Telescope Automation through CelestialGrid

Bret L. Little

A senior thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Bachelor of Science

J. Ward Moody, Advisor

Department of Physics and Astronomy

Brigham Young University

April 2011

Copyright © 2011 Bret L. Little

All Rights Reserved

ABSTRACT

Telescope Automation through CelestialGrid

Bret L. Little
Department of Physics and Astronomy
Bachelor of Science

CelestialGrid is an all-inclusive observatory system which automates the capture, retrieval, reduction, and long-term archival of astronomical data. In addition to providing a graphical interface for all observatory operations, CelestialGrid simplifies nightly observation planning by automating the capture of star standardization frames. This paper discusses the development, installation, and operation of CelestialGrid.

Keywords: Astronomy Software, CelestialGrid, Observatory Automation, Robotic Telescope, Remote Systems, Observational Astronomy.

ACKNOWLEDGMENTS

I would like to thank the Department of Physics and Astronomy for providing me with opportunities to be involved with exciting research at BYU. Especially I would like to thank J. Ward Moody for his instruction, advisement, and support throughout the last 6 years. He is an amazing mentor who incomparably influenced the course of this project and the course of my future career. Last of all and most importantly, I would like to thank my wife Tearsa for her continual love and support. This project would not have made it to this point without her.

Contents

Chapter 1

Introduction

Observatories at Brigham Young University (BYU) and around the world require extensive manual interaction to ensure continual operation. This is not necessary due to modern computer technology. A recently push has begun to automate the everyday operations of research observatories. The Canada France Hawaii Telescope [1], the Dominion Astrophysical Observatory [2], and the Fairborn Observatory [3] are now operating autonomously. These observatories have designed computer software to automate their research and logistical needs. After the Remote Observatory for Variable Object Research (ROVOR) at BYU started remote operation in Spring, 2008, we looked for a solution in automating the observatory. Software packages at other observatories are proprietary, complicated, and not entirely hardware compatible with ROVOR. For this reason I have written a simple observatory control package to automate not only ROVOR, but potentially other small observatories around the world.

CelestialGrid is a software package designed from the ground up to minimize human interaction with the observatory and telescope systems. Reduced interaction leads to fewer human introduced errors and enables researchers to focus on the science of astronomy and not the mechanics of nightly observation. Although CelestialGrid requires further development, its current release represents a significant step forward in automating observatories at BYU.

1.1 Background

An observatory is a facility used to observe astronomical events. Observatories have been built throughout history to better understand the heavens. Modern observatories primarily feature different types of telescopes. A robotic autonomous observatory is capable of automatically performing observatory tasks while adapting to observatory site conditions [4]. Historically the development of autonomous observatories has progressed with the development of computers. The first robotic observatory was built in 1969 at the University of Wisconsin and operated for three days before failing [5]. Afterwards and progressively through the 1970s and 1980s computer technology and robotic observatories improved. These more powerful computers enabled increased automation and control over observatory tasks. Why automate an observatory? The most obvious reason may be that an astronomer does not always want to stay up all night operating a telescope. Such operation is often tedious, time consuming, and error prone. Thus, an automated observatory with specific programmable routines simplifies observational astronomy.

Another reason for automating an observatory is to enable researchers to coordinate research efforts for variable targets of opportunity. Variable targets of opportunity are astronomical objects which periodically "flare" or "burst" in brightness. These changes in brightness are often hard to predict and last for brief time intervals. In particular Gamma-ray bursts (GRBs) are extremely energetic objects that flare in the high energy gamma spectrum. Although the GRBs are not observable from the surface of Earth, orbital space detectors transmit GRB coordinates to the surface for an optical spectrum followup. A GRB can last for as short as a few minutes or as long as multiple days. Because the time domain is uncertain, it is critical that ground based observatories quickly and accurately followup with GRB sightings; a robotic observatory is ideal for the task.

A blazar is an active galactic core. Blazars are similar to GRBs in that they periodically flare over widely varying time domains. Although it happens often, the cause of this flaring is unknown. Catching them while they are flaring is essential to understand the physics behind the

blazars. Undergraduate students Cameron Pace and Richard Pearson successfully monitored the blazar Markarian 501 throughout 2008 and 2009 (see Figure 1.3). Using data from ROVOR, both Cameron and Richard published their senior thesis' in 2010 [6] [7]. Because of robotic observatories, we are able to accurately monitor and quickly follow up on flaring blazars.

1.2 The Remote Observatory for Variable Object Research

The Remote Observatory for Variable Object Research (ROVOR) is a 16 inch RC Optical telescope sited 12 miles north west of Delta Utah. It has been built to remotely monitor bright objects that vary with time such as variable stars, cataclysmic variables, GRBs, and active galactic nuclei (AGN) including blazars, quasars, Seyfert nuclei and Low Ionization Nuclear Emission Regions (LINERS). ROVOR has been designed from the ground up with off the shelf materials, making it a cost effective tool for modern astronomical research.

CelestialGrid was originally designed to simplify the operation of ROVOR. Communication with ROVOR was originally done entirely through remote desktop. A user would remote desktop into the system, manually open the dome, turn on power to the telescope and equipment, and load an Orchestrate script to start observing. This system of operation continued for the first year of ROVOR's operation. Initially, problems primarily arose due to the limited internet connection. Often the observatory was inaccessible simply due to the slow internet connection. The slow internet connection also made remote operation troublesome and error prone.

Despite being problematic, throughout its first year of operation ROVOR successfully provided research quality data. Specifically, ROVOR successfully observed the active galaxy Markarian 501 throughout the year 2009. Figure 1.3 shows data from ROVOR of Markarian 501 being compared to data from other observatories from around the world. In addition to AGN research, previous undergraduate student Richard Pearson successfully recorded a GRB afterglow on April 30, 2010

Figure 1.1 The Remote Observatory for Variable Object Research (ROVOR). The observatory features a unique dome system called the Liffereth Dome. The Liffereth Dome is a pull-off roof designed for small telescopes and other observational equipment. It was specifically designed for the needs of the ROVOR project. The roof itself is completely removed from the observatory housing walls and cranked off to the side below the telescope horizon. Two swing arms on either side of the observatory work in unison to lift the roof off the structure over the telescope, and down and away into a cleared location. Pull-off torque is provided by a threaded rod connected to an electric motor at the back of the building. As the motor rotates, the threads turn through a threaded sleeve connected directly to the support arms.

(see Figure 1.2).

Due to the troublesome nature of operating ROVOR, development of CelestialGrid began during the Winter of 2010. The first successful operation of ROVOR took place during June, 2010. Since assuming full control of observatory operations in Fall, 2010, ROVOR, through CelestialGrid, has continued to provide data of AGNs and GRBs.

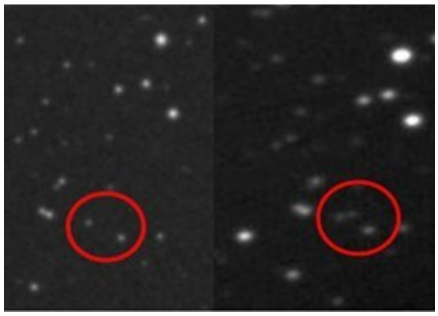

Figure 1.2 Recorded in the Gamma-ray burst coordination network as GCN Circular #9318, undergraduate student Richard Pearson used ROVOR to capture a Gamma-ray burst optical afterglow.

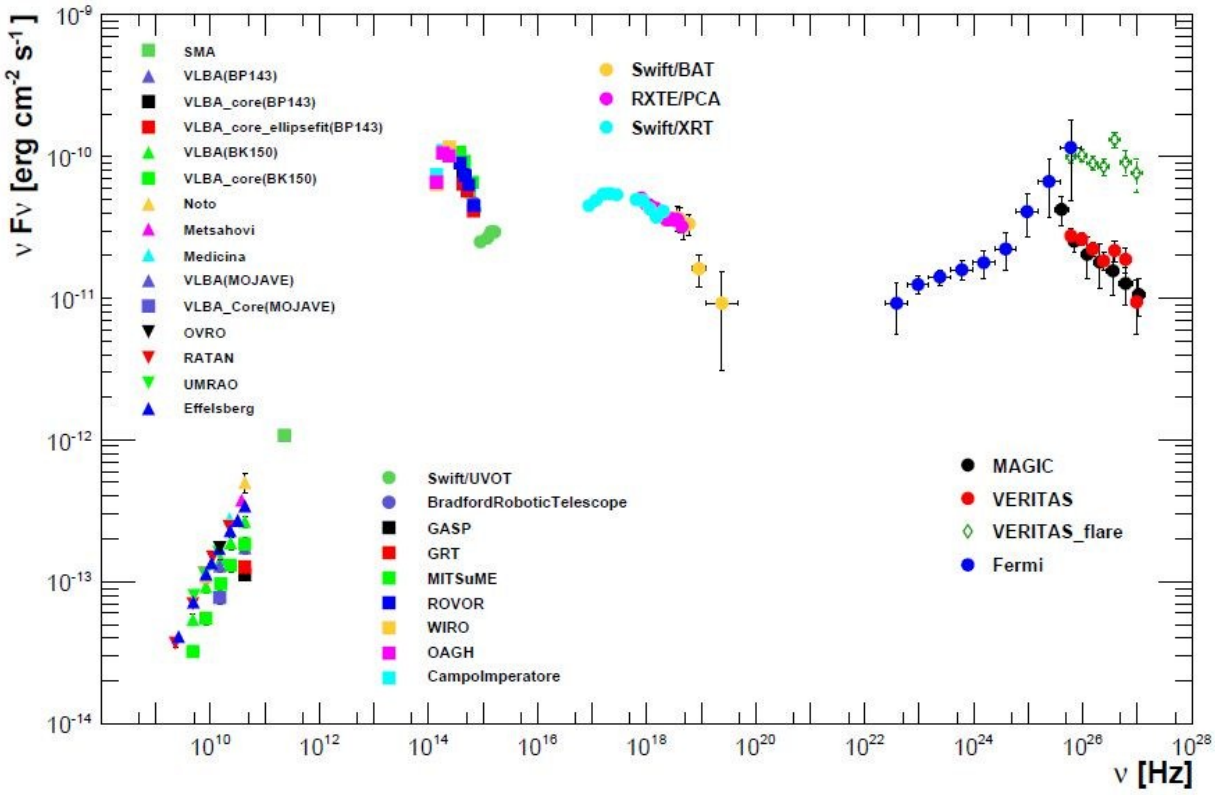

Figure 1.3 Data from the active galaxy Markarian 501. ROVOR data is compared to data from observatories around the world. [8]

Chapter 2

System Summary

In this chapter, I present a summary of the software architecture of CelestialGrid. CelestialGrid itself has two separate components: a client side application (CelestialGrid Client), and the main server software (CelestialGrid Server). The client needs to be installed on the observatory computers and the server software is run at BYU or on a personal computer. In addition to giving a technical software description, I will present the hardware and software requirements and instructions on how to install and configure a new installation of CelestialGrid. Throughout the technical discussion of CelestialGrid, refer to Figure 2.2 for a schematic on how the system is set up on ROVOR.

2.1 Technical Software Description

The software of CelestialGrid is almost exclusively developed in *Oracle Java* with access to a *MySQL* database. References to *Oracle Java*, *MySQL* and any other software packages are in Appendix A. This development choice enables the software to be platform independent on the server side of the application. The client side of the application requires *Microsoft Windows*. Software Requirements in Section 2.3 provides additional information on the separate software requirements

for CelestialGrid. The software is distributed and packaged as a *Java Web Start* application. *Java Web Start* technology enables all installations of the server software to automatically update on startup. The graphical user interface heavily relies upon the *Java* graphics libraries *Abstract Window Toolkit(AWT)* and *Swing*. The CelestialGrid source code is thoroughly documented through *Javadoc* (available in Appendix E Code Documentation).

The main purpose of the CelestialGrid client is to process remote commands from the CelestialGrid server by interfacing with local software installed on the observatory computers. Celestial Grid interfaces with the following five packages:

1. The Weather Station – The weather station software is managed by a *C++* application (available when installing the client, see Setup and Configuration Section 2.4) which periodically dumps local weather information to a text file. CelestialGrid retrieves weather information by periodically reading and parsing this text file. Table 2.1 describes the format of the weather text file. Conformity to this format is important for CelestialGrid to properly parse and retrieve weather information.

2. *LabView* – Because the system was designed for operation on ROVOR, the dome opening and closing system is managed by an application written in *LabVIEW*. This application only interfaces with ROVORs unique dome system (more on the dome system in Figure 1.1). More traditional dome systems can easily be controlled with added functionality through *Software Bisque AutomaDome*™ .

3. IP Power Switch – Through this physical device the power to separate vital components can be remotely turned on or off. Therefore, the power to the entire telescope system (mount, camera, filterwheel, focuser, and web cameras) can easily be managed. The IP Power Switch itself is a web server, and interfacing with it requires particular commands to be posted directly to its http web address. This is done by putting command variables into the address

Keyword	Data Type
Date	Tuesday Feb 22 10:34:45 2011
Wind Direction	Decimal (degrees)
Wind Speed	Decimal (m/h)
Temperature	Decimal (F)
Humidity	Decimal (percentage)
Pressure	Decimal (N/m^2)
Solar Intensity	Decimal (W/m^2)

Figure 2.1 The format of the text file written by the weather station. Conformity to this format is important for CelestialGrid to properly parse and retrieve weather information.

bar itself. If the power switch is given the address 192.168.0.1, then a command to query the power status would be: http://192.168.0.1/Set.cmd?CMD=GetPower. Code Documentation Appendix E has more information on posting to a web address.

4. Digital Video Recorder (DVR) – The observatory web cameras are all connected to a central DVR system. The DVR contains a web server, and interfacing with it requires particular commands to be posted directly to its http web address.

5. *Software Bisque Orchestrate* (or just *Orchestrate*) – Interfacing with *Orchestrate* enables communication to the mount, camera, filter wheel, and focuser. *Orchestrate* periodically reads and executes observing scripts. These scripts provide telescope control by tieing into all other *Software Bisque* products. Upon completion, *Orchestrate* outputs the scripts with specific comments about the success of each element within the script. CelestialGrid feeds *Orchestrate* individual scripts and reads the output results. Scripts fed to *Orchestrate* only image a couple objects at a time; therefore, CelestialGrid can easily interrupt the nightly routine with new observing parameters.

CelestialGrid is built around the server control system, with clients installed at observatory locations. Communication between the server and client instances is managed through the Celes-

Column Name	Data type	Options	Description
OBJ_ID	Int(50)	Primary	Unique ID
RESOLUTION	Int(16)	Null default	Field of view—Arc Seconds
RA	Varchar(20)	Null default	Right Ascension — 00:00:00.0
DC	Varchar(20)	Null default	Declination—00:00:00.0
FILTER	Int(8)	Null default	
EXPOSURE	Int(16)	Null default	Exposure time in seconds.
DATE	Date	Null default	Date image taken
DIRECTORY	Varchar(100)	Null default	File location.
OBS	Varchar(100)	Null default	Observatory ID
NOTE	Varchar(100)	Null default	Other information

Table 2.1 As images are taken, they are archived into this *MySQL* database within a table of the following format.

tialGrid Protocol. This network protocol defines how the system communicates through a Transmission Control Protocol/Internet Protocol (TCP/IP) connection. Table 2.2 describes the different commands available in the Celestial Grid Protocol.

CelestialGrid Server provides a batch utility (command-line) for archiving images from all connected observatories. Images may be automatically or manually retrieved from remote observatories. Because automatic retrieval is slow depending on the internet connection speed, data can also be retrieved manually and entered into long term storage. The image files themselves are archived on high capacity storage hard-drives. In order to minimize data loss due to hardware failure, these hard drives should be placed in a RAID configuration. Automatic retrieval is performed through the File Transfer Protocol (FTP). Files to be transferred include: web camera images, image data files, weather data files, and *Orchestrate* scripts. After images are taken and downloaded back to the university, they are archived into a *MySQL* database. Header information on each image is stored in this database. Table 2.1 describes the schema of the CelestialGrid database.

Command	Type	Value	Description
GetAAIObject	stat	string	Position of interrupt transient object
GetAAIStatus	stat	int	0 - Nothing found, 1 - Object found
GetDome	stat	int	0 - Closed, 1 - Open, 2 - Busy
GetFOV	stat	int	Get telescope field of view (arc-seconds)
GetHumid	stat	int	Query humidity
GetIAI	stat	int	Boolean, IAI observatory type
GetName	stat	string	Query observatory Name
GetPosition	stat	string	Telescope position
GetPower	stat	string	String of format "0:0:0:0"; Each digit represents a different port; 0 - Off, 1 - On
GetPress	stat	int	Query pressure
GetRain	stat	int	Query rainfall from last 10 minutes
GetSolar	stat	int	Query solar intensity (W/m^2)
GetTelescopePos	stat	int	Telescope position; 0 - parked, 1 - Home, 2 - tracking
GetTemp	stat	int	Query temperature (F)
GetType	stat	int	Observatory type: 0 - MQI, 1 - AAI
GetWindD	stat	int	Query wind direction (degrees)
GetWindS	stat	int	Query wind speed (mph)
LoadScript	exec	string	Script file name
SetDome	exec	int	0 - Close, 1 - Open
SetFOV	exec	int	Set telescope field of view (arc-seconds)
SetIAI	exec	int	Boolean, IAI observatory type
SetIAIObject	exec	string	Command AIA observatory to monitor this object.
SetName	stat	string	Set observatory name
SetPower1	exec	string	String of format "0:0:0:0"; Each digit represents a different port; 0 - Off, 1 - On
SetStop	exec	int	Stop all current operations: 1 - Stop
SetTelescopePos	exec	int	Telescope position; 0 - Park , 1 - Home
SetType	exec	int	Observatory type: 0 - MQI, 1 - AAI

Table 2.2 Different commands available in the CelestialGrid network protocol. Two types of commands are available execution(exec) and status(stat). Execution commands issue changes to the system while status commands only poll operating conditions. The value defines the expected datatype within the network packet. Available options are described in the description.

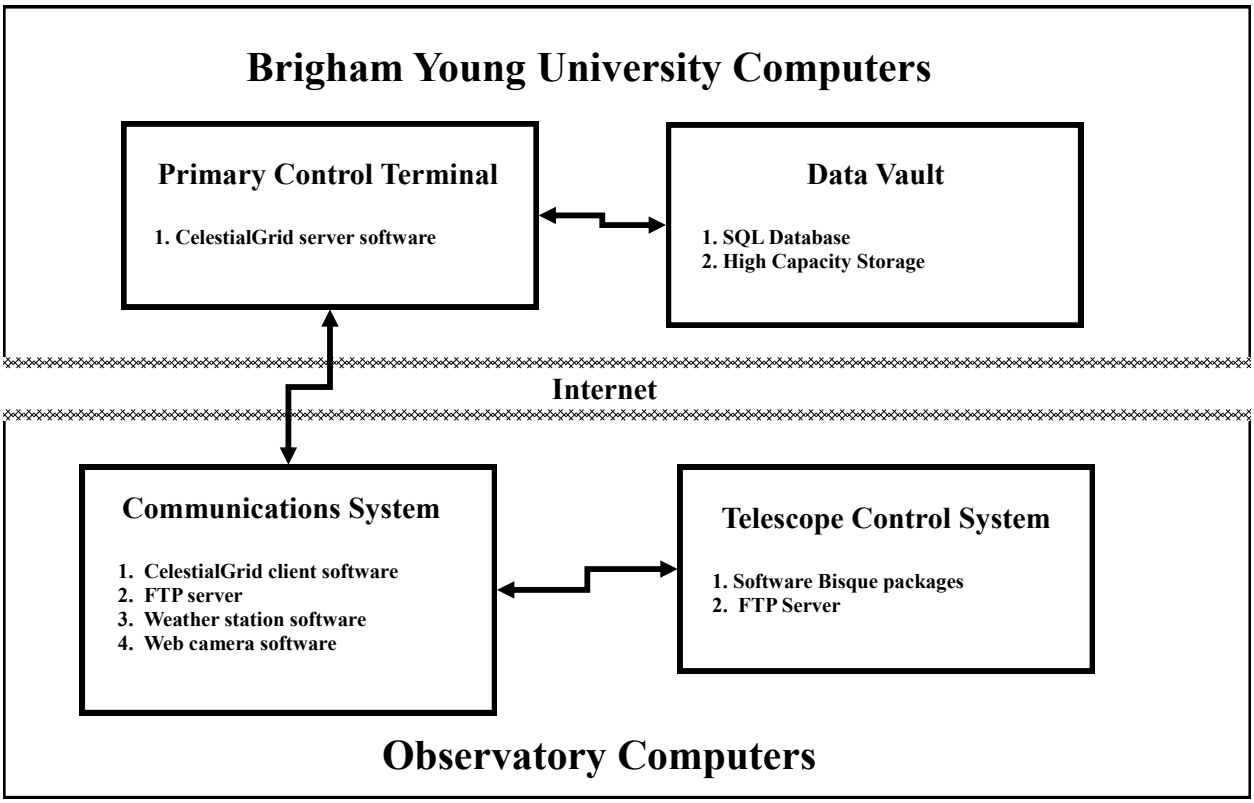

Figure 2.2 A schematic of how CelestialGrid is set up on ROVOR. Although the schematic says that CelestialGrid Server is run at BYU, the software can be installed and launched from any location.

2.2 CelestialGrid Hardware Requirements

CelestialGrid Client is the only part of the application that has specific hardware requirements. The client application connects to the telescope mount, camera, filter wheel, and focuser through *Software Bisque* products; therefore, the hardware requirements for CelestialGrid are only limited by the hardware requirements of *Software Bisque TheSky6*™ , *Orchestrate*, *CCDSoft*™ , and *AutomaDome*™ . As *Software Bisque* continually adds new modules to support new hardware, users should visit http://www.bisque.com for the latest supported hardware list.

In addition to *Software Bisque* hardware requirements, CelestialGrid Client also requires a weather station which will output a weather file according to software specification in Table 2.1. If this format is not available, the code within CelestialGrid will need modification (see Code Documentation Appendix E). The client system also requires both an IP power switch and a Digital Video Recorder (DVR) which CelestialGrid can post commands to.

2.3 CelestialGrid Software Requirements

CelestialGrid is built on top of multiple essential software applications. All the required software packages do not need to be installed on the same computer, but they do need to be installed on computers visible on the same local network. Before attempting to setup and install CelestialGrid, the user should make sure that required software is installed and the latest updates are applied. The required software components to be installed are described in detail below:

- CelestialGrid server software requirements:
  - *Java Runtime Environment* – at least version 1.6.
  - *MySQL* 5™ (optional) – Needed for archiving data.
  - *VNC*® Viewer – Remote desktop software package.

- CelestialGrid client software requirements:
  - *Microsoft Windows XP* or later – Needed because Software Bisque products require at least Windows XP.
  - *Java Runtime Environment* – at least version 1.6.
  - *Software Bisque TheSky6*™ – Controls the telescope mount.
  - *Software Bisque CCDSoft*™ – Controls the camera, filter wheel, and focuser.
  - *Software Bisque Orchestrate* – This software binds *Software Bisque* products through a common scripting language. The CelestialGrid client primarily communicates with Orchestrate.
  - *National Instruments LabVIEW 2010 – LabVIEW* is currently necessary for dome operation on ROVOR although other dome systems potentially can be controlled through *Software Bisque AutomaDome*™.
  - FTP Server – An ftp server is needed which is local to the observatory and will serve as a local storage area before data is transferred back to a permanent location at the university. ROVOR utilizes the free software *FileZilla Server*.
  - *VNC*® Server – Remote desktop software package.
  - *Software Bisque TPoint*™(optional) – Assists in building a pointing model which improves pointing and tracking accuracy of the telescope.
  - Weather station software (optional) – A *C++* application which interfaces with the ROVOR weather station.

Future software development will reduce the software requirements of CelestialGrid client. Specifically, we are developing a new adapter for CelestialGrid based upon the *Astronomy Common Object Model(ASCOM)*. Once finished, *Software Bisque* packages will only be optional and only

necessary for manual operation. Appendix D describes future software development in greater detail.

2.4 CelestialGrid Setup and Configuration

This section discusses how to set up and configure a new installation of both CelestialGrid Server and CelestialGrid Client.

2.4.1 Server

The following steps describe how to install and configure CelestialGrid cerver. Although CelestialGrid Server is permanently installed on BYU computers, the software can also be installed on a personal computer for controlling the observatory from home.

1. Begin the server installation by downloading the CelestialGrid Server package available at: http://rovor.byu.edu/CelestialGrid/CelestialGrid.zip

2. Unzip "CelestialGrid.zip" into the desired installation directory.

3. Start the application by double clicking on "CelestialGrid.jnlp" within the installation directory. (If Java is not yet installed, install it before starting CelestialGrid.)

4. A security warning will pop up. Select "Always trust content from this publisher" and click "Run". Figure 3.1 displays the main system graphical user interface.

5. ROVOR is the default observatory present. If adding a new observatory, select "File" and "New Observatory".

6. Select the "Edit" menu and click System Settings to set up the initial settings. These settings are global settings for all observatories managed through the server. Enter the IP Address

and login credentials to the MySQL database. (These fields can be disregarded if running the system from a location that does not have access to the database. This would usually be if the software is run from a personal laptop or computer not on the BYU network.) Enter a directory location for temporary files to be stored (web camera images). Make sure that this directory exists! If it does not, the web cameras will not work. The vault drive is the location where all data will be permanently stored. Once again, disregard this if not on the BYU network. Click "Save Settings". Settings are saved in the "Settings/CGSettings.txt" file within the installation directory.

7. Click the "Settings" button on the Observatory Control Panel to edit the Observatory Settings. Click "Apply Settings". Settings are saved in a file in the Settings directory according to the observatory name.

8. For both the main system settings and the observatory settings, if extensive settings were changed it might be necessary to reboot the application.

9. Setup is complete. Click "Connect" to connect to the observatory.

2.4.2 Client

The following steps describe how to install and configure CelestialGrid client. The client software should only be installed on one computer located at the observatory. If multiple observatories can have CelestialGrid client installed, a single installation of the server software is able to manage each.

1. Download the CelestialGrid Client package.

2. Unzip "CelestialGridClient.zip" into the desired installation directory.

3. Copy the Settings folder from a correctly configured CelestialGrid Server installation into the CelestialGrid Client installation directory. Some individual settings may need to be modified for the client version of the software (specifically Main Network Drive" and "Vault Drive Location"). Change these parameters by editing the text files in the Settings folder.

4. Double click the "CG Remote System.bat" file within the installation directory.

5. The client system is now running.

6. Before the client software will successfully manage the observatory, a copy of *Orchestrate* needs to be installed and running.

Chapter 3

Using the System

Using the system primarily involves operating the server interface. Although the server interface can be run from any location with an internet connection, operating the batch utility requires network access to the image archive and database.

The main graphical user interface (GUI) of the server system is divided into four separate panels: observatory control, status and weather conditions, observation tasks, and recent activity (see Figure 3.1). This chapter describes the functionality accessed through these panels.

3.1 Observatory Control

The CelestialGrid Server interface can manage multiple observatories. The controls to individual observatories are accessed using the tabs at the top of the interface window. The observatory control panel provides management to all observatory operations including: telescope control, power control, dome control, and web cameras.

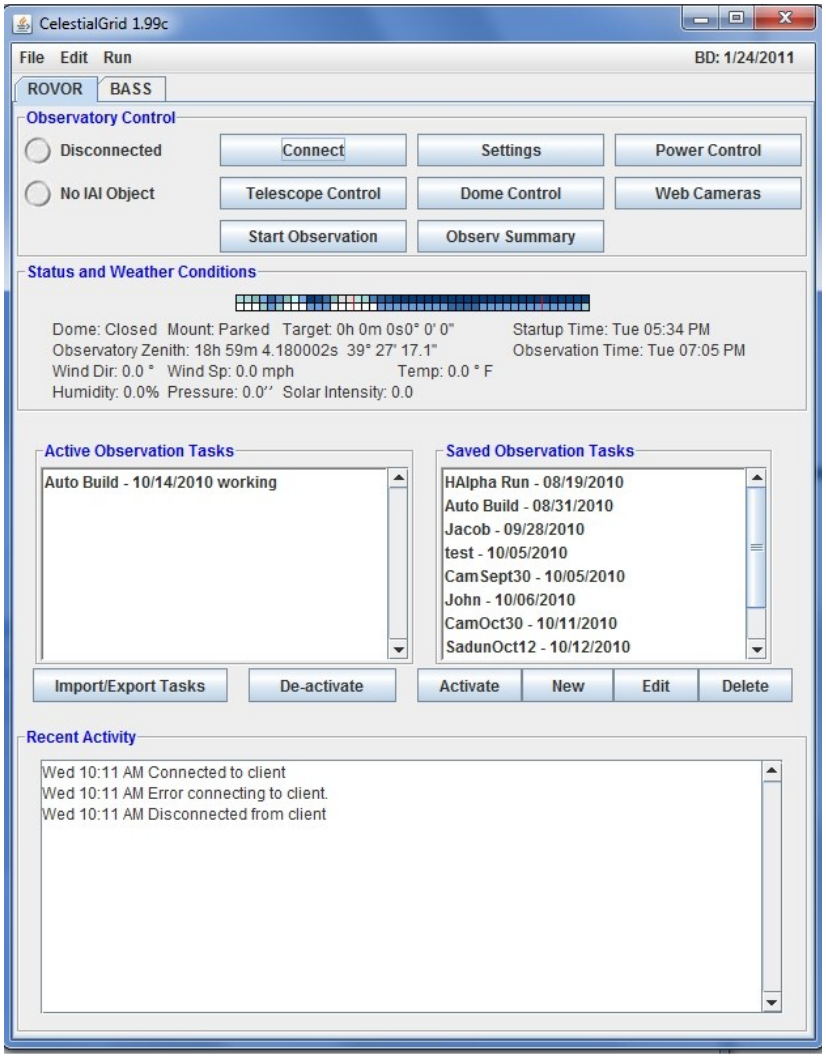

Figure 3.1 The primary graphical user interface of the CelestialGrid Server. The interface is divided into four main panels: observatory control, status and weather conditions, observation tasks, and recent activity.

3.1.1 Settings

The "Settings" button brings up the observatory specific settings window (see Figure 3.2). After changing observatory settings, restart the CelestialGrid server to apply the new setting. The available settings are:

Observatory Name – A unique identifier of the observatory. This value will appear in the header of all flexible image transport (FIT) files.

Observatory Type – The default is "Standard" but the field allows for future development of different observatory types (e.g. an all-sky survey).

Interrupt Automated Imaging (IAI) – If this option is enabled the observatory will respond to IAI feeds and interrupt scheduled operations for targets of opportunity.

Observatory IP – The internet address of the remote observatory site. This needs to be a static address.

Observatory Port – The port which CelestialGrid will communicate on. If the observatory computers are behind a firewall, this port needs to be opened. If the computers at the observatory location are behind a router, the router will probably need to be configured to forward this port to the computer running CelestialGrid Client.

FTP Path – The path on the FTP Server where the image FIT files are located.

Observatory Latitude and Longitude – Requires decimal formatting.

Power Port Names – These fields will label the power ports within CelestialGrid Server (see section 3.1.3.

Master-Flat Locations – The file paths of the master-flat calibration images for the batch utility to use in automatically calibrating downloaded images (see section 3.5).

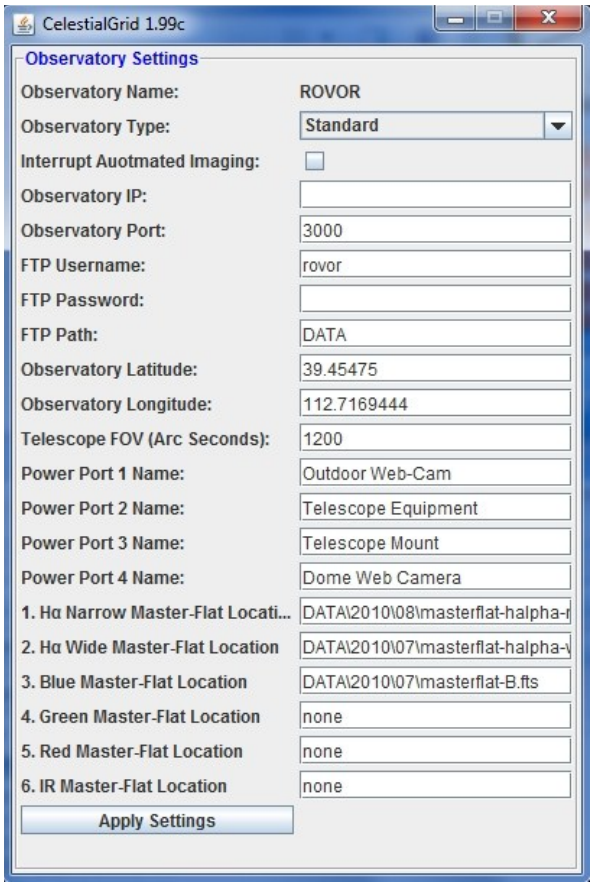

Figure 3.2 The observatory specific settings interface in CelestialGrid Server. If the settings are changed and saved, restart the application for changes to take effect.

3.1.2 Connection

Click the "Connect" button to connect the server and client systems. If the connection is successful, the connection light will turn from clear to green. It takes a few seconds for the connection to initialize. If successful, the weather information will automatically update in the status and weather conditions pane (see Section 3.2). If for some reason the connection cannot be established, the system will continue to retry for a few seconds before timing out. If the system times out, the connection status light will display red. (See Appendix Section B for troubleshooting.)

Beside the connection light is the interrupt automated imaging (IAI) indicator. This indicator will light up when the system has received a interrupt vector. These vectors usually indicate that a gamma-ray burst (see Section 1.1) is being imaged. Check the recent activity window for additional information on the object.

3.1.3 Power Control

Make sure that connection is established to the observatory before starting the power control. The power control interface manages power to a power control switch at the observatory location. This switch can automatically or remotely turn power on or off to four separate ports. At ROVOR, these ports have been configured to control power to the observatory mount, telescope equipment (camera, focuser, and filter wheel), and the separate web cameras. This configuration can be physically changed at the observatory location.

Click the "Power Control" button to bring up the power control interface (see Figure 3.3). Initially, the interface has no current information and should not be taken as accurate. Click the "Refresh" button to query the observatory and update the interface with accurate information. After the refresh button is clicked, the interface will deactivate until the system successfully queries the observatory power conditions. Activate or deactivate power to observatory components by selecting the desired port configuration and clicking the "Set Power Options" button. Immediately

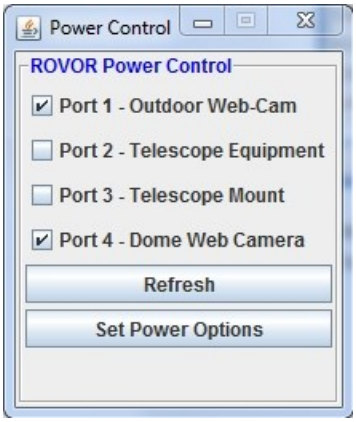

Figure 3.3 The power control interface. Initially, the interface has no current information and should not be taken as accurate. Click the "Refresh" button to query the observatory and update the interface with accurate information.

a signal will be sent to the observatory enabling or disabling power. (If manually starting the observatory, always deactivate power to the web cameras before any important imaging.)

3.1.4 Dome Control

Make sure that connection is established to the observatory before starting the dome control. Start the dome control interface by clicking the "Dome Control" button on the observatory control panel (see Figure 3.4). Initially, the interface has no current information and should not be taken as accurate. Click the "Refresh" button to query the observatory and update the interface with accurate information. The open, closed, and busy lights represent the current condition of the dome. If it is busy, the dome is neither completely open nor completely closed. As a general rule, the dome status lights are not perfectly accurate. If necessary, check the web cameras to verify the dome status.

Figure 3.4 The dome control interface. Initially, the interface has no current information and should not be taken as accurate. Click the "Refresh" button to query the observatory and update the interface with accurate information.

Figure 3.5 The web camera interface. Initially, the displayed images are buffered images and should not be taken as accurate. Click the "Refresh" button to begin downloading current web camera images. Click on an image to display a larger version.

3.1.5 Web Cameras

Click the "Web Cameras" button to open the web camera interface (see Figure 3.5). This interface is independent of the CelestialGrid Server connection and can be accessed as long as there is an internet connection available. Initially, the displayed images are buffered images and should not be taken as accurate. Click the "Refresh" button to begin downloading current web camera images. (Images may not download if the FTP server at the observatory is already uploading files.) Click on an image to display a larger version.

3.1.6 Telescope Control

Make sure that connection is established to the observatory before starting the telescope control. Clicking the "Telescope Control" button brings up the control interface (see figure 3.6). When it first opens, the interface has no current information and should not be taken as accurate. Click the "Refresh" button to query the observatory and update the interface with accurate information.

Before homing, parking, or slewing the telescope, *make sure that the observatory dome is open* and that the telescope and equipment both have power (see Section 3.1.4 for dome control and Section 3.1.3 for power control). Serious damage to the telescope may result if it is activated before the dome is open.

Click the "Find Home" button to slew the telescope into the home position. If the telescope has not been through the startup routine, this will be done before the slew operation. (The telescope startup routine starts CCDSoft and TheSky6 and takes around three minutes to complete.) The interface will deactivate while waiting for an operation to complete. If for some reason the operation fails an error will display (see Section B on troubleshooting). Clicking the "Park Telescope" button will slew the telescope to the park position and begin the shutdown procedure. (The shutdown procedure only shuts down CCDSoft and TheSky6.)

Slew the telescope to a specific position by entering right ascension and declination values into the interface. Equations 3.1 and 3.2 illustrate the format for right ascension and declination values respectively. Only seconds and arc-seconds can be decimal values.

$$Hours : Minutes : Seconds \tag{3.1}$$

$$\pm Degrees : ArcMinutes : ArcSeconds \tag{3.2}$$

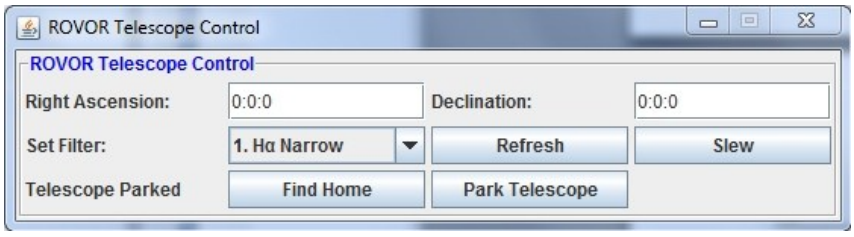

Figure 3.6 The telescope control interface. Initially, the interface has no current information and should not be taken as accurate. Click the "Refresh" button to query the observatory and update the interface with accurate information.

3.1.7 Starting Observation

Click the "Start Observation" button to start the nightly observation (make sure that connection is established to the observatory). Because this operation is entirely automated, no user input is necessary after the button is pressed. CelestialGrid automatically calculates the sunset time and begins a startup routine just before sunset. This routine opens the dome, turns on the telescope and mount, turns off the web cameras, and takes calibration frames. After the startup routine, CelestialGrid waits until astronomical twilight to begin the observation tasks.

3.2 Status and Weather Conditions

When the server is connected to the client, the status and weather conditions panel displays live information about the observatory (see Figure 3.7). This includes the status of the dome and telescope mount, the local weather conditions, and the calculated zenith at the observatory. The panel also displays the startup and observation times. The startup time is when the observatory will automatically begin its startup procedure. The observation time is when the observatory will actually begin imaging from the observation task queue. The difference between the two allows the observatory time to initialize the telescope and take bias, dark, and flat calibration frames.

The status and weather conditions panel also displays a preview of the ClearDarkSky report [9].

Figure 3.7 The status and weather conditions panel displays live information about the observatory, including the status of the dome and telescope mount, the local weather conditions, and the calculated zenith at the observatory. The panel also displays the startup and observation times.

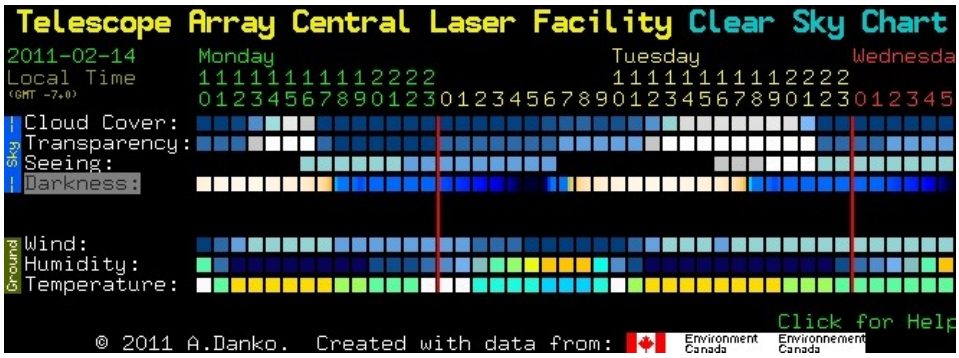

Figure 3.8 The Clear Sky Chart [9] features detailed weather information around the observatory. Each square represents an hour of the day. The darker the square, the better the conditions for astronomical imaging.

Click on the blue and white colored bar to bring up the ClearDarkSky report (see Figure 3.8). The chart displays detailed weather information for a two–three day period. The available information includes cloud cover, transparency, seeing, darkness, wind, humidity, and temperature. Generally cloud cover, transparency, and seeing are the main fields to monitor. Each square represents an hour of the day. The darker the square, the better the conditions for astronomical imaging. If there are lots of white squares, it generally means that the observatory should not be used for the night. See http://cleardarksky.com/csk/ for more information.

3.3 Observation Tasks

An observation task is a list of objects for the observatory to view. There are two types of observation tasks: active and saved. Active tasks are tasks that will be executed if the observatory is started (see section 3.1.7). Saved tasks are simply inactive tasks. The date associated with an observation task is the date that the task was first created. Creating a new or editing a task brings up the observation task editor. The observation task editor provides an interface for building and editing observation tasks.

Add a new object to an observation task by entering values into the right ascension and declination fields (see figure 3.9). The required format for the coordinates is given in equation 3.1 and 3.2. If an object is given a name, it will be saved under "Saved Objects". Previously added objects are available by clicking on the "Saved Objects" button.

Click the "Export Script" button to export the observation task as an orchestrate script.

3.4 Observation Summary

Click the "Observ Summary" button on the observatory control panel to open the observation summary interface (see Figure 3.10). This interface is a utility that allows the user to review all observation tasks planned for the night. The interface features a top heading with basic information about the nightly tasks, a panel listing all objects to be viewed, and another panel listing available Landolt standardization frames.

When launched, the utility loads all active observation tasks into the left panel (objects to view). These objects will be imaged in the order presented. The utility estimates how long each task will take, and calculates an approximate time of night that each object will be imaged. The zenith is calculated for each object in the list (with its corresponding observation time) and measures the angular distance δ, between zenith and the object's coordinates (zenith angle). If $\delta \geq 90^{\circ}$, the

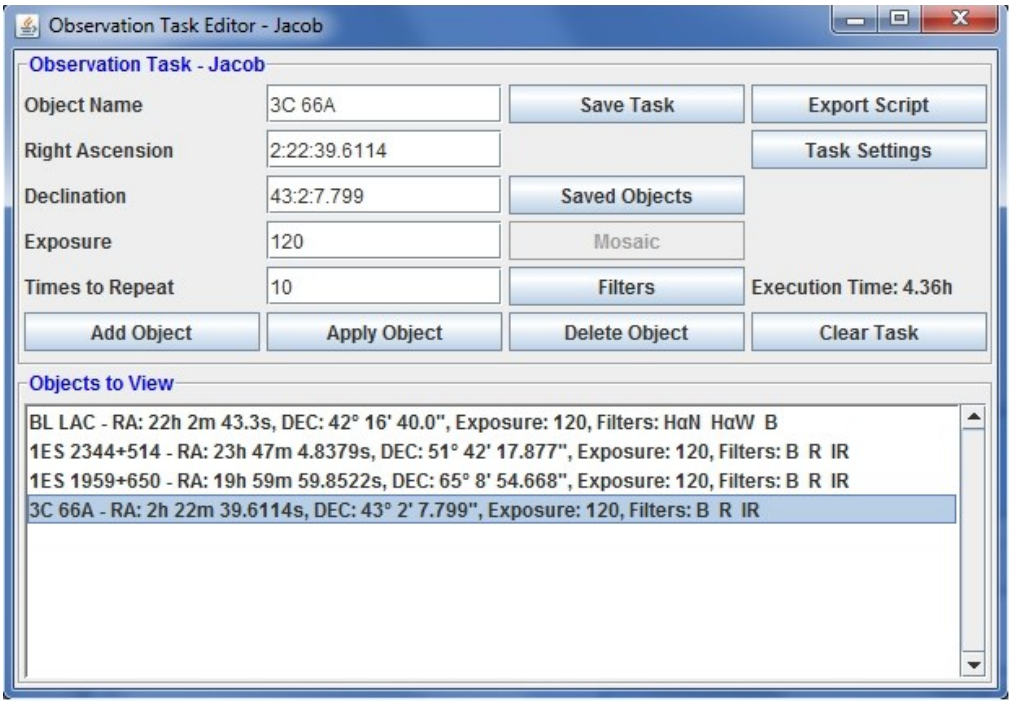

Figure 3.9 The observation task editor provides an interface for building and editing observation tasks.

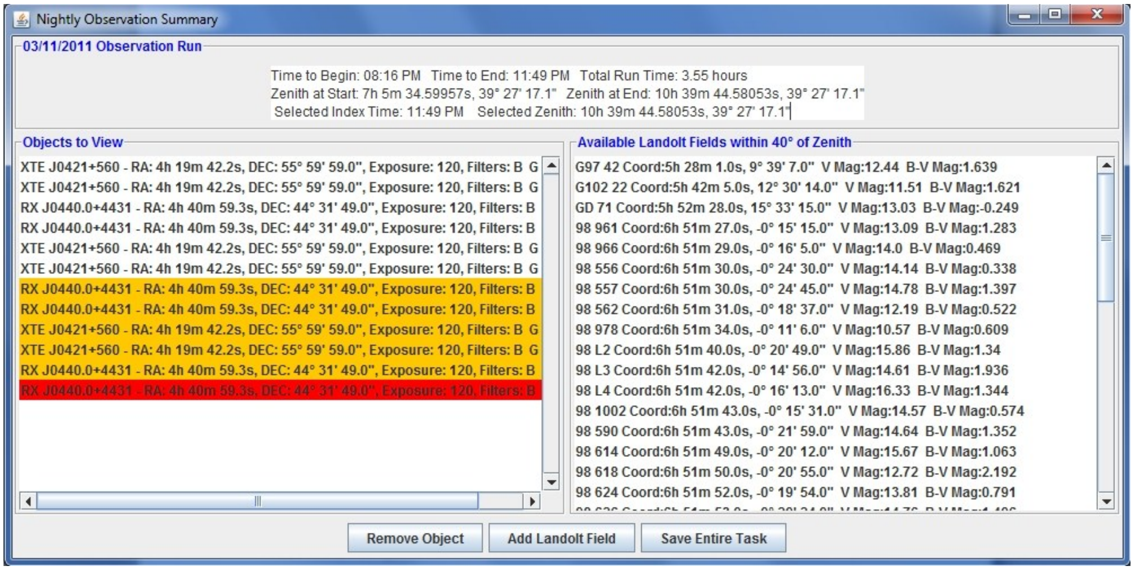

Figure 3.10 The observation summary interface. This interface is a utility that allows the user to review all observation tasks planned for the night.

object is below the horizon at that particular time and will be highlighted in red. If the measurement is $40^\circ < \delta < 90^\circ$, the object is highlighted in yellow as a warning.

If an individual object is selected in the objects to view panel. The information at the top of the screen will update with the selected object's estimated time to begin viewing, calculated zenith, and zenith angle. The list of available standardization Landolt fields is also updated. Click on any of the Landolt fields and a preview window will display. (Landolt images are downloaded from the internet; make sure an internet connection is available.) Click the "Add Landolt Field" button and the system will add the selected Landolt field directly after the selected object within the observation task. Click the "Save Entire Task" button and the entire task list will collapse into a single observation task entitled "Auto Build".

3.5 Batch Utility

The CelestialGrid batch utility is designed to automatically download, process, and archive flexible image transport (FIT) files. In the CelestialGrid Server installation directory, start the batch utility by entering at a command prompt

$$java \ -jar \ cgbatch.jar \ ROVOR \ 06 \tag{3.3}$$

The utility takes two parameters: the name of the observatory (ROVOR) and a time to begin downloading images. The parameter for the time is a two digit numeric representing the hour in the day between 00-24 (six in the morning is represented as 06). When started, the software waits until the designated hour to begin downloading FIT images from the observatory into a temporary directory (designated in the settings file). The remote instance of an image is deleted after it is successfully downloaded. Once all images are downloaded, the images are moved onto the vault drive into a directory structure representing the date they were taken. An image taken on January 1, 2011 would appear in the directory

$$/Data/2011/01/01/ \tag{3.4}$$

Immediately after the images are moved, master dark and bias calibration frames are created. The master calibration frames are then applied to all other images taken from the night. These final images are all stored together within another directory called "calibrated". The original frames are not deleted. Last of all, the software updates a *MySQL* database with header information from each image (including the path of each image).

Appendix A

Package References

Software and hardware packages utilized by CelestialGrid are displayed in Table A.1. Some packages are exclusively used by the client and others by the server system. See the given internet address for more information or to download software.

Package	Use	Reference Address
Astronomy Common Object Model	Client	http://ascom-standards.org/
ClearDarkSky	Server	http://cleardarksky.com/csk/
C++	Weather Station	http://www.cplusplus.com/doc/tutorial/
Digital Video Recorder	Web Cameras	http://www.aposonicccctv.com/index.php/standalone/4chdvr/79
Image Reduction and Analysis Facility	Image Reduction	http://iraf.noao.edu/
File Transfer Protocol	Client/Server	http://en.wikipedia.org/wiki/File_Transfer_Protocol
FileZilla Server	Client	http://filezilla-project.org/
IP Power Switch — RPS-ERP II	Client	http://www.remotepowerswitch.com/web-remotepowercontrol.html
Java Abstract Window Toolkit	Client/Server	http://download.oracle.com/javase/1.5.0/docs/guide/awt/index.html
Java Runtime Environment	Server/Client	http://www.oracle.com/technetwork/java/javase/downloads/index-jdk5-jsp-142662.html
Java Swing	Client/Server	http://download.oracle.com/javase/tutorial/uiswing/
Java Web Start	Client/Server	http://www.oracle.com/technetwork/java/javase/overview-137531.html
Javadoc	Code Documentation	http://www.oracle.com/technetwork/java/javase/documentation/index-jsp-135444.html
Microsoft Windows XP	Client	http://www.microsoft.com/windows/windows-xp/default.aspx
Mirametrics Mira®	Image Reduction	http://www.mirametrics.com
National Instruments LavVIEW	Client	http://www.ni.com/labview/
Oracle MySQL 5™	Batch Utility	http://dev.mysql.com/downloads/mysql/
RAID	Server	http://en.wikipedia.org/wiki/RAID
Software Bisque AutomaDome™	Client	http://www.bisque.com/sc/shops/store/automadome-box.aspx
Software Bisque CCDSoft™	Client	http://www.bisque.com/sc/shops/store/CCDSoftWin2.aspx
Software Bisque Orchestrate™	Client	http://www.bisque.com/Products/Orchestrate/orchestrate.asp
Software Bisque TPoint™	Client	http://www.bisque.com/sc/shops/store/tpointwinbox.aspx
Software Bisque TheSky™	Client	http://www.bisque.com/sc/shops/store/All+Products/TheSkyX+Editions/Professional/TheSkyXProWinDownload.aspx
TCP/IP	Server/Client	http://en.wikipedia.org/wiki/Internet_Protocol_Suite
VNC® Viewer and Server	Server/Client	http://www.realvnc.com/products/free/4.1/index.html

Table A.1 References to software and hardware packages utilized by CelestialGrid.

Appendix B

Troubleshooting

If we have learned one thing from the development of CelestialGrid and the operation of ROVOR, it is that problems regularly and inevitably occur. This section is intended to help a user with problems or errors that may arrive in operating CelestialGrid. I will simply provide a list of symptoms and possible problem resolutions.

- CelestialGrid Server does not start up – Make sure that you are starting the application by double clicking on CelestialGrid.jnlp. If it is the first time that you are running the application, make sure that the user has an internet connection (will fail on the first boot without an internet connection). If the application fails to startup after the user has modified the system settings, the settings themselves may be corrupt and causing problems. The settings can be changed without starting CelestialGrid by editing in a text editor Settings/ROVOR.txt (located within the CelestialGrid installation folder). If none of these fix the problem, delete the CelestialGrid installation directory and re-download and install the software.

- CelestialGrid Server errors in connecting to the client – First of all, make sure that the computer running the server application can connect to the internet. If it can, open up a command prompt and type ping 67.47.236.82 (or whatever the IP address of the observatory is). If this

command times out, then the observatory itself has no internet connection. If it is successful, try restarting the client application on the observatory computers.

- Telescope fails to slew or observation task execution errors – First of all, remote desktop into the observatory computers and check if the CelestialGrid client window is displaying any nasty errors. If it is, copy the entire error and report it to the developer. If no errors appear, restart CelestialGrid Client. Also close *Software Bisque CCDSoft* and *TheSky6* if they are open. Lastly, restart *Orchestrate*. Make sure Orchestrate is enabled to watch a folder for scripts to appear. Retry observation task execution or slew. If it still fails, verify that the telescope and mount are powered.

Appendix C

Development History

The original version of CelestialGrid simply helped in the creation of Orchestrate scripts. The main feature was that all objects passed into CelestialGrid would have their coordinates conformed to a predefined grid of the sky. This predefined grid had elements the size of ROVORs field of view. Having a predefined grid enabled every object imaged by ROVOR to have a corresponding grid number that the image could be archived under. The idea was to simplify archiving and aligning subsequent images. This original version of CelestialGrid did not have any direct telescope control or connection. Eventually I determined that archiving images this way would not be necessary if automatic photometry was stored in a SQL database. In Winter 2010, I began the design of an entirely new CelestialGrid system to automate all observatory functions. The following is a development history of the entire process, beginning May 2010. Development is also ongoing and this list may not represent the most up to date status.

1.90 - 05/20/2010

- Initial build finished.

1.93a - 05/25/2010

- Telescope control frame built and correctly communicating to remote system.
- Built GUI dome control frame.

1.93b - 05/28/2010

- Rebuilt network system to use ObjectOutput/InputStreams instead of PrintWriters and buffered readers. Write objects to network streams rather than simple strings.
- Simplified networking code on client end.
- New GUI option in TaskFrame to conform input coordinates to the predefined Celestial Grid. Added GUI tool-tips within TaskFrame.

1.93c - 06/01/2010

- Finished server interface for dome control.
- Complete rework and simplification of TelescopeControlFrame.java

1.93d - 06/02/2010

- Error dialog comes up when not connected and starting power, telescope, or dome control
- If power, telescope, or dome control frame already open and not connected, no function.
- Corrected filter labels.
- Power Control labels are now pulled from the observatory settings frame.

1.94a - 06/09/2010

- Built DomeController.java to interface with Labview - correctly opens/closes/stops dome
- Rebuilt OrchestrateWrapper to reference VBScripts for initializing/closing Orchestrate
- New algorithm correctly calculates Zenith.
- Added clear sky dark sky thumbnail to observatory GUI panel.
- 06/14/2010 - Web camera frame correctly displays updated images
- OrchestrateWrapper erases 002.txt file from the done folder after correctly reading it
- OrchestrateWrapper writes HANDSHAKE to the watch folder after every orchestrate cmd
- Removed rainfall reading

1.94b - 06/15/2010

- Built Orchestrate Initialize and De-initializing routines.
- Telescope control disabled while dome is closed.
- Orchestrate exported scripts correctly switch to light mode before object list.
- Various bug fixes and improvements in telescope slewing/homing/parking

1.94c - 06/16/2010

- Repaired filter algorithm.
- Built TelescopeThread - A class to write observation tasks to the OrchestrateWrapper
- Correctly retrieve status as observation tasks progress throughout the night.

1.94d - 06/18/2010

- Added AppStartup/AppShutdown commands to the OrchestrateWrapper start/stop orchestrate functions.
- Increased timeout period to (3 x Exposure) for takeimage TelescopeThread call.
- Changed GUI from bevel borders to etched borders.
- Added large DarkSkyClearSky window.
- Added Observation Task Settings Frame - options for timed execution, auto-darks, IAI interrupt for a task, and conformed coordinates option.

- Added an estimated execution time readout.

1.95a - 06/21/2010
- Every 30 minutes CelestialGrid server will automatically reset the connection to the client. Prevents timeouts after long connection periods.
- Built routine (StartObservation Button) to automate all observation functions.
- TelescopeThread processes multiple observation tasks according to an ArrayBlockingQueue
- TelescopeStatus messages are stored on the client-side on an ArrayBlockingQueue before being sent to the server - gaurantees all messages are sent.
- Created basic GUI for remote system client.
- Added splash about screen.

1.95b - 06/22/2010
- Increased TelescopeStatusMessage queue length to 120 - If the queue is full and a status message is added, the client currently lock up until the queue is free. – TOFIX
- Created FIT Processor thread for remote system client - Waits for FIT files to appear in CCDSoft temporary saved location. Moves files to permanent directly based upon date image was taken.
- Built an external batch application to perform FIT sorting on any location of FIT files. FIT Batch Processor - fit-sort.jar

1.95c - 06/28/2010
- Built Orchestrate VBScripts to call to set filenames to bias,dark,flat,light prefixes
- FITProcessor now can take the difference or quotient between two images. The process is done multi-threaded - these methods enable auto reduction of data and comparison of images

1.96c - 06/29/2010
- Added takeBiasFrames() method to OrchestrateWrapper. Called from TelescopeThread before an observation task is begun. Takes 40 bias images.
- Added takeDarkFrames() method to OrchestrateWrapper. Called from TelescopeThread before an observation task is begun. Takes 14 dark images for every exposure time throughout the observation task.

1.95d - 07/02/2010
- Build sqlClient class to interface with the sql database. Also built the sqlObject to hold values from the sql database.
- Built the DataDownloader class. Moved the functions to read and move FIT files from the FITProcessor class into the DataDownloader class. The DataDownloader class can run stand alone in batch form. Process:
    1. If the correct time of day logs into ftp server and downloads all FIT files to temp location
    2. Moves FIT files from temp location to an archive in a structure DATA/YEAR/MONTH/DAY/*.fit
    3. Updates a MySQL database with fit header information

  DataDownloader can be run standalone from cg-batch-dl.jar or from the new "Retrieve Data" button on the main observatory panel.

1.95e - 07/06/2010
- Modified DataDownloader to parse multiple formating options for FIT headers.
- Increased timeout time for auto taking DARK frames.
- Modified calibration frame taking algorithms. Will now execute at the start of an observation run, 4 Dark frames for every exposure, 15 total bias frames, flat frames not yet implemented.

1.95f - 07/09/2010
- Variety of GUI fixes relating to auto observation.
- Added a static function in CGApp to retrieve the date.
- Added pauses/sleeps to auto-dark routine and the orchestrate startup routine.
- The ObservationPanel will automatically save system setup every time an observation task is finished.

- Recent Activity Panel now displays a changed connection status.

1.96a - 07/13/2010
- If CGServer loses the connection to the client, will attempt to re-establish connection. Server connection should now persist throughout the entire night.
- Added an external class for calculating sunset/sunrise SunriseSunset.java from Jo Marie Green.
- Moved twilight time calculation methods to a static class: CelestialTime.java
- When the Start Observation button is clicked, observation will automatically begin 20 minutes prior to civil twilight.
- Built FLAT calibration routine. Flats are taken at civil twilight. Flat routine is followed by BIAS and then dark routines. Normal images are not taken until astronomical twilight.
- Added a static method to return the average value of a FIT file.
- FLAT fielding routine begins at civil twilight. Beginning exposure is 5 seconds and should definately flood the field. Flooded images are assumed to have an average of over 60,000 counts. Will wait 30 seconds at a time and repeat the 5 second exposure until twilight brightness has reduced the average image counts to below 40,000 and above 20,000 counts. Once this point is reached, will begin routine: .5 second intervals from 5 - 10 seconds, 1 second intervals from 10 to 20 seconds, and 3 second intervals from 20 to 60 seconds.
- Added writeImagePath.vbs visual basic script to Orchestrate. This script will save to most recently taken CCDSoft image path to lastfile.txt
- TelescopeThread now resized web-camera image - reduced filesize and download time.
- CameraFrame dynamically builds image windows based upon image resolution.
- Added a method in CelestialCoordinates to convert Epoch2000 to Epoch Now.

1.96b - 07/21/2010
- Built at JNLP - Java Web Start deployment system.
- All jar files are now signed before delivery. Building is now done through buildCGApp.xml - Ant Script.
- Added more GUI status updates during auto-observation run - including FLAT Average count values.
- Added Start Up and Observation Time displays in the main GUI.
- Built LandoltField class. Reads a CSV file and populates a database of possible standard Landolt fields. The function getListFromCoordinates() will find LandoldFields within 3 hours of a given coordinates.
- Fixed bugs in Spherical class where negative declination values weren't showing.
- Added drag and drop reordering in the TaskFrame editor. Conform coordinates option now defaults off.
- Fixed Apply Object button in TaskFrame editor to not conform applied coordinates to the CelestialGrid. Also the button doesn't default to no select after being pressed.
- Added the ability to give individual objects names within the TaskFrame editor. Named object coordinates are universally saved to be used in future observation tasks. Built the SavedViewingObjectFrame. Objects are saved to "Data\\savedObjects.dat"
- Cleaned up code between classes TaskFrame and ObservationTask. ObservationTask no longer includes a TaskFrame object, rather a TaskFrame is generated from a given ObservationTask.
- Estimated execution time in the TaskFrame now only displays two significant decimal digits.

1.96c - 07/26/2010
- Fixed a bug in TelescopeThread addStatusEntry function which caused remote app to seize up when disconnected from the server.
- Built Observation Summary GUI. Enables user to preview observation times for all activate observation tasks to run in a given night. Calculates the time to begin the task and the time to end. Also displays begining and ending zenith values. StandardStar Landolt fields are displayed within 3 arc-hours of selected viewing objects. If a viewing object is below the horizon, it is highlighted red and yellow if it is more than 3 hours from zenith.
- Built a static method to find the angle difference between two equatorial coordinates. CelestialCoordinates.getAngleDifference()

- Added a GUI option under TaskFrame to repeat individual viewing objects. Both ObservationSummary and TaskFrame correctly calculate times assuming 10 second image download time and 25 second slew times.
- Auto-flat routine now jogs east 1.5 minutes after every frame, periodically reseting to 6 degrees east of zenith.

1.96d - 07/30/2010
- Built MailerThread as the main Interrupt Automated Imaging(IAI) feed. The thread connects to an email account every 30 seconds. When connected, it parses emails that match the current day finding specific emails from the Gamma-ray bursts Coordinates Network. If a gamma ray burster is found, TelescopeThread will calculate if the coordinates are within the current horizon, if so, the current observation run is put on hold while the gamma-ray burster is imaged, 4x(B V R I) for 120 seconds each. At the end of the interrupt imaging the scope is re-slewed to it's original position and the normal nightly imaging is resumed. TODO This auto interrupt routine can also be initiated through the CGProtocol.
- Within the ObservationSummaryFrame changed the available landolt fields to be up to 40degrees instead of 45.
- Added image previews for landolt fields. Preview image files are loaded from http://rovor.byu.edu/lantolt/
- Added function in CelestialCoordinates to calculate Azimuth and Zenith from equatorial coordinates.
- Added functions in CelestialCoordinates which return whether or not a equatorial coordinate is east or west of the meridian.
- writeImagePath.vbs now dynamically rather than statically writes the lastfile.txt.
- setExtension.vbs replaces setFlat.vbs, setBias.vbs, setDark.vbs, and setLight.vbs. Call setExtension.vbs par where par equals the desired extension to appear on the fit files.

1.97 - 08/03/2010
- General code improvements amount most classes. Code documentation complete.

1.97a - 08/04/2010
- Begun official algorithm for reducing images after they are downloaded from the remote ftp server.
- Built methods for subtracting and averaging FIT files.

1.97b - 08/23/2010
- Rebuilt datadownloader application.
- Filenames now include object names.
- Fixed algorithm to calculate azimuth. Wrote additional functions for calculating Julian Date, LST, and GST.
- Function's CelestialCoordinates.isEastSideOfMeridian() and isWestSideOfMeridian now work.

1.99a - 09/21/2010
- Improved image auto/reduction algorithm
- Added a client side log of telescope status.

1.99b - 01/11/2011
- Added many improvements/bug fixes
- Improved web-camera interface
- Improved power control frame interface
- When taking dark calibrations, tracking is turned off
- ScreenShotThread now flushes the buffered image after each query to the webcameras, removing an out of memory bug.

1.99c - 01/24/2011
- Fixed the dome control interface to correctly report dome position.
- Removed ConformCoordinates setting within CelestialGrid until properly fixed.
- Disabled Mosaic GUI option in the task editor frame - until the mosaic generator is fixed.

Appendix D

Development Future

There are multiple features that are planned to be implemented within CelestialGrid. These features will improve the overall system functionality and enable the software to control observatories other than ROVOR.

- World Coordinates System – World Coordinates coefficients need to be implemented within every image that comes out of CelestialGrid. These coefficients provide an immediate translation between x/y pixel positions and right ascension and declination coordinates. Having these coefficients in the FIT header of each image is required for availability in the National Virtual Observatory.

- User login interface – Develop a user login interface that accepts multiple connections to the observatory. Each user would have a designated password. All images and command logging from CelestialGrid would be tied to the user that issued the command. This will improve security and functionality, as some users could have read only access.

- Automatic Scheduling Interface – Develop an interface which takes care of all object scheduling. CelestialGrid would simply be supplied a set of objects to image throughout the year.

The objects could have different priority levels. The scheduling interface would then automatically build observation tasks from night to night. It would know which, when, and for how long objects in the given set were viewable. This is a crucial step in making ROVOR not only robotic and automatic, but completely autonomous.

- Automatic Photometry - Our current setup is limited by the speed and bandwidth of our internet connection. Every night we download hundreds of images. This can take all day to finish. Instead of downloading images back to BYU, we would like CelestialGrid client to automatically calculate the photometry of every star on every frame each night. In the morning pure magnitude values could be downloaded back to BYU much more quickly. These values could then be archived in the *MySQL* database. Automatic graphs and charts of nightly results could also be emailed to users.

- Add support for traditional observatory domes (not like ROVORs). This should be a very simple feature to add, as *Orchestrate* and *AutomaDome* already have controllability to many different dome systems. All that really needs to be added is a new GUI for a different dome system and visual basic scripts for accessing the RASCOM dome controllers.

Appendix E

Code Documentation

The code for CelestialGrid is thoroughly documented through JavaDoc. Developers should visit http://rovor.byu.edu/CelestialGrid/doc/index.htm for more information. The code was also designed and built within the development environment Eclipse. If a user is attempting to access the code, it is encouraged that it is done through Eclipse. The following steps describe how to setup the Eclipse development environment:

1. Download and install Eclipse from http://www.eclipse.org/
2. Start Eclipse and create a new project from existing source. This should be done from the root directory of the CelestialGrid code base.
3. The code is now viewable and editable.

If changes are made, build the code through the Ant script build.xml within the root directory of the CelestialGrid code base. This can be done from within Eclipse. Right click build.xml and select build from Ant. The script will compile and build .jar files that can be distributed to the the client and server systems. Edit the script to modify where the output will be placed. The client jar file needs to be downloaded to the observatory control computer. The old jar file needs to be

deleted and replaced with the new. Distributing to the server systems is a bit more complicated because many users have the server system installed on many different computers. Java WebStart helps to fix this problem. Every user who has downloaded and installed CelestialGrid did not actually install the server jar file. Instead they downloaded a .jnlp file. Figure E.1 shows the contents of CelestialGrid.jnlp. The properties in this file point to a location on the internet where the CelestialGrid jar files are stored. Java WebStart looks at these files online and compares them to ones downloaded to the local machine. If the ones online are newer, it automatically downloads the newer files before starting CelestialGrid. This ensures that every user is running the most up to date version. In order to roll out a new update for every user of CelestialGrid, the developer needs to upload to the internet updated .jar files. Figure E.1 also lists the different jar files used by CelestialGrid. Generally the only jar file a developer needs to worry about is CG-Server.jar. All the other jar files are libraries, and do not generally need to be updated. If the developer references a new library in the CelestialGrid code, this library jar file will need to be added to every user's individual CelestialGrid.jnlp file. This can be troublesome, so only add libraries when they are absolutely needed.

```
<?xml version="1.0" encoding="UTF-8"?>
<jnlp spec="1.0+"  codebase="http://rovor.byu.edu/CelestialGrid/">
<information>
  <title>Celestial Grid</title>
  <vendor>Brigham Young University - Bret Little</vendor>
  <homepage href="http://rovor.byu.edu" />
  <description>Observatory Control</description>
  <icon href="icons/splash.jpg"/>
  <icon kind="splash" href="icons/splash.jpg"/>
</information>
<offline-allowed/> <security>  <all-permissions/> </security>
<resources>
  <j2se version="1.2+" />
  <jar href="CG-Server.jar"/>
  <jar href="CG-lib/httpmime-4.0.1.jar"/>
  <jar href="CG-lib/Jama-1.0.2.jar"/>
  <jar href="CG-lib/commons-logging-1.0.4.jar"/>
  <jar href="CG-lib/commons-codec-1.3.jar"/>
  <jar href="CG-lib/junit.jar"/>
  <jar href="CG-lib/edtftpj.jar"/>
  <jar href="CG-lib/jsch-0.1.41.jar"/>
  <jar href="CG-lib/commons-net-2.0.jar"/>
  <jar href="CG-lib/fits.jar"/>
  <jar href="CG-lib/httpclient-4.0.1.jar"/>
  <jar href="CG-lib/commons-httpclient-3.1.jar"/>
  <jar href="CG-lib/commons-vfs-1.0.jar"/>
  <jar href="CG-lib/mysql-connector-java-5.1.8-bin.jar"/>
  <jar href="CG-lib/flanagan.jar"/>
  <jar href="CG-lib/mail.jar"/>
</resources>
<application-desc main-class="sys.CGApp" />
</jnlp>
```

Figure E.1 The Java WebStart file format. These are the contents of the primary file, CelestialGrid.jnlp, that users double-click on to start CelestialGrid.

Bibliography

[1] S. Gajadhar, T. Vermeulen, and W. Cruise, *Telescopes from Afar* (2011).

[2] D. A. Bohlender, D. Monin, and L. Saddlemyer, *American Astronomical Society Meeting* (2007), Vol. 38, p. 278.

[3] J. Bresina, W. Edgington, K. Swanson, and M. Drummond, "Operational Closed-Loop Observation Scheduling and Execution," Technical report, NASA Ames Research Center, NASA Ames Research Center, M/S 269-2, Moffett Field, CA 84035 (1996) .

[4] A. J. Castro-Tirado, "Robotic Autonomous Observatories: A Historical Perspective," Advances in Astronomy 2010 (2010).

[5] J. F. McNall, T. L. Miedaner, and A. D. Code, "A Computer-Controlled Photometric Telescope," The Astronomical Journal **73,** 756–761 (1968).

[6] C. Pace, Senior thesis, Brigham Young University, 2010.

[7] R. L. Pearson, Senior thesis, Brigham Young University, 2010.

[8] A. Abdo, "Insights Into the High-energy γ-ray Emission of Markarian 501 from Extensive Multifrequency Observations in the Fermi Era," Astrophysical Journal **727,** 129–155 (2011).

[9] A. Danko, "ClearDarkSky,", Software, 2010, http://cleardarksky.com/.

Index