2017 Solar Eclipse

Bret Lorimore, Jacob Fenger, George Harder

October 26th, 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 spatial/temporal alignment of solar eclipse images before they are stitched into a movie, and a location-based eclipse simulator.

I. Problem Definition

A. Eclipse Image Pre-Processor

Solar eclipses offer researchers a rare chance to observe the Sun's corona. Normally only visible with highly specialized instrumentation, it becomes easily visible during a solar eclipse. The total solar eclipse in 2017 will offer solar researchers a unique opportunity to enlist the help of photographers across the United States to gather images of the Sun and its corona. In preparation, scientists are collaborating with Google to build the infrastructure to collect these eclipse observations and make them available to researchers.

In addition to collecting these raw images, one of the focuses of the Eclipse Megamovie project is to stitch them into a movie of the eclipse as it passes over the US. This data will enable researchers to observe the Sun's corona over an unprecedented time window. In order to stitch these images into a movie, they must be ordered temporally and resized/aligned spatially. The combination of a wide variety of locations and angles at which the crowdsourced images will be taken, the lack of reliable GPS and time-stamp data on most modern DSLR cameras, and the sheer volume of images that will be collected makes this a non-trivial problem.

B. Eclipse Simulator

The eclipse's path of totality is small relative to the continental U.S. As such, it is of interest to photographers and the public, what the eclipse will look like from a given location. One of the primary goals of the Eclipse Megamovie website eclipsemega.movie, is to give users access to information about this specific eclipse and eclipses in general. Enabling users to preview the eclipse will increase engagement before the eclipse and enable users to better select a viewing location.

II. Proposed Solution

A. Eclipse Image Pre-Processor

The eclipse image classifier application will ingest photos of the eclipse and align them spatially and temporally so that they are ready to be stitched into movies. With images sourced from photographers all with different camera equipment/configurations and with no guarantees of accurate timestamp and GPS coordinate data, this problem is challenging.

We plan to create a standalone C++ application using existing image processing implementations, including OpenCV and other libraries wherever possible. Specifically, this application will ingest eclipse images, perform the processing described above, and export them along with all meta-data required to compile them into movies. This application will not perform any of the stitching itself, nor will it interface with Google Cloud Storage, where the eclipse images are stored - it will simply be a binary that can be invoked by some larger system developed by Google.

B. Eclipse Simulator

To address the desire for users to be able to preview the eclipse, we will implement a JavaScript eclipse simulator as a standalone widget. It will be designed in a stylized, 2D manner. The visual design will be provided by Google along with all the visual assets necessary to build it - including images, SVG graphics, etc. 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 a scientifically accurate fashion.

III. Performance Metrics

A. *Eclipse Image Pre-Processor*

Given the large quantity of images users will upload to the site, each image must be processed accurately and in a reasonable amount of time. Each image should not take more than 5 seconds for pre-processing. While one second per image is ideal, quality is of high value for this project and we will likely face a performance-quality tradeoff.

From a quality perspective, the pre-processor must identify solar/lunar disks with 95-98% accuracy. It must also be able to classify images as being of full-disk/crescent/diamond-ring/total eclipses with 95% accuracy. These accuracy metrics will be computed from a set of hand labeled, golden data. Achieving these accuracy metrics is key to ensure the movie that is exported is smooth and accurate.

B. *Eclipse Simulator*

There are several metrics that will be used to evaluate whether our eclipse simulator a useable solution. These metrics deal with timing and the speed at which the simulator responds to user interactions. First, we want the simulator to load within 150 milliseconds. This quick load time makes users more likely to use our simulator and gives them a positive impression of the site. Next, our goal is for the simulator to run at 60 frames per second (fps). This metric is slightly hardware dependant. As such, frame rates down to 30fps are acceptable on slow hardware. Frame rates below 30fps are unacceptable.

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