Wavefront sensing and control software for a segmented space telescope

Scott A. Basinger[*a], David C. Redding[**a], Fang Shi[a], David Cohen[a], Joseph J. Green[a], Catherine M. Ohara[a], Andrew Lowman[a], Laura A. Burns[b]
[a]Jet Propulsion Laboratory, California Institute of Technology;
[b]Science Systems and Applications, Inc.

ABSTRACT

The Segmented Telescope Control Software (STCS) uses science camera information to align and phase a deployable segmented optical telescope. It was developed the for Next Generation Space Telescope (NGST) and has been successfully utilized on the Wavefront Control Testbed (WCT) for NGST and a portable phase retrieval camera (PPRC) system. The software provides an operating environment that will be used for the prime contractor's testbeds for NGST, and will eventually evolve into the Wavefront Sensing and Control (WFS&C) ground support software for NGST. This paper describes the engineering version of the STCS, the algorithms it incorporates, and methods of communicating with the testbed hardware.
Keywords: wavefront sensing, phase retrieval, Next Generation Space Telescope

1. INTRODUCTION

The NGST[1] is a deployable, cryogenic, actively controlled observatory that will have a primary mirror with a diameter of 6 meters. The optical telescope assembly (OTA) will be extremely lightweight and diffraction limited at 2 um. It is expected that after the initial deployment, the OTA will be highly aberrated and will need adjustments to bring the segments into phase. The baseline algorithms for this procedure have been described in previous publications[2]. The STSC has been developed to test the baseline algorithms on the WCT. It has been an invaluable tool in developing and verifying the performance of the WFS&C algorithms and in validating our computational models. In addition, the STSC has been expanded to work on similar testbeds and will eventually evolve into an operational software package that will be incorporated in the ground support software for NGST to perform the initial WFS&C and maintenance of the optical alignment of the OTA.

The STCS has an extensive graphical user interface that allows a user to aberrate a telescope simulator, or testbed, to mimic a deployable space telescope in its initial state. The user can then perform all the steps necessary to bring it into phase to the diffraction limit. It is hosted in Matlab, but has links to other software packages for optical modeling, TCP/IP connectivity, and allows parallel execution of code that is computationally intensive. The TCP/IP connections allow us to remotely control testbeds from around the country. The parallel programming greatly increases the speed at which the phasing algorithms can be executed and can run on a cluster of workstations or a Beowulf cluster.

The software incorporates several algorithms that incrementally reduce the residual wavefront error of the optical system. "Coarse Alignment" captures the starlight from individual segments of the telescope and places the light reflected from each segment on top of each other to maximize the optical intensity at the focal plane. After this, each segment is pistoned with respect to the optical axis to maximize the encircled energy at the center of the detector. "Coarse Phasing" uses a dispersed fringe sensor to resolve the residual piston errors between the segments, and white-light interferometry reduces the piston errors even further. "Fine Phasing" uses Gerchberg-Saxton[3] phase retrieval to get a high-resolution phase map of the system, which is then used to correct the error with actuators on the mirror or a

[*]sab@huey.jpl.nasa.gov; phone 1 818 354-3065; Jet Propulsion Laboratory, California Institute of Technology, M/S 306-451, 4800 Oak Grove Drive, Pasadena, CA, USA, 91109-8099;
[**]dcr@huey.jpl.nasa.gov; phone 1 818 354-3696; Jet Propulsion Laboratory, California Institute of Technology, M/S 306-451, 4800 Oak Grove Drive, Pasadena, CA, USA, 91109-8099;

deformable mirror. For the "Maintenance Mode," we use an algorithm called "In-focus Point Spread Function (PSF) Optimizer (IPO)[4]," which can estimate errors caused by segment drift and correct them. All the phasing is done with the science camera, greatly reducing the cost of the system by not requiring additional instruments for wavefront sensing. Another benefit of using the science camera for phasing is that no extra aberrations will be induced from non-common path optics.

We are using a model-based approach to design NGST. As such, the software was first written to run models of the hardware. As the hardware for the testbed was built, the software performed remarkably well, proving that our models were accurate and useful in the design process and in algorithm development.

Section 2 of this paper describe the components of the STCS and its overall design philosophy. Section 3 describes the architecture, standards, and organization we developed for the STCS. Section 4 contains a summary of a complete phasing session on the WCT architecture. And Section 5 describes future developments for the STCS.

2. PHILOSOPHY

The main purpose of the engineering version of the STCS is to 1) develop algorithms for phasing actively controlled telescopes using optical models, 2) verify the performance of these algorithms on hardware testbeds, and 3) validate the accuracy of the models so that similar methods can be used for model-based design of flight hardware. The STCS combines the optical models, algorithms for measuring aberrated wavefronts, and control methods for phasing segmented telescopes into a single package. It is important that the tasks mentioned above be easy to use and as automated as possible, especially for running Monte Carlo simulations on the hardware. In addition, remote access to the various testbeds is necessary to allow the user to run from outside that lab so as not to disturb the laboratory environment and also to allow access to testbeds from remote locations. The STCS is more than just a simple software package for running the NGST testbeds, it also provides an operating and programming environment for similar testbeds and to date has been successfully implemented on two other testbeds requiring image-based wavefront sensing.

We have developed software standards for the STCS to allow a certain homogeneity to the code we develop. This enables better readability for multiple developers and a basis for new developers to be more effective. In addition, the code is modular, which allows developers to concentrate only on the code they are currently working with.. Different algorithms and functions are well separated into different categories that are classified by their utility. Each module also has a data structure associated with it that contains all the data needed to display and perform the operation it is intended to do. Examples of different modules are GUI panels, algorithms, network communications, data archiving, and optical models.

In addition to modular functionality, each major algorithm developed for use in the STCS is well separated. As such, each algorithm has its own subsection and its own separate GUI panel to control and test the algorithm. These panels appear only when the user requests them and disappear when not in use so the computer screen does not become overly complicated with too much information. Each algorithm/panel typically has a large amount of data associated with it, and this data is collected into separate data structures for each. Having unique panels for each major algorithm leads to modularity so that functions can be added or removed without affecting the rest of the software.

Code reusability of the software is essential and provides for rapid development of the control software for new testbeds. We have been able to reuse some GUI panels for other testbeds without modification, while others can be used as templates for developing similar GUI panels quickly. In addition, most of the algorithms remain the same and can also be reused. Since we typically define the network communications protocols, we can use previous versions as a template for new hardware. The optical models, while quite complicated, can also be used as templates for new optical systems, greatly reducing the development time.

Incorporating our optical models directly into the STCS has several advantages. First, the algorithms for wavefront sensing and control can be entirely developed and matured using only the optics models. This saves time and effort when compared to developing them on the hardware. Our optical models have a high degree of fidelity and this approach has proven to be most successful when the algorithms were finally tested on the hardware. Secondly, the models themselves are typically incorporated in the algorithms to allow the most accurate knowledge of the optical

system to be utilized. Finally, having the optical models work in the same environment as the testbed gives the developer direct access to the models for tweaking various parameters and access to outputs of the model that can aid in the algorithm development, understanding results, or visualization for the GUI.

3. ARCHITECTURE

3.1 Software packages

The front-end computer "language" that we chose for the STCS is the MathWorks Matlab®, since rapid prototyping of algorithms is easily performed. In addition, Matlab has extensive GUI capabilities that we took advantage of to make the code easier to use. Matlab can also call externally compiled code, which allows us to link our optical modeling software into the STSC. Matlab itself is an interpreted language, but once routines are matured, they can be re-written in a compiled language and called from Matlab to speed up computations. It is also a goal that other developers can easily plug in new algorithms that they've written in Matlab, although this has not been exploited yet.

The optical modeling package that we use was developed at JPL and is called MACOS (Modeling and Analysis for Controlled Optical Systems)[5]. MACOS is written in FORTRAN and is therefore not immediately compatible with Matlab. However, we use MACOS as a library and write a "gateway" function to run an optical model from Matlab. Once this gateway function has been written, a user can exercise the optical model by a single function call that has as input parameters perturbations for the optical elements in the system, including commands for actuated segments and deformable mirrors. The gateway function then calls MACOS routines and returns a simulated image from the system as well as the aberrated wavefront as an optical pathlength difference (OPD). This methodology allows the user unfamiliar with MACOS to treat the optical models as a "black box."

The STCS is supported under Solaris and Linux operating systems. It runs under X-Windows, which is used extensively for the GUI. X-Windows allows another layer of remote accessibility, since the STCS can be run on a remote server and displayed locally using an X-Windows client. The latest production version of the code is maintained on a server in a special account created for the testbed. A typical user will set an environment variable to point to that account to run their code. Their preferences and individualized settings are stored in their local directory, however, allowing each user to personalize their environment. The code itself is maintained by CVS (Concurrent Version System), which allows multiple developers to each have their own copy of the code simultaneously. The above mentioned environment variable can be set to a developmental version of the code for testing purposes before new code is committed to the official version. CVS also allows remote access to the code repository, allowing developers to work across the country or at home on local computers.

3.2 Networking

The STCS uses TCP/IP to connect from a user's local computer to the computers that run the hardware. A Berkeley socket is set up between the computers. We have defined Interface Control Documents (ICD's) that specify commands that are sent between these computers. There are two basic commands for every piece of actuated hardware: query and move. "Query" asks a piece of hardware what its current status or setting is, for example, a filter wheel position. "Move" commands a piece of hardware to perform a certain function, for example, a translation stage should go to 44.5 mm. The commands are sent along with sequence numbers to make sure that no command is dropped. If a command is executed, a status byte is returned to confirm success. If there is a hardware or transmission error, an error code will be sent back for evaluation. The "query" and "move" commands are coordinated so that a piece of hardware is not sent a "move" command if it has been "queried" and is already in the desired position. In addition to filter wheels and translation stages, we also remotely control the white-light source and the camera, the latter returning images through the socket connections. Actuators for the mirror segments are controlled, too, and we plan to implement a closed-loop fast steering mirror in the future.

Figure 1 shows a schematic of the networking used by the STCS to control the WCT through the internet. The STCS uses externally compiled C code to make the Berkeley Socket connections to the other controller computers. These computers, in turn, use Lab Windows, Lab View, or C-code to communicate via sockets to the STCS. More information about the supporting hardware for the WCT can be found in Reference 6.

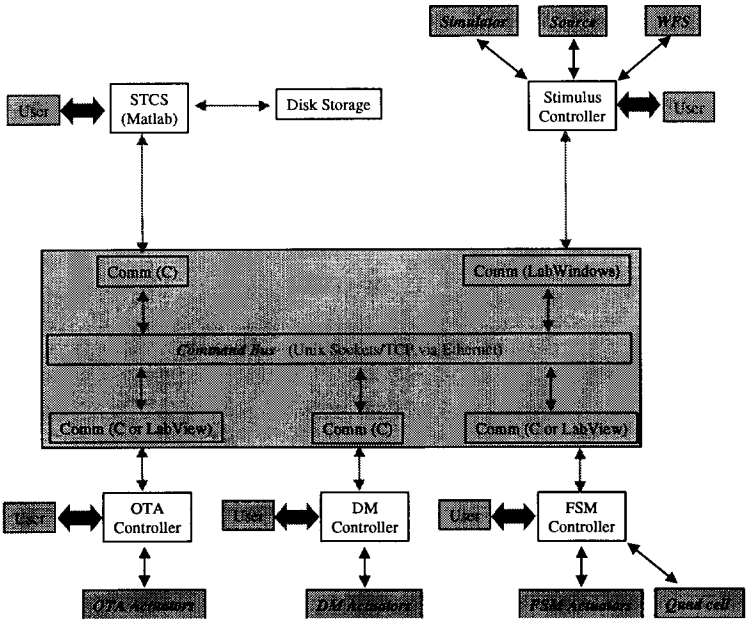

Figure 1. Network connections

3.3 Model server

In addition to using MACOS models that are compiled into Matlab executable (mex) functions and called directly from the STCS, we have also developed a network based model server. The model server has a C front end that uses Berkeley sockets to communicate with the STCS. It runs MACOS models, but all communications with the STCS go through the sockets and use the same interfaces that are described by the ICD's we developed for the hardware computers. All commands that are used for controlling actuated devices on the testbed are mimicked in the model server. Values of actuator positions and translation stage positions are stored and recalled when an image is taken, so that a proper image is returned. Noise models are also built into the model server to provide laboratory quality images. The noise models are updated by checking real images from the hardware and incorporating realistic values into the model server. The model server can also be "queried" as to the current locations of it's devices so that all functionality of the real hardware is imitated.

The model server has two major benefits. The first is that network communications can be tested an implemented before the actual hardware is ready. This allows for rapid integration of the STCS with the testbed, since debugging can be done purely with models. The other benefit is that the models are run in exactly the same way that the testbed is used, and a new user can be trained using the model server instead of the hardware, reducing cost by not running directly on the hardware. Since all communications with the model server must go through the socket stream, no extra information is passed between the algorithms and the models, allowing for a complete separation of the two and providing an environment that closely simulates the data received from the hardware.

3.4 Organization

The GUI panels are organized in a sequential hierarchy according to functionality. The first GUI panel that a user is presented with is the "Init," or initialization panel. It allows the user to choose from different hardware setups, optical light sources, and data archiving options. For an end-to-end phasing experiment, a user will run through four major steps: aberrate the system, coarse alignment, coarse phasing, and fine phasing. Each of these functions can be chosen from the "Init" panel in sequence. As each is chosen, the "Init" panel disappears and the corresponding GUI panel for the desired step becomes available.

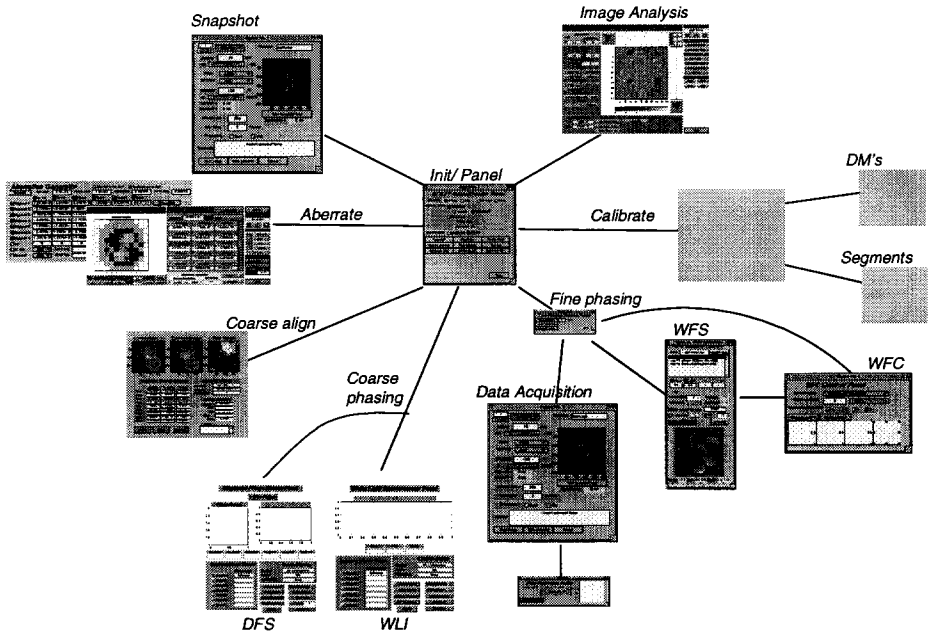

Figure 2. Spider picture

Figure 2 shows an overall layout of the GUI panels and their connectivity. The "Init" panel is in the middle, and the sub-functions are displayed radially about the "Init" panel. In the figure, lines between the various GUI panel depict their connectivity. In addition to the four main functions listed above, other functions such as calibration can be called from the "Init" panel. The various sub-panels for an end-to-end phasing experiment start with the "Aberrate" panels and proceed counter clockwise.

4. WALK-THROUGH

This section presents a brief walk-through of an end-to-end phasing experiment on the WCT using the STCS. A typical phasing experiment starts with aberrating the WCT. The "Aberrate" panels allow a user to aberrate the optical system to simulate a post-deployment state of a telescope. Mirror segments can be completely misaligned so that no light falls on the detector. Deformable mirrors can have actuators put into random states or have different Zernike modes applied to them. These aberrations are typically static, or "set and forget," since our algorithms assume that the space telescope's environment will be relatively stable over a period of several hours to days. The three main phasing steps can be executed and their progress can be monitored from the "Aberrate" panels, since the current alignment of the mirror segments and actuators of the deformable mirrors can be observed while corrections are taking place.

The "Coarse Alignment" panel allows the user to step through the algorithms that initially capture the light and then maximize the encircled energy to align the segments to within about five waves of wavefront error. The "capture" algorithm works by moving each segment in a spiral-pattern until the light falls on the detector. It detects the light this by differencing a reference image from an image with current state of the segment. These images are displayed on the panel. In addition, the values of the actuators for each segment are displayed and can be tracked while the segments are exercised. After the light from each segment is detected, each segment is individually pistoned to maximize the encircled energy, which can also be monitored from the panel. These operations can be performed automatically with the click of a single button, or stepped through manually to get a closer look at the progress.

The next step is "Coarse Phasing" which consists of two steps: using a dispersed fringe sensor and while light interferometry between pairs of segments to reduce the wavefront error in the system to about one wave. The

"Dispersed Fringe Sensor" panel allows a user to choose a reference segment and a segment to align to the reference. All other segments are moved out of the way. A grism is put into the optical train and fringes are formed on the detector with a period that is proportionate to the piston error between the two segments. This period is measured and the segment being aligned is moved to correct the error. This continues pair-wise until all segments have been aligned. The "White Light Interferometer" panel works in a similar fashion, but without the grism. Interference patterns from mis-aligned pairs of segments are analyzed to reduce the wavefront error even further. Both panels are currently run manually, with the user determining which pairs of segments to operate on. The user takes the data and presses buttons to analyze the fringes and a resulting number for the displacement is displayed along with a graphical representation of the data. The user then presses a button to correct the displacement and continues until all the segments have been phased.

The next step is "Fine Phasing," which incorporates two methods for phasing the system to $1/100^{th}$ of a wave. The first is our Modified Gerchberg-Saxton (MGS) phase retrieval. The second is In-focus PSF Optimizer (IPO). MGS phase retrieval takes several out-of-focus images and a pupil image and estimates the wavefront error. This allows us to correct the error using the actuators on the segments or the deformable mirrors. The user first uses the "Data Acquisition" panel to set up all the images, which are then taken in sequence. From there, the "Wavefront Sensing" panel is used to pre-process the data, run the MGS phase retrieval, and post-process the resulting wavefront estimate. The "Wavefront Control" panel is then used to determine the optimal control to correct the wavefront error and apply it to the WCT. We plan to use IPO for the maintenance of the figure of the system. IPO takes an in-focus image and measured modes of the segments and runs a parametric optimizer on the modes to match the image. This determines the state of the mirror segments, and allows for their correction. The "IPO" panel allows various parameters to be chosen for the optimizer and displays the image and the segment misalignments.

5. FUTURE DEVELOPMENTS

The wavefront sensing portion of the STCS is used for the portable phase retrieval camera (PPRC)[7]. The PPRC is an instrument that we can take to other facilities to measure the quality of their optics. This is most useful for segmented optics since a shearing interferometer cannot handle discontinuities. We initially plan to use the PPRC to test optics that are technology demonstrators for NGST.

The STCS has already proven an effective environment for wavefront sensing and controlling segmented telescope testbeds. We plan to expand its capabilities to work on other forthcoming testbeds for NGST as well as for other large optical telescopes currently planned. We would also like to allow other developers to contribute to algorithms for various testbeds and therefore we plan to standardize algorithm interfaces. Algorithms will not be required to be written in Matlab, but instead will be accessed through Application Program Interfaces (API's). Algorithm interface documents will be defined standardize the inputs and outputs of a particular algorithm. Rapid development of new GUI panels is facilitated by code re-use.

In the long term, the STCS will evolve into the ground support software to initially phase NGST after deployment and to perform ongoing figure maintenance. As the software becomes more mature, we plan to move it away from Matlab, preferring to use a compiled language instead. The current plan is to have the STCS be event driven, residing on a single server, with users accessing it through a web browser. This allows for more control of the software and the users accessing it.

REFERENCES

1. B. D. Seery, "Next generation space telescope (NGST)," SPIE Paper 3356-01, Kona, HI (1998).
2. David Redding, Scott Basinger, Andrew E. Lowman, Fang Shi, Pierre Bely, Richard Burg, Gary Mosier, "Wavefront Control for the Next Generation Space Telescope," SPIE Paper 3356-47, Kona, HI (1998).
3. Gerchberg-Saxton
4. C. Ohara, et al, IPO paper, SPIE Paper 4850-??, Waikoloa, Hawaii (2002).
5. MACOS reference
6. L. Burns, S. Basinger, et al, "Wavefront Control Testbed Integrated Software System," SPIE Paper 4850-??, Waikoloa, Hawaii (2002).
7. Phase Retrieval Camera reference