A Flexible Python Observatory Automation Framework for the George Mason University Campus Telescope

Michael Reefe[1], Owen Alfaro[1], Shawn Foster[2], Peter Plavchan[1,2], Nick Pepin[1], Monica Vidaurri[3], Kevin Collins[1], Justin Wittrock[1], Patrick Newman[1], Mary Jimenez[1], Michael Bowen[1], Kevin Eastridge[1], Taylor Ellingsen[1], Deven Combs[1], John Berberian[1,4], Natasha Latouf[1], Catilin Stibbards[1], David Vermilion[1,3], Kingsley Kim[1,5], Sudhish Chimaladinne[1,5], Shreyas Banaji[1,6]

1. *Department of Physics and Astronomy, George Mason University, 4400 University Dr, MS 3F3, Fairfax VA, USA, 22030*
2. *Department of Physics and Astronomy, Rochester Institute of Technology, 1 Lomb Memorial Dr, Rochester NY, USA, 14623*
3. *NASA Goddard Space Flight Center, 8800 Greenbelt Rd, Greenbelt MD, USA, 20771*
4. *Woodson High School, 9525 Main St, Fairfax VA, USA, 22031*
5. *Thomas Jefferson High School, for Science and Technology, 656- Braddock Rd., Alexandria, VA, USA, 22312*
6. *Flint Hill School, 3320 Jermantown Rd., Oakton, VA, USA, 22124*

Abstract

We present a unique implementation of python coding in an asynchronous object-oriented framework to fully automate the process of collecting data with the George Mason University Observatory's 0.8-meter telescope. The goal of this project is to streamline the process of collecting research data and monitoring weather, most often for follow-up observations for the TESS mission. We have automated slews and dome movements, CCD exposures, saving FITS images, focusing and guiding on the target, and taking calibration images (darks and flats). We also have automated periodically checking weather conditions to automate the decision-making involved in whether a shutdown is necessary. We are now able to input the specifications of the desired target in a user-friendly GUI that generates an input configuration file and launches the command-line code at the beginning of the night. The code, in its current state, has been tested and used for observations without error on at least 110 nights.

Introduction

The transit method became the most successful method for discovering new exoplanets with the launch of Kepler[1] in 2009, and more recently, the Transiting Exoplanet Survey Satellite (TESS) mission in 2018.[2] By observing the dips in star brightness as an exoplanet passes, or 'transits,' in front of its host star, we can acquire knowledge about exoplanet properties such as the radius, period, orbital eccentricity, atmospheric transmission properties, and in some cases additional planets in the same system[3]. TESS has found over 4000 exoplanet candidates during its first three years of scientific operation, monitoring ~500,000 relatively nearby and bright stars.

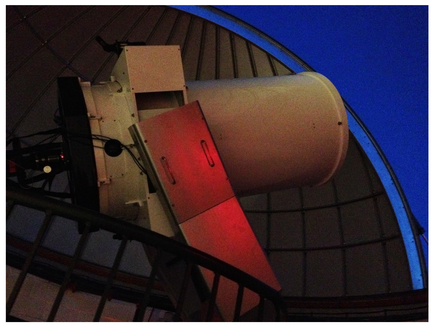

Figure 1:
Picture of our 0.8-meter Ritchey-Chrétien telescope

However, because the pixel scale of the on-board sky cameras of the TESS mission is 22 arcseconds per pixel, light from background stars inevitably contaminates the pixels that are also recording light from the star under investigation.[2] This may lead to false positives from a variety of eclipsing binary scenarios that are not transiting exoplanets. This necessitates photometric follow-up monitoring of TESS candidates at finer angular resolutions in order to further characterize these dips in brightness. This has prompted us to pursue full automation of the process with the 0.8-meter Ritchey-Chrétien telescope at George Mason University (GMU), using an asynchronous object-oriented approach in Python. The facility had previously been manually operated by a combination of webcams, TheSkyX, ASCOM Dome, MaxIm DL, RoboFocus, and a weather station. There are many existing software tools for the automation of telescope and observatory controls, including MaxIm DL, Robo-AO[4], the MINERVA control software,[5] DEMONEX[6], and other professional and commercial software packages. However, we chose to develop our own custom Python software optimized for both the George Mason Observatory hardware setup and science observation goals. The development of our automation software has been ongoing since 2018 and is available on GitHub at https://github.com/Kakon24/OmegaLambda.

References

1. W. J. Borucki, D. Koch, G. Basri, *et al.*, "Kepler planet-detection mission: Introduction and first results," *Science* 327(5968), 977–980 (2010).
2. G. R. Ricker, J. N. Winn, R. Vanderspek, *et al.*, "Transiting Exoplanet Survey Satellite," *Journal of Astronomical Telescopes, Instruments, and Systems* 1(1), 1 – 10 (2014).
3. D. Nesvorny and A. Morbidelli, "Mass and orbit determination from transit timing variations of exoplanets," *The Astrophysical Journal* 688, 636–646 (2008).
4. C. Baranec, R. Riddle, A. N. Ramaprakash, *et al.*, "Robo-ao: autonomous and replicable laser-adaptive-optics and science system," *Adaptive Optics Systems III* (2012).
5. J. J. Swift, M. Bottom, J. A. Johnson, *et al.*, "Miniature exoplanet radial velocity array i: design, commissioning, and early photometric results," *Journal of Astronomical Telescopes, Instruments, and Systems* 1, 027002 (2015).
6. J. Eastman, B. S. Gaudi, R. Siverd, *et al.*, "DEMONEX: the DEdicated MONitor of EXo-transits," in *Ground-based and Airborne Telescopes III*, L. M. Stepp, R. Gilmozzi, and H. J. Hall, Eds., 7733, 1243 – 1250, International Society for Optics and Photonics, SPIE (2010).
7. K. Collins and J. Kielkopf, "Astroimagej: Imagej for astronomy," (2013). Astrophysics source code library.

Software Structure & Design

We have structured the automation software for our campus telescope on the principles of object-oriented programming and multithreading, to run in Python on a Windows 10 operating system. Each hardware device, including the charge-coupled device (CCD) camera, telescope mount, dome, flat-field lamp, and focuser, was given its own Python class, each with their own attributes and methods, as well as their own CPU thread to allow for actions to be performed simultaneously and asynchronously across hardware devices. These hardware modules, along with the data input/output (I/O) module, form the foundation upon which our core systems and higher-level structures are built.

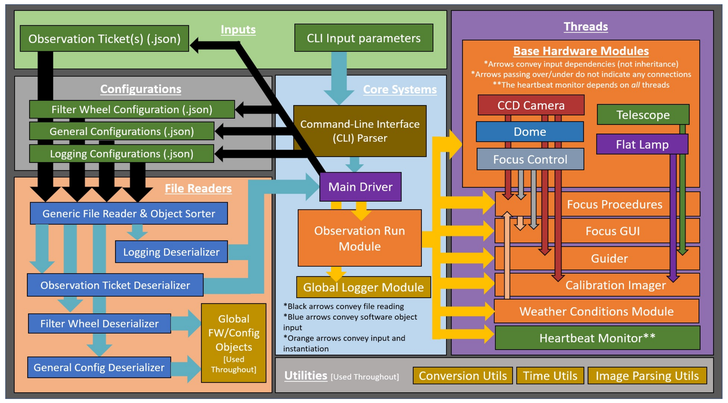

Figure 2: Automation code structure flowchart

These higher-level structures rely on the interplay of multiple hardware components, and they include automated focusing on the target, taking dark and flat-field images for data calibration and reduction, active guiding (i.e. keeping the stars in a stable position in the image frame by making corrections to the small errors between our telescope's passive tracking and the Earth's rotation), monitoring the local weather conditions, and even monitoring the status of each thread of the code itself. A flowchart of how all the modules of the code interact is shown in figure 2. In this poster, we assess two of these modules: the automated focusing and guiding, as well as the general quality of data collected.

Focusing

The goal of our focusing module is to efficiently achieve and maintain an optimal focus. Variable atmospheric conditions sometimes rapidly affect focus quality, and our software and hardware impose additional constraints on our measurements. The quantity of measure that we use to assess focus quality is the full-width at half-maximum (FWHM) of the starlight's point-spread function (PSF). However, because we want to avoid unintentionally driving the focus mechanism too far in one direction, and to save time on focusing, we have chosen to codify the assumption that the initial focus position is already close to the optimal position, which is true in most cases, and perform a simple grid search for the initial focus. Then, we adopt a linear temperature-dependent model to adjust the focus over the course of the night as the dome cools to atmospheric temperatures, where we parametrize the focal drift as 2 steps/°F inwards (1 step ≅ 12.7 μm). This is purely an empirical estimate, and we defer detailed characterization of its effectiveness and other parameters (i.e. the telescope's gravity vector) to future work.

Guiding

The guiding module tracks the movement of stars between images and adjusts the telescope's position to make corrections. This is done by reading the brightness values of the pixels in the images we collect, finding the peaks, and comparing them to the peak positions in previous images. The difference in position is calculated, converted into arcseconds, and a damping coefficient is applied before sending the telescope jog instructions to prevent overcorrection.

Conclusions

We have implemented an asynchronous and object-oriented framework where each piece of hardware and higher-level function is associated with a Python class. By building upon these base hardware classes and creating higher-level modules for important observing tasks, our software can monitor weather, acquire targets, focus, guide, and collect data and calibrations before shutting down. The reduction and analysis results from TOI 2081.01 have demonstrated the base functionality of our automation software, and its ability to collect data with a quality on par with, or even marginally above, our manual observations due to the increase in pixel stability provided by our active guiding module that has reduced red noise from pixel sensitivity variations. The focusing module, while it performs well in ideal weather conditions, has room for improvements in its interpretations of torus-shaped defocused PSFs.

Results & Discussion

TOI 2081.01

We have observed TESS candidate exoplanets with our automation software without error for at least 110 clear nights and counting (not including testing nights). We first present an overview of an example transit event attributed to TOI 2081.01, an exoplanet orbiting the star TOI 2081 (or UCAC 716-056861), on the night of UT 2021 May 13. The data was obtained using our automation software. Observations were performed in the R band, as is typical of our observatory, and observations began about 2 hours prior to the transit to achieve a sufficient photometric baseline. Midway through the transit, the code automatically determined that it was too cloudy to continue observations based upon satellite data and closed the dome. In about 1 hour, conditions cleared again, and the code was able to automatically resume observing the egress of the transit, allowing us to recover a robust detection of the transit. The data was reduced, and aperture photometry was performed using an AstroImageJ[7] software pipeline. The deep transit can be seen in figure 3.

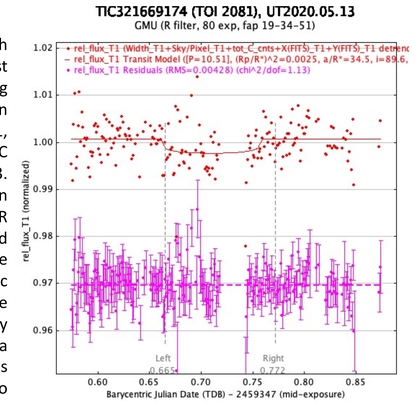

Figure 3: TOI 2081.01 transit light curve

Focusing

As shown in figure 4a, on a clear night with relatively stable weather conditions, the code is able to step through 11 unique focus positions, fit a positive-concavity parabola, and move to a minimum focus (the minimum point of the parabola). In this case, the minimum was relatively close to the initial position, which contributed to the success of the parabolic fit. During nights where the initial focus position is not close to the minimum position, however, our focus algorithm is unable to find a minimum because it obtains an incomplete picture of the parabola. An example of this is shown in figure 4b, where the data points form an almost linear fit shape.

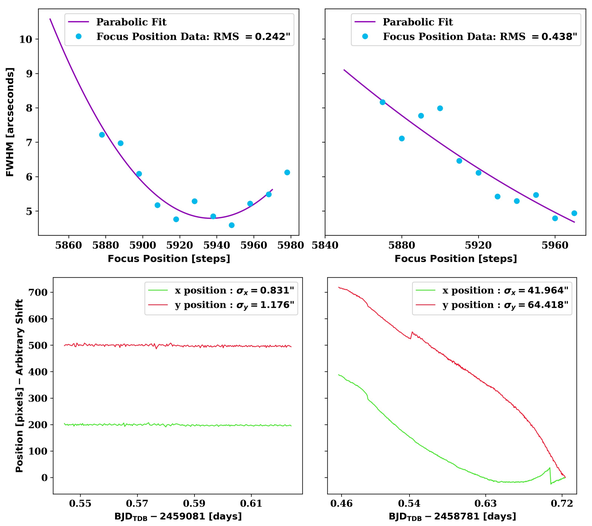

Figure 4:
***Left* – 4a, the focus FWHM positions on a night with ideal initial conditions**

***Right* – 4b, the same as 4a, on a night with less ideal initial conditions.**

Figure 5:
***Left* – 5a, the guider x (green) and y (red) pixel positions over the course of a night, while the active guiding was being used.**

***Right* – 5b, the same as 5a, on a night where the active guiding was *not* being used.**

Guiding

Figure 5 shows a comparison between a night when the automated guider was utilized (5a) against a night when it was not (5b). The large jump in 5b is likely due to a manual telescope jog to re-center the star. It is visible from 5a that we have been able to keep star positions steady and actively guide with an RMS as low as 3.23 pixels, or 1.2''. This is a vast improvement to the non-guiding RMS as high as 64.4'', a reduction of over 98% in both directions. This is also beneficial to our photometry and modeling efforts post-observation, as having a more stable pixel position will eliminate noise from differences in pixel sensitivities, bad pixels, and the gradual motion of the telescope. In general, the guider works this well in all but the most cloudy conditions. However, since we guide on the science images, we must rely on the cadence of the target's exposure times for guiding.