Publl. Astron. Obs. Belgrade No. 92 (2013), 99 - 104 *Contributed paper*

DEVELOPING A TELESCOPE SIMULATOR TOWARDS A GLOBAL AUTONOMOUS ROBOTIC TELESCOPE NETWORK

N. GIAKOUMIDIS[1], Z. IOANNOU[2], H. DONG[1] and N. MAVRIDIS[1]

[1]*Department of Computer Engineering, New York University Abu Dhabi, Abu Dhabi, UAE*
E–mail giakoumidis@nyu.edu

[2]*Department of Physics, College of Science, Sultan Qaboos University, Muscat, Oman*
E–mail zac@squ.edu.om

Abstract. A robotic telescope network is a system that integrates a number of telescopes to observe a variety of astronomical targets without being operated by a human. This system autonomously selects and observes targets in accordance to an optimized target. It dynamically allocates telescope resources depending on the observation requests, specifications of the telescopes, target visibility, meteorological conditions, daylight, location restrictions and availability and many other factors. In this paper, we introduce a telescope simulator, which can control a telescope to a desired position in order to observe a specific object. The system includes a Client Module, a Server Module, and a Dynamic Scheduler module. We make use and integrate a number of open source software to simulate the movement of a robotic telescope, the telescope characteristics, the observational data and weather conditions in order to test and optimize our system.

1. INTRODUCTION

A robotic telescope is an astronomical telescope that can observe the universe without being operated by a human (Wang et al. 2006). A robotic telescope system typically incorporates a number of subsystems, including a detector driving system (telescope pointing capability and dome control), focusing control system, weather station, etc. For an accurate observation and to maximize data collection the coordination of all these subsystems is essential. Moreover, as there is always an observation error introduced by a human operator, the autonomous robotic telescope can easily provide a more accurate observation with much higher efficiency. Furthermore, a single robotic telescope (regardless of its technical character, and construction cost) is limited by a number of constraints, such as telescope specification, target visibility, meteorological conditions, daylight, location restrictions, telescope availability etc.

A robotic telescope network normally consists of a number of robotic telescopes, preferably spaced over long distances in order to overcome location induced limitations. Based on the coordination of the sub robotic telescopes, the telescope network can be treated as a single observing instrument. Our final objective is to develop a system that is capable to manage and control a global robotic telescope network.

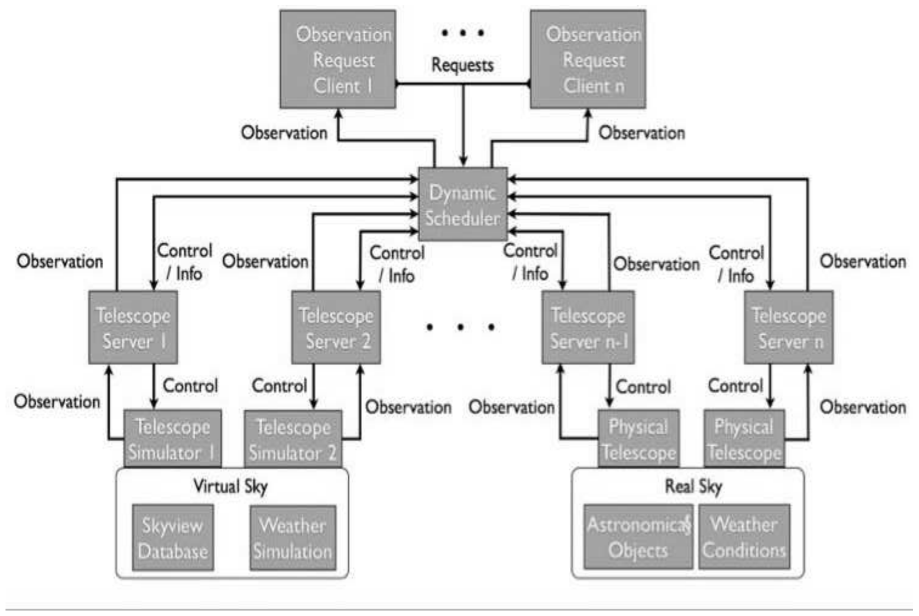

Figure 1: System architecture diagram

The robotic telescope network can be seen as a distributed sensor network with optimal estimation. It integrates the observation information from each robotic telescope. Currently there are a number of robotic telescope networks operating a wide range of telescope sizes and types (ROTSE, WET, HATNet, Robonet, Monet, LCGOT) with various degrees of human interaction involved in their operation.

Distributed sensor networks have been researched for many years (Garcia 2011). Nowadays, they have been successfully applied in forest fire detection (Ma et al. 2008), volcanoes monitoring (Werner-Allen et al. 2006), ocean measurements (Vasilescu et al. 2005), etc. While for the optimal estimation, we have many choices, like Kalman Filtering (Ribeiro et al. 2010), Information Filtering (Olfati-Saber & Shamma 2005), Viener Filtering (Kumar, Altman & Basar 2012), etc. However, the gap between the distributed sensor (i.e., single robotic telescope) for automatic optimal estimation is the information fusion. There are many kinds of robotic telescopes and each telescope's working condition is different. Here, we focus on how to combine these robotic telescope systems together. As a first step, we create a telescope simulator to act as a platform on which we will test our system

2. SYSTEM ARCHITECTURE

The proposed system that is able to manage and control the theoretical global robotic telescope network consists of three major sub-modules: a Client Module, a Server Module, and a Dynamic Scheduler Module (see Figure 1).

The following is a brief description of how the system architecture is implemented. First of all, an observation request is sent by the Client Module to the Dynamic

Scheduler Module. Then, the Dynamic Scheduler Module decides the available robotic telescopes for this specific observation request by considering the specifications of each of the telescopes, the target visibility at each telescope location, the meteorological conditions, etc. Afterwards, the Dynamic Scheduler Module communicates with the Server Module of each telescope and receives their current status. Once it finds the specific telescope that meets all the preconditions it then sends the command to the Server Module of the telescope to start the observation procedure. Later, once the telescope is on the desired position and has collected the observational data, the server module sends back to the Dynamic Scheduler Module the observational data. Finally, the Dynamic Scheduler Module forwards the observation data received from the Server Module to the Client module.

2. 1. THE CLIENT MODULE

The Client module is an application designed using the programming environment of *Labview*. It has as purpose to be the main User Interface (UI) of each client connected on the telescope network as well as the application, which can provide the observational data from the telescope (Feedback). The Client module is able to get inputs through user interface about all the necessary parameters for an observation, such as: target coordinates (declination, right ascension), desired date and time of observation, observation time, etc. When the user gives all the necessary parameters for an observation request to the Client Module following a start command, the Client Module software starts a TCP communication to the Dynamic Scheduler Module and sends all the above observation request data. The outputs of the Client module include the TCP communication data mentioned above, the real time information of the telescope (telescope position, weather condition, telescope specifications, site information, etc.) and the observational data. The algorithm of Client Module calculates the data from the user interface which is in string format, and then create bigger string containing all the data from the observation request in addition to some other data. This new-created data assists the Dynamic Scheduler Module algorithm to separate the useful information and to convert them into numeric values for further processing. After the data packet is ready, the Client Module opens a predefined network port and sends the data packet to specific IP address and port in a TCP protocol. In order to make our system more reliable, when Client Module sends the data, it leaves the same port open and wait to receive an announcement from Dynamic Scheduler Module successfully. We also add text to speech features to the Client Module in order to make a user-friendlier environment, by calling .NET classes from Windows operating system.

2. 2. THE SERVER MODULE

The Server module is an application designed using the programming environment of *Labview* and has a purpose to receive commands from the Dynamic Scheduler, to control the actual physical telescope, to send telescope status information to the Dynamic Scheduler and finally to send the Observation Data to the Client module. The Server Module is connected to the physical hardware of the telescope by using the AStronomy Common Object Model (ASCOM) driver libraries. The communication with the telescope hardware is bidirectional, which means that the Server module can give control commands to the telescope or to other instruments which are connected to the telescopes, such as cameras, filter focusers, filter wheels, domes, rotators, etc.

In addition, it receives current information for their status, such as telescope position, weather conditions, site information, dome position, selected filter, etc.

The Server Module as inputs all the necessary parameters to conduct an observation, such as, the target coordinates (Declination, Right Ascension), desired filter, etc. The inputs are made through a user interface (local control) or through a network by using TCP communication protocol (remote control). The outputs of the Server Module are the physical communication with the telescope hardware as we mentioned above. The information of the hardware state is transferred through user interface (local feedback) or through the network (remote feedback). The observation data is also transferred through user interface or through the network. To incorporate the ASCOM libraries to our software, we call .NET classes from the ASCOM open source code.

The Server Module algorithm has two operation modes: the local operation mode, and the remote operation mode. When the Server Module algorithm works on local mode, it takes the inputs values from the user interface fields and after some numeric transformations and calculation, it passes the results to the .NET classes of ASCOM libraries which communicate directly to the telescope hardware. Also, when the Server Module sends the slew command to the telescope, it tracks the telescope movement every 100 ms to make sure that the telescope moves to the right position. When the desire position is reached, it informs the user through the user interface that the telescope is on the desired position. When the Server module runs on remote mode, it opens a specific network port and waits to receive the data from the Dynamic Scheduler which contains the necessaries values to control the telescope. If the values are received successfully, it starts the same procedure described above to drive the telescope to the desired position. Meanwhile, it sends back to the Dynamic Scheduler module an announcement that the data has been received successfully. When the telescope is executing a movement the Server Module sends to the Dynamic Scheduler the current position of the telescope every 200 ms. For all the remote communication with the Dynamic Scheduler, the Server module sends and receives data by using the TCP protocol.

2. 3. THE DYNAMIC SCHEDULER MODULE

The Dynamic Scheduler Module is an application designed using the *Matlab* software package. Its purpose is to receive observation requests from the multiple Client Modules. By considering all the necessary parameters, it decides which telescope is appropriate for each observation request and makes the connection between the Client Module and the Server module.

We are experimenting with a host of scheduling algorithms at this stage. The incoming observation requests are partially specified, allowing for flexibility in their allocation; for example, a total observation duration might be specified, but within a flexible time window, and with the possibility for breaking the observation time to a number of sub-segments, which add up to the total required time. Most importantly, the scheduler might dynamically reschedule the observation in real time, given changing weather conditions or technical failures which necessitate the reallocation of observations that were assigned to telescopes which are now unavailable.

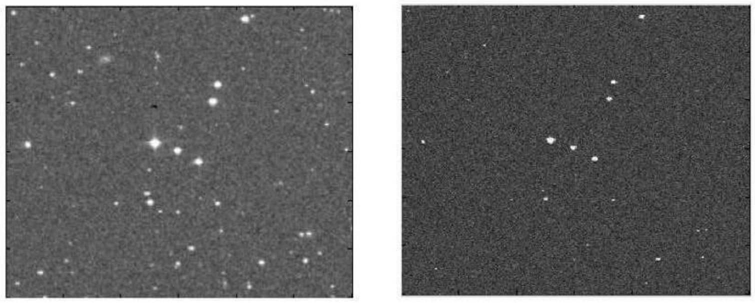

Figure 2: Image processing for adjusting observation images. Before adjustment (Left), after adjustment (Right)

3. THE SIMULATOR

To fully test our system and make the required experiments and optimization, we add some simulation features in our software. Those features include telescope movement, observation and observation image adjustment.

3. 1. TELESCOPE MOVEMENT

The simulation of the telescope movement is built using open source software. The open source software *Stellarium* is used to simulate the telescope movement and visualize the telescope pointing on the sky. When the Server Module receives the slew command and sends this command to the robotic telescope, the simulator receives back the actual position of the telescope every 100 ms until the telescope arrives at the desired position. When the Server Module receives a new position of the telescope, it transfers a value to the Stellarium software by using the UDP protocol.

3. 2. OBSERVATIONS

To be able to simulate the observation data we use the Digital Sky Survey (DSS) database from the *SkyView Virtual Observatory* by incorporating the *SkyView* Java interface into our Client Module. When the Client module receives that the virtual telescope is on the desired position, it runs an open source Java script code, which has access to the database of DSS images and retrieves the hypothetical observation data from the specific position of the simulated telescope. The retrieved images are FITS type images.

3. 3. OBSERVATION IMAGE ADJUSTMENT

To be able to simulate the effects of weather and sky conditions to the simulated sky images, we employ Gaussian filtering as well as artificial noise floor modulation, in order to create adjusted images, which correspond to what we would expect to have seen from the sky given specific telescope characteristics. An example set of images can be seen in Figure 2.

4. CONCLUSIONS AND FUTURE WORK

In this paper, a simulator was developed. It has the function of positioning to a desired target based on the client request. In the future we plan to implement a simulation of real time weather conditions in addition to real time data reduction, data quality evaluation and real time object classification on every acquired image with the final goal being to test the system on a real life telescope network.

References

Garcia P.: 2011, *IEEE Transaction on Knowledge and Data Engineering*, **11**, 4

Kumar, D., Altman, E., Basar, T. : 2012, *Proceedings of 9th Annual Conference on Wireless On-Demand Network Systems and Services*, 67

Ma, Y., Richards, M., Ghanem, M., Guo, Y., Hassard, J., : 2008 *Sensors*, **8**, 3601

Olfati-Saber, R., Shamma, J.S., : 2005, *Proceedings of IEEE International Conference on Decision and Control*, 6698

Ribeiro, A., Schizas, I., Roumeliotis, S., Giannakis, G.: 2010 *IEEE Control Systems*, **30**, 66

Vasilescu, I., Kotay, K., Rus, D., Dunbabin, M., Corke, P., : 2005, *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems*, 154

Wang, J., Mukherji, R., Ficocelli, M., Ogilvie, A., Liu, M., Rice, C. : 2006, *Modeling and simulation of robotic system for servicing HST, Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1026

Werner-Allen, G., Lorincz, K., Welsh, M., Marcillo, O., Johnson, J., Ruiz, M., Lees, J.: 2006, *IEEE Internet Computing*, **10**, 18