COAL: Coal and Open-pit surface mining impacts on American Lands

Taylor Alexander Brown, Heidi Ann Clayton, and Xiaomei Wang

Group 18

CS 461: Senior Capstone Fall 2016

Oregon State University

Abstract

Mining is known to cause environmental degradation, but software tools to identify mining impacts are lacking. Researchers studying this problem possess large imaging spectroscopy and environmental quality data sets as well as high-performance cloud-computing resources. This project provides a suite of algorithms using these data and resources to identify signatures of mining and correlate them with environmental impacts over time.

CONTENTS

1 INTRODUCTION

This is the design document for COAL. It outlines the design of our system as well as the research and development priorities at each step. The COAL suite of algorithms constitutes a pipeline of data processing and presentation, and is accompanied by API documentation as well as a website for end users and the general public. Specific implementation details remain to be determined based on research, but in general the program reduces a large set of hyperspectral and geographic data into comprehensible conclusions about mining and environmental impact in human and machine-friendly formats.

Each section was authored by the corresponding team member responsible for each component identified in the technology review: Sections 2.1 and 2.2 by Heidi; section 2.3 by Xiaomei; sections 2.4, 2.5, and 2.6 by Taylor; section 2.7 by Heidi; and sections 2.8 and 2.9 by Xiaomei. The research priorities and design choices for each part of the project are discussed in each section.

2 TECHNOLOGIES

2.1 Feature extraction in AVIRIS data

In order to process hyperspectral data, machine learning techniques will have to be used to process the raw AVIRIS images [1]. Further processing could be done using blob detection, as described below.

AVIRIS images cover a lot of land mass with a lot of different features, so a method of feature extraction is needed to automatically pick out the different regions on the image. In blob detection, an area of an image that is different from the surrounding pixels is known as a blob. This will make classification easier as instead of needing to classify every single pixel every single time, the classification algorithm will only need to process each blob, or feature, which typically contains many pixels. The blob detection algorithm that will be used is automatic scale detection. It is one of the more expensive methods of blob detection, but it is highly accurate, so it is the one that has been chosen as discussed in the technology review. The automatic scale detection algorithm is based on the Laplacian of Gaussian. As the name suggests, this is a convolution of the Laplacian and Gaussian filters [2]. The Gaussian filter is applied first to smooth out noise and then the Laplacian filter is used to detect the shapes in the image [3]. The equation for a Gaussian convolution is given below, where t represents scale [2].

$$g(x,y) = \frac{1}{2\pi t} e^{-\frac{(x^2+y^2)}{2t}}$$

The equation for a Laplacian filter is given below, where t represents scale [2].

$$\Delta^2 L = t(L_{xx} + L_{yy})$$

When the Laplacian filter is applied to the Gaussian filter, the equation becomes the one shown below [3].

$$LoG(x,y) = -\frac{1}{\pi t^4}[1 - \frac{x^2+y^2}{2t}]e^{\frac{-x^2+y^2}{2t^2}}$$

In automatic scale selection, the algorithm is looking for the scale that produces the largest Laplacian response in the blob center [4]. When looking at a kernel, the scale that is the best fit so to speak, is the maximum value in the kernel. Figure 1 is an example of this. The scale that would be chosen is five. Looking through the values, all of them correspond to specific behavior in the section of the image it represents. On sections that are fairly homogeneous, the

-1	0	0	0	0
0	1	2	1	0
0	3	5	1	0
0	2	2	3	0
0	0	0	0	-2

Fig. 1: An example of a kernel to be analyzed by automatic scale detection.

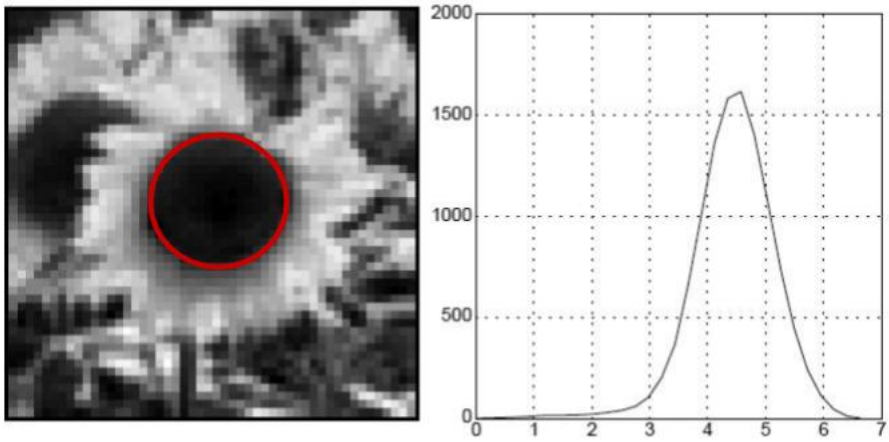

Fig. 2: A demonstration of the largest scale blob in the center of the flower corresponding with the local maximum of the curve [4].

individual values in the kernel will be zero or close to zero. When a difference occurs, the values will start to become negative on the lighter side of the blob and positive on the darker side [3]. Many implementations of the Laplacian of Gaussian method use a reversed sign. That is, the minimum is chosen instead. This does not alter the correctness of the algorithm, but we will be using a the positive signed implementation where the scale chosen is the largest number [3].

When graphed on a Cartesian coordinate system, the scale chosen is the local maximum. As illustrated in figure 2. with an example blob being the center of the flower, the largest and best fitting blob corresponds to the local maximum when graphed on the Cartesian coordinate system.

One important factor that needs to be considered for this design item is that the values being generated inside the Laplacian of Gaussian kernel can get very, very large [3]. It is very important that the environment used when running the automatic scale detection algorithm can support a large range of numbers, both positive and negative. The possibility of overflow is likely and this could very heavily skew the results of blob detection. The development team does not have experience with running this type of algorithm, but support for a minimum of 32-bit numbers is what will be used as a starting point.

2.2 Classification of AVIRIS data

After the data has been processed using feature extraction as detailed in the previous section, the resulting data will be put through a process of classification. The classification will take the AVIRIS images, which are essentially a map of

reflected light, and produce an image where each pixel is identified as belonging to a certain class [5]. This serves the purposes of making the data more readable to the development team and any other person who is using COAL and classifying the data in a useful way to aid in the identifying minerals, identifying mining, and correlating the effects with environmental impact design items. The classes will be types of land surfaces. This includes water (e.g. turbid versus less turbid), vegetation (e.g. dense versus less dense), basaltic lava flow, snow, and any types of minerals found in the image [1]. The algorithm used for classification will be a neural network.

The classification will be done by recording the differences of reflectance in the pixels of the image. For example, snow is highly reflective in the visible light spectrum but not in the infrared spectrum [5]. We will use both supervised classification and unsupervised classification to classify the different types of land surfaces in the images. That is, we will be training the neural network algorithm to look for specific minerals and land surfaces but will also try training the algorithm to classify pixels into any class deemed significant. In the training stage, COAL will use AVIRIS data on sites where the land surfaces are known. These images with known land surfaces will train the neural network algorithm to classify the land surfaces of AVIRIS images where the land surfaces are not known.

When testing the progress of the training on the neural network algorithm, the algorithm will be run on images with known land surfaces that were not used in the training stages. If the algorithm correctly classifies at least 80% of the pixels in at least 80% of the images used for testing, the training will be considered sufficient. Otherwise, more samples will be required in the training and the algorithm will need to be tested again. In general, small sample sizes do not produce reliable results when making predictions, so making sure the training has used enough AVIRIS images is important.

After the training stage, AVIRIS images with unknown land surfaces will be processed with the trained algorithm. The algorithm will classify each pixel into a class representing a land surface and that data will be used for further analysis (see design items 2.3, 2.4, and 2.5) and possibly be output into an image to be used on the homepage.

Something to consider when training and using this algorithm for analysis is spectral separability. Some types of minerals and other land surfaces have very similar relationships between reflectance and wavelength and can be cause the algorithms accuracy to suffer if they are separated into different classes [6]. It is up to the developers to determine if the particular minerals or other land surfaces that are not very separable should be put into the same class or kept separate depending on if the minerals or other land surfaces in question are important enough by themselves in some way to warrant a slightly lower accuracy.

2.3 Identify Mining

After we get the mineral identification and classification data, we need to identify the regions of mining activity. The data is further processed and possibly combined with geographic information about geology and mines. The goal of identifying mining is to accurately locate regions in which mining is taking place in a form that is suitable for both human inspection and further processing. This step creates input data for further processing of the project and the data evaluation.

To start our process of identifying mining, we need the output from the mineral identification step. The previous step has already located the regions that are containing mineral associated with mining. To distinguish mines from preexisting geology is a challenge; only rely on the existing geography data is not enough for the identification.

There are three methods mentioned after several group discussion sessions for identifying mining: mineral classification, using patterns of mineral deposits; GIS comparison, using geographic information such as mining permit

boundaries or geologic maps; and a combination of mineral classification and GIS comparison, using both sources of data. To locate regions with mineral deposits that indicate mining, we have decided that we are going to use the mineral classification methods. Then, we would move to a combination of mineral classification and GIS comparison methods once we are able to complete the first step without much trouble.

The mineral classification method can be very simple, mapping the presence of a mineral may be sufficient to locate mines: minerals with low natural concentrations can be used to identify mining activity. In this case, the result is expected to be straightforward, and the time spending on the identification is short. However, the outcomes are not always favorable under this method. Still, at the very beginning, we expect to rely on this to produce some useful outputs for further application. The combination of the mineral classification and GIS comparison can be costly; it would require the most research and time. Although the outcome can be most complete, so that is why we use mineral classification method at the beginning; simply mapping the presence of a mineral may be sufficient to locate mines due to minerals with low natural concentrations can be used to identify mining activity. GIS comparison can provide more accurate results.

2.4 Preprocessing Data to Correlate Mining with Environmental Impact

After mines have been identified, we are interested in analyzing their impact on the environment. Using environmental quality GIS data from agencies such as the EPA, we would like to be able to correlate mining operations with environmental degradation. This stage depends on the previous steps in the data processing pipeline and will require additional research to analyze and solve the problem.

Research and development of this step will focus on identifying and analyzing available literature and tools. The principal problem is to combine the spectroscopic-derived data with geographic data. Preprocessing this data will require using the metadata from the imagery to associate pixels with geographic coordinates. It will also require loading and handling geographic data. Researching means of accomplishing both of these processes will thus be an important and ongoing task.

In addition to literature, we will be identifying and learning to use tools and libraries for handling the data. Learning to use GIS applications such as ArcGIS is therefore a high priority. Using proprietary GIS software will require the team to arrange suitable licensing. Library methods for analyzing the imagery metadata and the geographic information will probably be used, so identifying suitable libraries will also be an important focus.

Having researched literature and tools, our development focus will turn to implementation. Decisions will need to be made about how to structure the application and how to pass the preprocessed data to the following feature extraction step. It may make more sense to combine both preprocessing and feature extraction of environmental impact into a single module, or we may find that the components are best separated into distinct procedures.

Ensuring that the input and output of our program is consistent is best accomplished by automated testing. It would be possible to apply test-driven development and write tests to specify program behavior, but we may find it more natural to implement procedures and tests alternately. At the end of the project, our tests will serve as an informal specification and an assurance of correctness.

The goal which the tests will verify is that the data is output in a suitable format for feature extraction. The spectrometry-derived data identifying mines will be associated with geographical locations. We may also find it desirable to simplify the data by discarding unnecessary information. The geographic information having been loaded into the

application may likewise be simplified. We will need to identify specific layers of interest to pass on to the next step. Combining both of the data or keeping them separate will be a factor in the design as well.

Preprocessing data to correlate mining with environmental impact thus depends on a sequence of steps. Initial research and development must be undertaken to analyze available literature and documentation as well as to find tools and libraries to facilitate our programming. Implementing the algorithms will make our assumptions explicit and require decisions about structure and format. Ensuring they meet our requirements will involve automated tests. The end product will be preprocessed imagery and geographic information that can be used to make correlations between mining and environmental impact.

2.5 Feature Extraction to Correlate Mining with Environmental Impact

Given preprocessed mine imagery and geographic information, feature extraction derives correlations between mining and environmental impact. Research will occupy a preliminary and ongoing part of implementing this module. Implementation will require design decisions as well as more complete specification of the desired output data. The result will be different representations and visualizations of data, such as tabular and graphical displays. Automated tests will help enforce our assumptions about the procedures, but data that is generated in human-readable formats will have to be analyzed in person and adjusted according to utilitarian as well as aesthetic preferences.

The research priority of this stage is to determine the best ways of deriving and presenting the results. One approach that has been described in our readings is to simply overlay the data sets in a geographical representation. This allows an end user to determine by eye the regions where mining and environmental problems overlap. However, we need to find out whether there are means of improving the accuracy of the correlations or removing superfluous results. Can we assign confidence values to sites where mining and pollution appear to be linked, or are the results more naturally boolean in character? What other ways of presenting the data improve the end user experience of processing the data? These are some of the problems and questions our research will seek to answer.

In addition to researching derivation and presentation, implementation will require referencing the documentation of software libraries used in this module. It may also depend on patching contributed libraries to correct bugs or add features. Imaging libraries, geographic information libraries, statistical libraries, and data visualization libraries are all likely candidates to contribute to our project. Identifying suitable libraries, becoming proficient with them, and fixing them are some of the steps implementation will entail.

The libraries we choose will influence the design choices of the module as well as the amount and nature of the code we write. As mentioned previously, this step and the preceding preprocessing step may be combined into a single module or they may be separated. Implementation of this module requires procedures for handling the data and generating representations for the end user. Separating these procedures would facilitate modularity and testability. They could be unified in the application program by means of command-line parameters or preference files selecting, for example, the desired graphical representation or the algorithm used to generate it.

The output as described may include tabular and graphical representations such as map overlays and charts. It will be necessary to specify the data formats to generate and to produce them to best meet the needs of the environmental data scientists using them. Tables, bar charts, line charts, geographical maps, and diagrams are all possible representations of different quantities in the result set. Probabilistic data using confidence values would require a means of conveying the different probabilities as well as potentially cutting off results that are below a certain level of accuracy.

Automated tests will help ensure the correctness of parts of the program, but the end result will be best analyzed in person. Components that could be tested include those responsible for converting input formats to a predictable binary output format. Sample data can be provided and tested to ensure that results such as tabular representations are consistent with expectations. However, the overall usefulness and quality of the data visualizations must be judged subjectively.

Deriving and displaying correlations between mining and economic impacts is the purpose of this feature extraction stage. Research into literature and libraries will be conducted, algorithms will be implemented, and graphical representations will be selected to convert preprocessed data into user-friendly formats. Automated tests will aid development, but the results will also depend on subjective determinations.

2.6 Rank and Document Changes Over Time

Ranking and documenting changes over time is the final step in the data processing pipeline. After identifying minerals, mines, and associated environmental impacts, further analysis is possible by providing a historical context to environmental changes. Implementation of this step could be as simple as instructing the end user to run the program several times over separate datasets, however this would not provide effective usability. A more sophisticated implementation would take as input a historical data set and generate trends automatically. Whatever approach is chosen, research into literature and libraries will again be a significant part of developing this feature.

Before development on this step can begin, the preceding stages of data processing must be complete. The design choices and implementation details made at each step will affect the options available for temporal analysis. Data formats and representations will influence the kinds of displays that can be produced. Program structure will determine how reusable certain software components are, a consideration that will be important when combining them to analyze changes over time.

The simplest interpretation of documenting changes over time is to process historical data and generate representations for each time period separately. A note in the user documentation would describe how to execute the program and possibly means of automating it. Shell scripts could be provided, for example, to apply the environmental correlation procedure to a collection of historical images and geographic information files. However, the first difficulty occurs when associating AVIRIS flight times with environmental measurements which were most likely produced at separate times and at separate intervals. The end user would have to make decisions about the nearest applicable data set, a requirement which would need to be noted in the documentation as well.

More sophisticated methods could be implemented to relieve the burden of some of these tasks at the cost of additional research and development time. For example, this module could receive command-line arguments or configuration files containing multiple data files and run the analyses automatically. Designing the program this way would allow new options for output such as generating temporal graphs which would not be possible if correlations are made in isolation. Research and development would be required to handle the time dimension of the data and generate it in a suitable and attractive format. Possibilities for output using this approach include animations which display the progression of environmental changes or images which display the same information statically.

It is likely that the libraries identified in previous stages will be used in this step, however new data representations such as animations could entail additional third-party components. Besides the specifics of implementation, more general points about the format of the data visualization will be necessary to produce a useful and attractive end product. Objective criteria for determining data visualization include pertinent trends or conventions in environmental

literature which would be desirable to emulate. Subjective criteria include choices about color and contrast which in addition to affecting the aesthetic quality of the application may also influence usability and accessibility.

The research, design choices, and implementation of this module will produce results which cap off the suite of data processing algorithms provided by COAL. Ranking and documenting environmental changes over time combines the information from all earlier steps into a unified whole for producing temporal analyses to inform environmental data scientists and other interested parties. Design and implementation could be simple, but depending on the time left available for developing this step after the others, more sophisticated approaches could be devised to enhance usability.

2.7 API Documentation

The Sphinx API documentation will be developed throughout the entirety of the project, with each team member documenting their code contributions as they develop. The code written will include docstrings for all functions that will include, at minimum, a short description of the purpose of the function, description of each parameter, and the input and output of the function. Further docstrings in the code will include explanations of algorithms and structures. We will then use reStructuredText (.rst) files to integrate any function descriptions with further elaboration and any examples, figures, and/or images. We will have a main page for the documentation that includes a table of contents, a search bar, and a general description of the API. We will also have a page regarding how to use COAL's API and a page for each library. The goal of the documentation is to be useful to a person who wants to use it in their software project but does not necessarily want to know everything that is going on in explicit detail. It is a description of the API as a black box for the most part, though the source code is available to those who want more detail and is linked on the homepage 2.9.

The main page has an auto-generated table of contents and search bar, so the only thing we will be writing on the main page is the general description of the API. The page regarding using the COAL API will include information on dependencies, compatibility with AVIRIS imagery, and examples showing how to get started with COAL. Each page documenting a library will have a short description of the library at the top, followed by a description of each function (documented in the code itself with docstrings). If applicable, any concerns regarding compatibility with other libraries or performance should be addressed here. This is particularly important since image processing takes a lot of computing power. The specifics of algorithms and statements are not necessary for the auto-generated documentation. We will put comments in the source code where necessary for people who want more detail, but the API documentation will only include specifics if they raise any kind of concern.

Testing will be relatively simple to make sure the API documentation is working correctly. Syntax errors are raised by Sphinx if they are encountered, so they are easy to check for. We will also be doing spell checking on all of our documentation to make sure it is clear and professional. We will also check to make sure that the documentation is output in a place that is accessible to users.

Members of this team have used Sphinx documentation before and found it elegant, flexible, and easy to read and write. It is also the method that Python uses for its documentation, which is frequented by members of this team and is seen as nicely formatted and organized. Sphinx is also compatible with Python 2 and Python 3, which makes it a good match for us if we ever decide we want to switch completely to Python 3.

2.8 Static site generator

We will use a static site generator to provide a fast and simple back-end for COAL's homepage. GitHub let users create one site per GitHub account or organization, which is convenient for us for that we have to keep our page content dynamically. What we will do first is create a GitHub page. The GitHub page can host personal, organization, or project pages directly from a GitHub repository. GitHub pages use github.io domains which are served over HTTPS. The master branch of the repository is responsible for t'she publishing; this can be selected from the GitHub Pages site repository under the settings option. Once we have it linked, we will start to build our website. The advantage of the static site generator is that it does not need to deal with databases. The static site generator mostly works with Javascript and HTML and that is why it has fast speed and relatively better performance. The main purpose of our website is to display the resources we need and make it convenient for our team to perform our tasks, so a static site generator is the perfect choice for us. Our site will not require many complicated functionalities, so a more complex back-end is not needed. Also, because we need to change the content on the home page often as the project continues, static site generators tend to have excellent version control for the content.

As the technology review states, we are going to use Jekyll as our static site generator. Jekyll is a simple, blog-aware, static site generator perfect for personal, project, or organization sites. We chose Jekyll as our static site generator because it has high compatibility with GitHub pages, which makes the work easier to do. To start to use Jekyll, we will simply download from the GitHub pages or the official website. Once we get the resources we need, we will create the back end to our home page. We do not have any experience with using Jekyll, but the guide on the Jekyll website is simple to follow.

Overall, using a static site generator is not a difficult task and there are a lot of resources online. Choosing a static site generator will save us time with maintenance and server resources.

2.9 Front-end framework

Due to the continuous nature of our project, we will need a website to show our work and progress both throughout the school year and possibly beyond. There will not be much content to put on the site toward the beginning of research and development, but the content will be filled out as the project progresses. The website serves the purpose of showing our work and presenting our project in a clear and succinct way.

We use plain HTML to build the skeleton of the website, then we put the basic information on, such as developer profiles, project information, and relevant links (Github, class website, etc.). Xiaomei will be using a template for a navigation bar that she has used before so that sh can build a simple page before filling everything inside.

Then, based on what is required, or what we want to put on our website later, we may need to alter the content. For the styling, we will use CSS. Our team has discussed that we would mostly use Bootstrap for our front-end framework, for the abundance of its themes and functionalities. Bootstrap is easy to use and works well with HTML, CSS, and JS; we choose it because of it saves us time so we can build a basic page very quickly. We also need some pictures related to mining or AVIRIS to make our site more aesthetically pleasing and to provide interest. There are many images of AVIRIS and its flight lines, so this should not be an issue. We will need to search for some techniques to fit different blocks for various using and make them look nice.

As the project continues, we will update our homepage on a weekly basis as well as link to our individual weekly updates for required for the computer science senior design course.

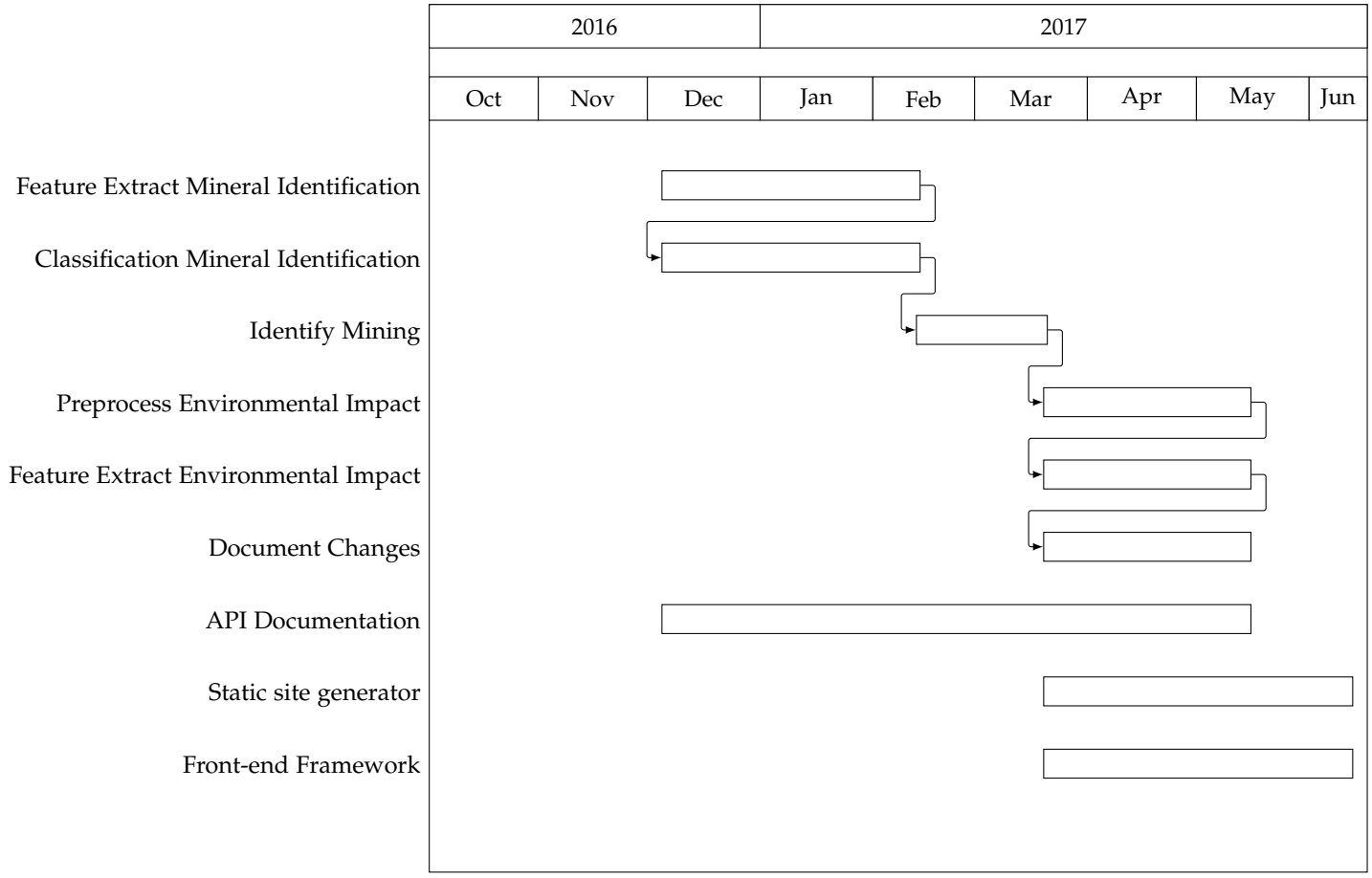

Fig. 3: Gantt Chart: Timeline of Project Tasks

3 Timeline

The Gantt chart pictured in Figure 1 describes the project timeline. Dependencies among each stage of the project are depicted by arrows and approximate implementation dates are indicated by the date range. Some components may be able to be developed independently while others may require a sequential approach. In general, however, our development will follow the data from initial processing to final representations. API documentation will parallel the development of our source code using comments to automatically generate browsable reference material. The website will provide a hub for developers, end users, and the general public describing our project, and its development will be relatively independent of the algorithmic components.

4 Conclusion

This document has outlined the design of the COAL project and the suite of algorithms and documentation it provides. The major components of the system were broken down and assigned to each team member who described in each section the research priorities and design decisions involved. Processing large datasets of imagery and geographic information require a significant investment to study and develop solutions. However, the results will provide a wealth of information for environmental data scientists and other concerned parties looking for associations between mining and environmental damage. Further revisions of this document will reflect specific choices made as the project matures.

5 GLOSSARY

AVIRIS	Airborne Visible/Infrared Imaging Spectrometer
COAL	Coal and Open-pit surface mining impacts on American Lands
docstring	Documentation string. In Python, they are comments that start and end with three quotation marks.
GIS	Geographic Information System

REFERENCES

[1] J. A. Benediktsson *et al*, "Classification and Feature Extraction of AVIRIS Data," IEEE Trans. on Geoscience and Remote Sensing, Vol. 33, No. 5, September 1995.

[2] T. Lindeberg, "Principles for Automatic Scale Selection," Handbook on Computer Vision and Applications, B. Jähne, et. all, eds., Academic Press, 1998.

[3] D. Matthys. (2001, March 21). *Laplacian of Gaussian Filter* [Online]. Available: http://academic.mu.edu/phys/matthysd/web226/Lab02.htm

[4] N. Snavely. (2012). *Features 2: Invariance and blob detection* [PowerPoint slides]. Available: www.cs.cornell.edu/courses/cs4670/2012fa/lectures/lec07_features2.ppt

[5] W. Amidon. (2014, March 17). *v25 introduction to classification in ArcMap and ENVI* [YouTube video]. Available: https://www.youtube.com/watch?v=OBJKGWocM6Q

[6] W. Amidon. (2014, March 17). *v26 evaluating spectral separability in ENVI* [YouTube video]. Available: https://www.youtube.com/watch?v=y685Km9njys