CS Capstone Design Document

May 25, 2018

Aerolyzer

Prepared for

Kim Whitehall

Prepared by

Group 19
Aerolyzer
Daniel Ross
Logan Wingard

Abstract

The Aerolyzer Project aims to deliver a new source of air quality and weather information through leveraging existing weather data and image analysis algorithms. When complete, this open-source project shall feature a Python library that uses image classification and third-party weather APIs, displayed with an intuitive web-based user interface. This document outlines the software design descriptions for the Aerolyzer Library

1 Table of Contents

Contents

REFERENCES

[1] B. Allen. (2015, Apr) Atmospheric aerosols: What are they, and why are they so important? [Online]. Available: http://www.nasa.gov/centers/langley/news/factsheets/Aerosols.html

[2] Introduction to support vector machines. [Online]. Available: https://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html

2 Overview

2.1 Scope

Aerolyzer mobile web application is intended to retrieve data from user images in order to analyze colors in the images using Wunderground data and image analysis software. This application proposed by Kim Whitehall is intended for aiding in the public understanding of air quality and of aerosols in the atmosphere.

2.2 Purpose

This document describes the three main components of Aerolyzer that will be designed, those being the library data interface, the horizon detection filter and the library structure.

2.3 Intended Audience

The intended audience for this software design document is the Aerolyzer developers and clients. It will serve to set design intentions so that the client can confirm the design plan will meet the requirements.

3 Definitions

3.1 Aerosol

Aerosols are tiny particles dispersed throughout the atmosphere. These particles originate from natural sources such as volcanic eruptions, and unnatural sources such as pollution. One can visibly see the effects of aerosols in the 'Rayleigh scattering effect' which visibly reddens sunsets and sunrises. [1]

3.2 Acceptable Image

An acceptable image is defined as an unedited image of the horizon. 3.3 Images used in data collection must have valid EXIF meta data, while the images used in training classifiers only require relevant image content. 3.4

3.3 Horizon

The horizon is defined as the line at which the sky and earth's surface appear to meet.

3.4 EXIF

Exchangeable image file format. This is the standard format for the metadata attached to jpg and png type images.

3.5 Local

Local is defined as the circular area around an observer with the radius being the distance from an observer's position to the horizon. For an observer on the ground, this distance is approximately 2.9 miles (4.7 km) and for an observer standing at an elevation of 100 ft (30 m) above ground level this distance is approximately 12.2 miles (19.6 km).

4 Design Description Information Content

4.1 Introduction

Aerolyzer is a Python library which hosts the functions called by the Aerolyzer_App framework. These function ultimately provide the user with information about the types of Aerosols that are most likely present in their area based on the colors of the image.

4.2 Software Design Description Identification

4.2.1 Image Data Preprocessing

When a user submits an image to the web framework it's immediately passed to a function called "check_image". The first portion of check_image reads the image's metadata to determine whether the image is suitable for aerosol analysis.

4.2.2 Horizon Detection Filter

The Aerolyzer library hopes to feature a module that infers aerosol size based on the hue distribution in images of the sunrise or sunset. To this end, a filter that intelligently recognizes acceptable images 3.2 is necessary to save computation time and guarantee reliable data.

4.2.3 Library Structure

The Aerolyzer library aims to be accessible to import for scientific research purposes. The structure of this library will determine the ease of access of this library.

4.2.4 Color Analysis

Once the image of a horizon at sunrise or sunset has passed all of the restraints put in place by the rest of the library it's passed to Color Analysis. This is the most time intensive function of the library since it performs a series of comparisons on the Color of each pixel in a large set.

4.3 Identified design stakeholders

4.3.1 Image Data Preprocessing

The primary stakeholders for the Image Data Preprocessing are the users and the development team. Image Data Preprocessing is meant to save the user's time by telling them that their image won't work for analysis before more time consuming calls are made. The other purpose is to make the image analysis functions more accurate for the user's sake and the development team, since it provides a standardized baseline for the images that are analyzed.

4.3.2 Horizon Detection Filter

The stakeholders for the Horizon Detection Filter python module include users who will be using the Aerolyzer web service. Users will ideally not have to wait for long periods of time for the server to process weather and air quality analysis. The horizon filter will expedite this process by filtering out bad data.

4.3.3 Library Structure

The stakeholders for the Aerolyzer librarys structure will be the developers and the Aerolyzers primary stakeholder, Dr. Kim Whitehall. The Dr. Whitehall requires the library to be accessible from Pythons package index.

4.3.4 Color Analysis

The stakeholders for the Color Analysis functionality of the Aerolyzer library will be Dr. Kim Whitehall and the end users. The main purpose of the Aerolyzer library is to provide an estimation of the likely aerosols in an image and the analysis of the image's color is the method by which this is accomplished.

5 SELECTED DESIGN VIEWPOINTS

5.1 Image Data Preprocessing

5.1.1 Design Elements

5.1.1.1 Design Attributes: Smartphone images all store EXIF data when the image is created. This data varies from device to device as EXIF data isn't entirely standardized, but there are certain tags that the Aerolyzer library will require the image has.

5.1.2 Design overlays

There are seven major tests in the image preprocessing. Each of these tests uses a different portion of the image's EXIF data. If the image fails any of these test then the image isn't verified. Without getting verified the color analysis functions won't be called on the image. The following are the constraints for each test in order from least time intensive to most time intensive.

1) The file type of the image must be .jpg or .png.
2) The image must be no larger than 4Mb.
3) The image must be in the resolution range 600X600-6000X6000.
4) The image must be a mobile image from a supported device.
5) The image cannot be edited or filtered in any way.
6) Location services must be enabled for the camera.
7) The image must be taken during sunrise or sunset.

5.1.3 Design rationale

Each test is meant to ensure the color analysis, to determine the aerosol content, is accurate. The first test checks the EXIF tag ”file type” to ensure that the image can be passed to OpenCV since it’s primary image types are jpg and png. The second and third test could potentially be adjusted depending on the processing power of the hosting device, but they check the tags ”file size” and ”exif exifimagewidth” respectively to put limits on the size of the image which helps standardize the speed of image analysis. The forth test checks that the exif tag ”image model” is in the list of acceptable models. The fifth test compares the time the image was originally taken to the last time it was modified, which are stored in the exif tags ”exif datetimeoriginal” and ”image datetime” respectively. The sixth test checks the exif tags ”gps gpslatitude” and ”gps gpslongitude” to make sure that the image had location services on and that it was taken in the United States. The seventh test uses the image’s location data and time data. A call is made to the WeatherUnderground API using the image’s location and original datetime. Then the sunrise and sunset times returned by tWeatherUnderground are compared to the time that the image was taken. If the image wasn’t taken within a certain time frame around sunrise and sunset, it’s deemed invalid.

5.1.4 Design languages

This code is part of the core functionality of the Aerolyzer Lbirary and as such it’s written in python.

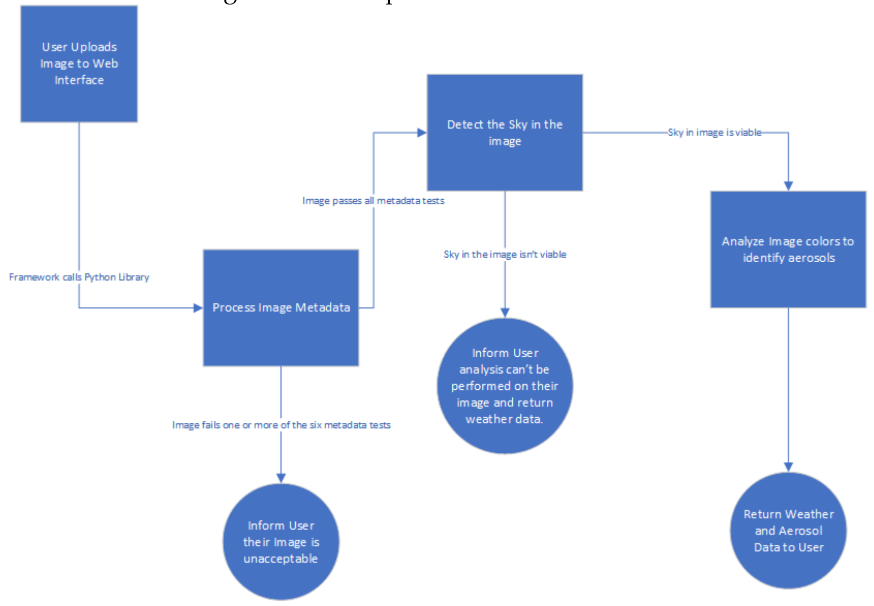
5.2 Horizon Detection Filter

5.2.1 Design Elements - OpenCV

5.2.1.1 Design Attributes: Outlined in the Software Design Document, a horizon detection filter shall be created using OpenCV and other python libraries as required. OpenCV is a BSD-licensed computer-vision library with many relevant sub-modules. Containing over 2,500 algorithms, OpenCV can analyze images and videos, recognize faces, identify objects, perform visual transformations, and more. One can use image transformations to normalize image data such that a classifier can more easily distinguish what is in a given image. [2]

Logan Wingard wrote the function is_landscape to identify the largest changes in color across the rows of an image. Ongoing work on the project can be viewed in the Aerolyzer librarys open-source Github.

5.2.1.2 Design Relationships: .



User Uploads Image to Web Interface
Detect the Sky in the image
Sky in image is viable
Image passes all metadata tests
Sky in the image isn't viable
Analyze Image colors to identify aerosols
Framework calls Python Library
Process Image Metadata
Inform User analysis can't be performed on their image and return weather data.
Image fails one or more of the six metadata tests
Inform User their Image is unacceptable
Return Weather and Aerosol Data to User

5.2.1.3 Design Constraints: Aerolyzers unique image criterion necessitates a predictive data model capable of detecting horizons with a minimum of 66% certainty and identifying aerosols with a minimum of 50% certainty. Efficient and expedient algorithm performance is necessary due to the expected user base who will be waiting for Aerolyzer to service their request.

5.2.2 Design overlays

The horizon filter produces an image that is the portion of the original image containing what is most likely to be the horizon. Then using this subimage the filter checks that the change in color intensity, which in this case is a measure of brightness, across the rows of an image is significant enough to be considered a horizon. If an image has objects in the foreground that obscure a horizon, the objects’ color will interfere with the overall drop in color intensity and make it more likely that the image will not be considered valid.

The process starts by creating a histogram of the color intensities across all of the pixels of the image. By finding the sum of these values across a row of the image, an average value for each color in each row is found. Then looking over every row's average color intensity and comparing it to the last, the largest drop in intensity in the image is found. If the intensity drop isn't large enough the image is considered invalid since there isn't a disguished horizon. If the intensity drop is large enough, the portion of the image around that drop is trimmed and isolated for analysis.

5.2.3 Design Rationale

Any image of the horizon will have a distinct drop in brightness where the sky ends and the ground begins. This is especially true when the image is taken during sunset and sunrise which is confirmed by the time and location of the image.

5.2.4 Design languages

An element in the greater Aerolyzer python library, the horizon detection algorithm shall be written in Python 2.7.12 while importing OpenCV 3.3.0 and Numpy 1.13.3. The Aerolyzer projects sponsor, Dr. Whitehall, chose Python as the librarys primary language.

5.3 Library Structure

5.3.1 Design Elements

5.3.1.1 Design Attributes: One goal of Aerolyzer is to create an accessible python package capable of being imported and giving access to all of the horizon detecting and color analyzing software written when creating a functioning Aerolyzer mobile web application. This will include an Apache licensed python script setup.py that ensures required packages such as numpy 1.8.0, exifread, opencv-python, and pyyaml are installed. The structure of this library is important to the usability of the library. To maximize simplicity and usability of the Aerolyzer library, the semantics of the python init file need to make it clear where each function is from. Aerolyzer is designed such that import aerolyzer will be the needed import to access the aerolyzer functions later on in the code by calling aerolyzer.¡function-name¿(). In the Aerolyzer package, __init__.py is a python file that determines how a module is imported. Aerolyzers __init__.py contains imports of the python scripts that supply Aerolyzer functionality. Setup.py is a python file that determines which packages will be installed when downloading the Aerolyzer module. The code for these files can be seen here. __init__.py

```
version = "0.0.0.4"
import image_restriction_functions
import image_restriction_main
import retrieve_image_data
from wunderData import *
from horizon import *
```

setup.py (The line that sets the required packages)

```
_requirements = ['exifread', 'numpy>=1.8.0', 'opencv-python', 'pyyaml',]
```

5.3.1.2 Design Relationships: When importing Aerolyzer, the __init__.py script executes and imports all python scripts with Aerolyzer functionality, including wunderground data script, image restriction script, image analysis script, etc.

5.3.1.3 Design Constraints: The setup script for importing Aerolyzer is licensed under the Apache License, Version 2.0 and is constrained to the compliance of said licence.

5.3.2 Design overlays

The Aerolyzer library is Under the Apache Licence Version 2.0 and follows Apaches permissions and limitations.

5.3.3 Design rationale

The choice to use pythons package index is to keep the Aerolyzer library open source enabling software developers, data analysts, etc. to access the Aerolyzer library and use its image analyzing software for their own research needs.

5.3.4 Design languages

The language used will be python 2.7.12 to write a setup script for when importing Aerolyzer.

5.4 Color Analysis

5.4.1 Design Elements

5.4.1.1 Design Attributes: The color analysis of an image is the most time intensive function of the Aerolyzer library.

5.4.1.2 Design Relationships: This functionality of the library requires that both the metadata preprocessing and the horizon filter functions have been passed successfully.

5.4.1.3 Design Constraints: The target accuracy is 50This will compare aerosol data from outside sources to the aerosol data produced by the library.

5.4.2 Design overlays

The color analysis of an image happens in three steps. First, 1000 random pixels are selected as a sample from the horizon subimage provided by the horizon filter. Second, Each of these pixels' hue is converted into a wavelength. This wavelength is indicative of the the aerosols in the air that the light is passing through due to the rayleigh scattering effect. The wavelength is found by comparing the hue of the pixel to an image of the visible color spectrum, which has been mapped to a range of wavelengths. Each wavelength is passed to a function that returns a likelihood, a p-score based on a normal distribution over an aerosol's expected size, that an aerosol of a certain type scattered the light captured in that pixel. The scores are then stored in an array for that pixel. Third, the scores for each type of aerosol are summed and sorted from greatest to least. This shows what aerosol is most likely to have caused the colors in the image.

5.4.3 Design rationale

The rayleigh scattering effect is the most direct relation between the size of particles and their effect on light. At this time the sample of pixels is 1000 pixels large because the wavelength conversion and aerosol comparison takes 4/1000 seconds per pixel.

5.4.4 Design languages

This portion of the library makes calls to the OpenCV library and the remainder is written in python.