Design and Implementation of the FocusGEO Software System (Postprint)

Authors: Wang Wei, Mao Yindun, Chen Guoping, Zhang Yongshuai, Yu Yong, Luo Hao

Date: 2018-05-29T00:00:00+00:00

Abstract

FocusGEO is a new-generation dynamic monitoring system for geosynchronous orbit independently developed by the Shanghai Astronomical Observatory. Compared with most commercial mature astronomical telescopes, it lacks standard low-level hardware drivers, and due to operational environment constraints, it is impossible to design and develop fully automated software systems such as Linux-based RTS2 or Windows-based ASCOM. To address the special requirements of FocusGEO, the research team independently designed and developed a Windows-based software system capable of implementing automated telescope observation, real-time data processing, data transmission, and data management. This paper introduces the implementation framework, overall workflow, and principles of data parallel processing of the software system, with focused discussion on the implementation methods for real-time image conversion and transmission, as well as real-time data processing and database insertion. Long-term observation operations demonstrate that the software system operates stably and reliably, and can provide a reference for the development of automatic control software for non-generic telescope systems.

Full Text

Design and Implementation of the FocusGEO Software System

WANG Wei, MAO Yindun, CHEN Guoping, ZHANG Yongshuai, YU Yong, LUO Hao
Shanghai Astronomical Observatory, Chinese Academy of Sciences, Shanghai 200030

Abstract

FocusGEO is a new-generation geosynchronous orbit dynamic monitoring optical system independently developed by Shanghai Astronomical Observatory. Unlike most commercially available astronomical telescopes, FocusGEO lacks standard hardware drivers and, due to operational environment constraints, cannot utilize conventional fully automated software systems based on Linux RTS2 or Windows ASCOM. To address these special requirements, the research team developed a comprehensive Windows-based software system that enables automatic telescope observation, real-time data processing, data transmission, and data management. This paper introduces the system architecture and overall workflow for the GEO dynamic monitoring telescope, with particular emphasis on the implementation principles of real-time image conversion and transmission, real-time data processing, and database storage. Operational results demonstrate that the software system is stable and reliable, providing a valuable reference for developing automatic control software for non-standard telescope systems.

Keywords: telescope; automatic observation; communication protocol; database

---

In telescope control software systems, both the ASCOM platform and RTS2 platform exhibit strong versatility and extensibility, providing interface standards for astronomical observation equipment. The former is based on the Windows operating system, while the latter operates on Linux. Currently, both platforms are widely applied in general telescope control applications.

ASCOM (Astronomy Common Object Model) is a free, open-source astronomical interface standard that introduces a driver layer between astronomical observation equipment and application software, serving as an intermediary bridge enabling free communication between them. Equipment manufacturers need only provide ASCOM-standard compliant drivers, while application software can operate through ASCOM without considering the specific characteristics of different equipment types or manufacturers. Since ASCOM employs the COM (Component Object Model) component model, it significantly enhances system extensibility and compatibility, facilitating remote operation and real-time control of telescopes [1-2].

RTS2 (Remote Telescope System, 2nd Version) is an open-source astronomical observation management system developed for Linux platforms using C/C++. Its modular component development technology ensures portability, and its plug-and-play design philosophy allows each module and component to enter or exit operation at any time without affecting overall system performance. This is particularly important when partial code errors occur and tasks cannot continue, as it prevents system crashes [3-4].

FocusGEO, short for a new-generation geosynchronous orbit (GEO) dynamic

monitoring optical system, was independently developed by the research team at Shanghai Astronomical Observatory over a two-year period. The telescope employs a single-mount, three-tube equatorial structure (see Figure 1 [Figure 1: see original paper]), repeatedly scanning the geosynchronous orbit belt above the station to achieve catalog management of space targets and space event monitoring. Due to FocusGEO's many specialized hardware devices lacking standard low-level drivers, and because the project's operational environment requires the driver layer and application layer to reside on different computers, conventional ASCOM- or RTS2-based automated software systems cannot be developed. Instead, design must begin from the lowest level, employing appropriate network communication to independently develop a Windows-based software system that implements automatic telescope observation, real-time data processing, data transmission, and data management.

The FocusGEO software system comprises six components: telescope control software, CCD camera control software, master control software, preprocessing software, GEO satellite measurement coordinate prediction software, and image compression and transmission software. The master control software communicates with each device driver using the User Datagram Protocol (UDP) while simultaneously acquiring video stream data from IP surveillance cameras to monitor telescope movement. The image compression and transmission software performs real-time image compression and format conversion, actively pushing images and preprocessing results to remote servers via FTP and immediately entering them into an Oracle database [4]. Unlike typical telescope control software that only serves observation functions, our system integrates multi-core parallel preprocessing, real-time image compression and transmission, and data archiving, offering more comprehensive functionality and higher automation.

[Figure 2: see original paper] shows the block diagram of the FocusGEO software system. The telescope control software and CCD camera control software have relatively simple functions, and their detailed design and implementation are not elaborated in this paper. This paper provides detailed descriptions of the remaining four software components, particularly the master control software that directly interfaces with observers.

1.1 Requirements Analysis

System software is a critical factor for achieving efficient and accurate operation of FocusGEO, and must satisfy several important requirements:

(1) **High degree of automation.** To achieve repeated scanning observation of the geosynchronous orbit belt above the station, data preprocessing, image compression, and data transmission, high automation is an essential requirement for the system software to minimize manual intervention frequency [5].

(2) **Strict timing requirements.** The system employs parallel operation of image acquisition and real-time data processing to ensure no temporal

conflicts between processes. This requires strict calculation of observation time for each sky area, telescope positioning time, and data processing time, as well as careful design of data flow and timing relationships. To effectively reduce data preprocessing time, multi-core parallel processing is adopted.

(3) **Strong robustness.** Multiple ports simultaneously listen for UDP messages to avoid conflicts during message transmission and reception. Since UDP communication and FTP transmission operate concurrently, each communication thread must work without interference. Every user operation must be validated to prevent system anomalies or crashes caused by improper operation. The master control software must communicate simultaneously with telescope, CCD camera, and preprocessing software, with one-to-one correspondence of port numbers to prevent misdirected commands. Command formats must be confirmed in advance, and received commands must be format-validated to exclude erroneous messages.

(4) **User-friendly interaction.** The interface must be concise and intuitive, providing clear operational requirements to enable rapid user operation, even for those unfamiliar with the system [5-6].

1.2 System Functions

Based on the above requirements, besides telescope control and CCD camera control software, the system includes five modules/software components: task parsing control, operating condition information processing, image compression and transmission, data preprocessing, and data archiving. Their functions are as follows:

(1) **Task parsing control:** Parses point information from task plans, sends position parameters such as hour angle and declination to telescope software, controls telescope positioning and camera exposure, and executes observation tasks.

(2) **Operating condition information processing:** Processes real-time status information from front-end observation equipment, including acquisition, display, and transmission of status information, and provides equipment fault alarm functions based on real-time status.

(3) **Image compression and transmission:** Monitors FITS (Flexible Image Transport System) images captured by the CCD camera, converts them to JPG format in real time, and actively pushes compressed images to the server.

(4) **Data preprocessing:** Receives preprocessing start commands from the master control software, processes images in parallel while the telescope observes the next sky area, and completes data processing before the next sky area's observation finishes.

(5) **Data archiving:** This module runs on the remote server terminal, establishing an Oracle database to receive FTP-transmitted processed data files and enable real-time data entry.

1.3 System Workflow

Before formal operation, FocusGEO requires proper configuration of IP addresses and port numbers for the master control software, telescope control software, and CCD camera control software according to established rules to ensure normal UDP message transmission and reception. The FTP server must also be activated on the server terminal to ensure smooth workflow for observation images and data processing results.

During normal operation, the master control software sends UDP messages to the telescope control software according to the task plan. The telescope positions to the first sky area, and after confirming positioning completion, the CCD camera collects images according to preset patterns (including exposure time and frame count). During image acquisition, the image conversion and FTP transmission module monitors newly generated FITS images, converts them to JPG format, and transmits them to the remote server via FTP.

After image acquisition, the CCD camera software sends a shooting completion message to the master control software. The system begins observation of the next sky area while simultaneously calling the data preprocessing module to execute data processing. This parallel working mode significantly reduces working time. After preprocessing, results are actively transmitted to the remote server via FTP and immediately entered into the database for management. The overall workflow is shown in Figure 3 [Figure 3: see original paper].

2 System Implementation

The software system employs C# as the programming language, with Visual Studio 2012 as the development platform and .NET Framework 4.5 as the development environment.

The master control software consists of two parts: (1) a GUI interface for human-computer interaction, and (2) a UDP control port for integrated control of the telescope, CCD camera, and dome equipment via UDP. Figure 4 [Figure 4: see original paper] shows a screenshot of the master control software's working status. After clicking the start observation button, the system automatically executes the task plan, sends equipment observation parameter commands to the telescope and CCD camera, and determines next operations based on received feedback status messages from the telescope and camera, achieving automated observation. Since equipment faults are inevitable in telescope system operation, to enhance system reliability, the GUI interface provides an operating condition information module that displays real-time status information for the telescope, CCD camera, and timing system, with equipment fault alarm functions in each status bar [7].

Within the local area network, considering the project's characteristics of low transmission frequency and small message size, combined with the need to communicate with six network ports, we carefully analyzed and compared UDP and TCP communication. UDP protocol was ultimately selected due to its low resource consumption, rapid response, flexibility, and avoidance of time-consuming connection establishment required by TCP. However, UDP communication has an inherent problem: packet loss can occur when transmission frequency is too high or messages are too large. To address this, packet retransmission and timeout mechanisms were implemented, proving highly effective in practice. To avoid conflicts when the master control software receives messages from multiple ports simultaneously, we adopted a delegate approach to process received UDP messages while listening to multiple network ports. When a port receives a message, it triggers a corresponding event for that port, and the Dispatcher.BeginInvoke function executes the triggered event asynchronously, ensuring the program's main thread continues working without interference. The implementation flowchart is shown in Figure 5 [Figure 5: see original paper] [8].

During CCD camera data acquisition, the image compression and transmission software monitors FITS file creation and performs compression conversion in real time. The software uses the FileSystemWatcher class to monitor .fit suffix file creation in the `D:\DATA` folder. When an image is created, it triggers a FileSystemEventHandler event that executes a delegated function for image compression and format conversion. This delegated function creates an EncoderParameter class instance to configure image quality ratio. Under the premise of ensuring clear display on large screens (1920×1080 resolution) while meeting FTP rapid transmission requirements, repeated testing determined that an 85% image quality ratio is optimal. Under this parameter, a 5M FITS image compresses to approximately 400K in JPG format, achieving a compression ratio of about 10:1. Compressed JPG images are pushed to the remote server via FTP, displayed on large screens for 12 sky areas, and immediately entered into the Oracle database. The workflow for observation image monitoring, compression, and transmission is shown in Figure 6 [Figure 6: see original paper].

Oracle has long been a leader in the database field and is currently one of the world's most popular relational database management systems, offering good portability, convenience, and powerful functionality suitable for various microcomputer environments. Applications developed using C# can conveniently utilize Oracle databases; the interface between C# and Oracle databases refers to operations such as calling Oracle data tables and modifying data [9-10].

The .NET Framework includes four data providers for different database standards. For Oracle databases, this software uses the Oracle.NET data provider, which includes five database objects: (1) OracleConnection for database connection, containing information for creating connections to data sources; (2) OracleCommand for issuing commands to data sources; (3) OracleCommandBuilder for constructing SQL commands; (4) OracleDataReader specifically for reading data; and (5) OracleDataAdapter for retrieving data and establishing a

bridge between data and datasets.

In the program, the database connection object establishes connection with the database. An OracleConnection instance object con is instantiated, with connection string parameters consistent with client program name resolution, and con.Open() opens the database. The OracleCommand object issues commands, with OracleDataReader used for data retrieval statements and Insert operations more commonly used for writing data to tables. After operations complete, con.Close() closes the database [10].

3 Conclusion

This paper introduced the overall framework of the FocusGEO system software, adopting the basic approach of master control software integrating telescope and CCD camera control to achieve automated system operation and reduce observer workload. To date, FocusGEO has begun normal observations at the Sheshan Science Park of Shanghai Astronomical Observatory, with the system software operating in good condition. With support from the system software, FocusGEO has achieved repeated scanning observations of the geosynchronous orbit belt above the station and real-time data processing, effectively improving system reliability and observation efficiency.

The entire system software was independently developed by the FocusGEO team. Through this project, the research team has mastered key technologies in remote control, observation and data processing, image compression, database design, and data management, providing technical support for automated control of non-standard systems.

Considering Shanghai's weather is not conducive to optical observations, plans are underway to relocate FocusGEO to the Lijiang Observation Station of Yunnan Observatories, where excellent night sky conditions and continuous dry season observations will enable regular observations and stable, sustained data acquisition. To enable remote control from Shanghai via the Internet, the software system's communication protocol will be upgraded to HTTP, laying technical groundwork for subsequent international observation networks.

We thank the observation assistants from the Optical Astronomy Technology Laboratory of Shanghai Astronomical Observatory who participated in this project. They identified numerous issues during the integration debugging and test observation phases of the FocusGEO software system, providing substantial valuable feedback for continuous software improvement.

References

[1] He ShouSheng, Xin Yuxin, Lun Baoli, et al. Astronomical dome control system based on ASCOM and Modbus/TCP standard[J]. Astronomical Research & Technology, 2017, 14(3):356-362.

[2] Wu Peng. Research on ASCOM development technology for astronomical telescopes[D]. Chengdu: Institute of Optics and Electronics, Chinese Academy of Sciences, 2015.

[3] Liang Bo, Yuan Zhi, Deng Hui, et al. A method to extend CCD operations across different operating systems in the RTS2[J]. Astronomical Research & Technology—Publications of National Astronomical Observatories of China, 2015, 12(4):466-472.

[4] Wei Shoulin, Cao Zihuang, Wang Feng, et al. A study of Web control of an RTS2 system based on the WebSocket[J]. Astronomical Research & Technology—Publications of National Astronomical Observatories of China, 2014, 11(4):404-409.

[5] He Shousheng, Fan Yufeng, Wang Chuanjun. An automatic CCD observation system based on the ASCOM Standard[J]. Astronomical Research & Technology—Publications of National Astronomical Observatories of China, 2013, 10(4):386-391.

[6] Lu Dongning, Huang Lei, Chen Yinwei, et al. Research and development of the dome/slit control system for the 85cm reflector of NOAC-BNU[J]. Astronomical Research & Technology—Publications of National Astronomical Observatories of China, 2008, 5(4): 386-391.

[7] Wang Wu, Wang Jinsuo, Jiang Congguo. Design and accomplishment of the Software for the control system of the 1.2-m horizontal Telescope[J]. Publications of Yunnan Observatory, 1991(1):53-60.

[8] Yao Zhengqiu, Zhou Bifang, Wang Daxing. 1.2m infrared telescope[J]. Acta Astronomica Sinica, 1990, 31(3): 284-290.

[9] Zheng Yujun. C# Object-Oriented Programming[M]. Beijing: Posts & Telecom Press, 2007.

[10] Liu Qiuxiang, Zhang Yongsheng. The technique of visiting SQL Server using ADO.NET in Visual C#[J]. Computer Systems & Applications, 2004(11):66-69.

Note: Figure translations are in progress. See original paper for figures.