Requirements

Totality AweSun

Bret Lorimore, Jacob Fenger, George Harder

November 4th, 2016

CS 461 - Fall 2016

Abstract

On August 21, 2017 a total solar eclipse will pass over the United States. The path of totality will stretch from Oregon to South Carolina. There has not been a total solar eclipse like this, crossing the country from coast to coast, since the eclipse of 1918. The Eclipse Megamovie Project is a collaboration between Google and scientists from UC Berkeley and several other institutions with the aim of compiling a large dataset of eclipse observations. Acquiring coronal data is of particular interest as the corona is not normally visible from Earth. Specifically, the project will crowdsource photos of the eclipse from photographers at various locations along the path of totality. These images will be aligned spatially and temporally and stitched into a unique movie that shows the eclipse over a period of 1.5 hours as it passes across the United States. Additionally, the complete photo dataset will be open sourced so that independent researchers may do their own analysis.

Google will contribute applications providing, among other things, backend image processing, photo upload capabilities, and static informational content. This senior capstone project will consist of two distinct sub-projects, specifically, improving/implementing an image processing algorithm facilitating the classification and alignment of solar eclipse images before they are stitched into a movie, and a location-based eclipse simulator.

David Konerding, Project Sponsor	Date
Bret Lorimore	Date
George Harder	Date
Jacob Fenger	Date

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Purpose

The purpose of this software requirements specification (SRS) is to describe in detail the Eclipse Image Processor and Eclipse Simulator that our group will produce. By writing these requirements down and agreeing to them with our sponsor both parties will have a clear understanding of what the finished product will be and what it will be able to do. The intended audience for this SRS is our sponsor, the Senior Capstone Instruction Team, and ourselves.

1.2 Scope

We are producing two products: an Eclipse Image Processor and an Eclipse Simulator. The Eclipse Image Processor will take images, find the Sun and the Moon in the images, crop the image around these bodies, and extract information about the relative position of the Sun and the Moon in these images. The Eclipse Simulator will provide users with a 2D visual representation of the eclipse from a specified location from 12 hours before it occurs to 12 hours after.

1.3 Definitions, acronyms, and abbreviations

Eclipse Megamovie Project: The Eclipse Megamovie Project is a collaboration between Google and scientists from Berkeley and several other institutions with the aim of collecting large quantities of observations of the solar eclipse that will pass over the United States on August 21, 2017. The project will crowdsource photos of the eclipse from photographers at various points along the path of totality.

EXIF: EXIF refers to the Exchangeable Image File Format, a standard media file format. Ancillary data tags associated with other media files are frequently referred to as ”EXIF fields”, ”EXIF data”, etc. This is how ”EXIF” will be used within the scope of this document.

GPS: GPS stands for global positioning system. We will refer to the latitude and longitude information obtained from a global positioning system as ”GPS coordinates”, ”GPS data”, etc. throughout this document.

JPEG/JPG: JPEG is a lossy compression technique for images. When we refer to JPEG/JPG files in this document we are referring to image files compressed in this method with the .jpeg or .jpg file extension.

PNG: PNG refers to the Portable Network Graphics image file format. Images in the PNG format are frequently referred to as ”PNGs” and are saved with the .png file extension.

1.4 References

This SRS makes reference to a report by Larisza D. Krista and Scott W. McIntosh titled ”The Standardisation and Sequencing of Solar Eclipse Images for the Eclipse Megamovie Project.” This technical report was produced as a

collaboration between scientists at the University of Colorado at Boulder, the National Center for Atmospheric Research and the National Oceanic and Atmospheric Administration. This paper can be found on arxiv.org.

1.5 Overview

The remainder of this SRS contains an overall description of the Eclipse Image Processor and Eclipse Simulator systems in section 2. Following these descriptions are specific requirements for the systems in section 3.

2 Overall Description

2.1 Product perspective

1) These products, the Eclipse Image Processor and the Eclipse Simulator, are both components of the larger Eclipse Megamovie Project. They are designed to operate as wholly independent modules that can be ”plugged into” the existing Eclipse Megamovie codebase. The Eclipse Image Processor will be a binary that receives images from another application, processes them, and exports them. The Eclipse Simulator will be a standalone JavaScript module that can be added to an existing webpage.
2) The Eclipse Image Processor does not directly interface with the user. It resides behind the front-end of the Eclipse Megamovie website. The Eclipse Simulator does interface directly with the user. It should function on most modern internet browsers (Chrome, Firefox, Safari). The simulator will appear to the user as a 2D animated depiction of the Sun and the Moon as they appear at the specified time and location. The simulator will also have background imagery in addition to the Sun and the Moon. Besides the images, the simulator will have a time slider, a location input, and a time display.
3) This system does not interface with hardware.
4) The Eclipse Image Processor will be designed to work on Ubuntu 16.04. It is necessary for the image processor to work on this operating system because the machines that will be running the processor use Ubuntu 16.04. The Eclipse Simulator is a JavaScript module that will work on modern browsers like Chrome, Firefox and Safari. We expect our users will use these popular browsers so it is necessary for our product to interface with them.

2.2 Product functions

1) The Eclipse Image Processor application will ingest photos of the eclipse, align them spatially, and categorize them to help recover temporal ordering. These aligned and categorized images will be cropped so that the Sun occupies the same amount of space in each image and will be exported. All data that is required to take the raw input images and produce the exported images will be saved to a data file. Additionally, select EXIF information will be extracted from the image files and included in this output data file.
2) The Eclipse Simulator will be a standalone JavaScript module enabling users to ”preview” the eclipse. It will be designed in a stylized, 2D manner. The simulator will incorporate a time slider that allows users to simulate the eclipse in a time window spanning from 12 hours before the eclipse to 12 hours after it. As users drag the time slider, the eclipse will animate in the simulator window. The view of the eclipse which users are presented will be specific to the selected location.

2.3 User characteristics

1) The Eclipse Image Processor application will be used by Google Engineers.
2) The Eclipse Simulator application will be used by the general public. No unusual technical/scientific knowledge is expected of these users. It is assumed however, that these users are familiar with the internet and web browsers.

2.4 Constraints

None.

2.5 Assumptions and dependencies

1) This SRS assumes the availability of Ubuntu 16.04.
2) This SRS assumes the availability of Google Cloud Platform n1- standard-4 virtual machines.
3) This SRS assumes the availability of the OpenCV computer vision library.

2.6 Apportioning of requirements

See Gantt Chart in Appendix.

3 SPECIFIC REQUIREMENTS

3.1 External Interfaces

3.1.1 Eclipse Simulator

1) The simulator is a standalone JavaScript module that can be included on an existing webpage.
2) Users can select the location from which to simulate the eclipse. This can be entered at any point while using the simulator.
   a) Location can be entered as: latitude/longitude, address, zip code, city name, state name.
   b) Initial simulator location can be programmatically set as initialization parameter.
3) Users will be able to adjust the simulator time from 12 before the eclipse to 12 hours after it.
   a) Time can be advanced via a draggable slider or clickable buttons.

3.1.2 Eclipse Image Processor

1) The image pre-processor will be compatible with Ubuntu 16.04 and will include a script that to install all dependencies/build the binary.
2) The application will accept the following input as command line arguments:
   a) Required: image_list_file
      i) Absolute or relative (to the directory the binary was invoked from) path to file containing a list of image filenames with no directory prefix.
   b) Required: output_dir
      i) Directory to write output files to.
   c) Optional: image_path_prefix
      i) Absolute or relative (to the directory the binary was invoked from) path to prepend to each image filename in image_file_list. Defaults to "./".
3) The application will accept JPEG (.jpeg/.jpg) and PNG (.png) image files.
   a) Images of an invalid format will be disregarded and an error message will be written to stderr.
4) The application will write the following output to the output_dir directory:
   a) image_transformations.txt
      i) File containing one line per image processed with the following values (comma separated):
         A) processed_image: processed image filename
         B) image_type: image type (FULL_DISK/TOTALITY/etc.), see requirement #2 of 3.2.2
         C) rot_angle: angle original image was rotated (degrees)
         D) crop_topl_x: x coordinate of top left corner of cropped image (refers to rotated image)
         E) crop_topl_y: y coordinate of top left corner of cropped image (refers to rotated image)
         F) crop_botr_x: x coordinate of bottom right corner of cropped image (refers to rotated image)
         G) crop_botr_y: y coordinate of bottom right corner of cropped image (refers to rotated image)

H) rel_center_offset: relative offset of solar/lunar disk centers, see requirement #3 of 3.2.2 (optional, only included for CRESCENT type images)

I) diamond_rel_size: size of diamond relative to the size of the solar disk, see requirement #4 of 3.2.2 (optional, only for DIAMOND_RING type images)

J) timestamp: timestamp at which image was taken (optional, only included when images have an EXIF timestamp field)

K) eclipse_path_percent: percentage through eclipse totality path at which image was taken, see requirement #8 of 3.2.2 (optional, only included for images with GPS coordinate EXIF field)

b) Processed image files

i) Processed image files will be saved into a sub-directory of output_dir called "images" and will be named as follows: *_pp.[png|jpeg|jpg].

5) Images that are rejected will not be added to the image_transformations.txt file and will not be cropped/exported to the images directory. When an image is rejected a message with this information will be printed to stdout.

6) The application will format all log messages as follows:

a) img_preproc:level:timestamp:message

i) Values of level: ERROR/WARNING/INFO/DEBUG

ii) Value of timestamp: current timestamp

iii) Value of message: specific logging message

3.2 Functional Requirements

3.2.1 Eclipse Simulator

1) Displayed solar/lunar placement will be based on location and time and will account for edge cases like when the location is not in the path of totality. For example, if the location is on the opposite side of the world as the eclipse, the simulator should shift to a night time display.

2) Simulator will display the local time that the simulator is set to, e.g. there is a well defined time associated with the user selecting Corvallis, Oregon as their location and a simulator time of -3:13 (3 hours 13 minutes) before the eclipse. This time should be displayed on the simulator.

3.2.2 Eclipse Image Processor

1) Invalid JPEG and PNG files (e.g. cannot be opened by OpenCV, width/height equal to 0px, etc.) will be ignored and an error message will be written to stderr.

2) The application will classify the input images as being one of the following types:

a) FULL_DISK

i) Image of an unobscured solar disk.

b) TOTALITY

i) Image of a total solar eclipse.

c) CRESCENT

i) Image of a partially eclipsed Sun, creating a ”crescent” shape.

d) DIAMOND_RING

i) Image of a nearly fully eclipsed Sun where there is one ”hot spot” on the Sun’s perimeter. This hot spot along with the Sun’s perimeter have the shape of a diamond ring.

3) For images of type CRESCENT, the application will compute/export a delta of the position of the center of the Sun and the Moon relative to the size of the solar disk. This delta will be a signed value based on the Sun’s position, i.e. if the Moon is to the left of the Sun (in the cropped/rotated image) the delta will be negative and conversely, if the Moon is to the right of the Sun the delta value will be positive.
4) For images of type DIAMOND_RING, the application will compute/export the size of the ”diamond” relative to the size of the solar disk.
5) Images where the solar disk has a radius of less than 50px will be rejected (see requirement #5 in 3.1.2).
6) The application will crop the images to be square with the Sun centered. The images will be cropped so that there is a 100px pad between the solar perimeter and the edge of the image on all sides.

a) Images that do not have enough room around the Sun to allow for the pad described above will be rejected, see requirement #5 in 3.1.2.

7) The application will rotate DIAMOND_RING and CRESCENT type images so that they are aligned horizontally, as described by Krista et al.
8) For images with GPS EXIF information, the application will compute/export the percentage through the eclipse’s path of totality at which the image was taken. 0% will be defined as the westmost point on that path of totality that is over land, this point is on the west coast of Oregon. 100% will be defined as the eastmost point on the path of totality that is over land, this point is on the east coast of South Carolina. The application will compute the point on the path of totality nearest the point where the image was taken. This point will be used to compute the percentage through the path of totality at which the image was taken.
9) Images with GPS EXIF information that are not on the path of totality will be rejected, see requirement #5 in 3.1.2.

3.3 Performance Requirements

3.3.1 Eclipse Simulator

1) All simulator resources will load in less than 500ms given a 1-10 Mbps internet connection.

3.3.2 Eclipse Image Processor

1) The application should take less than 5 seconds to process an image when running on a Google Cloud Platform n1-standard-4 virtual machine.

4 SUPPORTING INFORMATION

4.1 Appendix

1) Figure 1. Project Gantt Chart

ID	Task Name	Start	Finish	Duration
1	Eclipse Simulator: Basic UI	11/14/2016	11/25/2016	2w
2	Eclipse Simulator: Google Maps API Integration	11/21/2016	11/25/2016	1w
3	Eclipse Simulator: Sun and Moon Position Calculations	11/14/2016	12/2/2016	3w
4	Eclipse Simulator: Connect User Interface with Model	12/5/2016	12/16/2016	2w
5	Eclipse Simulator: UI Fit and Finish	1/9/2017	1/20/2017	2w
6	Eclipse Image Processor: Write Scripts for Application Build and Installation	1/9/2017	1/13/2017	1w
7	Eclipse Image Processor: Command Line Input and File Parsing	1/9/2017	1/13/2017	1w
8	Eclipse Image Processor: Output Writing and Formatting	1/9/2017	1/13/2017	1w
9	Eclipse Image Processor: Rough End to End Pipeline Implementation	1/16/2017	2/17/2017	5w
10	Eclipse Image Processor: Standard Logging Functionality	1/9/2017	1/13/2017	1w
11	Eclipse Image Processor: Accuracy and Speed Optimization	2/17/2017	4/27/2017	10w
12	Eclipse Image Processor: EXIF and GPS Timestamp Processing	3/6/2017	3/17/2017	2w
13				
14				
15