Open-source data analysis and visualization software platform: SAGUARO

Dae Wook Kim*[a], Benjamin J. Lewis[a], James H. Burge[a]
[a]College of Optical Sciences, Univ. of Arizona, Tucson, AZ 85721, USA

ABSTRACT

Optical engineering projects often require massive data processing with many steps in the course of design, simulation, fabrication, metrology, and evaluation. A MATLAB^{TM}-based data processing platform has been developed to provide a standard way to manipulate and visualize various types of data that are created from optical measurement equipment. The operation of this software platform via a graphical user interface is easy and powerful. Data processing is performed by running modules that use a proscribed format for sharing data. Complex operations are performed by stringing modules together using macros. While numerous modules have been developed to allow data processing without the need to write software, the greatest power of the platform is provided by its flexibility. A developer's toolkit is provided to allow development and customization of modules, and the program allows a real-time interface with the standard MATLAB environment. This software, developed by the Large Optics Fabrication and Testing group at the University of Arizona, is now publicly available.** We present the capabilities of the software and provide some demonstrations of its use for data analysis and visualization. Furthermore, we demonstrate the flexibility of the platform for solving new problems.

Keywords: Data analysis software, data visualization software, MATLAB^{TM}-based data processing platform

1. INTRODUCTION

Many high precision optical engineering projects require cutting-edge technologies to maximize the final optical system performance. Since the overall performance of such a system is not simply determined by a single component, more complicated and end-to-end type data processing is required through the entirety of the project. For instance, 8.4m mirror segments for the Giant Magellan Telescope (GMT) [1] have the capability of active shape control using the bending modes of the mirror substrates. The figure of merit defining the final performance of GMT is not the measured surface RMS from the interferometer, but the structure function of the measured surface after those bending modes are subtracted. Thus, various types of data processing, considering the opto-mechanics, optical design, optical simulation, and fabrication processes need to be performed on the raw data.

As a research and development group focused on precision optical engineering, the Large Optics Fabrication and Testing (LOFT) group at the University of Arizona continues to develop many technologies for various optical engineering challenges. While the final deliverables are usually hardware (e.g. optical components), the significant portion of the new technologies and intellectual assets are the software; not customized software to control project-specific hardware, but general data processing algorithms and visualization tools that are repeatedly used for different projects.

The challenges in maintaining and improving these software assets are obvious. In many situations, these pieces of software and algorithms follow their own formats, assumptions and interfaces. As a result, many data analysis and visualization codes are not compatible or interchangeable with each other. New data processing codes are often re-written for a new project simply because of the lack of standards. In many cases, large portions of the new processing codes have overlapping functionalities with the previously written codes except for new data protocols. As an example, a code calculating the surface RMS of a measured optical surface map after removing the first 4 Zernike terms was developed by James. If Benjamin needs to calculate the structure function of a map after removing the same Zernike terms, many parts of the code written by James will be very useful, but only if they use same data conventions and protocols. Indeed, if the Zernike term removal part was written as a module with a pre-defined interface, Benjamin can take it and just work on the additional structure function part for his data processing.

* letter2dwk@hotmail.com

** SAGUARO is a non-profit free-ware, and can be downloaded from the LOFT group website: www.loft.optics.arizona.edu.

Optical Manufacturing and Testing IX, edited by James H. Burge, Oliver W. Fähnle, Ray Williamson,
Proc. of SPIE Vol. 8126, 81260B · © 2011 SPIE · CCC code: 0277-786X/11/$18 · doi: 10.1117/12.894908

For these reasons, the demand for a standard platform that provides a convenient environment and uses proscribed data types and data processing modules has increased in the optical engineering area.

To address this, SAGUARO (Software Analysis Graphical-user-interface from University of Arizona for Research in Optics), a MATLAB^{TM}-based data processing platform, has been developed at the University of Arizona. It provides a standard way to analyze and visualize various types of data in the optical engineering field. The SAGUARO banner image, a saguaro cactus in Tucson, Arizona, is shown in Fig. 1. The saguaro is a well-known symbol of southern Arizona, and only grows in the Sonoran Desert area [2].

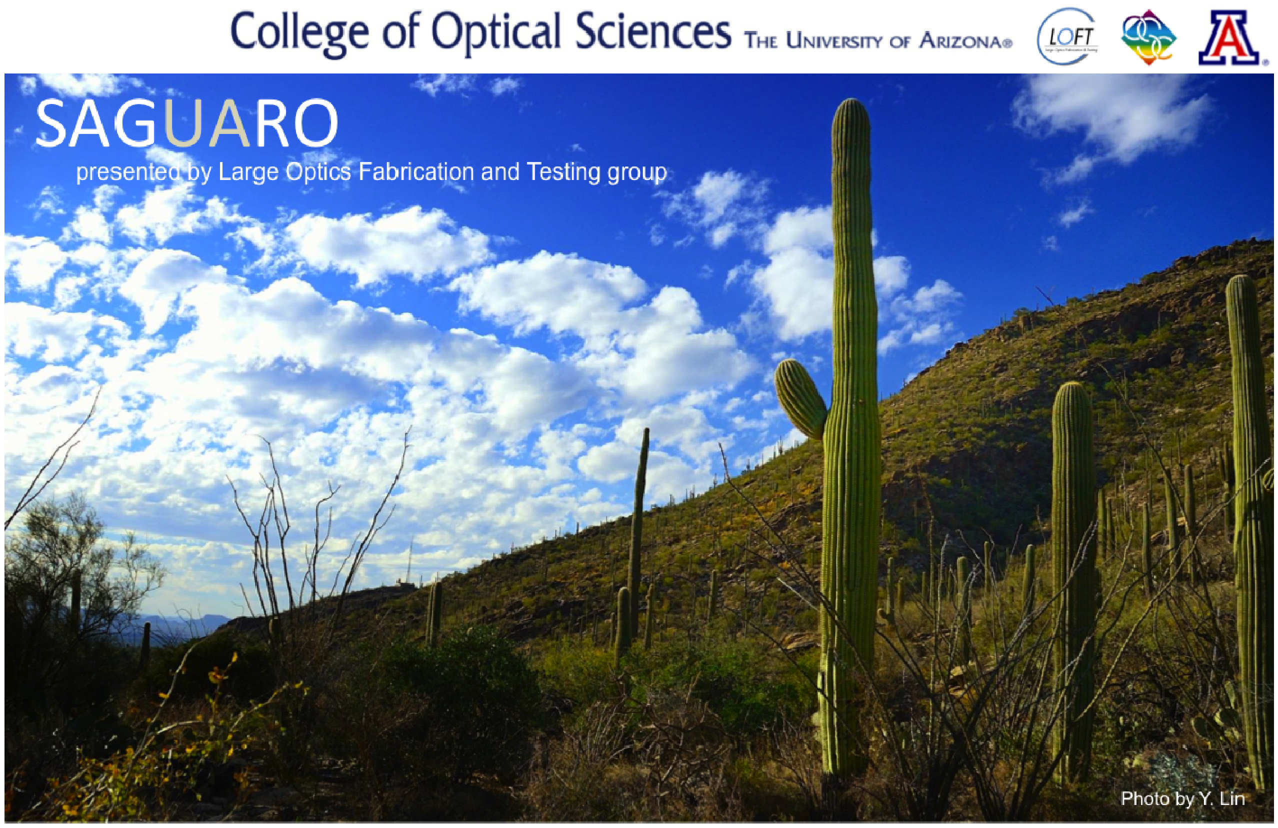

College of Optical Sciences THE UNIVERSITY OF ARIZONA®
LOFT
SAGUARO
presented by Large Optics Fabrication and Testing group
Photo by Y. Lin

Fig. 1. SAGUARO banner image showing Saguaro cactus in Tucson, Arizona. The Saguaro is a famous symbol of Arizona.

The schematic structure of SAGUARO is described in Section 2.1. Some standard data types are introduced in Section 2.2. Section 2.3 explains the module and macro features. The maintenance and acknowledgement plan for SAGUARO, based on MATLAB^{TM}, is described in Section 2.4. Three data processing examples using SAGUARO are presented in Section 3. Finally, Section 4 summarizes this discussion.

2. SAGUARO

2.1 Schematic structure of SAGUARO

SAGUARO is a data processing platform, using a Graphical User Interface (GUI) to provide an easy and efficient data manipulation environment. SAGUARO runs on top of the MATLAB^{TM} environment, provided by MathWorks, which is one of the most commonly used high-level computing languages. Although MATLAB^{TM} provides an interactive environment that enables users to perform computationally intensive tasks [3], it serves general purpose computing needs. This is not optimal for many optical engineering applications. Thus, SAGUARO can be regarded as a specialized platform on top of a general computing environment.

A schematic diagram of the overall platform structure is depicted in Fig. 2. SAGUARO runs on top of a MATLAB^{TM} session, color-coded in purple. Once SAGUARO is launched from MATLAB^{TM}, the main platform GUI in Fig. 3 shows up to provide users the data processing environment. Each users default setup is stored by the configuration feature. Users can manipulate the data stack using 'Load,' 'Save,' or 'Delete' buttons in the dataset panel in the left area of the main GUI.

On top of the multi-user configuration settings, modules and macros are mounted. The data processing in SAGUARO is achieved through running these modules or macros with the data. Each module is a piece of MATLAB^{TM} code, which follows the standard interface defined by SAGUARO. The standard interface is described in the developer's kit, which comes with the SAGUARO package. The macro feature enables pipelined modules for complex operations in a batch mode. Various modules are available either from the SAGUARO module library or from user's own module set. Available modules are listed in the 'Module' panel of the main GUI in Fig 3. More detailed explanations about the module and macro features are presented in Section 2.3.

Also, SAGUARO tracks customized parameter values for the modules and macros as a user-specific item. If some parameter values for a module are changed during a data processing, the values are automatically recorded and used in future SAGUARO sessions.

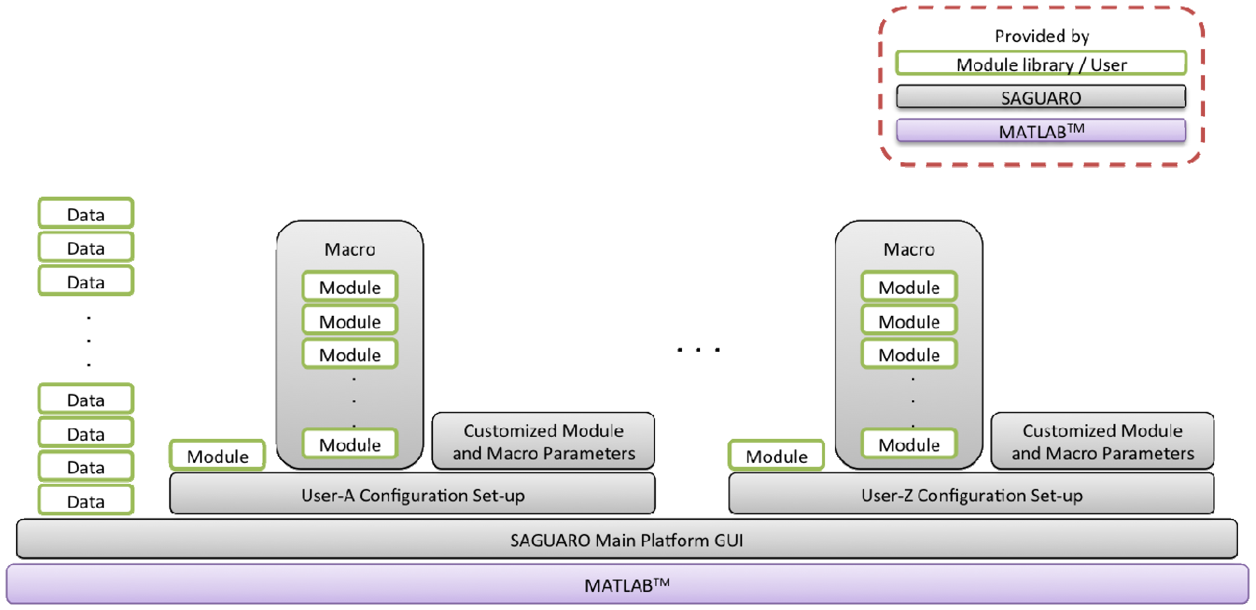
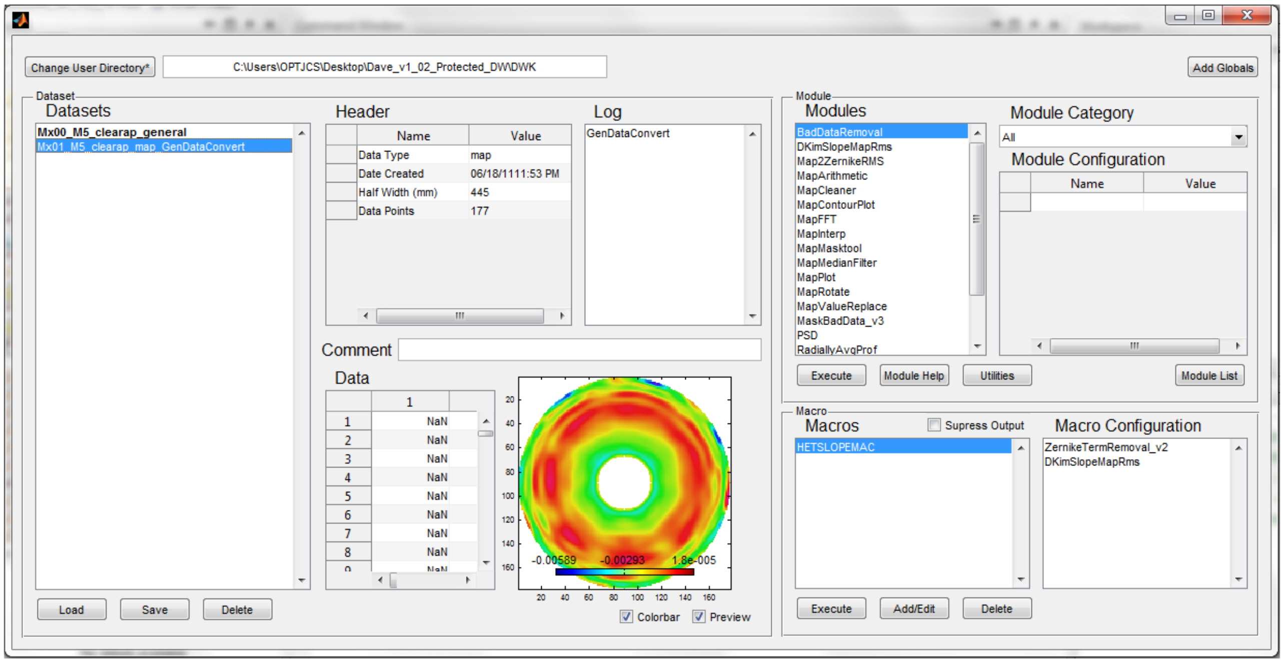

Provided by
Module library / User
SAGUARO
MATLAB™
Data
Macro
Module
Customized Module and Macro Parameters
User-A Configuration Set-up
User-Z Configuration Set-up
SAGUARO Main Platform GUI
MATLAB™

Fig. 2. Schematic SAGUARO structure showing the MATLAB^{TM}-based main platform GUI with configurable multi-user capability. Modules can be written by users, or are available from the existing module library. A Macro, a series of cascading modules, can be easily made using the macro feature in SAGUARO.

As an advanced option, SAGUARO allows a real-time interface with the MATLAB™ environment by using the 'Add Globals' button in the top-right corner of the main platform GUI in Fig. 3. Thus, the data in SAGUARO can be accessible in the MATLAB™ workspace for an unlimited data manipulation using MATLAB™ at any time.



C:\Users\OPTJCS\Desktop\Dave_v1_02_Protected_DW\DWK

Mx00_M5_clearap_general
Mx01_M5_clearap_map_GenDataConvert

Name	Value
Data Type	map
Date Created	06/18/1111:53 PM
Half Width (mm)	445
Data Points	177

GenDataConvert

	1
1	NaN
2	NaN
3	NaN
4	NaN
5	NaN
6	NaN
7	NaN
8	NaN

BadDataRemoval
DKimSlopeMapRms
Map2ZernikeRMS
MapArithmetic
MapCleaner
MapContourPlot
MapFFT
MapInterp
MapMasktool
MapMedianFilter
MapPlot
MapRotate
MapValueReplace
MaskBadData_v3
PSD
RadiallyAvgProf

Module Category: All

HETSLOPEMAC

ZernikeTermRemoval_v2
DKimSlopeMapRms

Fig. 3. SAGUARO main platform GUI with an example data set loaded. The user directory is shown in the top area, and the 'Dataset' panel is located in the left side. 'Module' and 'Macro' panels are located on the right side of the main GUI.

2.2 Standard data types

Standard data types for SAGUARO were defined with proscribed format including header information. While SAGUARO allows many types of data to be loaded, it uses standard data types for internal use. This ensures that every module can assume the standard data format and can be used in combination with other modules without worrying about file headers, units, etc.

Some standard data types with brief descriptions are listed in Table 1. A full list of all data types with templates and more information is available in the SAGUARO software package. The *Map* data type is used for data that exists as a matrix, and model a surface, such as an OPD map. The height units are defined to be mm. *Profile* data type is the 2D equivalent of *Map*. A *Profile* data might be a radially averaged profile, or other 2D data. *Profile* data include the length of the profile in mm in the header. *Standard Zernike Coefficient* data contain standard Zernike coefficients. The coefficients are defined to be RMS normalized. The header for *Standard Zernike Coefficient* data also includes the radius of the surface the data represents. *Mask* data is used for masking of *Map* data, for example to apply an annular pupil to the data. They share much of their definition with the *Map* data type. *Frequency Map* data is frequency space versions of *Map* data, used for Fourier analysis, etc. The maximum frequency of the data is specified in the header in units of mm^{-1}. *XYZ* data type can be used for a dataset with (x, y, z) values such as node information from a Finite Element Analysis (FEA) results.

Table 1. SAGUARO standard data types (Note: More standard data types will be added to support various needs in different optical engineering areas.)

Data types	File extension	Units	Descriptions
Map	.Map	mm	Data that exists as a matrix, and models a surface, such as an OPD map
Profile	.Profile	mm	2D equivalent of map datasets
Standard Zernike Coefficient	.ZernikeSTD	mm	Standard Zernike coefficients, and the m and n values of the terms
Annular Zernike Coefficient	.ZernikeAnnular	mm	Annular Zernike coefficients, and the m and n values of the terms
Mask	.Mask	N/A	Masking of map datasets
Frequency Map	.FreqMap	mm^{-1}	Frequency components of Map data (e.g. Fourier transform of a map)
XYZ	.XYZ	mm	Three columns of data containing x, y, and z values in each row
General	.General	Undefined	Undefined

There is also a "wildcard" data type named, *General*. This special data type was introduced to maximize the flexibility of the SAGUARO platform by allowing an undefined data type. This is a very useful concept for users who have one-of-a-kind data in the course of their data processing. For instance, an ASCII map file (e.g. surface.txt) generated by arbitrary interferometer software can be loaded to SAGUARO. The original map file may not follow the standard for *Map* data type. Thus, SAGUARO assigns *General* data type to the data, which is undefined. If no other standard data types are suitable to describe the data, it can remain as *General* type as long as needed. There are no restrictions about making modules work on *General* data types. It is envisioned that these modules will work on specific data is specific situations. In order to take advantage of many other modules in the library, user may want to convert the *General* data to one of the other data types (e.g. *Map*). Users can make his/her own data type conversion module, or simply use the default data conversion module, included in the module library.

These standard data types are the key to maintaining the compatibility between modules developed by independent developers all over the world. Additional data types will be defined as SAGUARO is used for more and more data processing situations in various optical engineering areas. Users may suggest new data types as official data types for SAGUARO by contacting the LOFT group. In order to guarantee the integrity of the new data types as a standard, they will be reviewed by the LOFT group before being announced to the public.

2.3 SAGUARO Module and Macro

The greatest power of SAGUARO is provided by its flexibility, and this flexibility mainly comes from the plug-and-execute module feature. The main platform of SAGUARO does not affect the actual data analysis or visualization processes used. It instead provides a convenient environment for various modules to be plugged in, and controls the data traffic between the modules. Numerous modules have been written and included in the module library; some of these are listed in Table 2. About 50 modules are included in the module library as of this writing in Aug 2011.

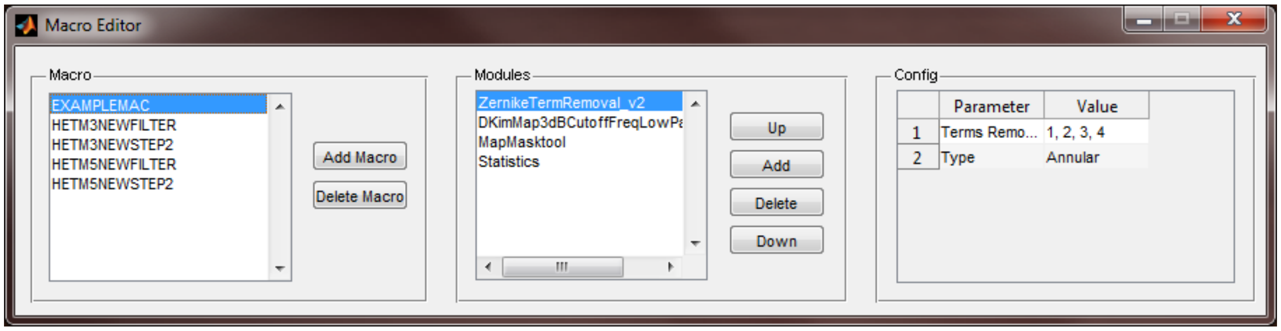
Table 2. Some SAGUARO modules in the module library (Acknowledgement: These modules have been developed by many module developers, and their names and contact information are available by pressing the ‘Module Help’ button in the SAGUARO main platform GUI.)

Module name	**Input data type**	**Output data type**	**Module descriptions**
MapCombine	*Map*	*Map*	Perform the arithmetic (-,+,*,/) combination of two map data.
Statistics	*Map* or *Profile*	N/A	Returns a set of statistics about a map or profile.
DKimSlopeMapRms	*Map*	N/A	Calculate and display the slope magnitude RMS of a surface map with the slope magnitude map.
DKimMap3dBCutoffFreqLowPassFilter	*Map*	*Map*	Perform the low-pass filtering using 3dB cut-off frequency.
MapFFT	*Map*	*Frequency Map*	Perform a 2D Fourier transform of a map.
FreqMapLogPlot	*Frequency Map*	N/A	Display the frequency map using log scale.
Zernike2Map	*Standard Zernike Coefficient*	*Map*	Converts a set of Zernike coefficients into a map.
ZernikeTermRemoval_v2	*Map*	*Map*	Removes standard (or annular) Zernike terms from a map.
XYZ2Map	*XYZ*	*Map*	Convert a non-uniform (i.e. scattered) x,y, and z data to a uniformly distributed map data.
Map2ZernikeRMS	*Map*	*Standard Zernike Coefficient* or *Annular Zernike Coefficient*	Computes a set of Zernike coefficients from a map.

All the actual data analysis or visualization processes are performed via running modules. If a user selects data from the ‘Dataset’ panel and a module (or macro) from the “Module” (or “Macro”) panel in Fig. 3, the SAGUARO main platform is ready to send the selected data to the module (or macro) as soon as the “Execute” button is pressed. SAGUARO does not limit any functionality of a module, and is only concerned with the output data from the module. In other words, every data analysis or visualization method or algorithms is available to the module developers.

A module is a user-defined MATLAB^{TM} function, which follows a proscribed format to communicate with the SAGUARO main platform. The module format is largely about the data handling protocol between the module and the main platform. In practice, this means there will be some extra lines of coding at the beginning and the end of the function code. Detailed descriptions and example modules are included in the SAGUARO package. Since this format was defined in such a way to minimize the efforts of module developers, it is also fairly easy to change an existing MATLAB^{TM} function into a SAGUARO module.

The Macro feature is provided for complex data processing cases, which can be accomplished by cascading modules in series. This avoids writing new modules for complicated data analysis processes. SAGUARO provides a Macro Editor GUI (Fig. 4) to generate macros using modules in the library. An example macro, EXAMPLEMAC, using 4 modules (ZernikeTermRemoval_v2, DKimMap3dBCutoffFreqLowPassFilter, MapMasktool, and Statistics) is shown in the Macro Editor in Fig. 4. The parameter values used in the modules can be initially assigned during macro generation.



EXAMPLEMAC
HETM3NEWFILTER
HETM3NEWSTEP2
HETM5NEWFILTER
HETM5NEWSTEP2

ZernikeTermRemoval_v2
DKimMap3dBCutoffFreqLowPa
MapMasktool
Statistics

	Parameter	Value
1	Terms Remo...	1, 2, 3, 4
2	Type	Annular

Fig. 4. The Macro Editor GUI in SAGUARO with an example macro EXAMPLEMAC selected. The modules in the macro are listed in the 'Modules' panel. The initial parameter values for the selected module ZernikeTermRemoval_v2 are shown in the 'Config' panel.

As a macro is used in SAGUARO, the latest parameter values are automatically tracked and stored for the next use. All modules may be included in macros, without additional work by the module developer, as long as the module follows the module format guidelines.

2.4 Maintenance and Acknowledgement Plan for SAGUARO based on MATLAB^{TM}

SAGUARO is a platform written in the MATLAB^{TM} language, and as such it requires a computer with MATLAB_R2009b or later version. Also, some modules may not work without some specific MATLAB Toolboxes (e.g. Image Processing Toolbox) if the module developer used a function in the Toolboxes. Thus, it is highly recommend that module developers specify the required Toolboxes in their module description section, so that the module can benefit more users in the society without confusion about any required Toolboxes.

The SAGUARO main platform code is released in the encrypted MATLAB^{TM} p-code to prevent unexpected sub-versions because such sub-versions may seriously harm the integrity of the standard data types and the compatibility between modules and macros. However, the module codes are fully opened to the public. This open-source policy for the modules is the key to making SAGUARO a mutually beneficial platform for everyone. As an example, many other researchers all over the world may garner benefits from a module developed by a researcher in University of Arizona. Since the code is open, it can be reviewed or checked before use. The module may be even improved by others, hopefully after proper contact with the original author. In this way, the intellectual assets of the whole optical engineering society can be accumulated and developed.

The whole concept of sharing modules is based on voluntarily actions. Of course, one may keep his/her own module, and not open it to the public. Thus, if someone chooses to share his/her module, he/she deserves credit. A module developer may or may not put his/her name and contact information in the module description section, and this information is displayed when the 'Module Help' button in the main platform GUI (Fig. 3) is pressed. Also, any use of SAGUARO and a module written by others to produce publications must be acknowledged. In this way, the module developer receives the appropriate credit for his/her public contribution. The technical discussions, suggestions, and module sharing, and announcements will be made in the SAGUARO Garden, which is a knowledge base for the SAGUARO users group across the globe. (The SAGUARO Garden will be accessible at the LOFT group official website: www.loft.optics.arizoan.edu [4].)

3. DATA ANALYSIS AND VISUALIZATION USING SAGUARO

Three data analysis and visualization example cases were assumed and demonstrated to show the flexibility and the capability of SAGUARO. These are only three cases out of unlimited number of possible data processing tasks. The maximum number of modules in these examples was limited up to 4 for a concise demonstration. In general, there is no limit in the number of modules that might be used to complete a given data processing task. The modules in these examples were written by a number of different authors, and this is a good example to show that various modules work together without causing compatibility issues.

The first example case, calculating the slope magnitude RMS (Root-Mean-Square) from a measured surface height map, is shown in Fig. 5. The first 11 Zernike terms were removed from the surface map (left) due to the measurement uncertainties in low order aberrations in this situation. Then, the slope magnitude map was generated from the residual surface map (middle) to calculate the slope magnitude RMS value. The final outcome from the data processing was the calculated slope magnitude RMS value, 59μrad, displayed with the slope magnitude map (right) on the computer screen. The overall process was performed using two modules, ZernikeTermRemoval_v2 and DKimSlopeMapRms, in Table 2.

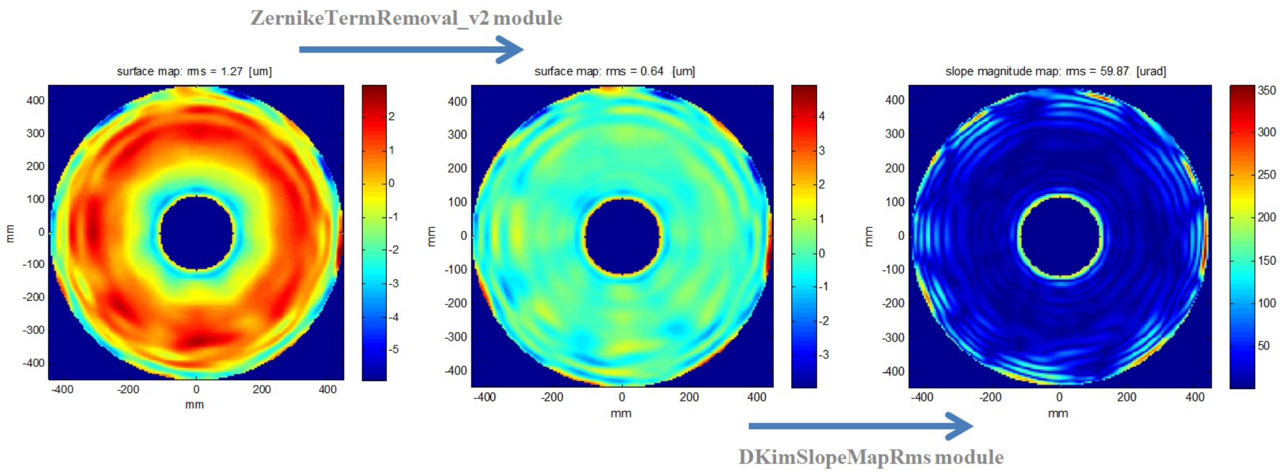
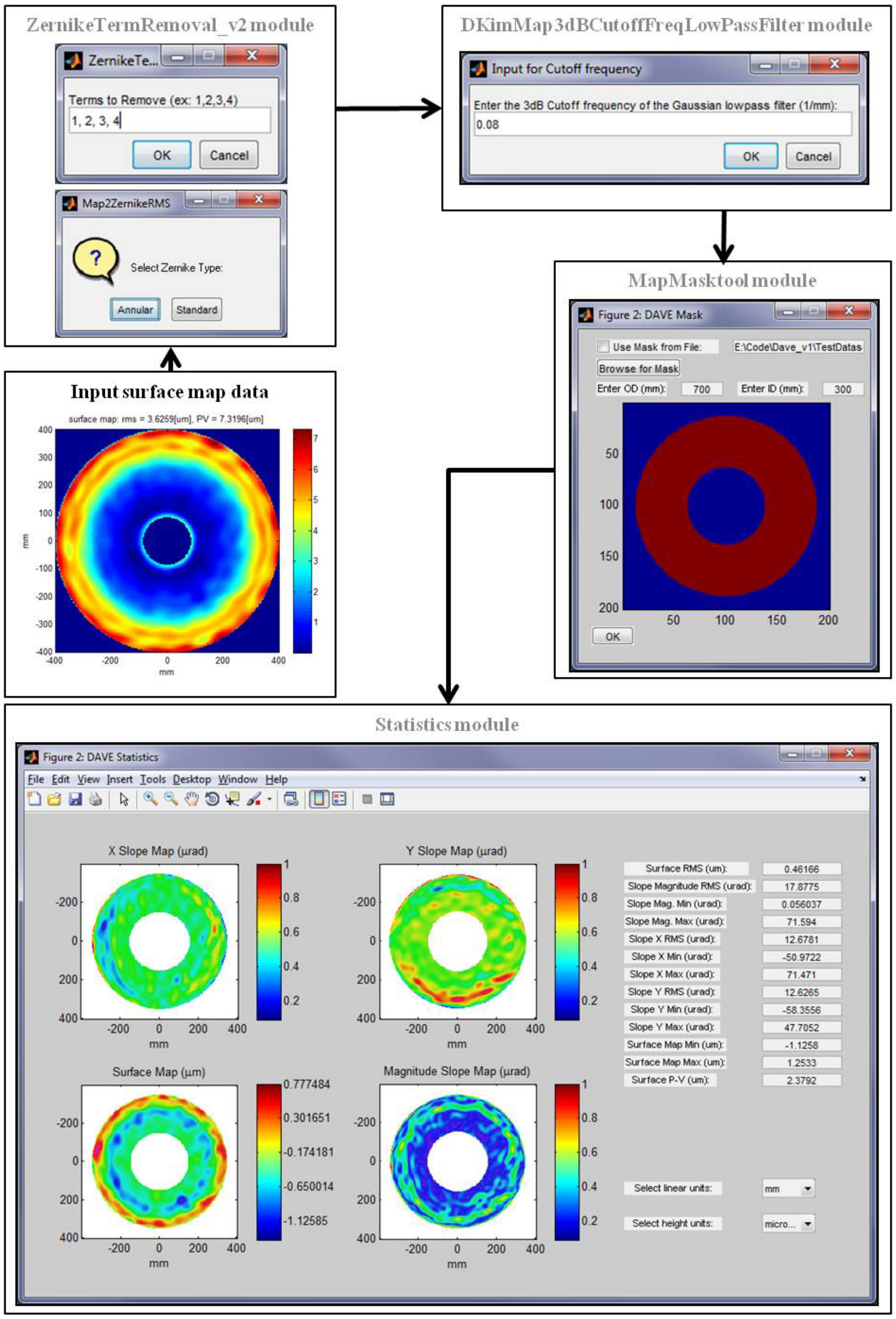


Fig. 5. Data processing example-1 using SAGUARO. The original surface map (left) was imported, and processed to remove the first 11 Zernike terms (middle). The slope magnitude RMS was calculated and displayed with the slope magnitude map (right).

The second example was calculating the surface RMS of an optical surface within the clear aperture (with an inside diameter 300mm and outside diameter of 700mm). Because the measurement was assumed to be blind to the piston, tip, tilt, and power measurement in this example case, the Zernike terms up to the 4th term were subtracted. Also, the surface map needs to be low-pass filtered with 3dB cutoff frequency of 0.08mm^{-1} as required in the specification. This data analysis process was achieved using the macro EXAMPLEMAC shown in Fig. 4, which cascades four modules, ZernikeTermRemoval_v2, DKimMap3dBCutoffFreqLowPassFilter, MapMasktool, and Statistics. The processed data in each step is presented in Fig. 6. The final surface RMS of the low-pass filtered surface map is shown as the final outcome (right).

The third example, shown in Fig. 7 was calculating the first 11 standard Zernike coefficients for a surface from FEA software. FEA analysis was done to determine the self-weight induced deflection of a flat mirror with three points axial support. The non-uniformly distributed node information from the FEA result was imported to the SAGUARO using the XYZ data type (left). The XYZ2Map module was used to convert the node information to a uniformly distributed *Map* data type (middle). Then, the Map2ZernikeRMS module was used to calculate the standard Zernike coefficients (right).


ZernikeTermRemoval_v2 module
ZernikeTe...
Map2ZernikeRMS
OK
Cancel
Annular
Standard
DKimMap3dBCutoffFreqLowPassFilter module
Input for Cutoff frequency
OK
Cancel
MapMasktool module
Figure 2: DAVE Mask
Browse for Mask
OK
Input surface map data
surface map: rms = 3.6259[um], PV = 7.3196[um]
mm
Statistics module
Figure 2: DAVE Statistics
File Edit View Insert Tools Desktop Window Help
X Slope Map (μrad)
Y Slope Map (μrad)
Surface Map (μm)
Magnitude Slope Map (μrad)
mm

Terms to Remove (ex: 1,2,3,4)
1, 2, 3, 4

Select Zernike Type:

Enter the 3dB Cutoff frequency of the Gaussian lowpass filter (1/mm):
0.08

Use Mask from File: E:\Code\Dave_v1\TestDatas

Enter OD (mm):	700	Enter ID (mm):	300

Surface RMS (um):	0.46166
Slope Magnitude RMS (urad):	17.8775
Slope Mag. Min (urad):	0.056037
Slope Mag. Max (urad):	71.594
Slope X RMS (urad):	12.6781
Slope X Min (urad):	-50.9722
Slope X Max (urad):	71.471
Slope Y RMS (urad):	12.6265
Slope Y Min (urad):	-58.3556
Slope Y Max (urad):	47.7052
Surface Map Min (um):	-1.1258
Surface Map Max (um):	1.2533
Surface P-V (um):	2.3792
Select linear units:	mm
Select height units:	micro...

Fig. 6. Data processing example-2 using SAGUARO. The input parameter windows were screen-captured as the macro EXAMPLEMAC was being executed. As the final outcome of the data processing, the statistics of the low-pass filtered surface map within the clear aperture are displayed (bottom). (Note: the arrow represents the data flow during the macro execution.)

The Z1 (piston), Z4 (power), and Z10 (Trefoil) shows significantly larger values than others as expected from the map shape. This example demonstrates a FEA result can be quickly fitted to the standard Zernike polynomials. Also, further data analysis or visualization can be easily performed by simply executing further relevant modules downstream.

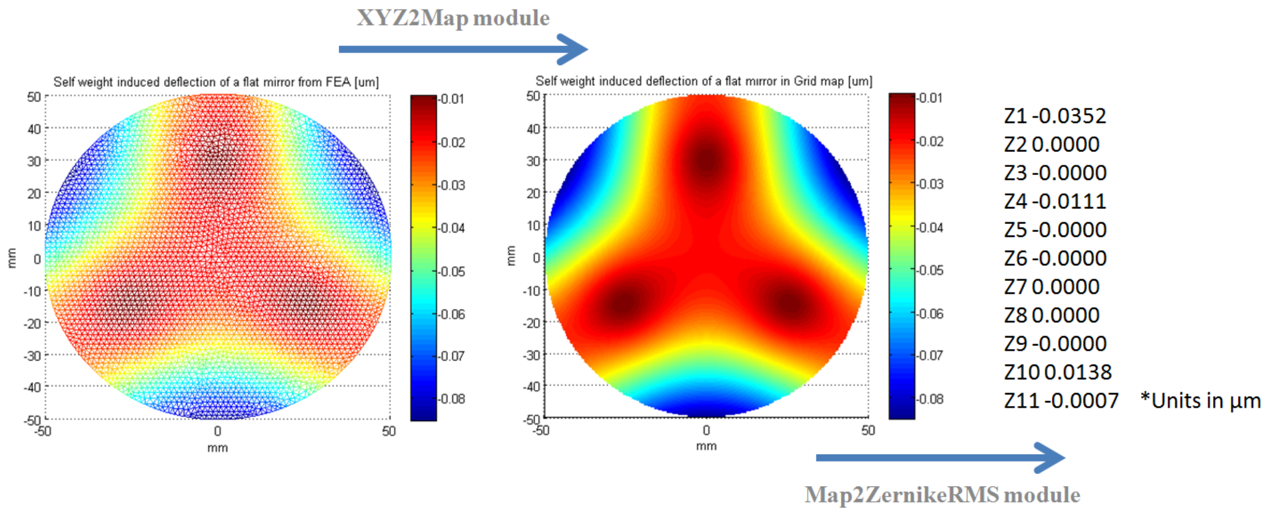


Fig. 7. Data processing example-3 using SAGUARO. Non-uniformly distributed node information from a FEA result (left) was converted to a uniformly distributed grid map (middle), and the standard Zernike coefficients (right) for the grid map were calculated.

4. CONCLUDING REMARKS

The LOFT group at the University of Arizona has developed a MATLAB^{TM}-based data analysis and visualization platform, SAGUARO, for the purpose of optical engineering research and development. This software is non-profit free-ware, and is available publicly through the LOFT group website: www.loft.optics.arizona.edu. A number of standard data types were defined and announced to ensure the compatibility between various modules. The flexibility of the platform due to the module-based structure and macro feature was well demonstrated using example cases. This powerful platform will provide a useful groundwork to accumulate the intellectual assets and data processing algorithms in the optical engineering field, and will provide mutual benefit among researchers and engineers around the globe.

The current version of SAGUARO is highly functional and useful as it is. We consider the public release as a beta test. We hope to get feedback on the functionality and performance before releasing the next full version. Also, there are additional features that are currently under development will be included in the next release, including the following:

- More complete programmers toolkit, including standardized protocol for testing and documenting modules
- Ability to create user buttons that run modules or macros
- Classification and naming conventions, to make it easier to select modules
- Centralized configuration control for the modules developed or adopted at the University of Arizona.

REFERENCES

[1] H. M. Martin, R. G. Allen, J. H. Burge, D. W. Kim, J. S. Kingsley, M. T. Tuell, S. C. West, C. Zhao and T. Zobrist, "Fabrication and testing of the first 8.4 m off-axis segment for the Giant Magellan Telescope," Proc. SPIE 7739, (2010)
[2] Saguaro national park service, "Where Saguaros grow", http://www.nps.gov/sagu/naturescience/location.htm.
[3] MathWorks, "MATLAB-The Language of Technical Computing," http://www.mathworks.com/products/MATLAB /?s_cid=global_nav
[4] Large Optics Fabrication and Testing group, http://www.loft.optics.arizona.edu