**NASA DEVELOP National Program**



NASA Langley Research Center

*Summer 2015*

CALIPSO Cross-Cutting

Interfacing CALISPO Data through a Graphical User Interface

 **Technical Report**

Final Draft – August 6, 2015

Grant Mercer (Project Lead)

Nathan Qian

Jeff Ely, NASA DEVELOP National Program (Science Advisor)

Dr. Kenton Ross, NASA DEVELOP National Program (Science Advisor)

Previous Contributors:

Jordan Vaa

Courtney Duquette

Ashna Aggarwal

# I. Abstract

The Cloud-Aerosol Lidar and Infrared Pathfinder Satellite Observation (CALIPSO) satellite is a NASA Earth observation that analyzes aerosol particles suspended in the Earth’s atmosphere. Researchers use visualized CALIPSO data to track the global distribution, dispersion, and source of aerosols. However, the current visualization tool for displaying CALIPSO data does not support needed features for tracking aerosols such as selecting areas of data and sharing those selected sections, making tracking specific airborne objects difficult for researchers. Adding these necessary features to the current CALIPSO visualization tool is difficult, as the tool is written in Interactive Data Language (IDL), a proprietary and obscure language, and writing additional features for the tool would require a specialized development team. For the 2015 summer term, our team is focused on the development of the *Visualization of CALIPSO* (VOCAL) Python program. VOCAL will serve as the successor to the current visualization tool for CALIPSO data. We will be building off the previous 2015 spring team’s work to implement a number of new features and offer new functionalities for Earth scientists to more easily identify the sources of aerosols and their impact on Earth’s climate.

**Keywords**

CALIPSO, CALIOP, LiDAR, Atmospheric Aerosols

# II. Introduction

**Background Information:**

Earth’s climate is a complex system that involves many variables and factors. One such factor is aerosols. Aerosols can come from anthropogenic (pollution or biomass burning) or natural (dust, sea salt, and volcanic) sources (Omar et al. 2009). Once in the atmosphere, aerosols can affect cloud formation and cloud radiative properties. Additionally, aerosols can suppress and enhance precipitation, simultaneously affecting the water cycle as well as radiative heating (Winker et al, 2010). Besides environmental impacts, aerosols can negatively affect human health in heavily populated regions (Winker et al, 2003). Unlike greenhouse gasses, aerosols are a major source of uncertainty in climate models due to their variable distribution and properties (Winker et al, 2003). Because of these effects, Earth scientists track and identify aerosols to analyze the aerosols’ impact on human health and Earth’s climate.

In April 2006, the Cloud-Aerosol LiDAR Infrared Pathfinder Satellite Observation (CALIPSO) was launched to provide measurements of clouds and aerosols and their interactions and roles in the climate system (Winker et al, 2010). CALIPSO carries three instruments for aerosol measurement. The primary instrument 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 the aerosol’s type through an algorithm by utilizing 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 (Omar et al 2009). With the addition of CALIPSO, researchers have access to more aerosol data and can better analyze aerosols’ effect on climate.

Prior to the spring of 2015, the CALIPSO science team used an Interactive Data Language (IDL) program written in 2007 to display CALIPSO data. Since 2007, the program has been maintained and updated with improvements. However, since IDL is a proprietary language, users outside of NASA cannot easily use nor customize the CALIPSO visualization tool. In 2010, a Python program, called CloudSat and CALIPSO plotting tool (ccplot), was written to mimic the IDL program to display CALIPSO data and was used by the CALIPSO science team. Unlike IDL, Python is an open source language, which allows users to more easily use and change programs written in Python. Despite the conversion to Python, ccplot proved to be obtrusive as CALIPSO data is displayed as an unadaptable image and can only be manipulated through a command line. In the spring of 2015, a team at NASA DEVELOP started development of a new CALIPSO visualization tool called Visualization of CALIPSO (VOCAL) that will address the issues of the previous CALIPSO visualization programs. The spring term provided the back end implementations that displayed the initial GUI and the three different CALIPSO plot types.

**Project Objectives:**

For the summer term, our team implemented a number of new features and started the process of releasing VOCAL. The new features added allow users to draw shapes that mask specific aerosols, and share these shapes through a database developed for VOCAL. Specifically, a user can trace a shape or multiple shapes in the backscattered or depolarized plots from CALIPSO data that represents a specific aerosol. The user can then tag the shapes with relevant data, such type of aerosol, the aerosol’s source, or the aerosol’s composition. When the user exports the shapes to the database, the shapes are saved in a JavaScript Object Notation (JSON) file with additional metadata, such as the time of creation and the corresponding CALIPSO data product file. When the shapes are saved in the database, the user can retrieve them using queries based on the CALIPSO data product or user-generated tags. The code base left over from the spring term already displays level one and level two CALIPSO data products. However, our team made additional improvements, such as bug fixes and code documentation, to make VOCAL more reliable and user friendly. With these additions, researchers can better collaborate on aerosol tracking and identification.

**Study Area:**

The scope of this project is global, as VOCAL can read any CALIPSO curtain plot data.

**Study Period:**

The timeframe of this project ranges from 2006, when CALIPSO was launched, to the present day.

**National Application:**

Our end-product addressed the Health and Air Quality application area. VOCAL will aid researchers in identifying, tracking, and documenting aerosols and their effect on the environment and air quality.

**Project Partners:**

Our end-user is the CALIPSO science team. Our main point of contact with our users were Dr. Charles Trepte and Dr. Amber Soja, who provided guidelines on the VOCAL features and use cases. The shape drawing and database functionality of VOCAL tool will help the CALIPSO science team to better track and identify aerosols and let them more easily share aerosol research with each other.

# III. Methodology

When the user launches VOCAL, the program displays a blank screen, awaiting further user instruction. The user can either import CALIPSO HDF files to create new shape objects, or import preexisting shape objects from the database to examine or edit. If the user loads CALIPSO HDF data, the user is prompted by a file dialog that explores the local file directory to retrieve the requested HDF file. After the selection, the user can select which plot to display (backscattered or depolarization ratio). If user selected the incorrect plot or needs to switch, the user can easily select another plot to display. The plot is then shown, and the user can draw shapes specific features of the plot or load shapes from a previous session. Whenever the user moves the cursor, the program displays and updates the coordinates to help with precision plotting. After a shape is drawn, it can be clicked and dragged to relocate it within the plot. If the user makes a mistake with the shape or no longer needs it, the shape can be erased. If the screen becomes too crowded with shapes, the user can temporarily hide the shapes to help focus on the plot. Besides the drawing functionality, the user has access to some navigation features. The user can zoom in on specific areas of the plot and can pan the plot to center a specific subsection of the plot. Lastly, the user can undo and redo navigation operations. If the user wants to clear navigation history and all drawings on the plot, the user can easily reset the plot with the press of a button. When the user is finished drawing, the user can save the shapes, which is stored locally as a JavaScript Object Notation (JSON) file or export them directly to the database. While the shapes are being saved, metadata such as the HDF file name and time of creation is included in the JSON file.

When the user loads shapes from the database, the user is prompted by a dialog displaying all the shapes in the database. The user can filter the shapes using queries based on shape attributes or user-generated tags. The user also has the option to conduct an advanced search to filter shapes based on the HDF file’s date, file name, time range, latitude range or altitude range. After selecting which shapes to use, the user can then download the shapes for personal use with access to the same functionality mentioned before. Additionally, the user can extract data from the aerosols bounded by the shapes. When the user selects a shape to extract data, VOCAL creates a subplot which displays the data. To find the data, VOCAL uses an interpolation search algorithm to find all the pixels that are bounded by the shape. The Inline Supplementary Code 1 provides VOCAL’s implementation of the interpolation search algorithm.

[Inline Supplementary Code 1]

To support all the features in the CALIPSO visualization tool, the program followed a strict object-oriented design. Several classes were created that handled the internal workings of the program to handle the various use cases that user can potentially create. Figure 1 shows the structure and layout of the CALIPSO visualization tool.

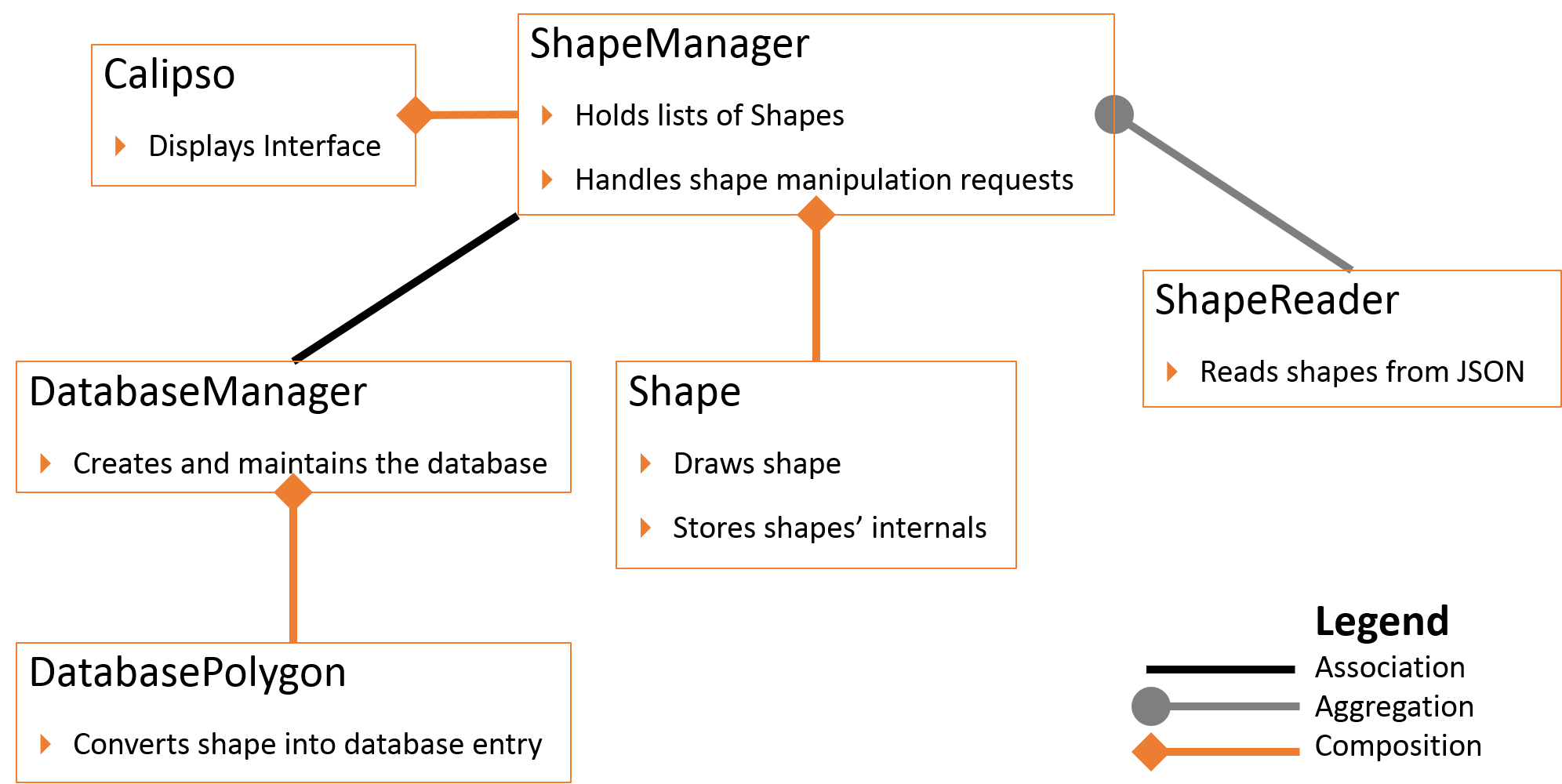


Figure 1. Unified Modeling Language (UML) class diagram of VOCAL

The main class of the program is the ShapeManager. It maintains and aggregates a list of Shape objects for each of the plots (backscattered, depolarized ratio, and vertical feature mask). Additionally, ShapeManager has another list that keeps track of the current plot and utilizes aliasing to update the list as additional objects are drawn onto the plot. The last item of the list is a blank Shape that will update when a user draws a shape. Whenever the user draws changes a shape (deleting, recoloring, moving, etc), ShapeManager will determine which shape the user requested and make the necessary change. In addition to Shape, ShapeManager has a ShapeReader and signals DatabaseManager. ShapeReader and DatabaeManager are used in ShapeManager to load and save shapes from JSON files and the database.

Within the ShapeManager, the Shape class stores the internal attributes of each shape. Every shape a user draws is an instance of a Shape. Shape saves attributes such as the coordinates of the shape’s vertices and the shape’s color. Shape then calls the canvas to draw the appropriate shape by passing in the attributes. After the shape is drawn onto the canvas, Shape continues to maintain the internal variables. When the user prompts the program to save the shapes, ShapeManager will request Shape’s internal variables to pass onto DatabaseManager for saving. When the user loads shapes into the program, ShapeManager will read in the necessary information from ShapeReader and DatabaseManager and Shape will save it and draw corresponding shape.

For file input and output (I/O), ShapeReader handles JSON reading for the program. When the user loads a JSON file, ShapeReader will open the requested JSON file and feed its information to ShapeManager, who will create Shape for each loaded shape. The Inline Supplementary Code 2 provides the implementation of the ShapeReader class and the inner workings of the JSON parser. For writing shapes, ShapeManager directly handles JSON writing. JSON writing works like ShapeReader in the opposite direction. First, ShapeManager will check if the JSON file already exists. If it does not exist, ShapeManager will automatically create the JSON file. Afterwards, ShapeManager will append all the requested shape data. The JSON files allow researchers to retain a local copy of their data and provides an alternative to VOCAL’s database feature.

[Inline Supplementary Code 2]

Like the two I/O operations, the DatabaseManager class handles shapes from input and output sources. The DatabaseManager has a DatabasePolygon, which holds the information represented in Shape for DatabaseManager to convert into a SQLite entry or vice-versa. The DatabaseManger makes extensive use of SQLAlchemy, which encapsulates SQLite queries and management by providing an object oriented interface consistent with the rest of the class structure. When the user saves shapes to the database, the ShapeManager class will pass the relevant information over to DatabasManager for saving. Likewise, when the user loads shapes from the database, DatabaseManager will pass the shapes’ information to ShapeManager, who will then save the data for local use. The DatabaseManager saves all entries onto a database where any researcher can access to download or upload their own shape files.

Unlike the previously mentioned classes, the Calipso directly interfaces with the user. The Calipso class displays the graphical user interface (GUI). The GUI displays CALIPSO data, the different CALIPSO data plots, and the shapes the user has drawn or loaded. The Calipso class has a ShapeManager which maintains all the shapes drawn and interfaces with local and database I/O for saving and loading shapes. Here, the user can draw shapes, zoom in the plot, pan around the plot, and saving and loading shapes. Additionally, the Calipso class interfaces with other auxiliary classes not displayed in Figure 1. These ancillary classes read the HDF data, draws the appropriate plot onto the screen, and populates the screen with menus, buttons, and toolbars for the user to use. Since the Calipso class is responsible for displaying the graphics of the program to the user, it is also responsible for booting the program upon launch.

The entire program was written in 32-bit Python 2.7. The CALIPSO visualization tool relies on several Python libraries and dependencies. The GUI utilizes the Tkinter library to display the graphical user interface and holds all the objects shown on the screen. For line calculations in shape drawing, the program used numpy methods. Database management relies on functions supplied by SQLAlchemy. The program imports ccplot and matplotlib to read CALIPSO HDF files and plot the data onto the screen. Since ccplot is only available in 32-bit version, all other Python modules and the Python interpreter itself must be in 32-bits as well. Additionally, matplotlib handles all the drawing displayed by VOCAL. During development, the program was written using the 32-bit Anaconda Python distribution, as it natively includes Tkinter, matplotlib, and numpy. However, the standard 32-bit Python 2.7 distribution can be used so long as the required dependencies are installed.

# IV. Results & Discussion

VOCAL builds upon and expands on the old IDL tool. With the new features, primarily the shape drawing and the database sharing, Earth scientists can better understand how aerosols affect Earth’s climate and human health. With the old tool, our partners would identify aerosols of interest, estimate the boundaries of the aerosols, and write them to an Excel spreadsheet. The spreadsheet would also contain other characteristics of the aerosol, such whether it was smoke, clouds, or pollution, and any notes by the researcher. Since the IDL tool was incapable of sharing specific data features, if a scientist wanted to share an aerosol, he or she would have to share the Excel spreadsheet instead. The receiving scientist would then have to open both the Excel spreadsheet and the IDL program to view the aerosol. With VOCAL, scientists now have a centralized way of viewing and sharing aerosol data.

In VOCAL, if an Earth scientist wants to highlight a specific aerosol within CALIPSO data, all he or she needs to do is draw a shape that bounds the aerosol, instead of approximating the boundaries and writing them down in an Excel sheet. The shape makes the aerosol of interest immediately noticeable to the researcher and other users compared to reading coordinates from an Excel spreadsheet and searching the shape from the coordinates. The scientist can then add attributes to the shape. VOCAL has a preset set of standardized attributes that the scientist can label the shape, such as smoke, volcanic plume, or cloud. Since VOCAL is an open source program, Earth scientists can add, remove, or change the existing list of attributes to meet their needs. Additionally, the scientist can add a note to the shape to specify additional characteristics or features of the aerosol. Figure 2 demonstrates the shape drawing and figure 3 shows the attribute-labeling feature of VOCAL.

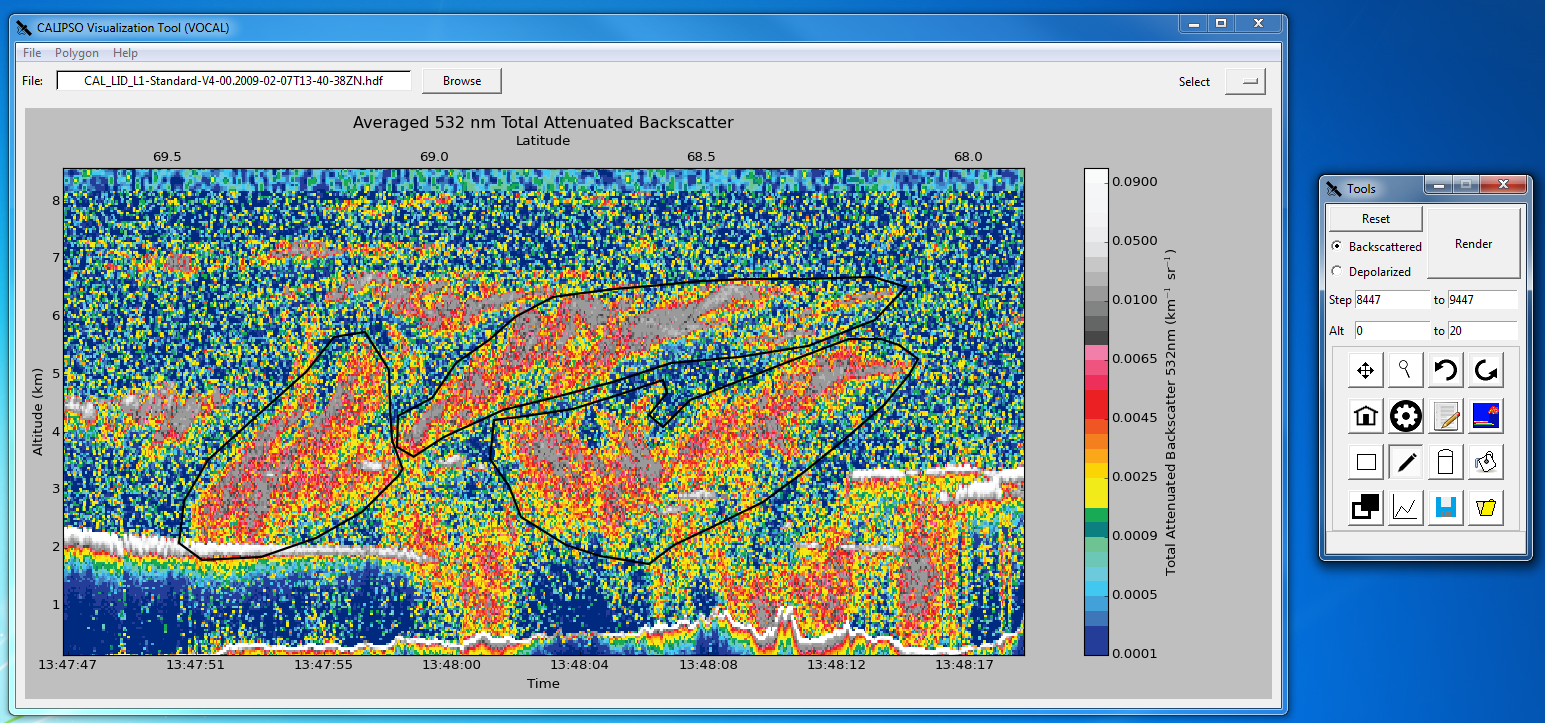


Figure 2. Sample of shapes

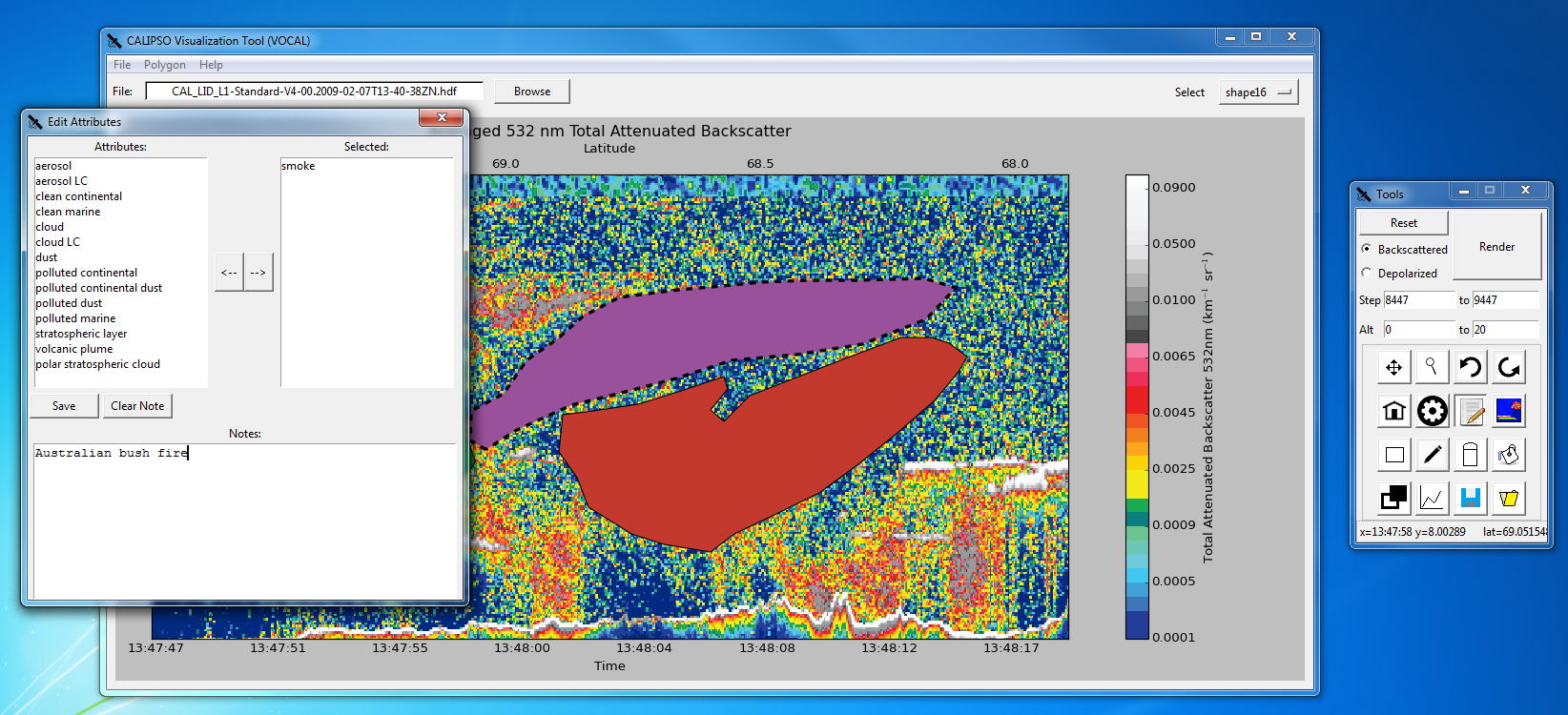


Figure 3. Sample of attributes

On its own, VOCAL’s shape drawing feature simply adds convenience to Earth scientists when identifying aerosols. However, the shape drawing functionality proves its merit in allowing researchers predict the trajectory of aerosols. A scientist can label an aerosol as a volcanic plume for one CALIPSO curtain plot. Then, for another CALIPSO plot at a different time and place, the scientist can label the same volcanic plume that has traversed the globe. After identifying all the volcanic plumes in all the curtain plots, the researcher can then export all these shapes into the database. Another user can view these shapes remotely and see all the volcanic plumes in the various CALIPSO data files. In the IDL program, the user would have to read off coordinates of the volcanic plumes for all the CALIPSO files and find the aerosols, whereas VOCAL makes the volcanic plume immediately noticeable with the drawn shapes. Additionally, the shapes display the volcanic plumes traversal graphic. As the user flips between the different CALIPSO .hdf files, the user can see in which the direction of the plumes travelling and can make predictions of the trajectory based on the shapes’ display. Based on the predicted trajectory, the user can analyze how the volcanic plume might affect local weather patterns, the global climate, and impacts on human health and air quality. The shape drawing and database sharing are not just simple quality of life additions; these new functionalities help Earth scientists identify and track aerosols within the atmosphere to better predict the aerosol’s role on global climate and health.

During the summer term, the team successfully implemented the shape drawing and database sharing features on time, the two main objectives of this project. Future development can refine and expand on these features. Future work may focus on minor bug fixes, optimizations, or other minor improvements. However, there are still areas on VOCAL in which the program can significantly be improved. Currently, VOCAL has functionality to extract data bounded by shapes, though the current implementation is not geared towards Earth scientists’ needs. Future work should focus on this data-extracting feature and tailor it to suit the researchers’ needs.

Another possible improvement is updating the ccplot library. Ccplot is integral for reading CALIPSO .hdf file and the original author of ccplot has discontinued support for the library. The biggest flaw with ccplot is that it only runs with 32 bit Python. This severely limits VOCAL’s computational power, as this forces VOCAL and all the library is uses to be in 32 bits as well. An ambitious improvement would be to reimplement ccplot on 64 bit Python.

In addition to the previously mentions changes, VOCAL’s pan feature could also be improved. Current implementation is constricted by the sheer amount of data to display on the screen. The typical CALIPSO plot contains 60,000 ticks of time data, each tick having a list of specific pixel values for each tick of altitude. Most useful data comes from the first 20 km, however with so many single points of time data a 32-bit python program simply does not have enough RAM to load all of the CALIPSO data. The current pan implementation divides the CALIPSO data into pages, the largest allowing only 15,000 ticks of time data to be displayed. When the user pans CALIPSO data, VOCAL loads the next page to display the new subsection of data. However, this operation is time consuming as VOCAL must load millions of data points. When the user pans, he or she sees a choppy animation as the plot moves horizontally. Because of this, the user has a hard time discerning the translation between dragging the plot and moving the plot horizontally. To improve panning, pages should be cached or preloaded so when the user pans the CALIPSO plot, the user sees smooth horizontal movement as the user drags the plot. This feature would require asynchronous operations and use of multiple threads to load the page into the cache.

The improvements listed may be accomplished in another term. However, since VOCAL is an open source project, anyone can contribute and improve VOCAL. The source code is hosted as a repository on GitHub, a popular version control system. Users can make changes and improvements to VOCAL, such as bug fixes or new features. To share these modifications, the user can send a pull request to the repository. Other users can comment and discuss whether to include the modifications into the source code and make it a standard feature within VOCAL. The owners of the repository can decide whether or not merge these changes. Figure 4 shows an example pull request.

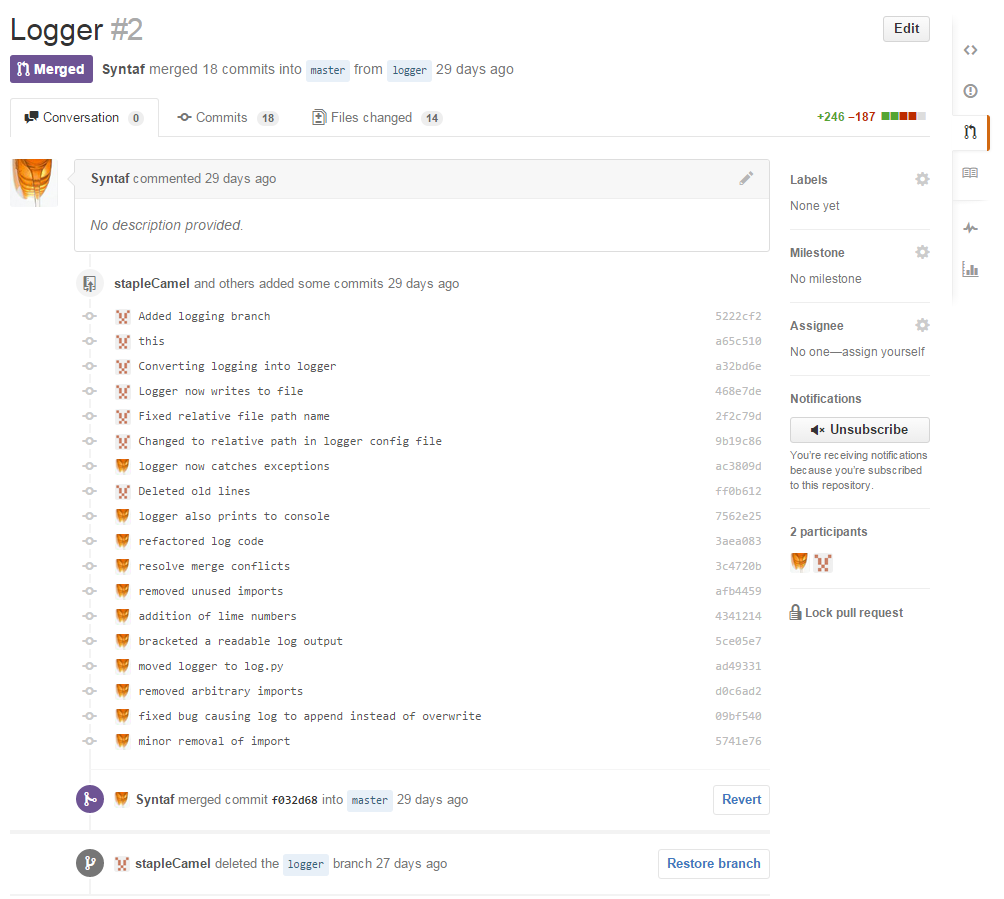


Figure 4. Example of a pull request

Additionally, GitHub provides convenient method of provided bug reports and requests. A scientist might find a bug in VOCAL, but may not have any programming knowledge to fix the bug. The scientist can post an issue on VOCAL’s GitHub page, where other users can see the issue, discuss it, and address it. VOCAL has a logger that reports on internal operations. If there is a crash or a bug, the scientist can post the log trace within the issue to help programmers and the owner fix the issue. Figure 5 demonstrates how issues work on GitHub. Issues posted on GitHub can also be requests for new features. If a scientist wants a new feature added into VOCAL and does not know how to implement it, he or she can post an issue on GitHub requesting this feature. Like any other issue, every user can view this issue and discuss whether or not to implement the request. The open source nature of VOCAL allows every user to collaborate on VOCAL’s development, regardless of programming knowledge. This allows scientists to easily adapt VOCAL to their needs and goals.

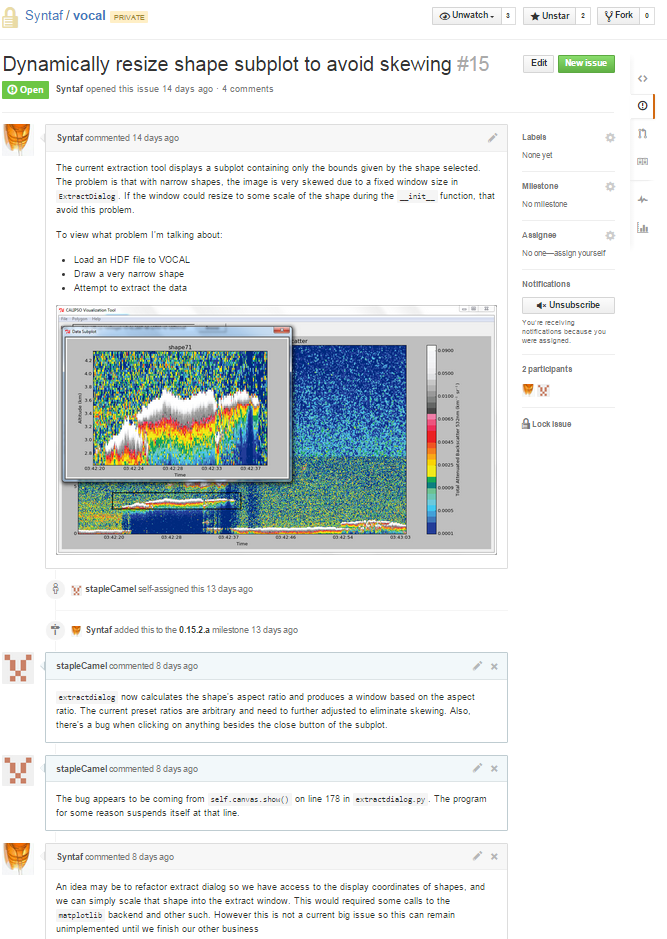


Figure 5. Example issue

# V. Conclusions

With the completion of VOCAL, Earth scientists will finally have a customizable and open source tool to help them analyze the impacts of aerosols on Earth’s climate and human health. Aerosols are a major factor in Earth’s climate; they can affect cloud formation and cloud radiative properties. The CALIPSO satellite, launched in May 2006, measures the global distribution of aerosols in Earth’s atmosphere and allows climate scientists to analyze aerosols and determine their potential impact on climate and human health. To view CALIPSO data, researchers used a tool written in IDL. However, this IDL tool is difficult to modify and lacked key functionality that would have aided researchers in identifying aerosols’ role in climate and air quality.

VOCAL is a major improvement over the old IDL tool, and includes new features, such as shape drawing and database sharing. VOCAL, written in Python, implements some of the old IDL tool’s features and adds additional functionality. The shape-drawing feature will allow scientists to highlight specific aerosol features of interests. The shapes additionally provide scientists the ability to view the data bounded by the shapes, enabling scientists to extract subsections of CALIPSO data. Researchers can then use these shapes to predict the trajectory of these aerosols within the Earth’s atmosphere. After selecting all aerosol features, the scientists can then share these shapes on the database, so that other researchers can view these aerosols and collaborate remotely on projects. Additionally, since VOCAL is an open source software, Earth scientists can change and modify the tool to meet their needs, another major improvement over the old IDL tool.

# VI. Acknowledgments

CALIPSO Health and Air Quality LaRC Spring 2015

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

CALIPSO Science Team

* Dr. Charles Trepte
* Dr. Amber Soja

NASA DEVELOP National Program

* Jeffry Ely
* Dr. Kenton Ross

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

# VII. 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, D. M., J. Pelon, J. A. Coakley, S. A. Ackerman, R. J. Charlson, P. R. Colarco, P. Flamant, Q. Fu, R. M. Hoff, C. Kittaka, T. L. Kubar, H. Le Treut, M. P. Mccormick, G. Mégie, L. Poole, K. Powell, C. Trepte, M. A. Vaughan, and B. A. Wielicki. "The CALIPSO Mission: A Global 3D View of Aerosols and Clouds." *Bulletin of the American Meteorological Society Bull. Amer. Meteor. Soc.* 91.9 (2010): 1211- 229. *Web of Science*. Web. 17 June 2015.

Winker, David M., Jacques Pelon, and Patrick McCormick. “The CALIPSO mission: Spaceborne Lidar for Observation of Aerosols and Clouds.” *SPIE Proceedings* 4893 (2003): 1-11. *SPIE*. 24 Mar. 2003. Web. 17 June 2015. <http://proceedings.spiedigitallibrary.org/>.

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.

# VIII. Content Innovation

VPS:

2015Sum\_Langley\_CALIPSOCrossCutting\_VPS.mp4

Inline Supplementary Material:

2015Sum\_LaRC\_TechPaper\_Inline\_Supplementary\_Material.txt

# IV. 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/>