University of Nevada, Reno

CSEWIS: Cloud and Snow Estimates from Webcam Image Streams

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science
in Computer Science and Engineering

by

Hannah Jane Muñoz

Dr. Sergiu Dascalu, Thesis Co-Advisor
Dr. Frederick C. Harris, Jr., Thesis Co-Advisor

August, 2018

THE GRADUATE SCHOOL

We recommend that the thesis
prepared under our supervision by

HANNAH J. MUNOZ

Entitled

CSEWIS: Cloud and Snow Estimates from Webcam Image Streams

be accepted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

Sergiu M. Dascalu, Ph.D., Advisor

Frederick C. Harris, Jr., Ph.D., Co-advisor

Scotty Strachan, Ph.D, Graduate School Representative

David W. Zeh, Ph.D., Dean, Graduate School

August, 2018

Abstract

Environmental monitoring is often accompanied by inexpensive camera technology to obtain automatic time series of of site conditions. However, the resulting pictures will often go unused for advanced analysis because of the lack of fast and reliable ways to quantify data from images.

This thesis presents CSEWIS, an image analysis software that aims to help fill the gap of missing analysis software by delivering accurate, repeatable coverage estimates of snow or clouds for use in environmental research. CSEWIS uses an image series from remote sensor networks and allows users to select regions of interest within the image and perform snow or cloud coverage analysis. Analysis algorithms can be done on the CPU, GPU, or on multiple GPUs for faster execution times. Results are delivered as a CSV file with a related XML metadata document. An optional AVI video file of the coverage can also be exported. By producing quantifiable results, CSEWIS aims to increase the usability of webcams commonly used in sensor networks.

This thesis details the background behind image processing, remote sensing, and webcams, overviews the system architecture, and how to use the resulting product. A comparison to related works and future work on the project are discussed.

Dedication

To my wonderful family who supported me throughout college and my fiancé who gives me the confidence and support to succeed.

Acknowledgments

I would like to thank my advisors, Dr. Sergiu M. Dascalu and Dr. Frederick C. Harris, Jr., and my committee member Dr. Scotty Strachan for their time and suggestions. They have helped and supported me throughout the development of this project and it would not have been possible without them.

This material is based in part upon work supported by the National Science Foundation under grant number IIA-1301726. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Webcams can be of great use in environmental studies but have not always been included in sensor networks. In fact, their addition to many networks is fairly recent. Typically sensor networks are a collection of different types of electronic sensors used to collect a variety of information about a local area. Connected sensors send data streams back from remote research sites to scientists to be interpreted as part of science investigations [37]. Cameras can be paired with other sensors in an array to ensure all sensors are working correctly and give visual context to other data streams. In the past, webcams were expensive and did not provide high-resolution photos. Advancements in modern day cameras has decreased the price and increased the resolution [4]. Since prices have dropped, webcams have become increasingly common in sensor networks. Images from webcams allow scientists to look at their test sites to make qualitative, visual analysis, like visually tracking environmental changes in landscape, plant growth, traffic density, and weather changes [4, 19].

Often, webcam images will go unused for more in-depth analysis as there has been little development of applications to quantify image information. As there are potentially many different uses for the images, there is a need for different types of scientific image analyzing software. Most research on utilizing images for environmental sensing goes into analyzing satellite images. Satellite images are good for large areas of land or clouds [20]. However, satellite images are not particularly useful at fine scales of time or space. For example, in times of harsh weather conditions or for smaller areas unable to be captured by satellite camera resolution. This is where

ground based webcams can be useful [20].

Current software available for environmental image analysis is usually sub-domain specific or difficult to use. Software that currently exists often requires users to be proficient in programming software like MATLAB or R. The need for coding knowledge can discourage users with no coding background.

CSEWIS expands the use of time series image analysis to new sub-domains within environmental science. CSEWIS allows users to quantify cloud and snow coverage within user-designated regions of interest within images. CSEWIS keeps its usage simple enough that no coding experience or knowledge of machine learning techniques is necessary.

Example use cases include calculating cloud coverage for renewable energies research. The cloud coverage determines the radiative balance in the atmosphere, which is used for transfer models [43]. Current satellites do not have a temporal resolution high enough to identify average coverage of clouds, which results in miscalculated coverage estimates. Tracking snow coverage helps determine snow water equivalent and surface albedo [3]. In high mountain areas where tracking snow coverage is important, forests or clouds often obscure the ground from satellites. This is where ground-based webcams become an important addition to satellite analysis. However, the resulting images from webcams in harsh weather conditions have their own host of problems that must be corrected before using the images [12, 38].

Traditional methods of determining coverage involves human observers analyzing ground-based images and estimating coverage qualitatively [21]. Human observers are usually experts in the subject, such as hydrologists or meteorologists [25]. This manual method is time consuming and has little scienctific repeatability. Measurements can often vary between observers [3]. Human estimations usually do not come with provenance and metadata, which are key to modern science. By using CSEWIS, the coverage estimates given are consistent and can be tested for accuracy. CSEWIS results produce metadata, making the results more appropriate for use in research.

CSEWIS is designed to be a simple analysis tool for scientists interested in an-

alyzing snow and cloud cover from webcam image streams. The software uses a graphical user interface (GUI) to allow users to select time series images and define regions of interest (ROI) within them. ROI are locations within the image where clouds or snow could be. Users receive quantifiable results of snow and cloud coverage as a CSV file which can be used to validate local observations or other remote sensing data. CSEWIS is easy to use and creates repeatable and accurate results.

In this thesis, we describe the development and utility of an application that estimates snow and cloud coverage from webcam images. This thesis is structured as follows: Chapter 2 gives more background on the problem the application tries to address; Chapter 3 discusses the software design; Chapter 4 shows the implemented software; Chapter 5 discusses the development and comparison of coverage algorithms on the GPU; Chapter 6 presents works related to the software developed; Chapter 7 contains conclusions and ideas.

Chapter 2

Background

2.1 Remote Sensing

Remote sensing is the measurement of environmental properties using data from a variety of devices but mainly cameras [41]. The field of environmental sensing is very old, dating back to World War II [29]. Since then, the domain has grown into a large field with a number of applications, from modeling urban environments to estimating water flux [29].

2.1.1 Cameras in Remote Sensing

Satellite cameras can only tell us so much about variables like ground cover, as the pictures taken by their cameras can be noisy with atmospheric aerosols like water vapor. Ground webcams are used to automatically infer information about the environment, such as whether it is a cloudy or snowy day [19]. The system uses a generic supervised learning technique to estimate the relationship between streaming images and weather signals. This allows there to be minimal human interaction with the webcams while still producing usable results. Another such algorithm developed for the webcams measures spring leaf growth over a series of days. However, this algorithm does involve human interaction, albeit, a tiny bit. A user must select an area in the image for the algorithm to measure the average green values in.

Most terrain images used in remote sensor research are taken from satellite images. Satellite images are often good for mapping and analysis of large areas [33].

However, ground-based cameras can have their own, separate use in sensor networks. As webcams become cheaper, they can be added to more and more arrays to help further analysis on the study sites. There are two types of environmental webcams: long term monitoring, where images series can be years or decades of data, or short term monitoring, where the image series contains only days or months of data [4]. Webcams at remote sites send back time series images which can be used to visually confirm other sensor values in the sensor array or downloaded later. Analysis on these images are often done on an image's color channels or using near-infrared camera data [4]. Cameras used in remote sites often, but not always, maintain a consistent field of view [35]. Large shifts in the field of view can cause problems for users trying to analyze the image as the viewshed of the camera changes.

2.1.2 Coverage

Climate models create mixed results for calculating cloud cover and solar radiation [43]. Cloud cover estimates for weather reporting are still commonly determined from human observation. To alleviate the need for meteorologists manually sifting through images, image analysis can be used to quickly determine cloud coverage. Satellite imagery can also be used for cloud coverage, but analysis on these images often leads to inaccuracy at smaller scales. Satellite cameras can only tell us so much about our local environment, as the pictures taken by their cameras can be noisy with aerosols and water vapor [19]. As such, we can use webcams to compliment the use of satellite images and get a more accurate assessment of cloud coverage.

Snow coverage can be estimated using LIDAR radar or satellite imagery [20]. However, in forests or areas temporarily obscured by clouds, satellite images cannot be used [19]. Because snow coverage can vary wildly over time and space, time series data set from multiple areas are often needed to make a good assessment of snow coverage in an area. Because of these issues associated with determining snow coverage, this is left to human interpreters to manually go through images and determine snow coverage.

2.1.3 Remote Sensing Site

The Nevada Research Data Center (NRDC) manages the data from several remote sensor network projects in Nevada [5]. Six of the research sites deploy web cams as a type of sensor. The Walker Basin Hydroclimate Project's Rockland Summit research site was chosen as our data set due to the southern camera's clear view of the sky and vast portion of land without large sagebrush or forest obscuring the view [5]. The Walker Basin Hydroclimate project began in 2012. The HD webcams on site collect images hourly. The project's goal was to track changes in the Walker Basin landscape and determine how those changes would affect the surrounding areas [5]. By using image analysis to track snow and cloud coverage in an area, time spent analyzing collected data can be reduced. The project has collected over 17,000 landscape images in the Western Great Basin.

Rockland Summit has 20 different camera angles, using pan-tilt-zoom presets in a Canon VB-H41 on site. The camera is set up to take HD pictures in 60-minute intervals from 10 AM PST to 5 PM PST. While the shading in the image do change throughout the day, the time intervals at which the camera is active mean that analysis on images does not have to take the changing colors from dawn and dusk into account. Images are taken in JPEG format with a resolution of 960 x 540. Each image has a bit depth of 24 bits, meaning that there are 16.7 million color tones possible on each image [45]. The southwest camera angle that was chosen has the best field of view for our use and does not have any sun interference.

2.2 Image Processing

Image processing is a method of analyzing and changing images [42]. Commonly, it is used for improving an image's quality by manipulating pixels of a digital image. Image processing algorithms are divided into two common categories; low-level and high-level. Low-level algorithms know little about the image's context. Low-level algorithms must be applied either by a human who knows the context of image or by

a high-level algorithm. High-level algorithms try to mimic human vision by identifying objects and uses machine-learning techniques. Our project uses low-level algorithms to make the software approachable to those without a machine-learning background.

2.2.1 Color Spaces

Colors in images are split into color channels. Each pixel in an image is given a set of values equal to the level of color occurring from that color channel. Color values range from 0 to 255. A pixel with a color value, or intensity, of 0 would be completely black. Similarly, a completely white pixel would have an intensity of 255. Different color channels are paired with relating color channels that create a whole image. A collection of color channels that make up properties of an image are known as a color space.

The most common color space is RGB: red, green, and blue. The three channels are related by how light hits the surface [15]. By viewing the color channels of an image separately, we can observe where that color is most vibrant in an image. Since the RGB color space is related by light, shadows and shading of an image causes the color channel values to change dramatically. When using the RGB color space, it's important to remember that lighting difference will change a color channel's values.

Dusk and dawn dramatically affect light and therefore have a great impact on analysis. Since our cameras turn off at dusk and back on at dawn, we did not consider this. We only needed to concern ourselves with shadows on the ground that might affect results. We were able to solve this problem by using another color space, HSL, in addition to RGB.

The HSL color space contains three different components; hue, saturation, and lighting [15]. HSL is related to the RGB color space and is considered another method of separating the color channels. Instead of each channel relating to a color, each channel in HSL relates to a luminance property of the image. Hue is the angle of the color on the color wheel. Unlike other color channels, it's value ranges from 0 to 360. Saturation shows the intensity of the color, or how high the value of a single RGB

color channel is in a given area. The lighting color channel is simply the brightness of the image and is not related to the two other channels.

2.2.2 Histogram

Histograms are visualizations of different pixel intensities in an image [17]. A Histogram is a chart where the x-axis is a number of bins that you want to sort your values into. The y-axis of the graph is the number of values in a bin. Typically for images, there are 256 bins, once for each pixel intensity. By looking at an image's histogram, we can determine properties of the image, like brightness, contrast and saturation [17]. Histograms can also be used to manipulate the image itself [6]. For example, redistributing pixels among the bins equally can increase the contrast of an image.

2.2.3 OpenCV

OpenCV is a computer vision/image processing library for various languages and operating systems [27]. With over 2,500 available algorithms and 2.5 million downloads, OpenCV is widely used in many types of computer vision projects [7]. OpenCV uses matrices as a way of storing images, where each element of the matrix is a pixel in an image. Most computer vision algorithms work by analyzing pixel intensity values in an image. Since images can have thousands of pixels, computation on large images can become costly. However, OpenCV's algorithms have been highly optimized and as such have significantly outperformed other popular computer vision libraries [28].

In 2011, OpenCV introduced Graphical Processing Unit (GPU) accelerated algorithms [26]. Current goals of OpenCV GPU project is to provide a GPU computer vision framework consistent with the CPU functionality, achieve high performance with the GPU algorithms, and to complete as many algorithms as possible so image analysis can be done completely on the GPU [26]. OpenCV's GPU implementation was written using CUDA, so developers could take advantage of previously developed CUDA libraries.

Algorithms developed for GPU often differ than algorithms implemented on CPU. GPUs can do hundreds of independent calculations simultaneously, which lends to immense speedup over CPU implementations. Because of the architecture differences between CPU and GPU, there is no guarantee that algorithms implemented on the GPU will necessarily be faster than those done on the CPU [39]. There can be many factors, such as image size or algorithm technique, that could increase execution time on GPU compared to CPU. If there are any dependencies between pixels in an algorithm, such as a blurring mask, the time spent trying to sync pixel values could attribute to low performance [22].

There is currently no multiple GPU (multi-GPU) support for the GPU algorithms beyond a method to manually switch GPU devices [10]. OpenCV does not recommend using multiple GPUs on smaller images, as added overhead of data transfers between GPUs could negate any speedup achieved from using GPU. However, there is no mention on whether multiple GPUs could attain speedup on streams of images

2.3 Metadata

As data sets move into the digital age, it becomes harder to keep track of it. If data sets continuously move through labs and change format, the origin can be difficult to determine or it's data could be corrupted [11]. Metadata is a method of trying to fix that. Metadata is described as being data about data. Metadata gives context to a data set by describing information about it, such as creation date, measurement device, research labs, or dedicated metadata handling software [11, 31].

Although there is no metadata standard for all data types, some developed standards usually tend to rise above others. Popular metadata standards are sometimes written in XML because it is easily customized and machine parsed [11]. A good, well-developed metadata standard should be well documented and have their standards easily searched [4].

Currently, there is no set metadata standard for webcam imagery. To develop a robust set of metadata standards, research must be done to determine what mea-

surement factors are constant among camera types, camera specifications, and how to classify images to disseminate them among scientists [4].

Chapter 3

Software Design

3.1 Overview

CSEWIS is a software designed to easily calculate snow and cloud coverage in images. The software allows users to create regions of interest within an image and choose what type of coverage to calculate. This chapter discusses the requirements, use cases, and the design of CSEWIS.

3.2 Requirements

Functional requirements for CSEWIS are presented in Table 3.1 and non-functional requirement are discussed in Table 3.2. The functional requirements describe the desired use of CSEWIS. Non-functional requirements describe the frameworks that were used to build CSEWIS, as well as high-level desires.

Table 3.1: Functional Requirements for CSEWIS.

Functional Requirements	
Name	Description
FR1	The user shall be able to load multiple pictures for analysis
FR2	The user shall be able to view all loaded pictures
FR3	The user shall be able to view RGB color channel values of an image
FR4	The user shall be able to select and manipulate a region of interest to track coverage within the image
FR5	The user shall be able to select and manipulate multiple regions of interest within a single image stream
FR6	The user shall be able to analyze snow coverage in a region of interest
FR7	The user shall be able to analyze cloud coverage in a region of interest
FR8	The user shall be able to save their image stream coverage estimates as an AVI file with regions of interests highlighted
FR9	The user shall be able to output their coverage results to a CSV file
FR10	The user shall be able to configure user metadata and have it saved for future use
FR11	The user shall be able to view coverage results as a binary image

Table 3.2: Non-Functional Requirements for CSEWIS.

Non-Functional Requirements	
Name	Description
NFR1	The software shall run on Windows 8
NFR2	The software shall be built in Qt for the user interface
NFR3	The software shall use OpenCV for image processing
NFR4	The software shall use CUDA to reduce execution time on large image series
NFR5	The software shall support common image types
NFR6	The software shall produce accurate and repeatable results
NFR7	The software shall not need coding or machine learning knowledge to use

3.3 Use Cases

Figure 3.1 shows the Use Case Diagram for CSEWIS. The use case diagram shows how the user can interact with the software and what actions are available to them at any given moment. The user, displayed on the left hand side of the image, can perform a variety of different action within the GUI. The different actions are shown as ellipse inside the software GUI, shown as a rectangle.

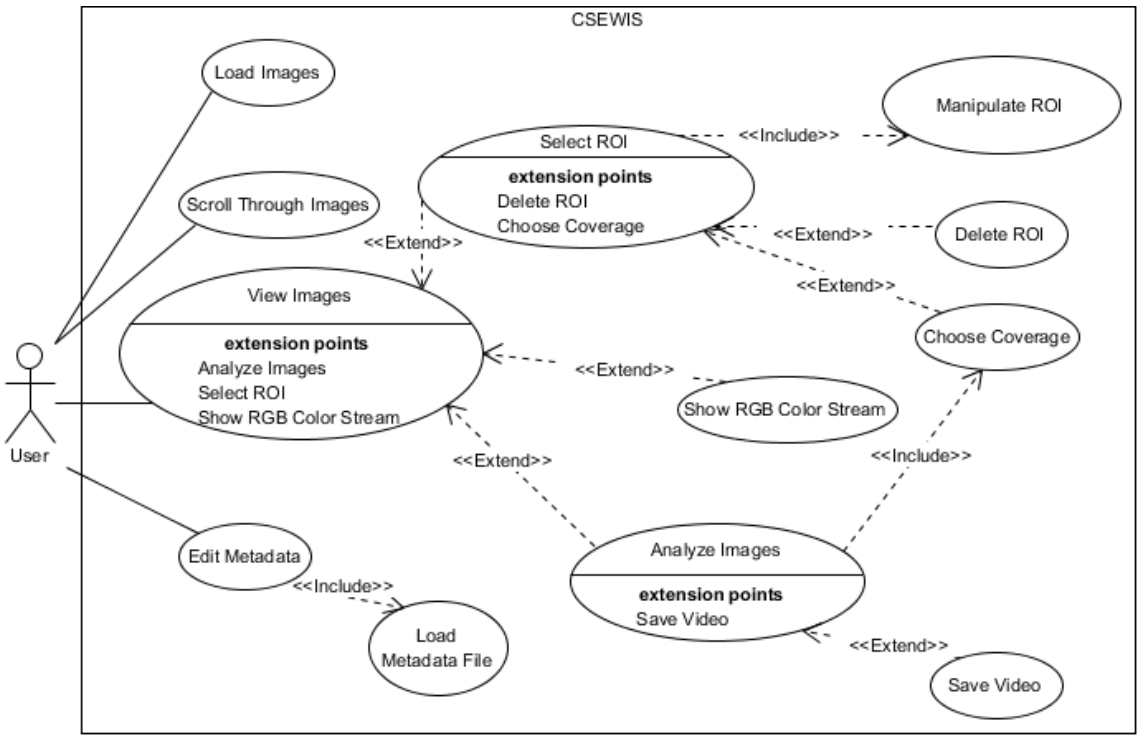

Figure 3.1: The use case diagram of CSEWIS.

In the rest of this section, we will discuss the use cases presenting Figure 3.1.

Load Images

The user will go to the menu bar item and select "Open". This will open the Windows File Explorer. The user will be able to select either a single image or multiple images for processing. The image series should be selected sequentially when possible.

Scroll Through Images

The user will scroll through the loaded images using the scroll bar at the bottom. Inside the scroll area, the user will be able to view the two previous and two next

images in the series. The current image will appear in the working area.

View Images

Once images have been loaded, the user will be able to view the images they have just loaded. One "main" image will be loaded into the main area of the screen. The user will be able to interact with this image by selecting regions of interest and viewing information about the image.

Select ROI

The user will select "ROI" from the image manipulation options. The user can then click and drag their mouse on an image to create a ROI. The ROI will be applied to all pictures loaded. There is no limit to how many ROI may be applied

Delete ROI

If the user decides they no longer want a previously created ROI, the user can delete it. The user right clicks within the ROI and selects "Delete" from the pop up menu. The ROI is removed from the image.

Choose Coverage

The user will choose what sort of coverage the image is calculating by right clicking in a ROI and choosing an available coverage method from the pop up menu.

Show RGB Color Stream

The user will view the RGB color stream by clicking in the image in the working area and selecting "Show RGB". A histogram with the red, blue, and green color channel will appear in a pop up window. The histogram reflects the image in the working area.

Analyze Images

The user will start the coverage analysis by selecting "View" from "Export" in the main window toolbar. The user will then select whether the coverage will be calculated on CPU, GPU, or multiple GPU.

Save Video

The user will save their results as a video by checking "Export Video" in the export menu. The user will select an appropriate amount of frames per second from the

spinner.

Load Metadata File

If previous metadata exists, the data will be loaded from a locally saved file. The previous information will fill a pop up window where the user can input new data or edit old metadata values.

Some of the use cases are more complicated than others. In order to clarify, two of these use cases have been provided with more details. Table 3.3 goes over the use case for "Edit Metadata". Table 3.4 goes over the use case for "Manipulate ROI".

Table 3.3: A detailed use case for "Edit Metadata".

Use Case: Edit Metadata
Brief Description: The user will go to the menu bar item and select "Metadata". A new GUI will appear with options for adding new metadata to the images. The user can then save their metadata edits or exit.
Primary Actors: User
Secondary Actors: None
Preconditions: None
Main Flow: 1. The user selects "View" from "Metadata" in the main window toolbar 2. The software creates a window with optional metadata text boxes 3. The user will fill out their desired metadata options 4. The user will select "Save" 5. The metadata window will close
Post-conditions: An XML document with the user's metadata will be created
Alternative Flows: 2.b The software will fill windows with previously saved metadata 3.b The user will edit desired metadata options
Alternative Flows: 4.b The user will select "Cancel"

Table 3.4: A detailed use case for "Manipulate ROI".

Use Case: Manipulate ROI
Brief Description: The user will select points on the ROI and drag to their desired location within the image. This allows the user to make non-rectangular ROI.
Primary Actors: User
Secondary Actors: None
Preconditions: At least one ROI must be available
Main Flow: 1. The user selects a corner edge of a ROI 2. The user drags the edge to cover the new location 3. The software adjust the ROI size to fit the user's specifications
Post-conditions: The resulting ROI will be a different size.
Alternative Flows: None

3.4 System Architecture

3.4.1 High Level Architecture

Figure 3.2 provides a high-level architecture of the system. Each subsystem is dependent on the subsystem above it. Subsystems can have users interact with them. Processes are automated and are done with minimal to no user interaction. Many of the subsystems are dependent on classes unique to Qt. To help clarify what these subsystems are and how they work, a brief description of each is given.

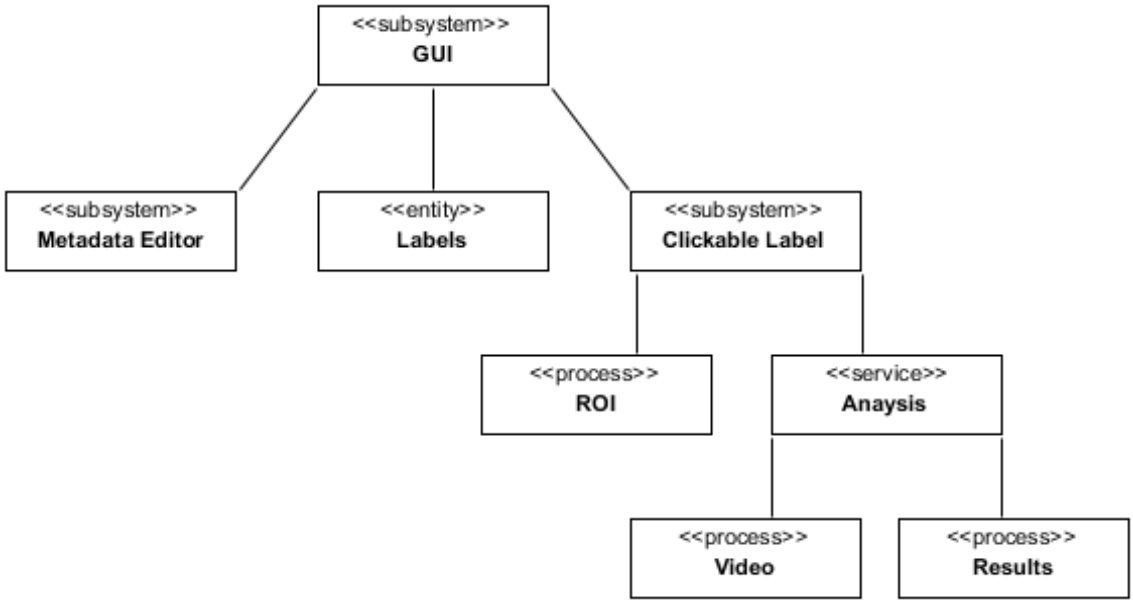

Figure 3.2: A high-level overview of the CSEWIS system architecture.

Subsystems Descriptions

GUI

The GUI subsystem is the main window of the program. It is designed using the Qt libraries. This subsystem is the main window, where the user can choose options from the Menu subsystem, view images from the Qlabel subsystem, or use the ROI process to create ROI in the ClickableLabel subsystem.

Label

Label is the subsystem that handles images within the program. Qlabel is a Qt specific class that allows for visualization of text and images. Qlabel objects do not allow mathematics operations to be performed on them. Any analysis that is done on an image must first be converted to an OpenCV matrix. Qlabel classes are uses simply for viewing images, not for interacting with.

Clickable Label

The Clickable Label subsystem creates and handles user interaction with Qlabels. This subsystem allows for more use of the Qlabel class. The subsystem detects users mouse clicks and drags making it possible to create ROI in the image.

Metadata Editor

The Metadata Editor subsystem allows the user to view and edit the metadata. This subsystem handles the creation of a new GUI interface that the user can input information into. The GUI has text lines for user information.

ROI

The ROI process creates the ROI within the Clickable Label subsystem. The process takes the positions within the Clickable Label that the mouse was clicked and released. It also handles the storage of the ROIs selected and deletion.

Analysis

The Analysis process handles the image processing of the image times series. This process calculates the snow and cloud coverage in ROIs and sends the resulting value to the Results process.

Video

The Video process outputs an AVI video file of the image series. The ROI in the image series is highlighted in red. The calculated coverage of each area is shown inside each ROI.

Results

The Results process creates a CSV file with the computational results of the Analysis process.

3.4.2 Class Diagram

The class diagram in Figure 3.3 expands on CSEWIS' design. Figure 3.3 shows the main classes of the system and how they relate to one another.

The Qt software system handles interactions between functions as a system of "signals" and "slots." Signal programs send out alerts that trigger slots functions to activate. Functions that are signals or slots are labels as such in their name.

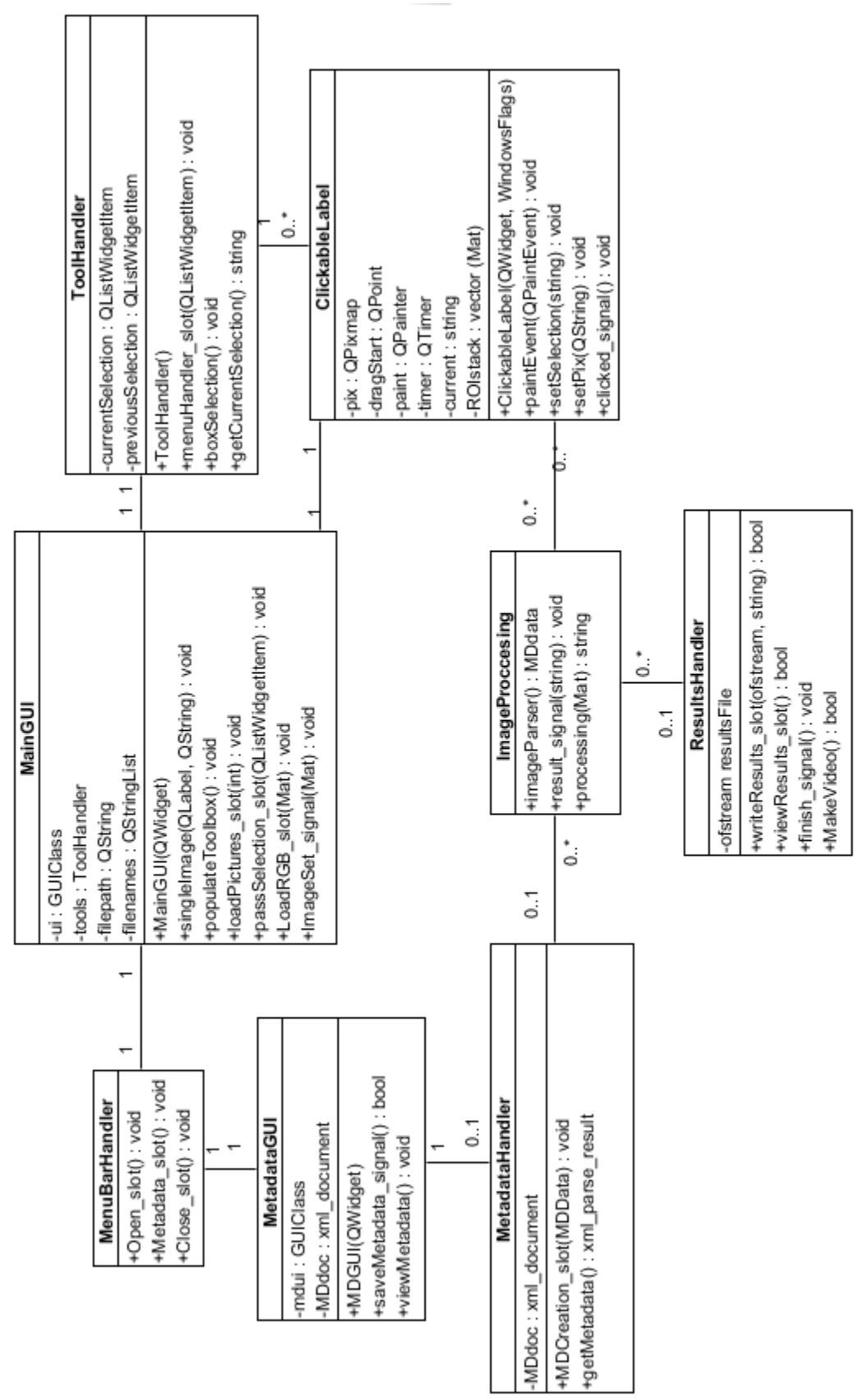

Figure 3.3: The overview of the system class architecture.

3.4.3 Data Design

The coverage results, metadata, and optional video are exported from the software as a collective file as seen in Figure 3.4.

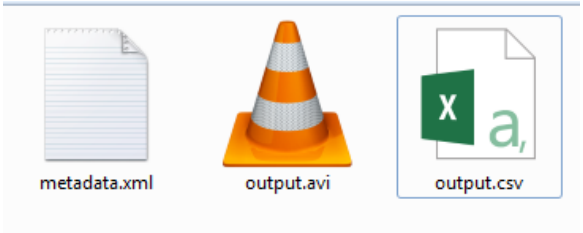

Figure 3.4: The collected results output file.

Results CSV File

The results CSV is shown in Figure 3.5. The first row shows which region of interest the resulting data in the column is from. ROI will be labeled in what order the user selected them in. The second row shows the size of the ROI and the third row, the type of coverage algorithm run. The first column is the number of the image in the series. The following columns are the resulting coverage percentage in the ROIs in the image.

	A	B	C
1	Picture	ROI 1	ROI 2
2		551 x 120	557 x 94
3	Detection Type	Cloud Coverage (%)	Snow Coverage (%)
4	Tower_Camera-Southwest-2014_05_17-09_01_52-PST.jpg	99.9319	69.1203
5	Tower_Camera-Southwest-2014_05_17-10_02_05-PST.jpg	99.9667	57.1852
6	Tower_Camera-Southwest-2014_05_17-11_02_30-PST.jpg	93.5829	52.6701
7	Tower_Camera-Southwest-2014_05_17-12_01_57-PST.jpg	96.1086	66.4712
8	Tower_Camera-Southwest--2014_05_17-13_01_50-PST.jpg	68.2063	68.5358
9	Tower_Camera-Southwest-2014_05_17-14_01_48-PST.jpg	99.5266	64.2462

Figure 3.5: The results CSV design with a row of example data

Metadata XML File

The metadata is formatted into three main parts as in Figure 3.6: The researcher metadata, the institution metadata and the project metadata. All metadata is input

from the metadata GUI window. “Researcher” contains about the software user themselves. “Institution” has information about where the researcher works. “Project” indicates where the image series has come from and from what funding.

```
<?xml version="1.0" encoding="utf-8"?>
<Researcher>
    <Name>Hannah Munoz</Name>
    <Email>hannahmunoz@nevada.unr.edu</Email>
    <Phone>775-555-1324</Phone>
    <Website>https://github.com/hannahmunoz</Website>
</Researcher>
<Institution>
    <InstitutionName>University of Nevada, Reno</InstitutionName>
    <Address>1664 N Virginia St, Reno, NV 89557</Address>
    <Department>Computer Science and Engineering</Department>
</Institution>
<Project>
    <ProjectName>Walker Basin Hydroclimate</ProjectName>
    <ProjectURL>http://sensor.nevada.edu/WalkerBasinHydro/</ProjectURL>
    <Funding>This material is based in part upon work supported by the National
    Science Foundation under grant number(s) IIA-1301726. Any opinions, findings,
    and conclusions or recommendations expressed in this material are those of
    the author(s) and do not necessarily reflect the views of the National
    Science Foundation.</Funding>
    <SiteName>Rockland Summit</SiteName>
    <Location>Northern Nevada</Location>
    <Latitude>38.650556</Latitude>
    <Longitude>-119.0925</Longitude>
    <Elevation>0</Elevation>
</Project>
```

Figure 3.6: Metadata about an image series from the Walker Basin Hydroclimate project.

Video

Selections of an output video can be seen in Figures 3.7, 3.8, and 3.9.The regions of interest are highlighted in red. The calculated coverage is displayed inside of each region of interest. Two regions of interest are shown in 3 different images from the video. In Figure 3.7 there are no clouds and no snow. The coverage estimates in both ROI show in red that there is 0% snow or cloud coverage. Similarly, Figures 3.8 and 3.9 show the same scene at different levels of snow and cloud coverage. The estimates within each region of interest changes accordingly.

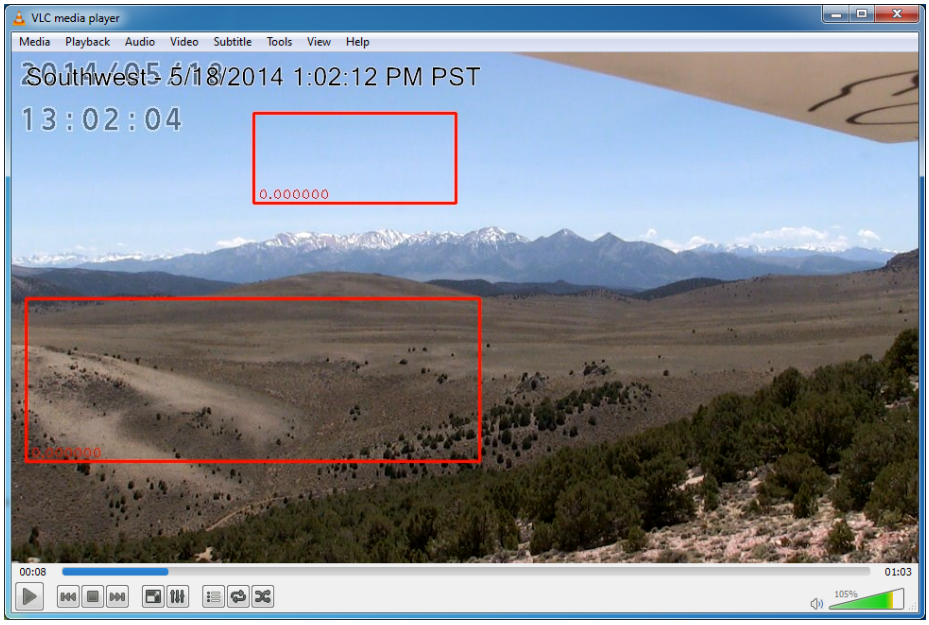

Figure 3.7: A frame from the output video with no snow or cloud coverage. The coverage results in both is 0%.

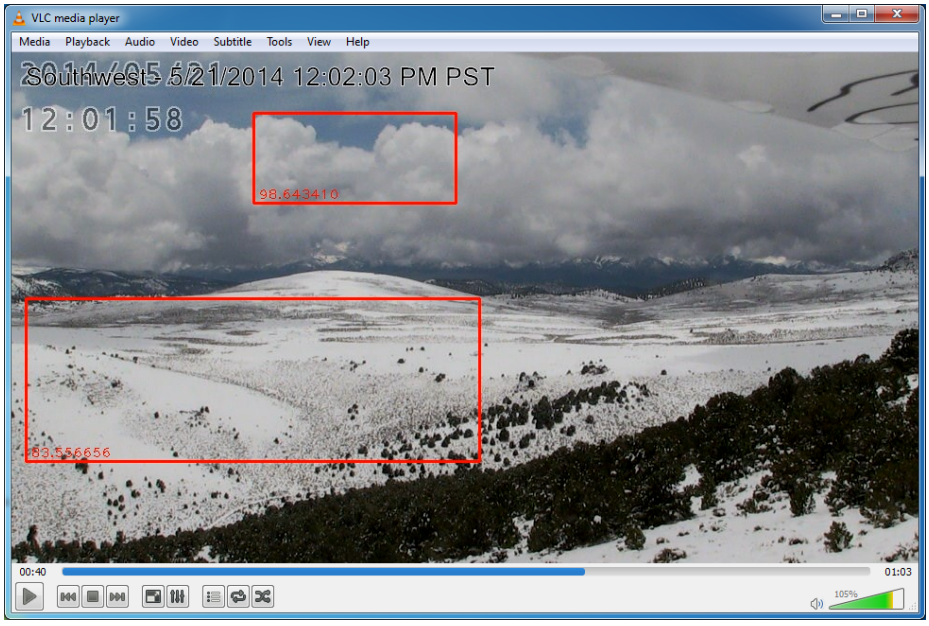

Figure 3.8: A frame from the output video with partial snow and cloud coverage. There is 98.64% cloud coverage and 83.55% snow coverage within the regions of interest.

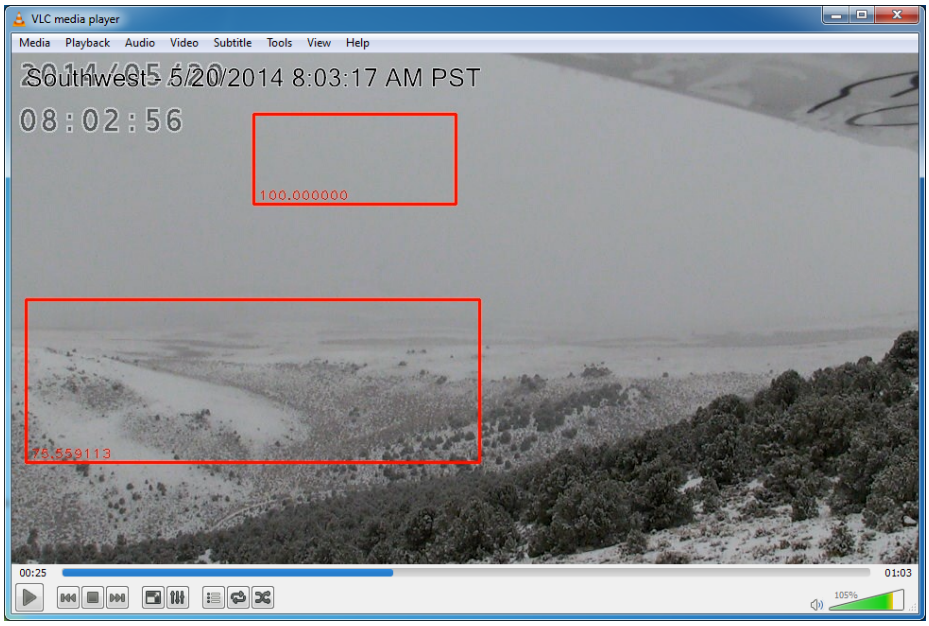

Figure 3.9: A frame from the output video with snow and cloud coverage. There is full cloud coverage at 100%. There is less snow coverage than in Figure 3.8 at 75.56% coverage.

Chapter 4

Implementation

This chapter discusses the user interface design for CSEWIS. This section has been set up to be a user manual for those interested in using the created software.

4.1 Main Screen

Figure 4.1 show CSEWIS after initial startup. The interface was kept simple to streamline the user experience, especially for users with little to no coding experience. The majority of the work can be accomplished in the main window, whose features have been numbered for further discussion.

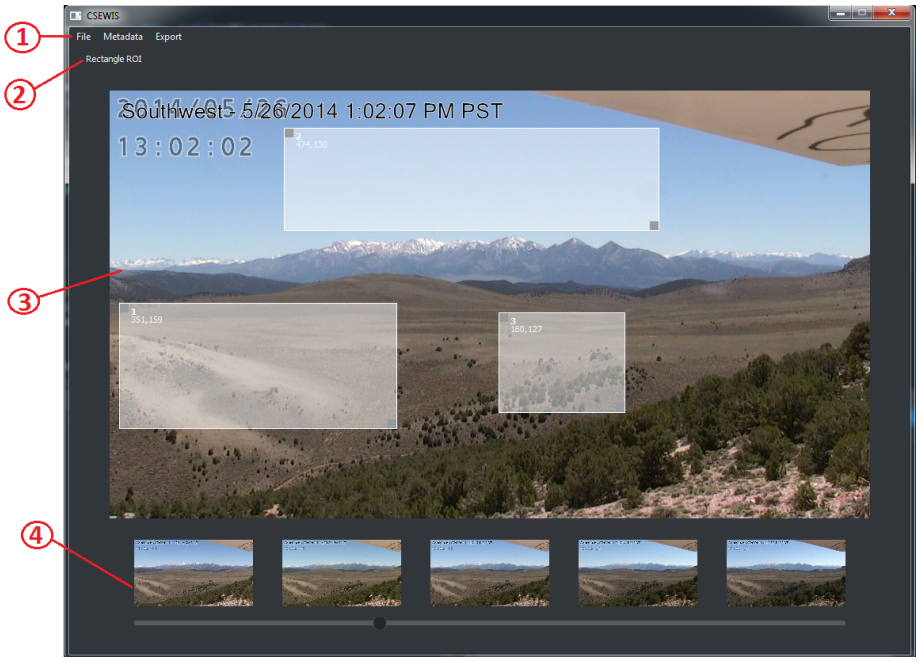

Figure 4.1: The main user screen. The numbered elements are explained below.

1. The main menu, which has three different options.

   - **File:** Allows the user to select images using their local computer file browser.
   - **Metadata:** Brings up the metadata window in Figure 4.2. Allows the user to view and edit the user's metadata
   - **Export:** Brings up the export window in Figure 4.3. Allows the user to choose algorithm execution methods and export options.

2. Allows the user to select a region of interest within the image

3. The main in Figure 4.1 working image allows the user to interaction by setting regions of interest within it.

4. The scrolling image area allows the user to scroll through the pictures within the image stream. This is helpful for locating regions where snow or cloud coverage are likely to occur.

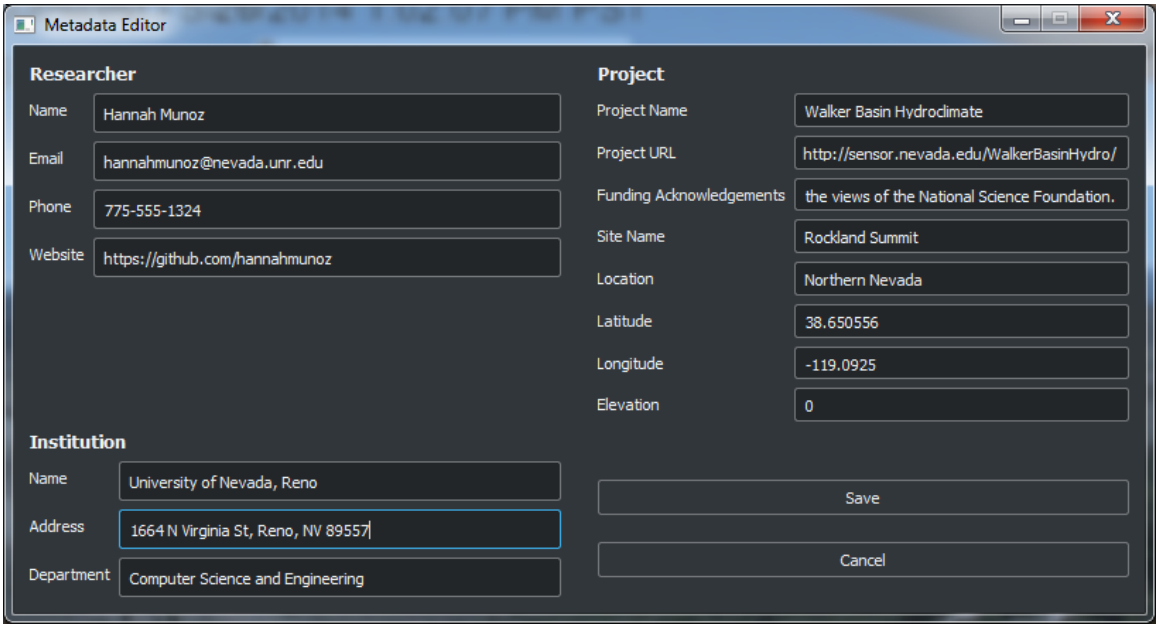

Figure 4.2: The metadata viewing and editing screen.

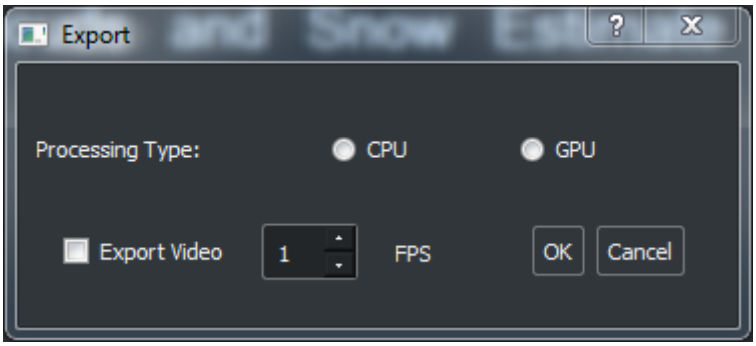

Figure 4.3: The export window, which allows the user to select the method of algorithm execution and video export options.

4.2 Histogram

After loading an image series, the user can view the histogram of the image by right clicking on the main working image. The image's histogram will appear in a new window. Histograms are useful for determining dominant colors in an image and can be used to determine how elements of scenery affect the color space. Figures 4.4 and 4.6 are two images from the same webcam. Figure 4.4 show a bright sunny day whereas Figure 4.6 shows a cloudy, snowy day. Their related histograms show that

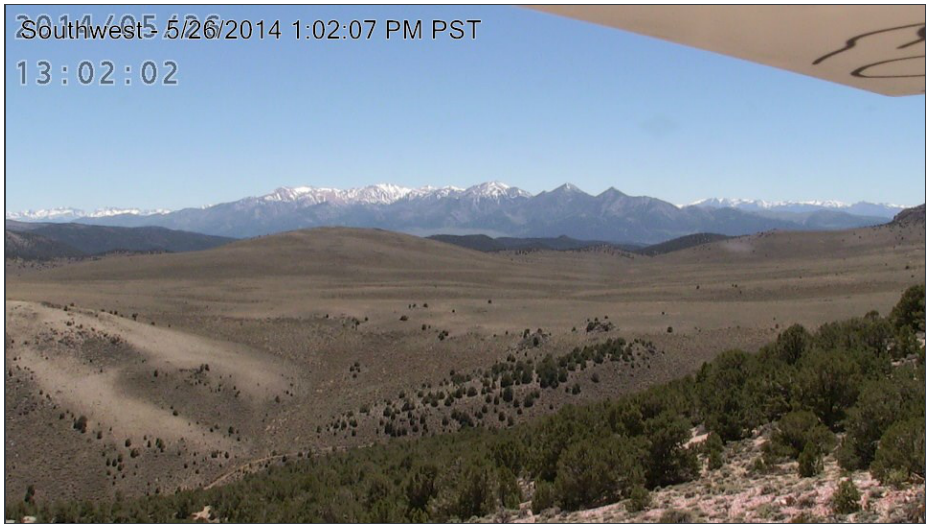

Figure 4.4: A bright sunny day at Rockland Summit. This image and its related histogram in Figure 4.5 give a good feel for what the color channel looks like on blue skies and clear ground.

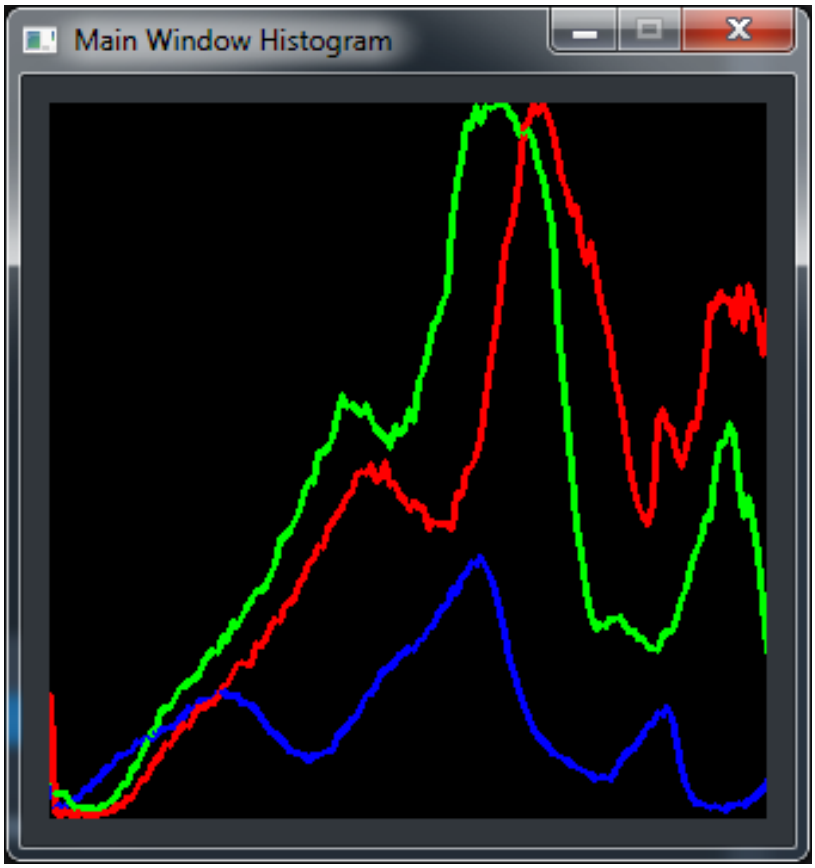

Figure 4.5: The histogram of the image in Figure 4.4. The color streams are clearly different from each other and are well spread out among the bins.

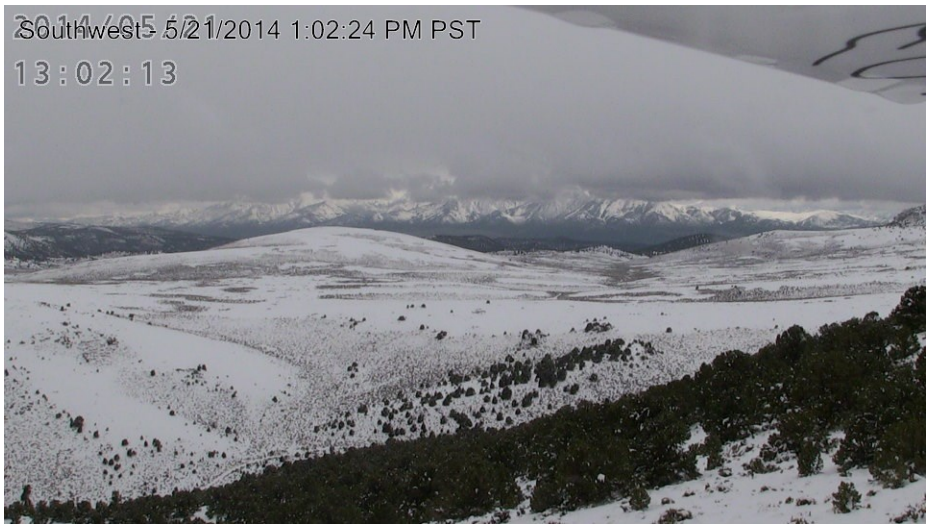

Figure 4.6: An overcast, snowy day at Rockland Summit. Its related histogram in Figure 4.7 shows the color space of the image.

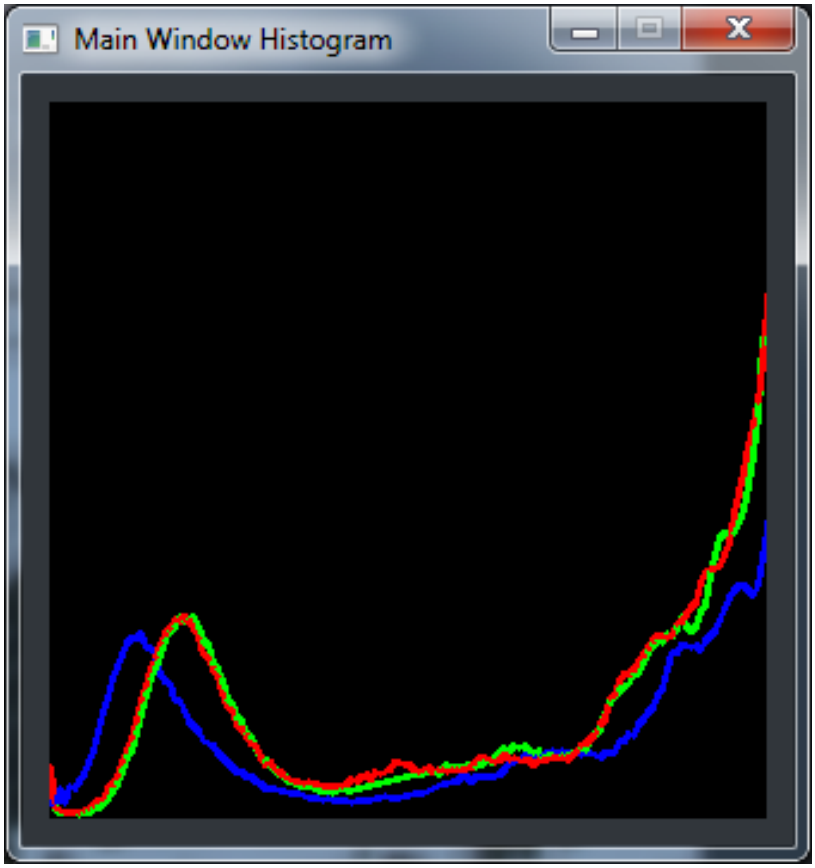

Figure 4.7: The histogram of the image in Figure 4.6. The color channels begin to converge to the same value.

the color channels are different within the two images. Figure 4.7 has mostly RBG values in the 255 bucket, meaning the resulting pixel would be white. Figure 4.5 on the other hand shows a white variation in RBG colors.

4.3 Regions of Interest

ROIs can be added to the image using the "Rectangular ROI" button. The user simply clicks and drags until their ROI is the appropriate size. The user can add as many ROIs as needed. In Figure 4.1 there are three regions of interest, shown as slightly opaque white boxes with its identifying number and size in the top left corner.

Once an ROI is placed, a user can choose to adjust the size, as seen in Figure 4.8. The dark grey boxes in the top left and bottom right corners allow users to click and grab to adjust the size. When a user selects an ROI, it turns red to provide user feedback. Once the user is happy with their region of interest placement, they can choose which algorithm to apply by right clicking within the region and using the menu in Figure 4.9.

If a user decides they no longer want a certain region of interest, it can be deleted. The user right clicks on a ROI, as shown in Figure 4.9 and chooses delete. The remaining ROI will be renumbered to account for the removal.

4.4 Coverage Algorithms

4.4.1 Cloud

Algorithm 4.1 was based off a hybrid threshold algorithm previously developed by Li [25]. Little computational work is done in determining cloud coverage, however it was determined to be is very accurate with a $Threshold_c$ value of 32. A comparison between an ROI from the data set and the results of the algorithm are in Figure 4.10.

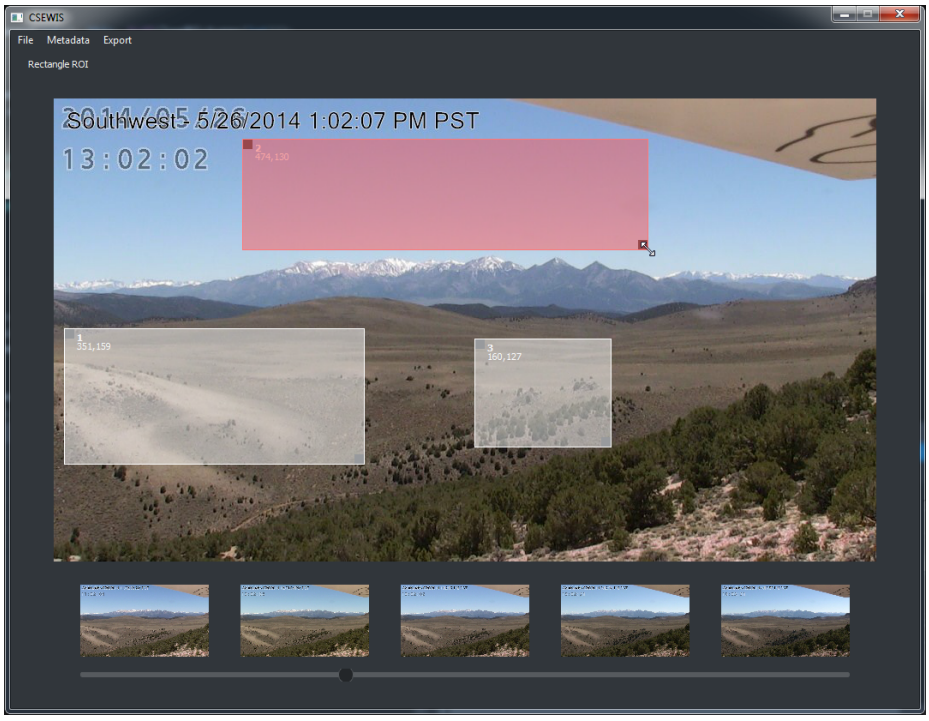

Figure 4.8: An example of re-sizing a region of interest.

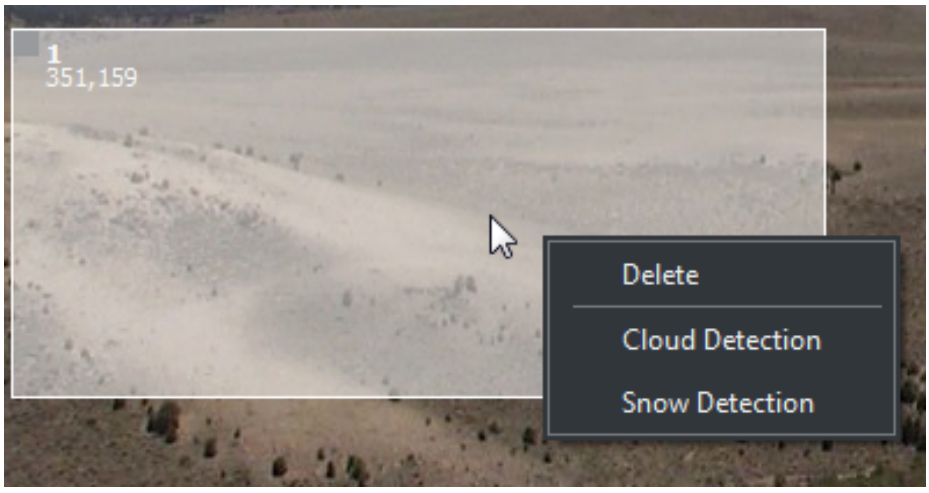

Figure 4.9: A menu from the region of interest.

Algorithm 4.1 Cloud Coverage Algorithm

```
Split Image into RGB Color Channels
for Pixel i in Image do
    if Blue_i - Red_i > Threshold_c then
        i is Cloud
    else
        i is Sky
    end if
end for
```

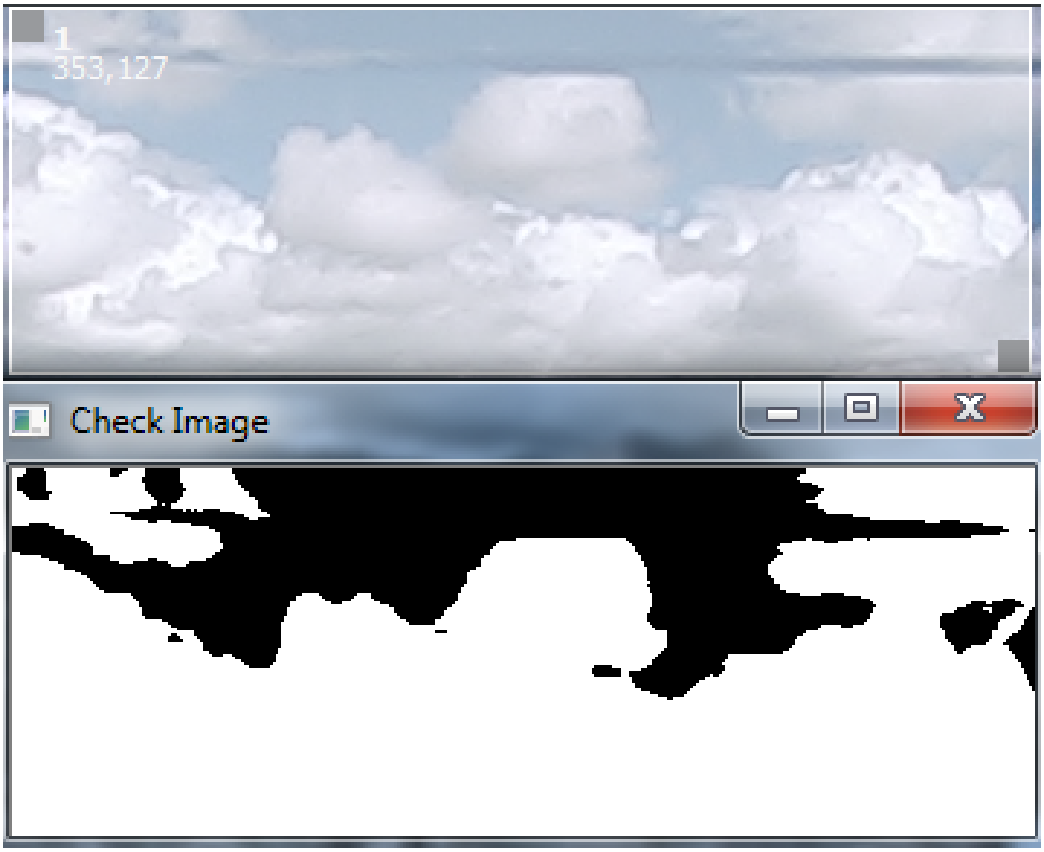

Figure 4.10: A comparison of the cloud coverage algorithm's output.

4.4.2 Snow

The snow coverage algorithm requires converting color streams, blurring, and several elemental operations making its execution a bit more intensive than the cloud algorithm. The snow detection algorithm was based off work previously done by Salvatori [40]. In our version, $Threshold_h$ is set to 20 and $Threshold_b$ is set to 127.

Figure 4.11: A comparison of the snow coverage algorithm's output.

Figure 4.11 shows how the output of Algorithm 4.2 compares to the original image.

Salvatori's algorithm did not have high accuracy results when tested on our dataset. This may be because the image set tested in Salvatori's project was from mountainous regions, where as our image set is from a desert. Salvatori's algorithm had problems distinguishing snow from sand.

We were able to fix this problem by adding a test of another color channel. The difference in pixel intensity in the HSL color space for snow and sand pixel was significant enough to distinguish between the two. This helped improved the accuracy of our snow coverage algorithm.

Algorithm 4.2 Snow Coverage Algorithm

```
1: Split Image into RGB Color Channels
2: Convert Image to HLS Color Space
3: Split Image into HLS Color Channels
4: Gaussian Blur (5x5) Hue Channel
5: for Pixel i in Image do
6:    if Hue_i > Threshold_h then
7:       if Blue_i > Threshold_b then
8:          i is Snow
9:       end if
10:   else
11:      i is Ground
12:   end if
13: end for
```

Chapter 5

GPU Analysis

5.1 Overview

In this chapter, we discuss the development of the snow and cloud cover algorithms for the GPUs. We compare execution time, speedup factor, and throughput of our snow and cloud coverage algorithms implemented on CPU, GPU, and multiple GPU. By varying the number of images in the stream, and the size of the images, we can determine the best method of processing image streams in minimal time.

5.2 Implementation

Two major implementations were developed; Windows and Linux executables. The Windows implementation ran the algorithm through the GUI seen in Chapter 4 for CPU or single GPU. The Linux implementation was implemented as a command line implementation and can run for CPU, single GPU or multi-GPU.

The cloud coverage algorithm was used as our low computation test. As such, it does very little calculations to determine cloud coverage. Lines 3 and 4 of Algorithm 4.1 are completely independent and can be run simultaneously on GPU causing speedup.

The snow coverage algorithm was our high computation test. The algorithm requires converting color streams, blurring, and several elemental operations. Because some of these operations rely on pixels related to them, the actions cannot always be done in parallel.

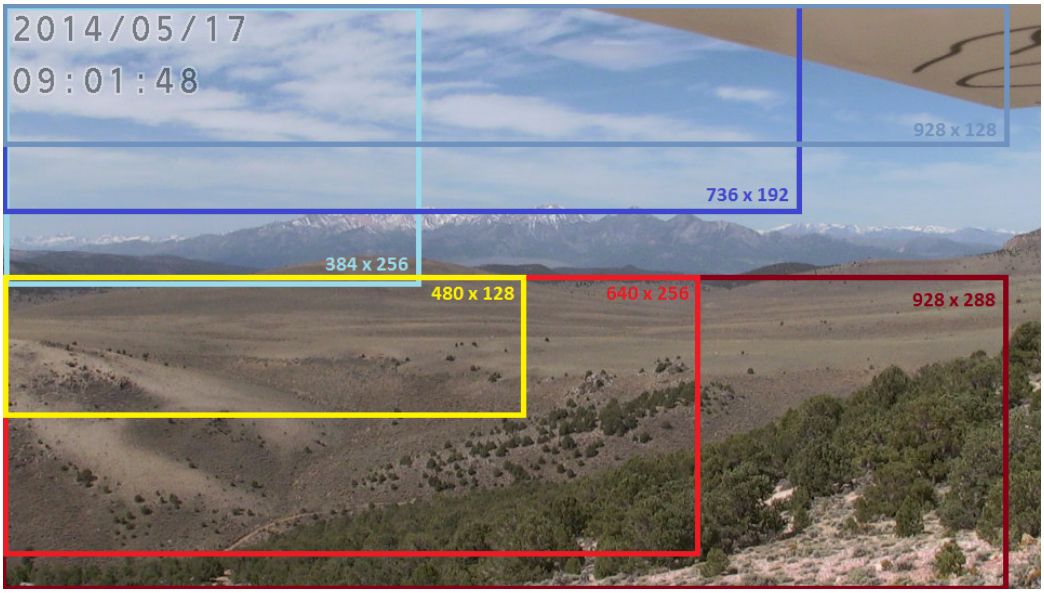

Figure 5.1: The various region of interest on which the cloud and snow algorithms were applied.

Although some functions must have algorithm changes to increase performance of the GPU, our implemented CPU algorithms did not need to be changed. The only difference between implementations is that the GPU algorithms have an added overhead of transferring the image to and from the graphics card. All developed algorithms were made to be as similar to the CPU implementation as possible. This allowed for more accuracy between comparisons of the algorithms. OpenCV's design allows for an almost direct correlation between its CPU and GPU code. Figure 5.1 shows regions of interest chosen for testing.

The various algorithms implementations were applied on a Ubuntu 16.04 machine with eight GeForce GTX 1080 (Pascal Architecture v6.1) graphics cards, two Intel Xeon Processors E5-2650 CPU and 64GB of RAM. Each card has 8114 MiB of memory, 65536 bytes of constant memory, and 49152 bytes of shared. Concurrent copying and execution is enabled with 2 copy engines running.

5.3 Execution Times

5.3.1 CPU

The CPU implementation for Algorithm 4.1 and Algorithm 4.2 are applied during separate runs to each image in the series. Image sizes and number of images in a series were varied over five runs and an average execution time was calculated in Figure 5.2.

Both algorithms run in linear time. as the number of images in a series increases so does the execution time. The image size does affect execution time, as seen from the increase in execution times in Cloud 736x128 and Snow 928x288, but not until a significant amount of images are added to the series. Algorithm 4.2 has higher execution times than Algorithm 4.1 because there are more steps that need to be done during it.

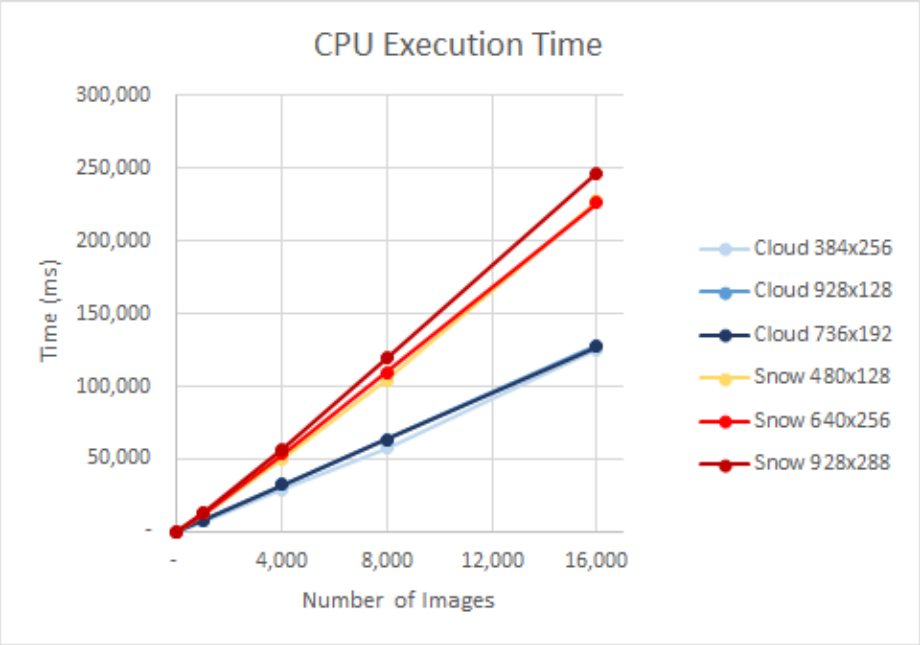

Figure 5.2: The execution time in milliseconds of both the cloud and snow algorithm run on the CPU.

5.3.2 GPU

Similarly, the GPU algorithm was implemented using OpenCV's GPU algorithm. In this implementation, after the image is read in from the file, it must be "uploaded" to the GPU. This take a significant amount of time initially to set up. To try to alleviate the initial overhead, a 1x1, single channel GPU matrix is created before processing the image series. This reduced the execution time of a single image using Algorithm 4.2 from an average of 755.822 ms to 23.356 ms; 32 times speedup. Once either algorithm has been applied to the image series, the resulting image is "downloaded" from the GPU to the CPU and the coverage percentage is pushed to a vector. The average execution time over 5 runs can be seen in Figure 5.3.

When moved to the GPU, OpenCV's matrices are divided up into $(c+15)\,/\,15$ by $(r+15)\,/\,15$ grids where c and r are the columns and rows in the image. Each block in the grid has 16 threads in both the x and y direction, resulting in 256 threads per block. Since 256 is divisible by 32, it takes advantage of the GPU's warp scheduling to increase speedup.

Once the GPU, all the execution times begin to converge together. The snow coverage algorithm has a significant amount of speedup. However, the cloud coverage algorithm actually takes longer on the GPU than it did on the CPU. This is because the added overhead of sending the images to the GPU and back takes more time than execution on the GPU could decrease. In other words, there is not enough computations done to justify sending it to the GPU.

5.3.3 Multiple GPU

The multiple GPU implementation utilizes threads to control multiple GPUs at once. Each thread is given all the file names of the images in the series and the region of interest height and width. The number of images in the series is divided among available GPUs. Each thread then runs the previously implemented single GPU algorithms on their own GPU stream. This allows for all image analysis to happen concurrently and images are uploaded and downloaded from the GPU asynchronously. The aver-

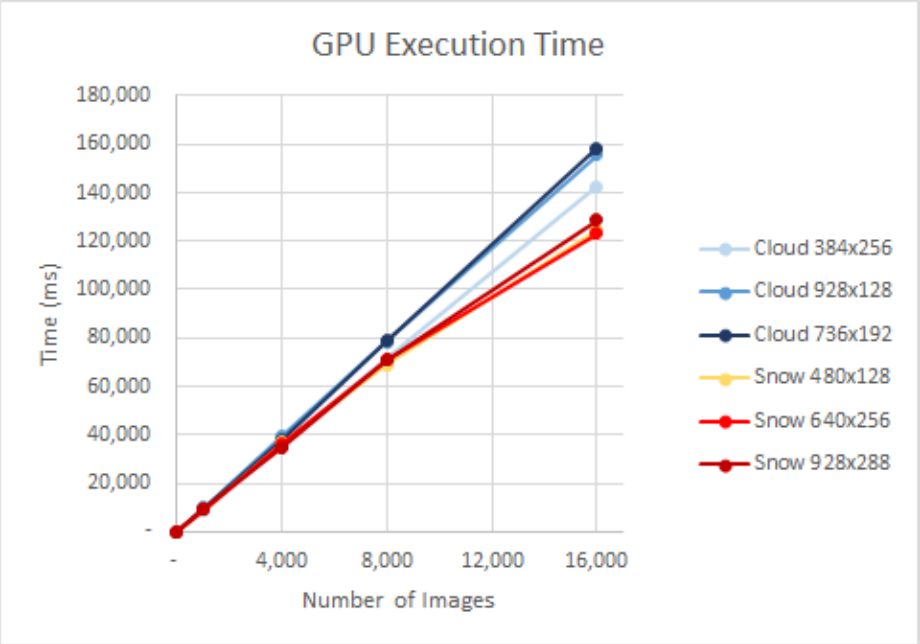

Figure 5.3: The execution time in milliseconds of both the cloud and snow algorithm run on the GPU.

age execution time over 5 runs can be seen in Figures 5.4 and 5.4. For simplicity's sake, Figures 5.4 and 5.4 show the largest of the ROI run for both algorithms. The full data set, however, is available [30].

As the number of GPUs increases, the execution time decreases. Even in the cloud detection algorithm, which did not achieve speedup when run on a single GPU, decreases its run time when more GPUs are added. Once 8 GPUs are added, however, execution time begins to increase again. Once again, the issue is the overhead resulting from sending the images to the GPUs. The amount of work being done on each GPU is not efficient and the images are going to so many different GPUs that they are bogging down the PCEI bus that connects the GPUs together.

5.4 Comparison

While Algorithm 4.1 does not increase speedup on a single GPU, speedup is gained during multi-GPU computing. Figure 5.6 shows the speedup factor of the 736x192 ROI over 8 GPUs. For GPUs 2 through 7 a steady increase of speedup occurs. At 8

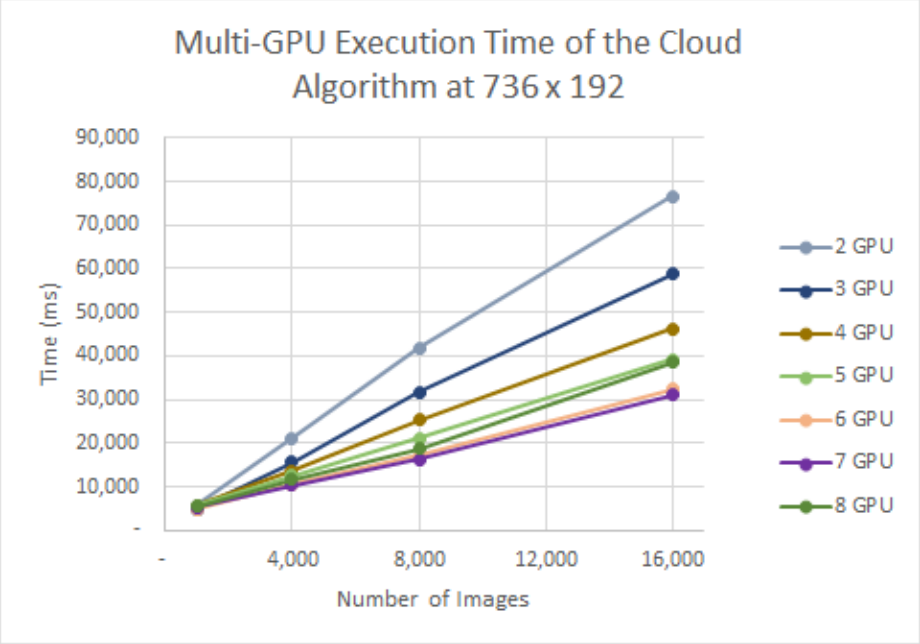

Figure 5.4: The execution time in milliseconds of the cloud algorithm with a 736x192 ROI.

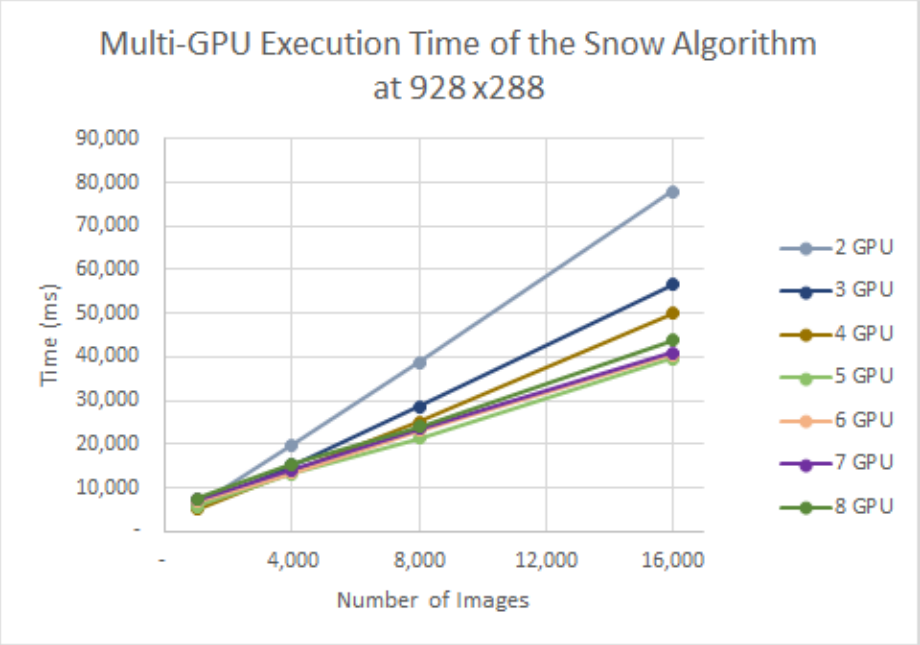

Figure 5.5: The execution time in milliseconds of the snow algorithm with a 928x288 ROI.

GPUs, speedup begins to decrease. As discussed previously, this is because there is too much data transfers happening between GPUs.

Algorithm 4.1's throughput in Figure 5.8 is quite clear. As the number of GPUs increases, up to 7 GPUs, the throughput increases as well.

Algorithm 4.2's speedup graph in Figure 5.7 is a little different. A steady increase in speedup factor occurs from the use of 1 to 5 GPUs, not 7 GPUs. This is because of the extra workload that needs to be done to calculate snow coverage in the region. At a smaller number of images, anything less than 4,000, it becomes confusing to determine the most efficient number of GPU to use. The speedup for such a small amount of images does not increase with more GPUs. This could be because the machines the algorithms were run on were either closer or farther from each other, thus affecting execution time.

Algorithm 4.2 Figure 5.9 is similar, in that it reflects trends previously discussed in its speedup graph. The throughput also increases as more images are added to the series, because large amounts of images can be more efficiently handled on more GPUs.

Figure 5.7 displays a superliner speedup characteristic of Algorithm 4.2. This is an expected characteristic, as OpenCV algorithms can have up to 100x speedup over their original implementation [26].

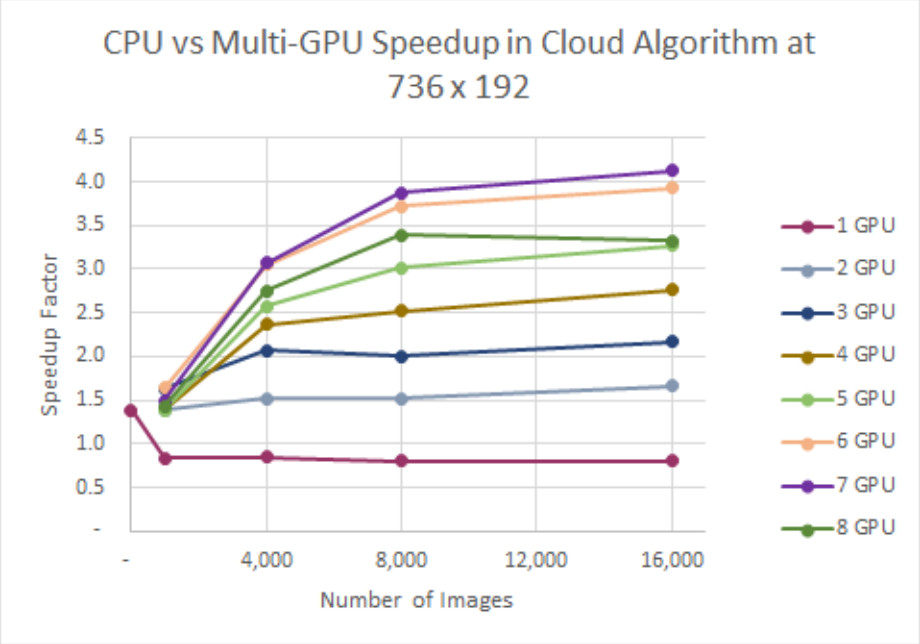

Figure 5.6: The speedup factor of the cloud algorithm with 736x192 ROI.

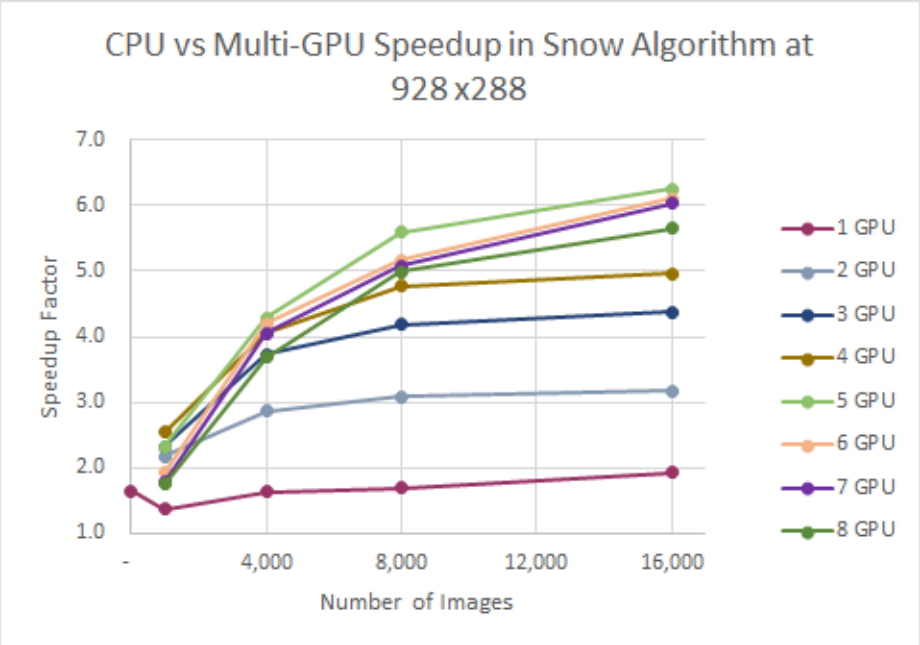

Figure 5.7: The speedup factor of the algorithm with 928x288 ROI.

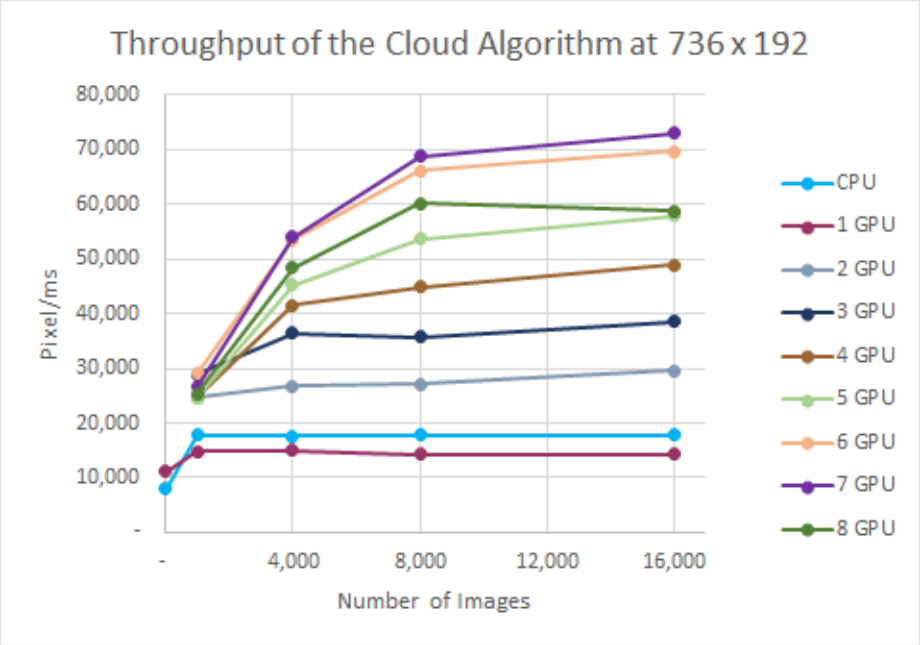

Figure 5.8: The throughput of the cloud algorithm with 736x192 ROI on CPU, GPU, and multiple GPUs.

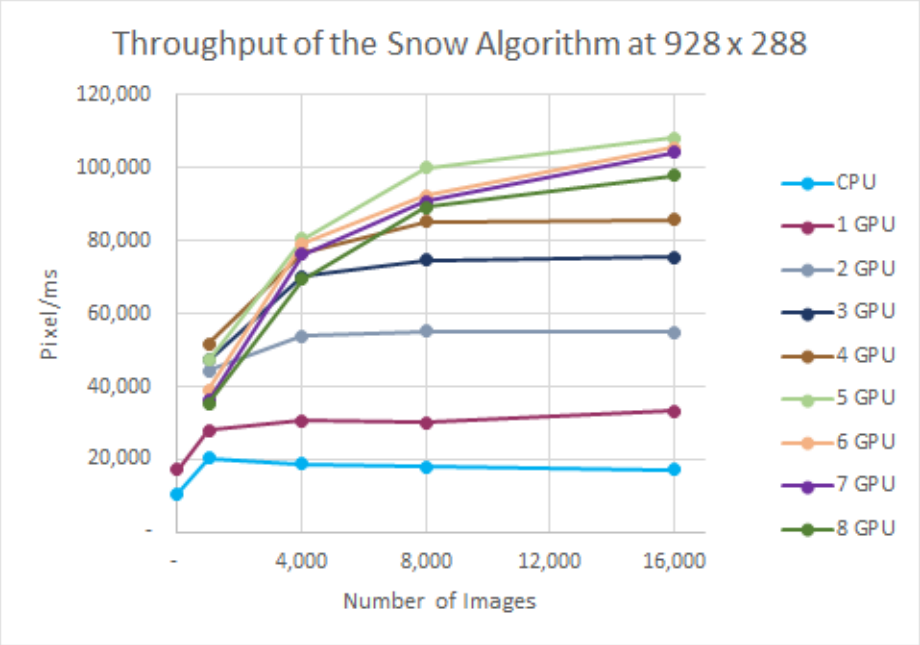

Figure 5.9: The throughput of the snow algorithm with 928x288 ROI on CPU, GPU, and multiple GPUs.

Chapter 6

Comparison with Related Works

In this chapter we compare CSEWIS with several popular environmental analysis software. The advantages and disadvantages of each are listed.

6.1 PhenoCam Network

PhenoCam is a webcam network that both collects images from webcams for analysis and provides tools by which to do the analysis [32]. Phenocam has a set of requirements that scientists must configure their webcams to before they can be a part of the network. This ensures that all images in the network will have similar properties and analyses on all images can be done without configuring any settings. PhenoCam's analysis tools are available for Windows, Linux, and Mac operating systems. The provided tools are written with a pre-compiled MATLAB back-end and R routines.

The Phenocam group has done continuous work on the usability of webcams in phenological studies [36]. In their study the placed a webcam with an infrared filter at a specified location. The camera was not moved, except to be refocused, so all images were of the same location. The resulting images were not altered or enhanced in any way. An analysis on the image stream was done using MATLAB on an ROI of the tree canopies. They were able to use the images to find when the tree canopies became green for spring. However, there are many downfalls to doing analysis with webcams. A difference in color balance levels can change the results of the image.

To be a part of the Phenocam network, cameras must adhere to certain protocols

or get a waiver [35]. All cameras must be white balanced, securely mounted, and pointed away from the sun to minimize noise in the image [35]. Waived cameras are typical camera that were set up before joining the PhenoCam network and deemed to produce good images [35].

Image analysis in PhenoCam is often done on user defined region of interests. The ROI is stored as a TIFF and a series of masks are applied to the RGB color channel of the area to determine vegetation coverage. The angle of the sun is taken into account to filter out images that where too bright or too dark [35]. Citizen Science is used to determine quality of images and remove outliers. Outliers are determined as images with qualities that could affect the output of the analysis. In the case of PhenoCam, this could be images with too much snow or are foggy and blurry.

PhenoCam allows users to analyze tree canopy phenology and nothing else [36]. Research labs, such as NEON, have modified PhenoCam to allow for analysis of snow, but there is no dedicated software available [14].

6.2 Finnish Meteorological Institute Image Processing Tool

Finnish Meteorological Institute Image Processing Tool (FMIPROT) is part of the phenological camera network as part of the MONIMET project in Finland [44]. The project overlooks a webcam network of 14 sites with 27 cameras. The software can track any color value from its three processing algorithms; color fraction extraction, vegetation index, and custom color index [34]. Additionally, FMIPROT can be used with any camera or camera network. The software can be run as either a GUI or through the command line. The results are output as an interactive HTML plots.

A recent study using FMIPROT focused on tracking snow coverage in three northern sites in Finland [3]. Much of the area used in the study is covered with forest and wetlands, making it hard to use satellite data to assess snow coverage. Analysis of the research areas was done on $2592x1944$ JPEG images from four cameras. A new picture of the area is taken every hour or half hour between 10:45 and 12:45 so as

to ignore illumination differences. ROI are selected within the image series to avoid trees and non-ground objects in the image. Results were compared the results from 7 human experts to determine coverage accuracy.

Snow coverage was determined by comparing the blue histogram of an image to either a threshold value of 127 or the first local minimum. A smoothing filter is applied to the histogram. Errors in the results were due to blue lichen or rainwater sometimes being classified as snow. The histogram is also susceptible to shadows and vegetation, which can cause the threshold to become too high a number and classify pixels incorrectly.

6.3 Sky Coverage of Brazilian Antarctic Station

Research was done to create a methodology to determine cloud coverage in remote sensor locations [43]. This study takes place in Antarctica and uses the webcams from the Brazilian Antarctic Station Ferraz. The methodology uses a camera from remote sensor arrays at ground level, unlike other cloud detection systems that point directly into the sky [21]. Shadows and solar elevation were not taken into account, as the camera was mounted on a building roof and would not run into any of their ill effects.

The methodology uses both the RGB color space as well as HLS color space to determine whether a selected pixel is a cloud. Clear sky can be assumed to be mostly blue with high saturation. Cloudy skies will be very low saturation with little blue. The software classifies pixels into three separate categories, clear sky, cloudy sky, and undefined sky. Undefined pixels are often along the edges of cloud, transitioning into sky, or pixels affected by aerosol noise. Pixels are classified based on upper and lower threshold based on group averages and deviations of the pixel saturation in the image.

Two professionals compared results of the system to manual observations. It was observed that both observers produced similar cloud coverage estimates while the system often over estimated when low amount of clouds were present. Undefined estimations from the classifier was often the most inconsistent when compared to the

professional observations.

6.4 Features List

Table 6.1 shows how CSEWIS compares to the previously discussed software. Further work done on CSEWIS to expand its available features could make it a worthy competitor.

Table 6.1: A comparison of features available in the related works and CSEWIS.

	CSEWIS	FMIPROT	PhenoCam	SCBAS
Orthorectification		x		
Free to use	x	x		
Solar Elevation Correction			x	
Network of Cameras		x	x	
Phenology Coverage			x	
Snow Coverage	x	x		
Cloud Coverage	x			x
Multiple ROI	x	x	x	
GPU Algorithms	x	x	x	x
Coding Experience Needed		x	x	
Metadata Generated	x		x	

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this thesis, we introduced CSEWIS, a software for determining snow and cloud coverage from a webcam image series. The system allows a user to create regions of interest where they believe there could be instances of snow and cloud coverage. The user can choose which detection algorithm (snow or cloud) they would like to run within the area. After making their selection, the user can export estimates as a CSV file or an AVI video for further analysis.

Ground based webcams can be used in remote sensor research to validate and improve upon coverage estimates done with satellite imagery. While satellite images can give a more exact coverage estimate, noise such as aerosol vapor and tree canopies can make it hard to determine coverage on the ground. Ground based webcams are able to track activity in hidden areas. The addition of webcams to sensor networks is still fairly new and is lacking usable software and algorithms for the detection of environmental properties.

Calculating snow and cloud coverage is important for environmental science. Coverage estimates are used for theoretical models and for tracking changes in landscape over time. Image processing algorithms can be used to replace the traditional human element in coverage calculation. By taking humans out of the process, coverage estimates can be calculated faster and more consistently.

The main goal of this thesis was to create software that let the user take advantage

of webcams by creating a user friendly way to determine coverage of snow and clouds. The results are accurate and repeatable, which allows them to be used in research. By allowing for analysis on the GPU, a speedup factor of up to 6x can be achieved.

7.2 Future Work

CSEWIS is limited in its coverage detection type. Other detection types, such as green value for phenology, could be added [32]. This would improve the robustness of the project. The Phenocam network has 385 sites on four continents tracking changes in green values. Although we chose to develop a less popular coverage algorithm, it is obvious that there is a greater need for phenology tracking. Adding a phenology coverage algorithm to CSEWIS could increase its usefulness in the field.

Another area of improvement could be in creating polygon regions of interest. Rectangular regions of interest are useful for capturing large areas, but can be hard to use in small areas. The user may be force to choose areas that include trees or rocks, which will result in errors when determining coverage. Polygon regions of interest would allow the user to select around noisy areas and expand ROI to exact locations. Region based detection algorithms could help select regions of interest with minimal user input [23].

CSEWIS could also be improved by adding camera rectification techniques to images. Two of the previously discussed software include orthorectification and solar elevation corrections [3, 36]. Allowing for corrections of camera lenses distorting its image and noise due to the sun can improve the accuracy of the results. The lens distortion at our camera site may not seem noticeable, however many research sites use fisheye lens cameras which should be corrected before doing analysis [21]. Adding this feature would improve CSEWIS' usability, however it comes at a cost, as all camera lens and solar elevations are different among model types and site locations. Adding this feature would mean the user would have to assure the correction algorithms work with their camera type.

Bibliography

[1] Zeinab Nazemi Absardi and Reza Javidan. Classification of big satellite images using hadoop clusters for land cover recognition. In *2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI)*, pages 0600–0603, December 2017. DOI: 10.1109/KBEI.2017.8324870.

[2] Rahul Agrawal, Soumyajit Gupta, Jayanta Mukherjee, and Ritwik Kumar Layek. A gpu based real-time cuda implementation for obtaining visual saliency. In *Proceedings of the 2014 Indian Conference on Computer Vision Graphics and Image Processing*, ICVGIP '14, 1:1–1:8, Bangalore, India. ACM, 2014. ISBN: 978-1-4503-3061-9. DOI: 10.1145/2683483.2683484. URL: http://doi.acm.org/10.1145/2683483.2683484.

[3] Ali Nadir Arslan, Cemal Melih Tanis, Sari Metsämäki, Mika Aurela, Kristin Böttcher, Maiju Linkosalmi, and Mikko Peltoniemi. Automated webcam monitoring of fractional snow cover in northern boreal conditions. *Geosciences*, 7(3):55, July 2017. ISSN: 2076-3263. DOI: 10.3390/geosciences7030055. URL: http://dx.doi.org/10.3390/geosciences7030055.

[4] Tim B. Brown, Kevin R. Hultine, Heidi Steltzer, Ellen G. Denny, Michael W. Denslow, Joel Granados, Sandra Henderson, David Moore, Shin Nagai, Michael SanClements, Arturo Sánchez-Azofeifa, Oliver Sonnentag, David Tazik, and Andrew D. Richardson. Using phenocams to monitor our changing earth: toward a global phenocam network. *Frontiers in Ecology and the Environment*, 14(2):84–93, 2016.

[5] Nevada Research Data Center. Nevada research data center, 2017. URL: https://sensor.nevada.edu/NRDC/. Accessed: 09-01-2017.

[6] Arthur Coste. Project 1 : histograms. URL: http://www.sci.utah.edu/~acoste/uou/Image/project1/Arthur_COSTE_Project_1_report.html. Accessed: 07-05-2018.

[7] Ivan Culjak, David Abram, Tomislav Pribanic, Hrvoje Dzapo, and Mario Cifrek. A brief introduction to opencv. In *2012 Proceedings of the 35th International Convention MIPRO*, pages 1725–1730, May 2012.

[8] Soumyabrata Dev, Yee Hui Lee, and Stefan Winkler. Color-based segmentation of sky/cloud images from ground-based cameras. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 10(1):231–242, January 2017. ISSN: 1939-1404. DOI: 10.1109/JSTARS.2016.2558474.

[9] Soumyabrata Dev, Bihan Wen, Yee Hui Lee, and Stefan Winkler. Ground-based image analysis: a tutorial on machine-learning techniques and applications. *IEEE Geoscience and Remote Sensing Magazine*, 4(2):79–93, June 2016. ISSN: 2473-2397. DOI: 10.1109/MGRS.2015.2510448.

[10] OpenCV API Documentation. Gpu module introduction, 2018. URL: https://docs.opencv.org/2.4/modules/gpu/doc/introduction.html. Accessed: 2-11-2018.

[11] Paul N. Edwards, Matthew S. Mayernik, Archer L. Batcheller, Geoffrey C. Bowker, and Christine L. Borgman. Science friction: data, metadata, and collaboration. *Social Studies of Science*, 41(5):667–690, 2011.

[12] Roman Fedorov, Alessandro Camerada, Piero Fraternali, and Marco Tagliasacchi. Estimating snow cover from publicly available images. *CoRR*, abs/1508.01055, 2015. arXiv: 1508.01055. URL: http://arxiv.org/abs/1508.01055.

[13] Jozef Gerát, Dominik Sopiak, Miloš Oravec, and Jarmila Pavlovicová. Vehicle speed detection from camera stream using image processing methods. In *2017 International Symposium ELMAR*, pages 201–204, September 2017. DOI: 10.23919/ELMAR.2017.8124468.

[14] Leslie Goldman. Phenocam data for plant phenology and snow cover dynamics now available, 2016. URL: https://www.neonscience.org/observatory/observatory-blog/phenocam-data-plant-phenology-snow-cover-dynamics-now-available. Accessed: 12-14-2017.

[15] Vikas Gupta. Color spaces in opencv (c++ / python), 2017. URL: https://www.learnopencv.com/color-spaces-in-opencv-cpp-python/. Accessed: 06-27-2018.

[16] Batuhan Hangün and Önder Eyecioğlu. Performance comparison between opencv built in cpu and gpu functions on image processing operations. In volume 1, pages 34 –41, Nişantaşı Üniversitesi.

[17] Sneha H.L. Pixel intensity histogram characteristics: basics of image processing and machine vision, 2017. URL: https://www.allaboutcircuits.com/technical-articles/image-histogram-characteristics-machine-learning-image-processing/. Accessed: 07-05-2018.

[18] Sunhee Hwang, Youngjung Uh, Minsong Ki, Kwangyong Lim, Daeyong Park, and Hyeran Byun. Real-time background subtraction based on gpgpu for high-resolution video surveillance. In *Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication*, IMCOM '17, 109:1–109:6, Beppu, Japan. ACM, 2017. ISBN: 978-1-4503-4888-1. DOI: 10.1145/3022227.3022335. URL: http://doi.acm.org/10.1145/3022227.3022335.

[19] Nathan Jacobs, Walker Burgin, Nick Fridrich, Austin Abrams, Kylia Miskell, Bobby H. Braswell, Andrew D. Richardson, and Robert Pless. The global network of outdoor webcams: properties and applications. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '09, pages 111–120, Seattle, Washington. ACM, 2009. ISBN: 978-1-60558-649-6. DOI: 10.1145/1653771.1653789. URL: http://doi.acm.org/10.1145/1653771.1653789.

[20] Akira Kato, Justin Morgenroth, David Kelbe, Christopher Gomez, and Jan van Aardt. Ground truth measurement of trees using terrestrial laser for satellite remote sensing. In *2013 IEEE International Geoscience and Remote Sensing Symposium - IGARSS*, pages 2106–2109, July 2013.

[21] Andreas Kazantzidis, P. Tzoumanikas, Alkiviadia F. Bais, Spiros Fotopoulos, and George Economou. Cloud detection and classification with the use of whole-sky ground-based images. *Atmospheric Research*, 113:80 –88, 2012. ISSN: 0169-8095. DOI: https://doi.org/10.1016/j.atmosres.2012.05.005. URL: http://www.sciencedirect.com/science/article/pii/S0169809512001342.

[22] Jing Ke, Tomasz Bednarz, and Arcot Sowmya. Optimized gpu implementation for dynamic programming in image data processing. In *2016 IEEE 35th International Performance Computing and Communications Conference (IPCCC)*, pages 1–7, December 2016. DOI: 10.1109/PCCC.2016.7820646.

[23] Shawn Lankton and Allen Tannenbaum. Localizing region-based active contours. *IEEE Transactions on Image Processing*, 17(11):2029–2039, November 2008. ISSN: 1057-7149. DOI: 10.1109/TIP.2008.2004611.

[24] Jiwan Lee, Bonghee Hong, Yongdeok Shin, and Yang-Ja Jang. Extraction of weather information on road using cctv video. In *2016 International Conference on Big Data and Smart Computing (BigComp)*, pages 529–531, January 2016. DOI: 10.1109/BIGCOMP.2016.7425986.

[25] Qingyong Li, Weitao Lu, and Jun Yang. A hybrid thresholding algorithm for cloud detection on ground-based color images. *Journal of Atmospheric and Oceanic Technology*, 28(10):1286–1296, 2011. DOI: 10.1175/JTECH-D-11-00009.1. eprint: https://doi.org/10.1175/JTECH-D-11-00009.1. URL: https://doi.org/10.1175/JTECH-D-11-00009.1.

[26] OpenCV library. Cuda, 2018. URL: https://opencv.org/platforms/cuda.html. Accessed: 2-11-2018.

[27] OpenCV library. Opencv library, 2017. URL: https://opencv.org. Accessed: 10-11-2017.

[28] Slavomir Matuska, Robert Hudec, and Miroslav Benco. The comparison of cpu time consumption for image processing algorithm in matlab and opencv. In *2012 ELEKTRO*, pages 75–78, May 2012. DOI: 10.1109/ELEKTRO.2012.6225575.

[29] Assefa M. Melesse, Qihao Weng, Prasad S. Thenkabail, and Gabriel B. Senay. Remote sensing sensors and applications in environmental resources mapping and modelling. *Sensors*, 7(12):3209–3241, 2007.

[30] Hannah Munoz. Final run times, 2018. URL: `https://github.com/hannahmunoz/MultiGPUOpenCV/blob/master/FinalRuntimes.xlsx`. Accessed: 05-10-2018.

[31] Hannah Munoz, Connor Scully-Allison, Vinh Le, Scotty Strachan, Jr. Fredrick C. Harris, and Sergiu M. Dascalu. A mobile quality assurance application for the nrdc. *Proceedings of the ISCA 26th International Conference on Software Engineering and Data Engineering*, October 2017.

[32] The University of New Hampshire. Phenocam, 2012. URL: `https://phenocam.sr.unh.edu/webcam/`. Accessed: 10-11-2017.

[33] Nathalie Pettorelli, Kamran Safi, and Woody Turner. Satellite remote sensing, biodiversity research and conservation of the future, 2014.

[34] MONIMET Project. Fmiprot: fmi image processing toolbox, 2018. URL: `http://fmiprot.fmi.fi/`. Accessed: 07-04-2018.

[35] Andrew D. Richardson, Koen Hufkens, Tom Milliman, Donald M. Aubrecht, Min Chen, Josh M. Gray, Miriam R. Johnston, Trevor F. Keenan, Stephen T. Klosterman, Margaret Kosmala, Mark A. Friedl Eli K. Melaas, and Steve Frolking. Tracking vegetation phenology across diverse north american biomes using phenocam imagery. *Scientific data*, 5:180028, 2018.

[36] Andrew D. Richardson, Julian P. Jenkins, Bobby H. Braswell, David Y. Hollinger, Scott V. Ollinger, and Marie-Louise Smith. Use of digital webcam images to track spring green-up in a deciduous broadleaf forest. *Oecologia*, 152(2):323–334, May 2007. ISSN: 1432-1939. DOI: `10.1007/s00442-006-0657-z`. URL: `https://doi.org/10.1007/s00442-006-0657-z`.

[37] Andrew D. Richardson, Stephen Klosterman, and Michael Toomey. Near-surface sensor-derived phenology. In *Phenology: An integrative environmental science*, pages 413–430. Springer, 2013.

[38] Dominic Rüfenacht, Matthew Brown, Jan Beutel, and Sabine Süsstrunk. Temporally consistent snow cover estimation from noisy, irregularly sampled measurements. In *2014 International Conference on Computer Vision Theory and Applications (VISAPP)*, volume 2, pages 275–283, January 2014.

[39] Vigneswaran Saahithyan and Somaskandan Suthakar. Performance analysis of basic image processing algorithms on gpu. In *2017 International Conference on Inventive Systems and Control (ICISC)*, pages 1–6, January 2017. DOI: `10.1109/ICISC.2017.8068687`.

[40] Rosamaria Salvatori, Paolo Plini, Marco Giusto, M Valt, Roberto Salzano, Mauro Montagnoli, Anselmo Cagnati, Giuseppe Crepaz, and Daniele Sigismondi. Snow cover monitoring with images from digital camera systems. *Italian Journal of Remote Sensing*, 43, June 2011.

[41] Robert A. Schowengerdt. *Remote Sensing: Models and Methods for Image Processing*. Elsevier Science, 2006. ISBN: 9780080480589. URL: https://books.google.com/books?id=KQXNaDH0X-IC.

[42] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image Processing, Analysis, and Machine Vision*. Cengage Learning, 2014. ISBN: 9781285981444. URL: https://books.google.com/books?id=QePKAgAAQBAJ.

[43] Mariza Pereira Souza-Echer, Enio Bueno Pereira, Bins Leonardo, and M. A. R. Andrade. A simple method for the assessment of the cloud cover state in high-latitude regions by a ground-based digital camera. *Journal of Atmospheric and Oceanic Technology*, 23(3):437–447, 2006. DOI: 10.1175/JTECH1833.1. eprint: https://doi.org/10.1175/JTECH1833.1. URL: https://doi.org/10.1175/JTECH1833.1.

[44] Cemal Melih Tanis and Ali Nadir Arslan. Fmiprot: fmi image processing toolbox, 2018. URL: http://fmiprot.fmi.fi/. Accessed: 06-25-2018.

[45] Cornell University. Moving theory into practice: digital imaging tutorial, 2003. URL: http://preservationtutorial.library.cornell.edu/intro/intro-04.html. Accessed: 06-27-2018.

[46] Mi Wang, Liuyang Fang, Deren Li, and Jun Pan. Using multiple gpus to accelerate mtf compensation and georectification of high-resolution optical satellite images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 8(10):4952–4972, October 2015. ISSN: 1939-1404.