**NASA DEVELOP National Program**



NASA Langley Research Center

*Spring 2016*

LaRC CALIPSO Cross-Cutting III

Interacting with CALIPSO Data through a

Graphical User Interface

 **Technical Report**

Final Draft – Mar 25, 2016

Kathleen Moore (Project Lead)

Jordan Vaa

Kenton Ross, Ph.D., NASA Langley Research Center (Science Advisor)

Grant Mercer, University of Nevada, Las Vegas (Science Advisor)

Michael Bender, NASA Langley Research Center (Science Advisor)

Previous Contributors:

Grant Mercer

Nathan Qian

Ashna Aggarwal

Courtney Duquette

# I. Abstract

Atmospheric scientists analyze satellite data as part of their assessments of atmospheric health. One such satellite, the Cloud-Aerosol Lidar and Infrared Pathfinder Satellite Observation (CALIPSO), outputs images of longitudinal cross-sections of the Earth’s atmosphere. Depending on the wavelength of light used in the observation, it is possible to detect the presence of various aerosols. Some are natural sources, like dust storms, and sulfur dioxide from volcanos. Others are human-made, like sulfates from factories and smoke from burning biomass. It is important to track aerosols because an abundance of atmospheric aerosols contributes to accelerated cooling and warming of the Earth. The first instance of a visualizer for this data was written in an obscure, proprietary language, Interactive Data Language (IDL), making further modification of this tool virtually impossible. Since then, DEVELOP has produced new visualization software, *Visualization of CALIPSO* (VOCAL), written in Python.

As of the completion of the previous term, in addition to displaying CALIPSO images, the team added the ability for the user to “select” regions of interest by drawing shapes and assigning attributes to them. This information can subsequently be pushed to a backend database for the purposes of sharing and collaboration. However, the tool still needed to be enriched with other features such as better exception-handling and descriptors for annotations, and it needed to have cross-platform compatibility. Consequently, we have added these and other features to enrich the user experience, and we streamlined installation of the software on the Windows operating system. These updates have greatly improved VOCAL’s usability.

**Keywords**

Atmospheric Aerosols, CALIPSO, CALIOP, LiDAR, Python 2.7

# II. Introduction

The Earth’s atmosphere is a complex and dynamic system, with many factors contributing to and interacting with different ecological systems and processes. Some of the major factors in this system are clouds and aerosols, which consist of various particulates that permeate the layers of the atmosphere. Understanding how clouds and aerosols form, travel, and extinguish in the atmosphere helps us to understand the composition of the atmosphere with much greater clarity (Omar et al. 2009). Aerosols are created by a variety of sources – both anthropogenic and natural – and as they are located in the atmosphere, can travel across large geographic areas. As a result, understanding their life cycle is important to understanding how various environmental entities interact. There are many terrestrial and airborne means of studying aerosols, but these lack the range and the longevity to properly study the effects of aerosols on the environment. Given these limitations, researchers employed a space-based approach to alleviate both of these issues (Winker et al, 2009). Thus, a satellite program was chosen to monitor atmospheric health.

On April 20, 2006, the Cloud-Aerosol LiDAR Infrared Pathfinder Satellite Observation (CALIPSO) was launched as a follow-up to the shuttle based LiDAR In-Space Technology Experiment (LiTE) to serve as a more long-term platform for continuing this study, and it continues to collect data. Its purpose is to provide measurements of clouds, aerosols, and their interactions and roles in the climate system (Winker et al, 2010). CALIPSO carries three instruments as its payload. The primary active sensor is the Cloud-Aerosol LiDAR with Orthogonal Polarization (CALIOP), a new, nadir-viewing, two-wavelength polarization-sensitive LiDAR sensor (Winker et al, 2009). The remaining two are passive sensors: a wide field camera and an infrared imaging radiometer (Winker et al, 2009). CALIPSO can determine an aerosol’s type by measuring backscatter and volume depolarization ratio measurements (Omar et al. 2009). The algorithm takes altitude, integrated attenuated backscatter, location, surface type, and volume depolarization ratio as parameters. With the advent of CALIPSO output, researchers have access to more aerosol data and can better track aerosols through the atmosphere as well as determine crucial information about where aerosols travel.

The CALIPSO mission is pertinent across a wide variety of the NASA Applied Sciences National Applications Areas, most specifically, Health & Air Quality, Climate, Weather, and Disasters. Because of this multidisciplinary focus, it is considered a “Cross-Cutting” mission. A major challenge that researchers face when working with CALIPSO data is the data presentation style, which is different from many other commonly used datasets. CALIPSO collects vertical profile data of the atmosphere, rather than wide-swath data, which means that many common tools for visualization have difficulties displaying CALIPSO data. To overcome this obstacle, a proprietary visualization tool was developed specifically to view and manipulate CALIPSO data. However, this tool was written in IDL, an uncommon proprietary language, which makes it difficult for new developers to update and maintain. In the spring of 2015, a DEVELOP group created an updated visualization tool in Python to alleviate this difficulty. They integrated a Python-based tool known as the CALIPSO and CloudSat Plotter (CCplot) that takes CALIPSO data as an input and outputs two-dimensional plots. However, this tool had no user interface and had limited ability to manipulate the data once plotted. In two terms of DEVELOP, this tool has been updated to reflect the needs of researchers, becoming the Visualization of CALIPSO (VOCAL) tool. Continuing to update this tool to better fit the needs of the researchers, we have enabled the tool to be easily installed in Apple’s Mac OS X, and we have integrated web-based data acquisition in a client-server relationship. This will allow researchers greater flexibility when working with the tool as well as alleviate the need to download and store the large data-sets.

VOCAL is a graphical user interface (GUI) that enables a user to load a Hierarchical Data Format (HDF) file, which is the output of the CALIPSO satellite. The user can then visualize the data and interact with it by tagging target regions in the image in a “lasso”-style. Because the previous DEVELOP teams emphasized the object-oriented, organized, and expandable nature of the code, much of the infrastructure and functionality of the current VOCAL version have been rolled into the current term. Upon this code base, our team has improved the system in two major spaces: cross-platform functionality and efficiency. Here, we give a brief overview of how VOCAL works, followed by our specific additions to the project and their importance to the study of Earth observations as a whole.

**A. System Overview**

At the core of the VOCAL system is the interaction between three general components: shapes (and their manager), the database (and its manager), and the GUI. A user that wants to view CALIPSO data launches the VOCAL GUI and either loads in an HDF file or previously-created shapes from a JavaScript Object Notation (JSON) file. These shapes are each associated with a particular HDF file so that shapes cannot become “orphaned.”

Whether or not the user loads previous shapes, the user can draw on the image, creating a polygon. On completion of the drawing, VOCAL associates attributes with the shape, including the polygon’s coordinates and an assigned color for identification purposes. The user can draw an unlimited number of shapes. However, if these crowd the image space, the user can hide selected shapes from view while continuing to work in the same view. Because VOCAL also accommodates zooming in/out and panning, the user has multiple options for focusing on a desired region of the image. Lastly, shapes can be moved and erased. They are purely a means for denoting regions of interest on the image and enabling this data to persist after the current VOCAL session.

To this end, VOCAL has an integrated database for storing and loading shapes. Utilizing the SQLAlchemy database manager, VOCAL saves previous shapes, each in a JSON file, and this file is also associated with the HDF file from which it was originally taken. Because attributes such as the local image coordinates are saved, the user can immediately return to work on this particular shape or share it with another user of VOCAL. Herein lies one of the strongest features of VOCAL: the facilitation of collaboration.

The current features of VOCAL are in place largely to serve the purpose of unambiguous discussion of atmospheric aerosol data. Earth observations such as those produced by the CALIPSO satellite, are pinned down in location on a global scale, but this vantage potentially loses accuracy as well as intuitiveness when one starts to work on a subset of a cross-section of the atmosphere. Now, multiple scientists can be remotely located but certain that they are discussing the same region of data, stored on the same file. Our task, then, became a matter of how to increase the collaborative capabilities of the software and its data.

**B. Term Objectives**

There are several trajectories along which improvements could have been made to VOCAL: usability, portability, and functionality are one subset. As we narrowed our focus for this term, we identified the most important features to improve that would best orient this software for release. These features encompassed improving usability and error-logging, and enabling a first release to the end-user.

The priority for the term was to enable a smooth installation process on Windows. The software was originally built for the Windows operating system, so it would be easiest to focus on a stand-alone installer for Windows with an eye towards deployment. Hence, we sought to build an installation package that would be simple to use for anyone that wished to install the software.

One feature that should always be emphasized concurrently with deployment is error-logging, or, bug-tracking. As long as we expect users to push this software to its designed limits, we should expect unexpected user-actions. These actions may be completely reasonable within the context of the software’s intended usage but may have been overlooked by the development team. With this in mind, we knew that we needed to prioritize improving the error-logging of the software by ensuring that exceptions could be caught in logs and that logs with errors in them would persist after program termination.

At the start of this term, VOCAL was already very user-friendly. It had a thorough documentation website, and on the GUI itself, it incorporated useful “tool-tips” that would appear when a user hovered over most click-able components of the GUI. There were some subtleties remaining that would improve the experience for a user, especially for a first-time user. One of these would be to better annotate the names of shapes in the attribute pane. For each drawn shape, its name would be listed in a drop-down menu at the top, but it was unclear to which shape each name belonged on the actual drawing pane. It was also unclear for which HDF files shapes could be loaded. We wanted to tightly associate each set of shapes to the original HDF file on which they were drawn.

# III. Methodology

Through close collaboration with one of our science advising consultants and previous CALIPSO Cross-Cutting team lead at DEVELOP, Grant Mercer, at the University of Nevada, Las Vegas, we were able to add increased functionality to VOCAL. We frequently communicated over e-mail, voice- and video-conferencing, and the Slack web-based messaging client that is geared towards software development teams.

When a software development team is independently working on one code-base in a distributed manner, version control is critical to incorporating changes from all coders. The use of GitHub as our code repository and version control system was critical to our efficiently addressing bugs and implementing these new features.

# IV. Results & Discussion

This is currently the third generation of this project. Since the first term, the CALIPSO Cross-Cutting teams have built and continued to add functionality to the visualizer. It has become an increasingly useful program through which scientists can interact with satellite image data. Here, we discuss the contributions that we have made to the project during this term.

**A. Stand-Alone Windows Installer**

Before the current term, we had an involved, multi-step process for installing VOCAL on its primary operating system, Windows 7. Because it is limited by the compatibility of the CCPlot library, the installation on Windows is 32-bit. To facilitate installation and usability of the software, we created a VOCAL installation executable (EXE) file that would streamline the installation process, as depicted in Fig. 1.

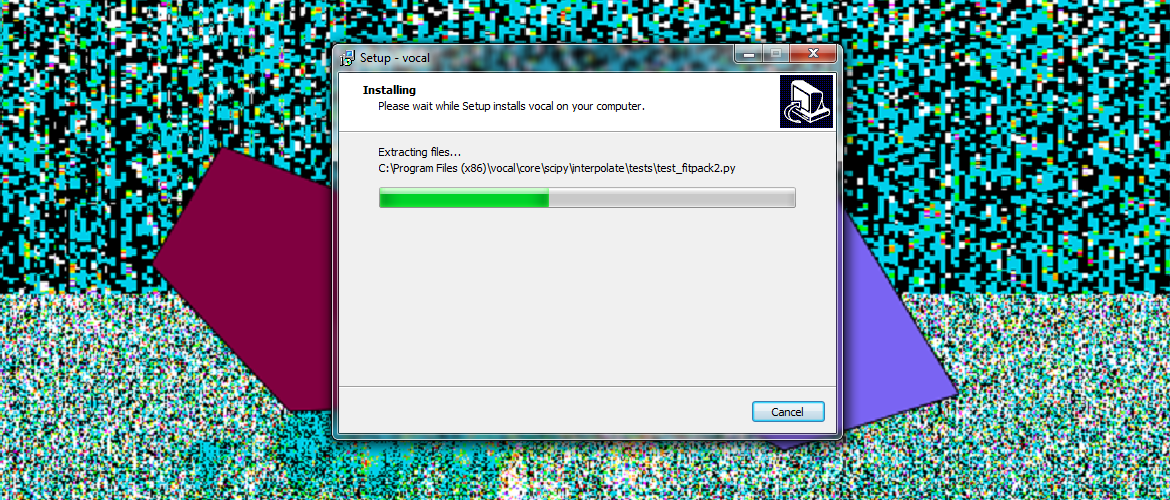


Figure 1. Screenshot of the stand-alone Windows installer.

The installation package includes dependencies that the system may not already have, and it installs data files, such as the database, into the following system folder:

C:\Users\[username]\AppData\Local\Roaming\

The installation package also includes an uninstallation script that will remove runtime files and data files associated with VOCAL, should the user wish to do this. We encountered a challenge writing the installer because the original procedure did not account for the user not having administrative privileges on his or her own system, as is often the case for a user on a NASA machine. We had to write a work-around so that the only time the installer requested administrative rights was initially, and not every time the user tried to run VOCAL. Ultimately, we built an easy-to-use installer for Windows.

**B. Shape Attributes**

The user-experience is critical to this software because it is interactive. Through the GUI, the user visually analyzes the loaded HDF file for important regions. It is these regions that the user will then want to annotate for future reference. Before this term, the attribute window listed several attributes pertinent to the shape, including the scale of the HDF file at which it was drawn, its color (just to distinguish it from other drawn shapes), and any other attributes added to the shape through the attributes pane of the program. However, there was no way to distinguish shapes apart in the drop-down menu of the program. We associated a name to each shape in the attribute window that would also appear in the drop-down list of all shapes. See Fig. 2.

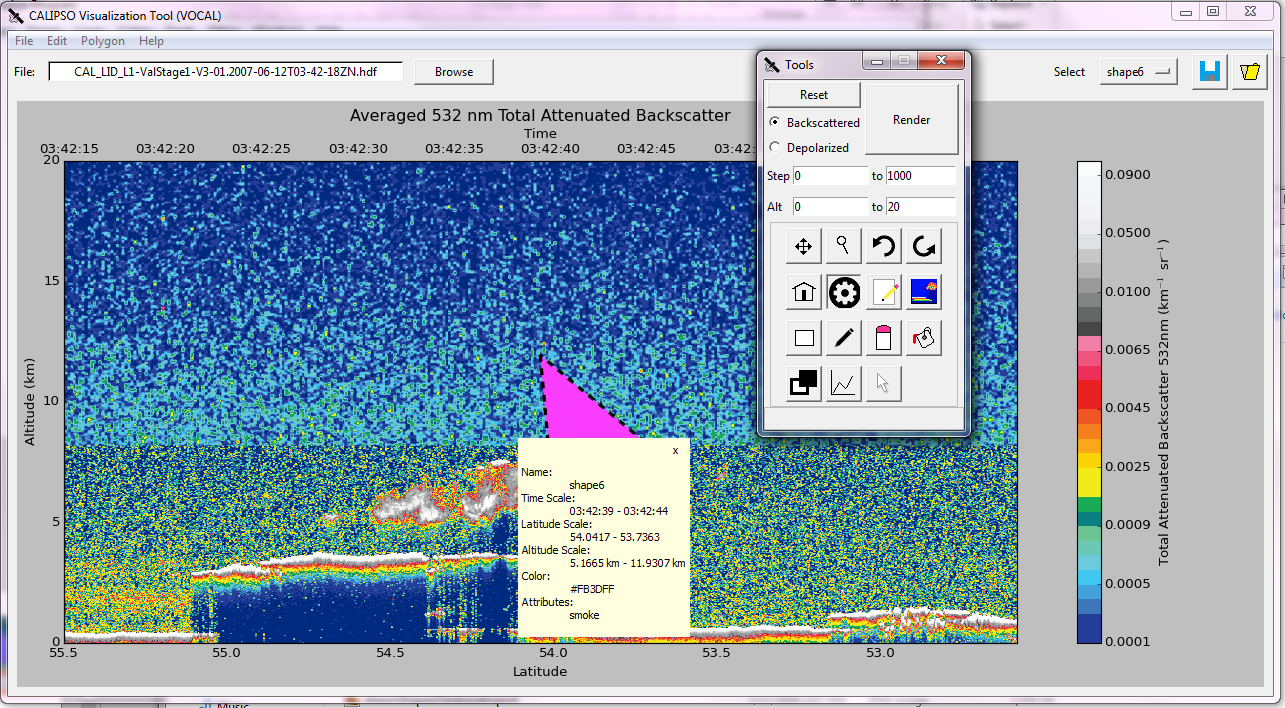


Figure 2. Screenshot of VOCAL depicting the attributes pane with the shape name (lower-left) and the shape name in the drop-down menu of shapes (upper-right.)

Secondly, one problem with VOCAL had been that the program would fail “silently” if the user tried to load shapes onto an HDF file on which they had not initially been drawn. A user, especially a novice user, would not know why, while he or she would insist on trying to load a set of shapes, they would not display. To ameliorate this, we built in a more rigorous check on the currently displayed HDF file and target shapes. If the HDF file associated to each of these entities did not match, a useful error would be shown to the user, as shown in Fig. 3.

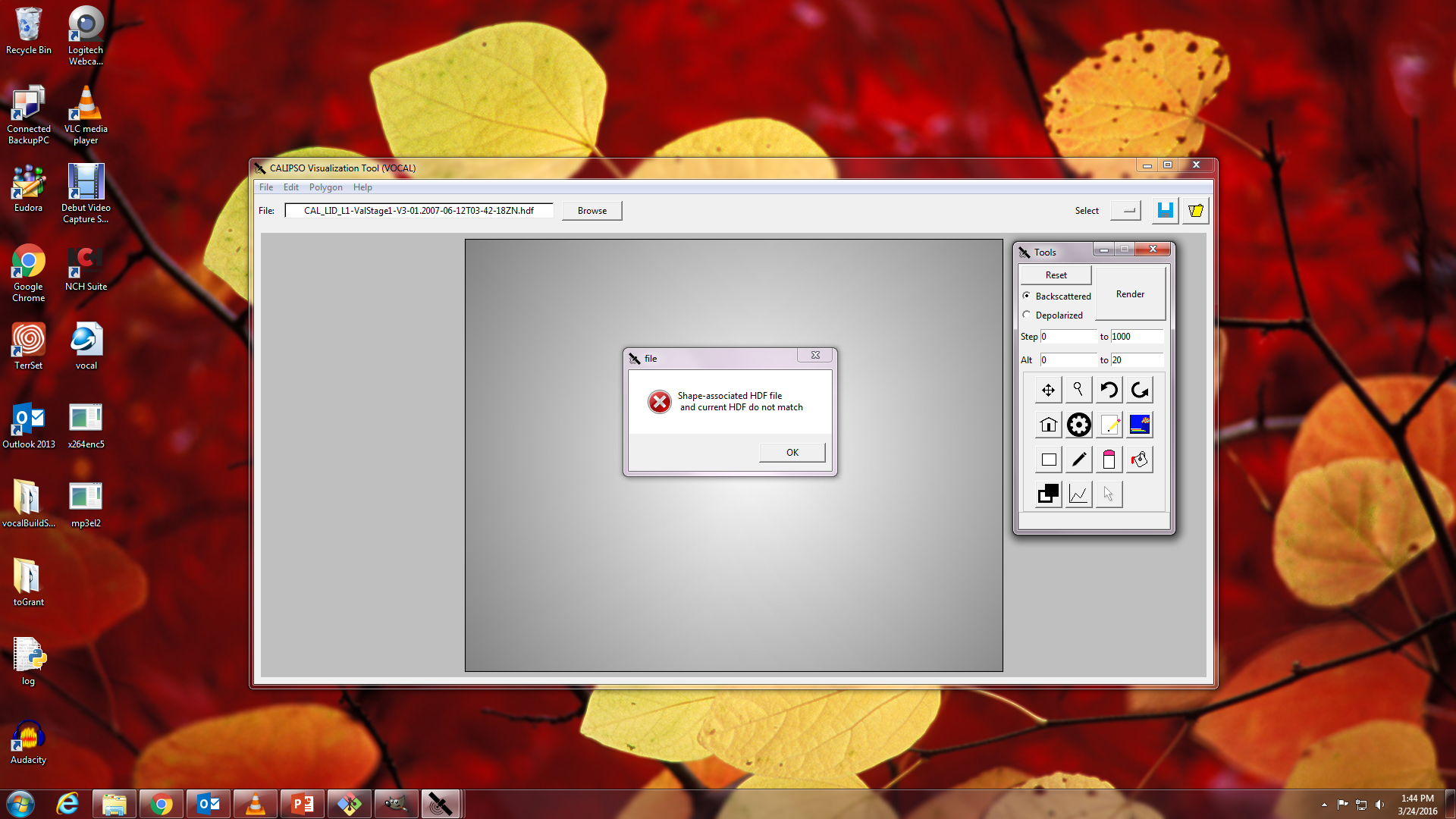


Figure 3. Trying to load shapes to an inappropriate (in this case, empty) HDF file.

**C. Error-logging**

In addition to the front-end features added this term to improve the user-experience, the team also overhauled the error-logging system in VOCAL. Prior to this term, the error-logging system was used to create a log of events throughout the run-time of the program, which the developers specifically tagged to be caught. However, this version of the log was erased and rewritten each time the program ran. This allowed the developers to check that the program was running correctly, but provided little ability to effectively and accurately track errors that arose during the run of a release-version of the software, and it also ran the risk of being accidently over-written if the program was re-opened after an error was encountered.

In order to improve on this system, the current team implemented a feature that caused the log to write an additional, “error-log” if an error was raised on the logging system, as well as capture a stack-trace should such an exception be raised by the program. In addition, unexpected exceptions were not being caught in the log at all. We incorporated a way to capture these unanticipated exceptions in the main log file and error-logs. Both of these contributions are shown in Fig 4. The new system now maintains a separate log, per-error, should an error occur, and each log should provide enough information so as to allow developers to trace the source of the error.

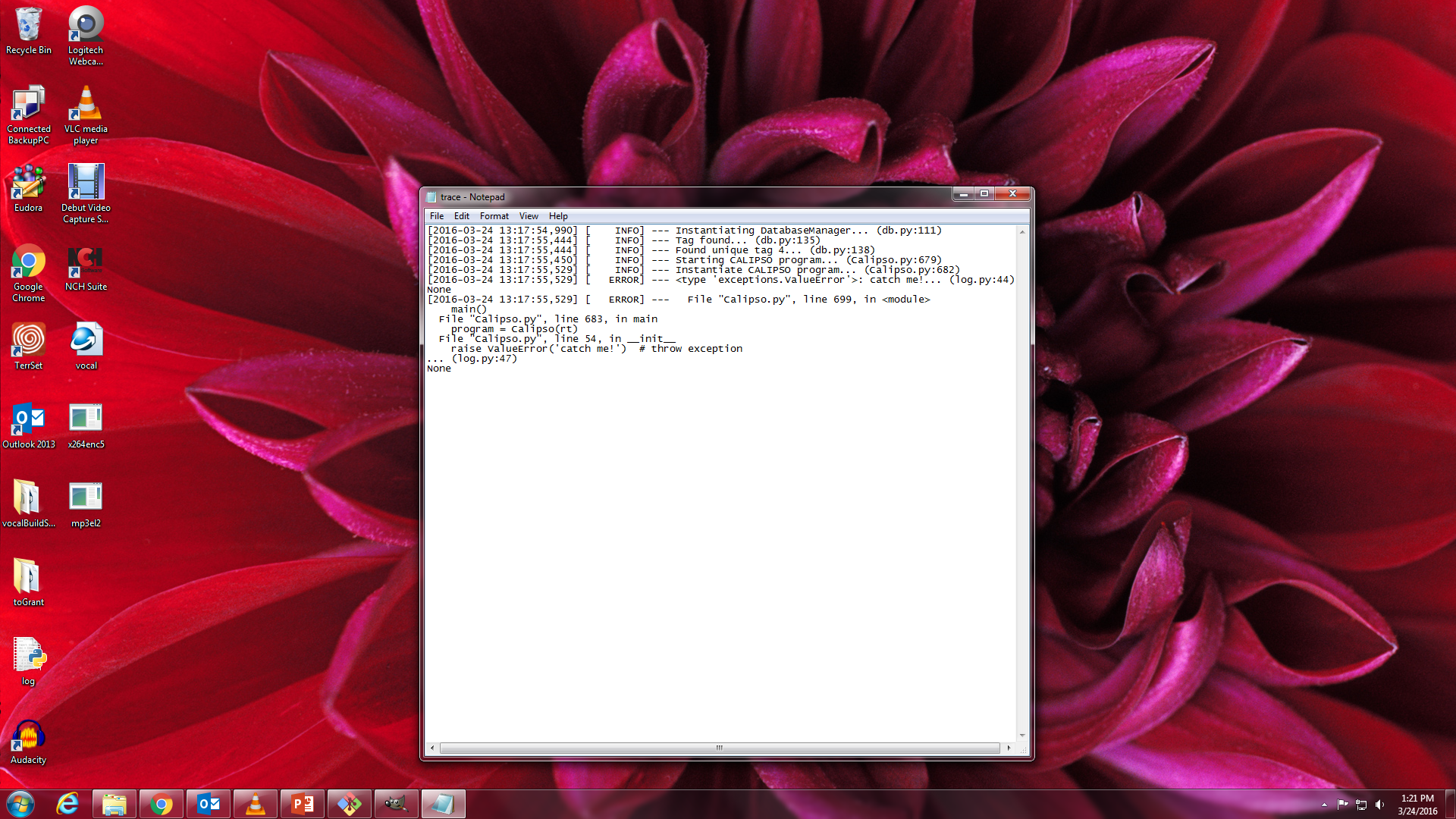
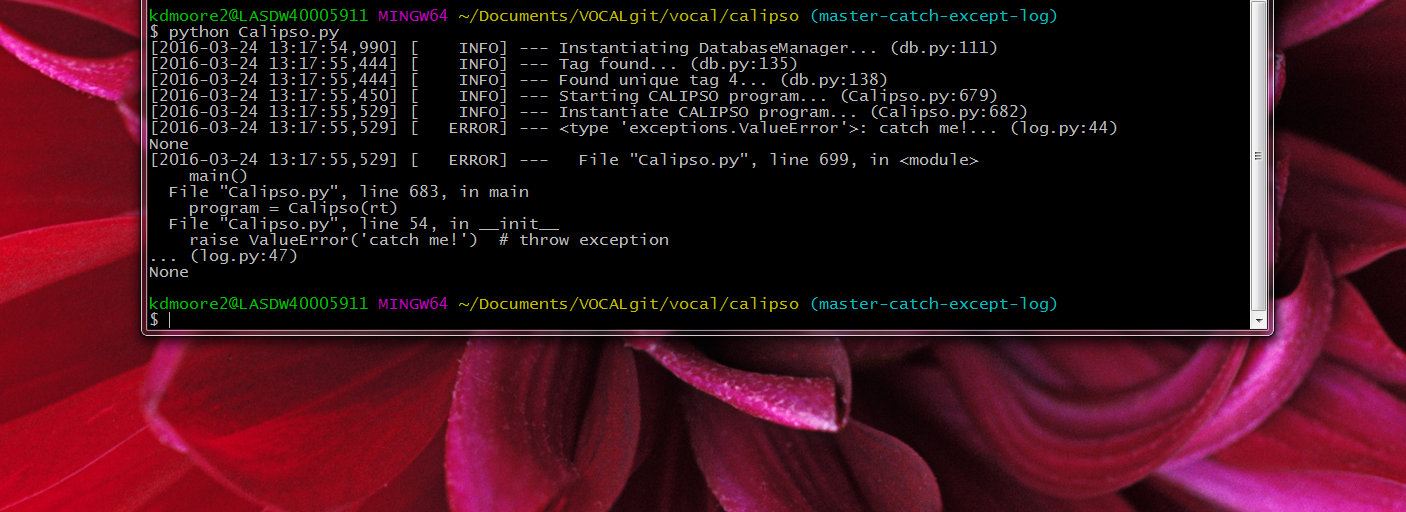


Figure 4. The exception thrown from the program is not only displayed to the command-line interface, it is pushed to a separate log-file.

Now that the first Windows release of the program has been deployed to the end-user, we await feedback from our partner, Dr. Charles Trepte, and the rest of the CALIPSO science team as they have a chance to try out the software. Users can submit feedback through the VOCAL documentation website (See the Appendix), and these issues can then be formalized as tasks in the code repository. Thus, the software lifecycle will persist for VOCAL, as shown in Fig. 5:

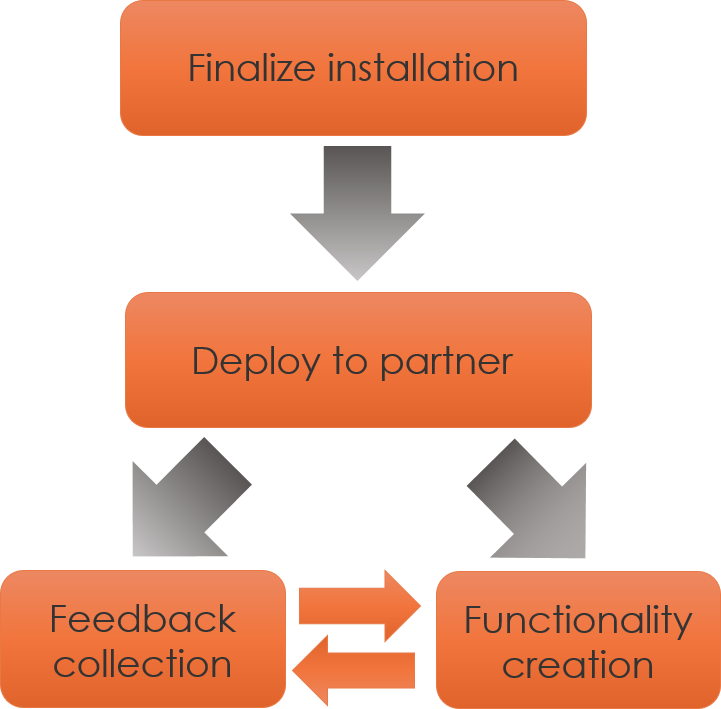


Figure 5. Diagram of the software lifecycle as it relates to VOCAL. The first two, sequential stages are now complete, and we are currently in the feedback loop stage.

# V. Future Work

There are many exciting avenues down which the project can proceed in future terms. They revolve around extending VOCAL to usage on multiple platforms as well as making both the input and output data more flexible.

**A. Cross-platform Compatibility**

The primary users of the VOCAL system are expected to be the CALIPSO science team, a selection of whom work on Apple computers running Mac OS X. As we have built a stand-alone Windows installer, we will work to integrate Mac functionality into the VOCAL installation by creating a separate installer for Mac. Similarly to the Windows installation, the major obstacle will again be ensuring backwards-compatibility to accommodate CCPlot and its associated dependencies. As the methods and means of installation on a Mac differ from those on a Windows system, the process will have to be fine-tuned and tested for operability.

**B. OPeNDAP Integration**

One of the bottlenecks in using VOCAL is the need for locally-stored HDF files, which can oftentimes become very large, very quickly. This both hinders storage capacity by taking up a large amount of memory, as well as creates lengthy download speeds for gathering data. Future collaboration with the Atmospheric Science Data Center (ASDC), the organization responsible for collecting and storing CALIPSO data, will hopefully lead to the utilization of the Open-source Project for a Network Data Access Protocol (OPeNDAP). OPeNDAP is a protocol for accessing scientific data over the Internet. OPeNDAP integration will allow researchers using VOCAL the option of selecting only the portion of the dataset in which they are interested, directly from the VOCAL interface, rather than having to download the data ahead of time and then load it into the program in its entirety. This will reduce the storage and download requirements on the researcher's part, as well as facilitate easier sharing of findings and data among researchers.

**C. Database Flexibility**

Our partner has expressed interest in making the database, which is currently an amalgamation of attributes of JSON shape files, more extendable and flexible. When attributes of new shapes are exported, this data is concatenated onto the existing database. If the user is simply interested in maintaining a centralized list of shapes from which to load JSON files, this strategy suffices. However, the CALIPSO science team is interested in maintaining separate databases with arbitrary areas of focus. For example, one database may be dedicated to regions of clouds across many HDF files while another is devoted to all types of aerosol regions for a particular HDF file only. This would be a valuable term-project for the next generation of CALIPSO Cross-Cutting.

# VI. Conclusions

We have improved upon VOCAL (Visualizer of CALIPSO), software for visualizing and interacting with satellite image data from the CALIPSO satellite. It enables scientists to identify regions in the image and tag them by drawing polygons over the regions and assigning attributes. The crux of the software is facilitating collaboration among scientists because these target regions can be exported, imported, saved to databases, and archived for distributed and future use.

This term, we focused on a select group of improvements, including increasing the usability of the software. On the front-end, this meant incorporating more sanity-checking into interaction with shapes. We made the names of shapes more explicit in their associated attribute panes. We also required that the proper HDF file on which the shapes were originally drawn be loaded into the program before the system attempts to load any shapes from a JSON or database file.

Secondly, we improved the error-logging capability of the software. Until this term, exceptions were not caught in the log, and the error-log consisted of a single, legacy file that persisted until the termination of the program. If an error was encountered, it would be logged in this way, but there would be no persistent record of it so that it could be addressed later, if the program were terminated. We added the automatic generation of separate error-log files for encountered errors that include trace-backs for exceptions, if they occurred.

Lastly, we packaged VOCAL into an easy-to-use, stand-alone installer for Windows that only requires one click from a user with administrative rights to the machine. This and the previous contributions increase the legitimacy of this software and encourage heavy use of VOCAL. The more people that try VOCAL, the more feedback we hope to receive, and the better we can make this software in upcoming terms.

# VII. Acknowledgments

CALIPSO Health and Air Quality, LaRC, Spring 2015

* Jordan Vaa (Team Lead)
* Ashna Aggarwal
* Courtney Duquette

CALIPSO Cross-Cutting II, LaRC, Summer 2015

* Grant Mercer (Team Lead)
* Nathan Qian

Dr. Charles Trepte (Project Partner), CALIPSO Science Team

NASA DEVELOP National Program

* Dr. Kenton Ross (Science Advisor)
* Michael Bender

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Aeronautics and Space Administration.

This material is based upon work supported by NASA through contract NNL11AA00B and cooperative agreement NNX14AB60A.

# VIII. References

Omar, Ali H., David M. Winker, Mark A. Vaughan, Yongxiang Hu, Charles R. Trepte, Richard A. Ferrare, Kam-Pui Lee, Chris A. Hostetler, Chieko Kittaka, Raymond R. Rogers, Ralph E. Kuehn, and Zhaoyan Liu. "The CALIPSO Automated Aerosol Classification and Lidar Ratio Selection Algorithm." *Journal of Atmospheric and Oceanic Technology J. Atmos. Oceanic Technol.* 26.10 (2009): 1994-2014. *Web of Science*. Web. 17 June 2015.

Winker, David M., Mark A. Vaughan, Ali Omar, Yongxiang Hu, Kathleen A. Powell, Zhaoyan Liu, William H. Hunt, and Stuart A. Young. "Overview of the CALIPSO Mission and CALIOP Data Processing Algorithms.” *Journal of Atmospheric and Oceanic Technology J. Atmos. Oceanic Technol.* 26.11 (2009): 2310-323. *Web of Science*. Web. 17 June 2015.

# IV. Content Innovation

VPS: 2016Spring\_LaRC\_CALIPSOCross-CuttingIII\_VPS\_FD.m4v

# X. Appendices

**Installation and User Guide**

For instructions on installing VOCAL’s required dependencies and how to use VOCAL properly, visit the documentations website at:

<http://syntaf.github.io/vocal/>