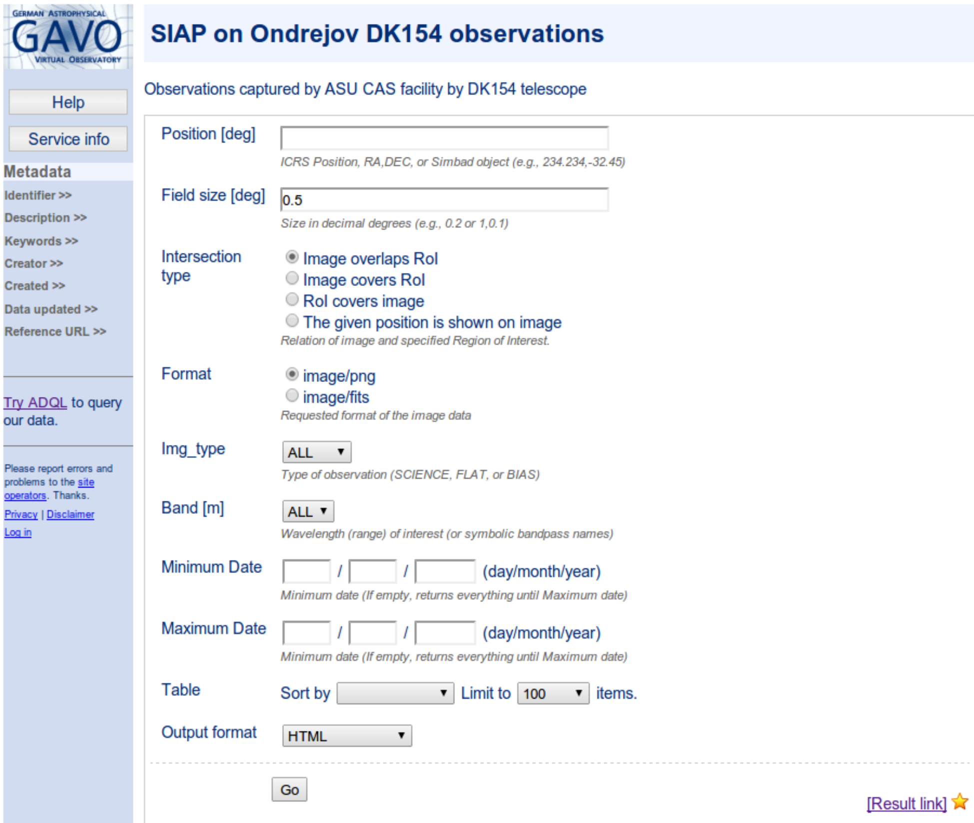

Figure 5.3: GAVO DaCHS SIAP query form

On the left you see a column with metadata, these are the metadata that you define at the begining of your descriptor, in my example in Appendix C theses are lines 4 to 7.

You can fill in html form fields or simply hit enter to display everything that is stored in the database. You will get a list of results. You can download images directly from here by clicking on their links. In case of FITS images you will be asked for login details.

Conclusion

The task of creating an Image server with SIAP support was accomplished. All observation images of DK154 survey are published and are accessible online to public. FITS images are protected and are available for access just for members of the science group who work with them. Preview images, created in the best possible quality in png format, are available for public use. They hold the same astrometrical information and therefore are very good for illustrational purposes.

Ingestion mechanism for publishing pictures is flexible, full support of SIAP based requests as well as TAP and ADQL is present. I am using some new SIAP features which have not been yet added to official standard, however it is planned.

The main goal, which is to present IVOA stadards and an IVOA compliant service has been achieved. This thesis fully explains how one can create such a service and how to manipulate with it. It is a good piece of work which will help many astronomers to take a step into new technological advances and standards which are presented by International Virtual Observatory Alliance.

Results of this thesis was presented on IVOA Interoperability meeting in Heidelberg to astronomical community, which was held from 12 to 17th of May 2013.

Bibliography

[1] ImageMagick @ONLINE. [Cited 2013-05-12]. Available at WWW: <http://www.imagemagick.org/>

[2] Independent JPEG group @ONLINE. [Cited 2013-05-12]. Available at WWW: <http://www.ijp.org>

[3] JPEG compression @ONLINE. [Cited 2013-05-12]. Available at WWW: <http://www.fileformat.info/mirror/egff/ch09_06.htm>

[4] Russian Virtual Observatory @ONLINE. [Cited 2013-05-12]. Available at WWW: <http://www.inasan.rssi.ru/eng/rvo/>

[5] UCD resolver @ONLINE. [Cited 2013-05-12]. Available at WWW: <http://dc.zah.uni-heidelberg.de/ucds/ui/ui/form>

[6] Demleitner, M.: The GAVO DaCHS Tutorial @ONLINE. [Cited 2013-05-12]. Available at WWW: <http://vo.ari.uni-heidelberg.de/docs/DaCHS/tutorial.html>

[7] Demleitner, M.: The GAVO Data Center Helper Suite @ONLINE. [Cited 2013-05-12]. Available at WWW: <http://vo.ari.uni-heidelberg.de/docs/DaCHS/>

[8] Hroch, F.: Fitspng @ONLINE. [Cited 2013-05-12]. Available at WWW: <http://integral.physics.muni.cz/fitspng/>

[9] Hroch, F.: Munipack, a general astronomical image processing software @ONLINE. [Cited 2013-05-12]. Available at WWW: <http://munipack.physics.muni.cz/>

[10] IVOA: VOTable Format Definition Version 1.3 @ONLINE. [Cited 2013-05-12]. Available at WWW: <http://www.ivoa.net/documents/VOTable/>

[11] Pierre Fernique, F. B., Thomas Bosch; Oberto, A.: Aladin Sky Atlas @ONLINE. [Cited 2013-05-12]. Available at WWW: <http://aladin.u-strasbg.fr/>

[12] Richmond, M.: Photometric systems and colors @ONLINE. [Cited 2013-05-12]. Available at WWW: <http://spiff.rit.edu/classes/phys445/lectures/colors/colors.html>

[13] Roelofs, G.: PNG, the definitive guide @ONLINE. [Cited 2013-05-12]. Available at WWW: <http://www.libpng.org/pub/png/book/>

Appendix A

Acronyms

ADQL Astronomical Data Query Language

DaCHS Data Center Helper Suite

FITS Flexible Image Transport System

GAVO German Astrophysical Virtual Observatory

GUI Graphical user interface

HDU Header Data Unit

IVOA International Virtual Observatory Alliance

NASA National Aeronautics and Space Administration

OSPS Ondřejov Southern Photometry Survey

PQL Parameterized Query Language

RA Right Ascension

ROI Region of Interest

SCS Simple Cone Search

SIAP Simple Image Access Protocol

TAP Table Access Protocol

UCD Unified Content Descriptor

URL Unified Resource Locator

VO Virtual Observatory

XML Extensible markup language

Appendix B

Tips for GAVO DaCHS installation

Installation of GAVO DaCHS is very well described in its documentation, which is accessible on `http://vo.ari.uni-heidelberg.de/docs/DaCHS/`. Here I will describe a concrete example of how it is installed on our sever.

We installed DaCHS from its source which is obtainable from `http://vo.ari.uni-heidelberg.de/soft/dachs`. It is also possible to install DaCHS from a package, see info at `http://vo.ari.uni-heidelberg.de/soft/repo`.

B.1 Source

GAVO DaCHS source is located in folder */usr/share/pyshared/gavo*. But it basically can be anywhere you put it. You can later use the source to debug if you come accross any problems.

B.2 Database

Database location: */home/gavoadmin/gavodb/* (administrator's home folder).

Database installation folder: */usr/share* (default).

Database configuration files: */etc/postgresql/8.4/pgdata* (default).

User *gavoadmin* is database administrator and the one who runs data ingestion and administers the whole server.

B.3 File system

System files are located at */var/gavo/* folder. Resource descriptors are in */var/gavo/inputs/*. My resource descriptors are in folders *dk154_rawdata* and *dk154_reduced*. Resource descriptor of my colleague Jiří Nádvorník, who works with me on this project, is located at *extract* folder. Folders */var/gavo/etc/* contains configuration files. Log files are stored at */var/gavo/logs/*

Appendix C

Resource descriptor

RAW images

```
<?xml version="1.0" encoding="iso-8859-1"?>

<resource schema="dk154_rawdata">
  <meta name="title">DK154 Chile telescope</meta>
  <meta name="creationDate"> 2012-04-27T00:00:00 </meta>
  <meta name="description">Observations captured by ASU CAS facility by DK154 telescope</meta>
  <meta name="subject">Stars: Variable</meta>

  <table id="raw_images" adql="True" onDisk="True" mixin="//siap#pgs">
      <mixin
              facilityName="'ASU CAS - NBI'"
              collectionName="'ASU CAS - NBI/DFOSC_FASU'"
              calibLevel="1"
      >//obscore#publishSIAP</mixin>
      <column name="exposure" tablehead="Exp. time" unit="s"
              ucd="time.duration;obs.exposure"/>
      <column name="telescope" type="text"
              ucd="meta.id;instr.tel"
              description="Telescope used for the observation"
              verbLevel="25"/>
      <column name="img_type" type="text"
              ucd="meta.code.class"
              description="Type of observation (SCIENCE, FLAT, or BIAS)"
              verbLevel="1"/>
      <column name="accref" type="text"
              description="internal file reference"/>
  </table>

  <data id="import_image_content" auto="false">
    <sources recurse="True" pattern="data/*.fits"/>
    <fitsProdGrammar qnd="True">
      <rowfilter procDef="//products#define">
        <bind key="owner">"beusers"</bind>
        <bind key="embargo">'2018-12-31'</bind>
        <bind key="table">"dk154_rawdata.raw_images"</bind>
      </rowfilter>

        <rowfilter name="addPNG">
          <code>
```

```
                yield row
                row["prodtblAccref"] = row["prodtblAccref"].replace("/data/", "/dataPNG/", 1)
                row["prodtblPath"] = row["prodtblAccref"].replace("/data/", "/dataPNG/", 1)
                row["prodtblEmbargo"] = '2010-12-31'
                row["prodtblPath"] = row["prodtblPath"][:-5]+".png"
                row["prodtblAccref"] = row["prodtblAccref"][:-5]+".png"
                row["prodtblMime"] = "image/png"
                yield row
              </code>
            </rowfilter>

        </fitsProdGrammar>

        <make table="raw_images">

          <rowmaker id="makeRawFrame">
            <var name="dateObs">dateTimeToMJD(parseTimestamp(@DATE_OBS.split(".")[0]))</var>
            <var name="obs_collection">"%s/%s"%(vars["ORIGIN"], vars["INSTRUME"])</var>
            <var name="cleanedObject">@OBJECT.split("_")[0]</var>
            <var key="filt">vars.get("FILTB") or vars.get("FILTA")</var>

            <apply procDef="//siap#computePGS">
                    <bind key="missingIsError">False</bind>
            </apply>
              <var key="canonicalFilterName">
                {
                "R": "Bessel R",
                "I": "Bessel I",
                "U": "Bessel U",
                "B": "Bessel B",
                "V": "Bessel V",
                }.get(vars.get("filt"))

              </var>

            <apply name="Halpha">
              <code>
                if vars["FILTB"]=="H-alpha_narrow":
                vars["bandpassLo"] = 6600
                vars["bandpassHi"] = 6700
                vars["bandpassRefval"] = 6654
                vars["bandpassUnit"] = "m"
                vars["filt"] = "Ha"
              </code>
            </apply>

            <apply procDef="//siap#setMeta">
              <bind key="title">"OSPS %s %s %s"% (vars["DATE_OBS"], vars["filt"], @OBJECT)</bind>
              <bind key="instrument">@INSTRUME</bind>
              <bind key="dateObs">@dateObs</bind>
              <bind key="bandpassId">vars["FILTB"]</bind>
              <bind key="bandpassLo">vars["bandpassLo"]</bind>
              <bind key="bandpassHi">vars["bandpassHi"]</bind>
              <bind key="bandpassRefval">vars["bandpassRefval"]</bind>
            </apply>

            <apply procDef="//siap#getBandFromFilter">
              <bind key="sourceCol">"canonicalFilterName"</bind>
            </apply>

            <map key="bandpassId" source="filt"/>
            <map key="telescope" source="TELESCOP"/>
```

```
    <map key="exposure" source="EXPOSURE"/>
    <map key="img_type" source="IMAGETYP"/>
    <map dest="accref">\inputRelativePath</map>
   </rowmaker>

  </make>
</data>

<data id="update_image_content" original="import_image_content" updating="True">
  <sources recurse="True" pattern="data/*.fits">
   <ignoreSources fromdb="select distinct accref from dk154_rawdata.raw_images"/>
  </sources>
</data>

 <service id="sia" allowed="siap.xml,form">
  <meta name="title">SIAP on Ondrejov DK154 observations</meta>
  <meta name="shortName">dk154_rawdata raw_images</meta>
  <meta name="sia.type">Pointed</meta>
  <siapCore id="sia" queriedTable="raw_images">

   <condDesc original="//siap#protoInput"/>
   <condDesc original="//siap#humanInput">
    <inputKey original="//siap#base_FORMAT" name="hFORMAT" tablehead="Format">
     <property name="notForRenderer">siap.xml</property>
     <values default="" id="base_Format_values">
       <option title="image/jpeg">image/jpeg</option>
       <option title="image/fits">image/fits</option>
     </values>
    </inputKey>
   </condDesc>

   <condDesc>
    <inputKey original="img_type" showItems="5" id="ImgType_input" type="text">
     <values default="" id="base_Format_values">
       <option title="ALL">ALL</option>
       <option title="FLAT">FLAT</option>
       <option title="BIAS">BIAS</option>
       <option title="LIGHT">LIGHT</option>
     </values>
    </inputKey>
     <phraseMaker>
      <code>
        key = inputKeys[0].name
        val = inPars.get(key, None)
        if val=="ALL":
            yield "img_type IN (select distinct img_type
                      from dk154_rawdata.raw_images)"
        else:
            yield "img_type = '%s'"%(val)
      </code>
     </phraseMaker>
   </condDesc>

   <condDesc id="bandCond">
    <inputKey name="BAND" type="text" description="Wavelength (range)
        of interest (or symbolic bandpass names)" unit="m"
         std="True" utype="ssa:DataId.Bandpass">
    <values default="" id="base_BAND_values">
        <option title="ALL">ALL</option>
        <option title="B">B</option>
        <option title="I">I</option>
        <option title="R">R</option>
```

```
                <option title="U">U</option>
                <option title="V">V</option>
                <option title="Ha">Ha</option>
        </values>
        </inputKey>
          <phraseMaker>
              <code>
                key = inputKeys[0].name
                lit = inPars.get(key, None)
                if lit is None:
                        return
                if lit=="ALL":
                        yield "bandpassId IN (select distinct bandpassId
                                                    from dk154_rawdata.raw_images)"
                else:
                        yield pql.PQLPar.fromLiteral(lit, key).getSQL("bandpassId", outPars)
              </code>
          </phraseMaker>
      </condDesc>

      <condDesc combining="True">
        <inputKey name="date_min" type="date"  ucd="pos.eq.date"
                description="Minimum date (If empty, returns everything until Maximum date)"
                tablehead="Minimum Date">
        </inputKey>

        <inputKey name="date_max" type="date" ucd="pos.eq.date"
                description="Minimum date (If empty, returns everything until Maximum date)"
                tablehead="Maximum Date">
        </inputKey>

        <phraseMaker id="DatePhrase" name="dateSQL">
          <code>
            if (inPars["date_min"]):
                    minTS = dateTimeToMJD(datetime.datetime.combine(
                            inPars["date_min"],
                            datetime.datetime.strptime("0:0:0", "%H:%M:%S").time()))
            else:
                    minTS = 0.0

            if (inPars["date_max"]):
                    maxTS = dateTimeToMJD(datetime.datetime.combine(
                            inPars["date_max"],
                            datetime.datetime.strptime("23:59:59", "%H:%M:%S").time()))
            else:
                    maxTS= dateTimeToMJD(datetime.datetime.now())

            yield "dateObs BETWEEN %%(%s)s AND %%(%s)s"%(
                    base.getSQLKey("date_min", minTS, outPars),
                    base.getSQLKey("date_max", maxTS, outPars))
          </code>
        </phraseMaker>
      </condDesc>
    </siapCore>
  </service>
</resource>
```

Appendix D

Data examples

An example of RAW FITS primary header

```
SIMPLE  =                    T / conform to FITS standard
BITPIX  =                   32 / unsigned short data
NAXIS   =                    2 / number of axes
NAXIS1  =                 2148 / length of data axis
NAXIS2  =                 2048 / length of data axis
EXTEND  =                    T / this is FITS with extensions
HISTORY Created with RTS2 version 0.9.4 build on Oct  4 2012 11:01:12.
CTIME   =           1349667167 / exposure start (seconds since 1.1.1970)
USEC    =               212423 / exposure start micro seconds
JD      =      2456208.6477662 / exposure JD
DATE-OBS= '2012-10-08T03:32:47.212' / start of exposure
OBJECT  = 'ngc330field1'       / object name
EXPOSURE=                 120. / exposure length in seconds
EXPTIME =                 120. / exposure length in seconds
INSTRUME= 'DFOSC_FASU'         / name of the data acqusition instrument
TELESCOP= 'DK-1.54 '           / name of the data acqusition telescope
ORIGIN  = 'ASU CAS - NBI'      / organisation responsible for data
FOC_NAME= 'FASC    '           / name of focuser
UTSTART = '2012-10-08T03:32:47.212'
EQUINOX = '2000.   '
CCD_TYPE= 'E2V44-82'           / camera type
CCD_SER = 'Michelle'           / camera serial number
SCRIPREP=                   16 / script loop count
SCRIPT  = 'exe /etc/rts2/multiexp.py' / script used to take this images
SCR_COMM= '        '           / comment recorded for this script
COMM_NUM=                    0 / comment order within current script
BINNING = '1x1     '           / [pixelX x pixelY] chip binning
BINX    =                    1 / [pixels] binning along X axis
BINY    =                    1 / [pixels] binning along X axis
CCD_TEMP=              -119.87 / CCD temperature
HIERARCH DATA_CHANNELS =     1 / total number of data channels
CHAN    =                    2 / channels on/off
CHAN1   =                    T
CHAN2   =                    F
GAIN1   =                   1. / Individual digital gain in channel 1
GAIN2   =                   1. / Individual digital gain in channel 2
CDSG    =                50000 / Fundamental cds-gain
SECPPIX =                0.396 / Arcseconds per pixel
CCDPSIZ =               0.0135 / pre-binning pixel size [mm]
```

```
IMAGETYP= 'LIGHT   '           / LIGHT, DARK, BIAS or FLAT
SHUTTER =                    0 / shutter state (0 - opened, 1 - closed)
FILTA   = 'empty   '           / used filter number
FILTB   = 'V       '           / used filter number
MNT_NAME= 'TASC    '           / name of mount
LATITUDE=              -29.263 / observatory latitude
LONGITUD=             -70.7403 / observatory longitude
ALTITUDE=                2363. / observatory altitude
ORIRA   = '00:56:46.530'       / original position (J2000) RA
ORIDEC  = '-72:27:12.00'       / original position (J2000) DEC
OFFSRA  = '00:00:00.000'       / object offset RA
OFFSDEC = '+00:00:00.00'       / object offset DEC
OBJRA   = '00:56:46.530'       / telescope FOV center position (J200) - with off
OBJDEC  = '-72:27:12.00'       / telescope FOV center position (J200) - with off
TARRA   = '05:38:30.148'       / target position (J2000) RA
TARDEC  = '-29:15:08.14'       / target position (J2000) DEC
CORR_RA = '00:00:00.000'       / correction from closed loop RA
CORR_DEC= '+00:00:00.00'       / correction from closed loop DEC
HIERARCH MO_RTS2RA = '00:00:00.000' / [deg] RTS2 model offsets RA
HIERARCH MO_RTS2DEC = '+00:00:00.00' / [deg] RTS2 model offsets DEC
TELRA   = '00:56:46.530'       / mount position (read from sensors) RA
TELDEC  = '-72:27:12.00'       / mount position (read from sensors) DEC
JD_HELIO=     2456208.64962137 / heliocentric JD
TEL_ALT =              46.1063 / horizontal telescope coordinates altitude
TEL_AZ  =              353.721 / horizontal telescope coordinates azimuth
PARKTIME= -9.1191291391491E-36 / Time of last mount park
AIRMASS =     1.38682453082697 / Airmass of target location
HA      = '23:01:43.596'       / Location hour angle
LST     = '23:58:30.126'       / Local Sidereal Time
MOVE_NUM=                    0 / number of movements performed by the driver; us
CORR_IMG=                    0 / ID of last image used for correction
MNT_ROTA=                 180. / mount rotang
MNT_FLIP=                    1 / telescope flip
HIERARCH aguider_offsRA = '24:00:00.000' / Offset of telescope get from autoguid
HIERARCH aguider_offsDEC = '+00:00:00.60' / Offset of telescope get from autogui
HIERARCH aguider_stop =      F / STOP Signal for autoguider script
TELSTAT = 'SKY TRACK'          / Telescope status at end of observing
PRESSURE=                767.1 / Atmospheric pressure [mbar]
TEMPERAT=                   5. / Outside temperature of enviroment [Celsius]
MNT_INFO=     1349667166.28534 / time when mount informations were collected
SUN_ALT =    -52.2165131185685 / solar altitude
SUN_AZ  =      23.577500383514 / solar azimuth
FOC_TYPE= '        '           / focuser type
FOC_POS =               38.714 / focuser position
FOC_TAR =     38.8324165344238 / focuser target position
FOC_DEF =        38.8324165981 / default target value
FOC_FOFF=                   0. / offset from focusing routine
FOC_TOFF=                   0. / temporary offset for focusing
CAM_FILT=                    0 / filter used for image
BZERO   =           2147483648 / offset data range to that of unsigned long
BSCALE  =                    1 / default scaling factor
AVERAGE =     12852.3593786371 / average value of image
STDEV   =     2039726.04348368 / standard deviation value of image
CTYPE1  = 'RA---TAN'           / WCS transformation type
CTYPE2  = 'DEC--TAN'           / WCS transformation type
CRVAL1  =            14.193875 / reference value on 1st axis
CRVAL2  =    -72.4533333333333 / reference value on 2nd axis
CRPIX1  =                1074. / reference pixel of the 1st axis
CRPIX2  =                1024. / reference pixel of the 2nd axis
CDELT1  =             -0.00011 / delta along 1st axis
CDELT2  =              0.00011 / delta along 2nd axis
CROTA2  =              359.488 / rotational angle
```

```
DETSIZE = '[0:2148,0:4612]'    / unbined detector size
DATASEC = '[1:2148,1:2048]'    / data binned section
DETSEC  = '[0:2148,2049:4097]' / unbinned section of detector
TRIMSEC = '[52:2099,1:2048]'   / TRIM binned section
CCDSUM  = '1 1     '           / CCD binning
LTV1    =                  -0. / image beginning - detector X coordinate
LTV2    =               -2049. / image beginning - detector Y coordinate
LTM1_1  =                   1. / delta along X axis
LTM2_2  =                   1. / delta along Y axis
DTV1    =                    0 / detector transformation vector
DTV2    =                    0 / detector transformation vector
DTM1_1  =                    1 / detector transformation matrix
DTM2_2  =                    1 / detector transformation matrix
FILTER  = 'empt    '           / camera filter as string
DATE    = '2012-10-08T03:35:11.067' / creation date
END
```

An example of astrometric reduction in reduced FITS file primary header

```
COMMENT === Astrometric Solution by Munipack ===
COMMENT Type: absolute
COMMENT Reference catalogue: UCAC4 Catalogue (Zacharias+, 2012)
COMMENT Projection: GNOMONIC
COMMENT Number of objects used = 255
COMMENT RMS =    417.6E-03 [arcsec]
COMMENT Scale =     0.3961254026     +-   1.3E-02 [arcsec/pix]
COMMENT cos(pa) =    0.9999441176 +-   1.5E-03
COMMENT sin(pa) =    0.0105717408 +-   1.4E-01
COMMENT Position Angle (pa)  =    0.6057274146 +-   8.2E+00 [deg]
COMMENT Alpha center projection (CRVAL1) =   14.1966487365 +-   1.2E-03 [deg]
COMMENT Delta center projection (CRVAL2) =  -72.4533758281 +-   1.2E-03 [deg]
COMMENT Horizontal center (CRPIX1) =  1074.000 [pix]
COMMENT Vertical   center (CRPIX2) =  1024.000 [pix]
COMMENT Catalogue RA,DEC [deg]        Data X,Y [pix]     Residuals [arcsec]
COMMENT   14.47896650 -72.49588530  297.995  644.079    72.9E-03  -192.6E-03
COMMENT   14.18106240 -72.50419250 1111.298  561.995   194.9E-03   -83.5E-03
COMMENT   14.01397180 -72.48556340 1569.740  725.872   257.3E-03  -100.0E-03
COMMENT   14.56600150 -72.48636280   60.315  732.131   148.7E-03  -157.6E-03
COMMENT   14.51512890 -72.42387530  202.314 1299.361   173.3E-03    -1.2E-03
COMMENT   14.09406560 -72.47661920 1352.198  809.386    62.4E-03   -68.5E-03
COMMENT   14.06064090 -72.46182950 1445.254  943.118   208.3E-03    -1.3E+00
COMMENT   14.52733920 -72.34999230  173.664 1970.153  -382.9E-03   232.2E-03
COMMENT   14.41530060 -72.51405700  470.253  478.246   265.9E-03  -157.0E-03
COMMENT   14.07048800 -72.46307700 1418.178  931.963   140.5E-03  -225.2E-03
COMMENT   13.82985710 -72.43501590 2082.827 1176.993  -472.7E-03    24.6E-03
COMMENT   13.97428950 -72.47474530 1679.368  821.230   344.7E-03   426.5E-03
COMMENT   14.10550890 -72.47172120 1321.294  854.343   154.1E-03    46.3E-03
```

An example of VOTable structure


```xml
<VOTABLE>
  <DESCRIPTION>
  Observations captured by ASU CAS facility by DK154 telescope
  </DESCRIPTION>
    <RESOURCE type="results">
      <DESCRIPTION>
      Observations captured by ASU CAS facility by DK154 telescope
      </DESCRIPTION>
      <INFO name="QUERY_STATUS" value="OK"/>
      <TABLE name="dl154_rawdata">
        <DESCRIPTION>
        Observations captured by ASU CAS facility by DK154 telescope
        </DESCRIPTION>
        <DESCRIPTION>Type of observation (SCIENCE, FLAT, or BIAS)
        </DESCRIPTION>
        </FIELD>
        <FIELD ID="accref" arraysize="*" datatype="char" name="accref"
        ucd="VOX:Image_AccessReference" utype="Access.Reference">
        <DESCRIPTION>Access key for the data</DESCRIPTION>
        </FIELD>
        <FIELD ID="mime" arraysize="*" datatype="char" name="mime"
        ucd="VOX:Image_Format" utype="Access.Format">
        <DESCRIPTION>MIME type of the file served</DESCRIPTION>
        </FIELD>
        <DATA>
        <TABLEDATA>
          <TR>
            <TD>0.002</TD>
            <TD>BIAS</TD>
            <TD>
            http://vos2.asu.cas.cz/getproduct/dk154_rawdata/
            data/20121208/bias_000021.fits
            </TD>
            <TD>image/fits</TD>
          </TR>
        </TABLEDATA>
        </DATA>
      </TABLE>
    </RESOURCE>

</VOTABLE>
```

Appendix E

GLU record

```
%ActionName Gavo_sia_raw
%Description Ondrejov DK154 SIAP raw
%Owner ASU CAS
%DistribDomain ALADIN
%VersionNumber 1  2013/24/02 19:04
%Url http://vos2.asu.cas.cz/dk154_rawdata/q/sia/siap.xml?POS=$1,$2&SIZE=$3
&band=$4&FORMAT=$5&img_type=$6&date_min=$7&date_max=$8
%Param.Description $1=Right Ascension
%Param.Description $2=Declination
%Param.Description $3=Radius (arcmin)
%Param.Description $4=Filter
%Param.Description $5=Format
%Param.Description $6=Image type
%Param.Description $7=Obs min date (ISO)
%Param.Description $8=Obs max date (ISO)
%Param.DataType $1=Target(RAd)
%Param.DataType $2=Target(DEd)
%Param.DataType $3=Field(RADIUSd)
%Param.DataType $6=Input(IMG)
%Param.Value    $3=0.17
%Param.Value    $4=R - Filter R
%Param.Value    $4=I - Filter I
%Param.Value    $4=B - Filter B
%Param.Value    $4=U - Filter U
%Param.Value    $4=V - Filter V
%Param.Value    $4=Ha - Filter Ha
%Param.Value    $4=ALL - All filters
%Param.Value    $5:image/png
%Param.Value    $5:image/fits
%Param.Value    $6:LIGHT
%Param.Value    $6:BIAS
%Param.Value    $6:FLAT
%Param.Value    $6:ALL - All types
```

```
%Param.Value    $7:
%Param.Value    $8:
%ResultDataType Mime(sia/xml)
%Doc.User http://vo.ari.uni-heidelberg.de/docs/DaCHS/
%Aladin.Label Gavo-Ondrejov (Asu DK154 SIAP RAW)
%Aladin.LabelPlane OSPS $1 $2 $4
%Aladin.Menu DK154
%Aladin.Logo ivoa.gif


%ActionName Gavo_sia_reduced
%Description Ondrejov DK154 SIAP reduced
%Owner ASU CAS
%DistribDomain ALADIN
%VersionNumber 1  2013/24/02 19:04
%Url http://vos2.asu.cas.cz/dk154_reduced/q/sia/siap.xml?POS=$1,$2&SIZE=$3
&band=$4&FORMAT=$5&date_min=$6&date_max=$7
%Param.Description $1=Right Ascension
%Param.Description $2=Declination
%Param.Description $3=Radius (arcmin)
%Param.Description $4=Filter
%Param.Description $5=Format
%Param.Description $6=Obs min date (ISO)
%Param.Description $7=Obs max date (ISO)
%Param.DataType $1=Target(RAd)
%Param.DataType $2=Target(DEd)
%Param.DataType $3=Field(RADIUSd)
%Param.Value    $3=0.17
%Param.Value    $4=R - Filter R
%Param.Value    $4=I - Filter I
%Param.Value    $4=B - Filter B
%Param.Value    $4=U - Filter U
%Param.Value    $4=V - Filter V
%Param.Value    $4=Ha - Filter Ha
%Param.Value    $4=ALL - All filters
%Param.Value    $5:image/png
%Param.Value    $5:image/fits
%Param.Value    $6:
%Param.Value    $7:
%ResultDataType Mime(sia/xml)
%Doc.User http://vo.ari.uni-heidelberg.de/docs/DaCHS/
%Aladin.Label Gavo-Ondrejov (Asu DK154 SIAP REDUCED)
%Aladin.LabelPlane OSPS $1 $2 $4
%Aladin.Menu DK154
%Aladin.Logo ivoa.gif
```

Appendix F

Contents of enclosed CD

```
readme.txt ...................... the file with CD contents description
exe ............................. the directory with executables
src ............................. the directory of source codes
    wbdcm ....................... implementation sources
    thesis ...................... the directory of LaTeX source codes of the thesis
text ............................ the thesis text directory
    thesis.pdf .................. the thesis text in PDF format
    thesis.ps ................... the thesis text in PS format
```