



WORCESTER POLYTECHNIC INSTITUTE

# Mid-latitude All-sky-imager Network for Geophysical Observation

*A Major Qualifying Project submitted to  
the Faculty of Worcester Polytechnic Institute in partial fulfilment of the requirements for the  
Degree in Bachelor of Science in Electrical and Computer Engineering by*

---

Fabrice Fotso Kengne

---

Rohit Mundra

---

Maria Alexandra Rangel

6th March, 2013

Advised by

---

Professor Andrew Klein

Supervised by

Elizabeth Kendall

Asti Bhatt

Sponsored by

## **SRI International**

This report represents work of WPI undergraduate students submitted to the faculty as evidence of a degree requirement. WPI routinely publishes these reports on its web site without editorial or peer review. For more information about the projects program at WPI, see <http://www.wpi.edu/Academics/Projects>.

# Acknowledgements

*“We would like to thank our mentors and friends, Asti and Elizabeth, for their guidance, support, and ability to answer our incessant questions with so much enthusiasm and passion. This project could not have succeeded without their presence. Without them, we would not have pushed ourselves to accomplish so much.*

*We would also like to thank our project advisor, Professor Klein, for his encouragement and professional advice during this project. It was his patience and his confidence in us that motivated us to step out of our comfort zones and broaden our horizons.*

*Finally, we would like to extend our gratitude to Steven Chen and Dr. Todd Valentic for investing their time and sharing with us the wealth of knowledge they possess.”*

-Alex, Fabrice and Rohit

# Abstract

This project presents the design of an all-sky imager network system. The designed system is capable of taking all-sky images of upper atmospheric airglows and low-latitude auroras, and applying image processing algorithms to correct distortions and to eliminate background noise. This design incorporates novel astronomical image processing routines designed in Python. The processed images are then geo-located on a map of the United States for real-time display on a website. This system will be used to study macroscopic-scale auroras and airglows over the country and to characterize the dynamics of charged particles in the ionosphere. SRI International also intends to get high-school students involved in space-science education using this system.

# Executive Summary

The Mid-Latitude All-Sky-Imager Network for Geophysical Observation, also known as MANGO, is a project sponsored by SRI International, a research institute headquartered in Menlo Park, California. SRI International's MANGO project has two main goals: an educational goal to get high school students involved in space science and upper atmospheric phenomena, and a scientific goal to further the understanding of the consequences of upper atmospheric phenomena on communication systems. To accomplish these goals, the institute needs to make real-time observations of upper-atmospheric phenomena. Our team was responsible to architect and formulate a system which allows researchers and students to observe mid-latitude airglow and auroras from multiple locations across the United States.

Before we architected a system to meet this purpose, we started with background research on upper-atmospheric phenomena such as airglow and auroras. We specifically studied the underlying science behind them to understand why they occur, at what altitudes they occur, and the variations they exhibit. We also looked into the different layers of the ionosphere and the consequences they have on radio wave propagation. Through this process, we learned that airglows occur at night due to the recombination of photoionized particles, which causes electrons to change energy levels leading to the emission of photons. While auroras mostly occur at night as well, they are produced by the collision of solar electrons with oxygen and nitrogen atoms in the upper atmosphere. Depending on the ionosphere's properties at a certain location, radio waves of certain frequencies can either refract into space or reflect back to Earth's surface. A study of methods to image ionospheric phenomena revealed that fisheye lenses are widely used to observe upper atmospheric phenomena. Although fisheye lenses have the advantage of a wide field of view (FOV), they introduce optical distortions such as barrel distortion and are prone to luminosity distortions such as atmospheric extinction and Van Rhijn effect.

In order to design a system that can make real-time observations of upper-atmospheric phenomena, it was important to compartmentalize the overall system into three subsystems: a data acquisition system, an image processing system, and a visualization system. The data acquisition system is responsible for image acquisition and remote system information acquisition. In our

design, image acquisition was done using <undisclosed astronomy grade> CCD cameras which come with Windows drivers and software support for Windows-based platforms. The cameras also come with a software development kit (SDK) which allows developers to implement custom programs to operate the camera. The programming language we chose to build the camera control program is Python; we used Python's *ctypes* module to interact with the camera's driver dynamic linked libraries (DLLs). Since image acquisition happens at remote sites, it was necessary to implement a program that monitors the camera enclosure's ambient temperature, the power status of the computer controlling the camera, and its disk space. This requirement was met by writing a routine that constantly checks for these critical metrics, and saves them to CSV files (a single relation relational database) for visualization later.

The next step in the design was to process the images from the CCD camera by rectifying the distortions introduced and mitigating the consequences of the Van Rhijn effect and atmospheric extinction. The first important step in the image processing system was all-sky spatial calibration. Spatial calibration was done for the purpose of converting all-sky-images from their standard coordinate system to a more appropriate geographical coordinate system. In other words, it is done to align the top of the image with the true geographical North. This was done by physically identifying the stars on the image, and then finding their corresponding azimuths and elevations using a star catalog for that location and time. Since the purpose of the entire design was to observe airglow and auroras, other astronomical objects such as stars were to be disregarded. Thus to remove stars from images, we created a star removal algorithm that detects and removes stars. This routine goes through each of the image's pixels and detects sudden changes in brightness; when detected, the location is identified as a star. The values of star locations are then interpolated over. As previously stated, all-sky imagers use fisheye lenses and these lenses introduce luminosity distortions such as Van Rhijn effect and atmospheric extinction. Van Rhijn effect occurs because objects that are located at low elevation angles yet far away from each other get projected and superimposed on outer pixels of all-sky images. This causes them to appear brighter than the pixels located closer to the lens optical axis. Atmospheric extinction counters the effect of the Van Rhijn effect in a sense since light from astronomical objects located at low elevation angles is attenuated more by the presence of more air mass causing reduced flux to reach the imager. To correct these effects, we used mathematical models of the Van Rhijn effect and atmospheric extinction and design a correction filter which we apply to the captured images. Barrel distortion on the other hand was corrected using an unwarping method. The method transformed an image from hemispherical projection to a rectilinear projection. A key observation in this transformation was

the way fisheye images are recorded; we observed that fisheye lenses record angular view per unit distance recorded, while rectilinear lenses record unit distance per unit distance recorded. This method was extended to map the distances in the image to precise geographic coordinates for projection on a web-based Mercator map.

To integrate data acquisition system with image processing system, it was essential to design a program that determines whether or not new images are available for image processing. Thus, we implemented a program that frequently checks for newly acquired images. As new images arrive, the program temporarily queues the images and sends them for image processing using multicore capabilities of the processor. This program was critical for real-time processing of all-sky images.

The visualization system displays the images processed in the previous step. Visualization was done by designing a website to which students, scientists, and SRI International's researchers will have access. The website was built according to SRI International's web development requirements. We used a Python-based web development framework called Flask. The website has two parts: a public section for non-SRI International affiliated members such as students and scientists, and an administrator page for SRI International's researchers. The public page contains information about the MANGO project, descriptions of airglow and auroras, and a United States map on which processed images representing airglows and auroras are overlaid. The administrator page enables SRI International's administrators to add/modify/remove sites, to add/modify calibration data, and to monitor remote site information such ambient temperature, disk space, battery life, and CCD temperature. All parties have the ability to register to the website.

Once all the systems were designed, we integrated them and tested the final product. We were able to acquire the images from the CCD camera. The intermediary image detection program was able to detect new images, and send them to the image processing system. After several tests, the outcome of the overall system met all the design requirements: images were properly obtained, processed, and overlaid onto a web-based map. We were also able to view remote system's health in the administrator website.

Through this project, the team acquired knowledge of the consequences of geophysical phenomena on contemporary communication systems. To meet SRI International's project requirements, a problem statement was established and a plan of action was developed accordingly. The project required knowledge of image and signal processing algorithms, power systems, computer networks, and higher-level object-oriented design; skills that are instilled in us by the faculty at Worcester Polytechnic Institute (WPI). As a result of applying such expertise, the team

developed one of the first astronomical image processing programs using Python. Although this project has successfully advanced SRI International's goal to learn more about mid-latitude airglow and auroras, there are many other techniques that can be used to improve the MANGO project. We have identified the following fields of improvements: firmware/hardware, back-end software, and front-end software. During the design, the team was limited to using a Windows notebook to run the system because the <undisclosed astronomy grade> CCD camera lacked Linux support. Once a Linux platform is developed to interface with the camera, the system can be imported into a compact Linux computer such as Raspberry Pi. A Linux platform will be beneficial in terms of portability, cost (free operating system, and low cost hardware), and power consumption. The team also found that image calibration was tedious. As we have already discussed, we had to manually identify stars in an image to calibrate the image. However, this method can be replaced with an automated back-end method such as a randomized Monte-Carlo search to increase calibration precision. Monte-Carlo search can also be used to generate an accurate lens function, thus increasing the precision of the map projection routine. Another change we recommend is the improvement of the front-end software by implementing a more scalable database system. The current database (SQLite), although easy to set up and good for rapid development, is limited in size. It also lacks sophisticated features that larger databases such as PostgreSQL and MySQL offer. Our last recommendation is to find a better method to rank the overlaid images on the Earth map. In our design, a specific image rank protocol has not been developed. Near-real-time cloud cover databases are available online and free of charge. They can be integrated with the MANGO system to determine the image layer ranks, and the cloudy regions can be assigned a lower layer rank since airglow and auroras cannot be observed.

# Table of Contents

1	Introduction .....	1
1.1	Project Description .....	1
1.2	Sponsor Description .....	3
2	Background .....	5
2.1	Ionosphere.....	5
2.1.1	Aurora .....	6
2.1.2	Airglow .....	8
2.2	Ionospheric Effects on Radio Communication Systems.....	9
2.2.1	Ionospheric Variations.....	11
2.2.2	Radio Waves and the Ionosphere .....	12
2.3	Optics and Imaging Techniques.....	13
2.3.1	Factors Affecting Choice of Optics.....	13
2.3.2	Use of Fisheye Lens in Auroral Imaging.....	15
2.4	Image Processing.....	16
2.4.1	Characteristics of All-Sky Images .....	16
2.4.2	Read-out Noise and Data Binning .....	17
2.4.3	Van Rhijn Effect and Atmospheric Extinction .....	17
2.4.4	Required Image Processing.....	19
3	Methodology.....	22
3.1	Data Acquisition.....	22
3.1.1	Image Acquisition.....	22
3.1.2	System Information Acquisition.....	24
3.1.3	Integrating Image and System Information Acquisition on Remote Site .....	24
3.2	Image Processing.....	27
3.2.1	All-Sky Spatial Calibration .....	28
3.2.2	Star Removal.....	31
3.2.3	Flat Field Correction: Countering Van Rhijn Effect and Atmospheric Extinction...32	



3.2.4	All-Sky Projection onto a Uniformly Spaced Grid .....	33
3.3	Real-time Image Processing .....	40
3.4	Display and Visualization.....	42
3.4.1	Setting up the Website .....	43
3.4.2	Public Webpage.....	44
3.4.3	Administrator Webpage.....	46
3.5	Conclusion.....	51
4	Results .....	53
4.1	Data Acquisition.....	53
4.1.1	Image Acquisition.....	53
4.1.2	System Information Acquisition.....	54
4.2	Image Processing.....	54
4.2.1	All-Sky Spatial Calibration .....	54
4.2.2	Star Removal.....	56
4.2.3	Flat Field Correction: Countering Van Rhijn Effect and Atmospheric Extinction...57	
4.2.4	Hemispherical to Mercator Projection .....	58
4.3	Visualization .....	59
4.3.1	Public Page.....	60
4.3.2	Administrator Page.....	63
5	Conclusion.....	88
5.1	Recommendations on Future Work .....	88
5.1.1	Firmware/Hardware Recommendations .....	89
5.1.2	Back-end Software Recommendations.....	89
5.1.3	Front-end Software Recommendations .....	89
	References .....	91
	Appendix A: MANGO System Architecture .....	95
	Appendix B: Image Acquisition Program.....	96
	Appendix C: Data Acquisition Program .....	100
	Appendix D: Image Processing Core .....	105
	Appendix E: Image Processing Guide .....	109

Appendix F: Web Interface Code.....	111
Appendix G: Web-based Image Processing .....	112
Appendix H: MANGO User Guide .....	116

# Table of Figures

Figure 1.1: All-sky-imager prototype for the MANGO remote system.....	2
Figure 2.1: Layers of the Ionosphere [7].....	5
Figure 2.2: Northern hemisphere aurora [10] .....	7
Figure 2.3: Aurora [12].....	8
Figure 2.4: Earth's airglow [14].....	9
Figure 2.5: Aircraft Communication via Ionosphere [16].....	10
Figure 2.6: Ionosonde [17] .....	10
Figure 2.7: Solar storm sends charged particles toward Earth's atmosphere [18].....	11
Figure 2.8: Radio waves reflection and absorption [20] .....	12
Figure 2.9: Angle of View by Focal Length for Full-Frame Sensors.....	14
Figure 2.10: An Image with Noticeable Barrel Distortion [22] .....	15
Figure 2.11: ESO's VLT image taken with a circular fisheye lens [24] .....	15
Figure 2.12: All-sky image before any post-processing [23] .....	17
Figure 2.13: Van Rhijn Effect.....	18
Figure 2.14: Atmospheric Extinction because of Increased Air Mass .....	19
Figure 2.15: All-sky image with star calibration [23] .....	20
Figure 2.16: All-sky image after star removal [23] .....	20
Figure 2.17: All-sky image after being projected onto a geographic grid [23] .....	21
Figure 3.1: Camera Control Flow of Control .....	27
Figure 3.2: Projecting an all-sky image onto a geographic coordinate system [23].....	28
Figure 3.3: Lens Function Generated from Star Data.....	29
Figure 3.4: Star removal method.....	31
Figure 3.5: Representation of a star in pixels [32] .....	32
Figure 3.6: Fisheye Projection of 2D objects on to a 1D recorded image.....	34
Figure 3.7: Projecting a fisheye image orthographically .....	35
Figure 3.8: Mercator Projection World Map.....	36
Figure 3.9: Airglow Layer Geometry.....	37
Figure 3.10: Latitudes and Longitudes on a Mercator Image.....	38
Figure 3.11: Parallel Scaling in Mercator Projection.....	38
Figure 3.12: Interpolation requirement to estimate white (undefined) pixels .....	39

Figure 3.13: Processing multiple images in real-time efficiently .....	42
Figure 3.14: MANGO Website Structure .....	43
Figure 3.15: Push-Pull Flask Framework .....	44
Figure 3.16: Public Section Website Structure.....	44
Figure 3.17: Administrator Section Website Structure.....	46
Figure 4.1: Image Acquisition Raw Image Result.....	53
Figure 4.2: Image Acquisition Raw Image Compared to TheSky6 Software.....	55
Figure 4.3: Raw Image Result versus Calibrated Image Result .....	56
Figure 4.4: Star Removal Results.....	56
Figure 4.5: Overlaying the background correction filter on the all-sky image .....	57
Figure 4.6: All-sky image after the background correction filter has been applied.....	58
Figure 4.7: Hemispherical to Mercator Projection Result.....	59
Figure 4.8: Public Page Top Menu Bar .....	60
Figure 4.9: MANGO Public Main Page .....	61
Figure 4.10: What's MANGO? Information page .....	61
Figure 4.11: Ionosphere Information page.....	62
Figure 4.12: Space Phenomena Page.....	62
Figure 4.13: Register Page .....	63
Figure 4.14: Login Page .....	63
Figure 4.15: Administrator Top Navigation Bar .....	64
Figure 4.16: Administrator Side Navigation Bar.....	64
Figure 4.17: Logos .....	65
Figure 4.18: MANGO Main page .....	65
Figure 4.19: Admin Login Page.....	65
Figure 4.20: MANGO System Management Page .....	66
Figure 4.21: Site System Information Status Page.....	67
Figure 4.22: Data Set Viewing Options Button .....	67
Figure 4.23: Ambient Temperature System Graph .....	68
Figure 4.24: CCD Temperature System Graph.....	68
Figure 4.25: Disk Space System Graph .....	69
Figure 4.26: Battery Level System Graph.....	69
Figure 4.27: All Site Settings Page.....	70
Figure 4.28: Add a New Site Option.....	71
Figure 4.29: Modify a Site Option .....	71

Figure 4.30: Remove a Site Option.....	71
Figure 4.31: Add New Site Page .....	72
Figure 4.32: Coordinate Verification .....	73
Figure 4.33: Modify Site Page.....	74
Figure 4.34: Remove Site Page .....	74
Figure 4.35: Calibration Settings Page.....	75
Figure 4.36: Add New Calibration Data.....	76
Figure 4.37: Modify Calibration Data .....	76
Figure 4.38: Add New Calibration Data.....	77
Figure 4.39: Image Information Form .....	77
Figure 4.40: Star Information Form.....	78
Figure 4.41: Lens Function Graph for Star Input Data Check.....	78
Figure 4.42: Calculated Data for Star Input Data Check .....	79
Figure 4.43: Modify Calibration Data Page.....	80
Figure 4.44: Schedule Settings Page.....	81
Figure 4.45: Add Schedule Option .....	81
Figure 4.46: Modify Site Option .....	82
Figure 4.47: Add New Schedule Page.....	82
Figure 4.48: Add New Schedule Form .....	83
Figure 4.49: Modify Schedule Page .....	84
Figure 4.50: Account page .....	85
Figure 4.51: Modify Account Settings Option.....	85
Figure 4.52: Add New Administrator Option.....	85
Figure 4.53: Administrator Change Password Page .....	86
Figure 4.54: Add New Administrator Page.....	86

# 1 Introduction

This chapter introduces the project including the goals and tasks that were accomplished. We also introduce the sponsoring company of our project along with their business domains.

## 1.1 Project Description

Atmospheric science is the subset of earth science which deals with the study of the atmosphere, its processes, the effects other systems have on it and the effects it has on other systems. Since temperature profile is empirically identifiable and uniform around the earth, it is often used as a metric to divide the atmosphere into five main layers based on their distinct properties. These are, from highest to lowest in altitude, the exosphere, thermosphere, mesosphere, stratosphere, and troposphere [1]. Although altitude is one metric for classification of the atmospheric layers, several layers are classified based on other metrics such as ozone density, solar ionization and gas mixture properties, for example the ionosphere and the magnetosphere [2]. The ionosphere, which lies between 85 km and 600 km in altitude, is part of the atmosphere which comprises the mesosphere, thermosphere and exosphere. One distinct property of the ionosphere is the vast presence of ions and electrons in it [2]. Because of the charged nature of the particles that exist in the ionosphere, the Earth's atmosphere is susceptible to turbulence triggered by enhanced flux of energetic electrons emitted by the Sun. This turbulence is often referred to as an ionospheric storm. Such ionospheric storms cause emission of photons in the Earth's upper atmosphere causing a natural light to be seen. This phenomenon is known as Aurora. Airglow is another upper atmospheric phenomenon which is caused by solar radiation. Solar radiation energizes the atmospheric atoms and molecules during the day which relax later at night, thereby emitting photons [1].

A clear understanding of these phenomena is important to scientists because solar storms can disrupt satellite telecommunications as a result of large disturbances in the upper atmosphere. Due to their potential to cause telecommunication disruptions, the effects of solar storms on Earth deserve public attention and the topic should be addressed in the high school science curriculum. However, current high school science curriculum does not discuss these topics in detail. This project intends to address this societal issue by facilitating a system to observe airglow and auroras

## INTRODUCTION

to increase awareness so that students consider careers in space science and upper atmospheric phenomena.

SRI International has initiated an educational outreach program called MANGO (Mid-latitude All-sky-imager Network for Geophysical Observation) which is geared toward getting students interested in careers in space science and upper atmospheric phenomena. This program relies on data collection from real time observations of airglow and aurora. The institute has developed a camera prototype along with an outdoor enclosure and dome assembly. This camera, tested at the McDonald Observatory in Texas, was capable of making useful observation of ionosphere airglow emissions. This first system (as shown in Figure 1.1) is in the process of being installed at Pescadero High School. However, data from the system needs to be collected and processed so that observed upper atmospheric phenomena can be localized on the map of the Earth. Subsequently, these observations can be used fruitfully by participating high schools as well as interested scientists in the community.



Figure 1.1: All-sky-imager prototype for the MANGO remote system

## INTRODUCTION

Thus, the goal of the project was to architect and formulate a system which allows researchers and students to observe mid-latitude airglow and auroras from multiple sites in the United States. To accomplish the project's goal the three objectives were formulated. The three objectives are data acquisition, image processing, and visualization. Data acquisition allows gathering of real-time camera data and other system health metrics and transfers the data to a computer for image processing. The image processing techniques transform the spherical image into an understandable image with geographical coordinates eliminating noise and other undesired effects. Lastly, the visualization objective allows the processed images and the acquired system information to be displayed on a website in quasi-real-time. The website developed was composed of a public and administrator web interface. SRI International will be using the public website as their main way to interact with the schools and to communicate its program to the larger scientific community. Upon completion of this project, the group has advanced SRI International's goals of enabling the means of educational outreach of the MANGO system.

### **1.2 Sponsor Description**

SRI International, founded as the Stanford Research Institute in 1946 by Stanford University, is an independent nonprofit organization located in Silicon Valley, California. The institute gained its independence from Stanford University in 1970 and changed its name to SRI International in 1977. The institute is committed to the discovery and to the application of science and technology for knowledge, commerce, prosperity, and peace. SRI International conducts research that is client-sponsored and develops solutions for government agencies, commercial businesses, foundations, and other organizations. The 63-acre headquarters is fully equipped with specialized equipment and laboratories [3].

SRI works with many companies in Silicon Valley and in the world. Because of its experience with multiple clients, SRI International has helped launch many spin-offs such as Nuance Communications, Intuitive Surgical, and Trapit among others. Apart from conducting cutting-edge research and promoting entrepreneurship, SRI also focuses on licensing patents. Some of the well-known patents that SRI has licensed are those of drugs that have treated infectious diseases such as malaria. They have also licensed computer algorithms for companies such as Google and L3 [4].

SRI has about 2,500 employees worldwide that are composed of corporate, engineers, policy researchers, scientists and technologists. By having a broad variety of employees, the institute has a base knowledge of engineering, science, and technology which is being combined with policy,



## INTRODUCTION

growth, and business strategy. Having such a broad range of employees, SRI has organized the institution into divisions with different focuses and specialties to meet its clients' needs. These divisions are the biosciences, information and computing sciences, physical sciences, policy, and the engineering and systems group. Our group will be working under the Engineering and Systems Division which focuses in the discovery, invention and innovation in the fields of geospace studies, robotics, communications, radar and sensing, marine technology, space technology and integration, and quantum sensing [5].

## 2 Background

The following chapter is a collection of our background research. We first describe the ionosphere and its important components. We also discuss phenomena that occur within the ionosphere such as aurora and airglow. We then discuss how telecommunication depends on the ionosphere followed by concluding the chapter by discussing the importance of optics and image processing, and why it is required for ionospheric imaging. We describe the different techniques of image processing that are applied.

### 2.1 Ionosphere

The ionosphere is a region of the upper atmosphere where charges, both positive and negative exist in quantities large enough to influence radio waves [6]. The ionosphere is composed of different layers which are identified by the alphabets D, E, and F. The F layer is divided into two regions, F<sub>1</sub> and F<sub>2</sub>. The division of the layers within the ionosphere is shown in Figure 2.1.

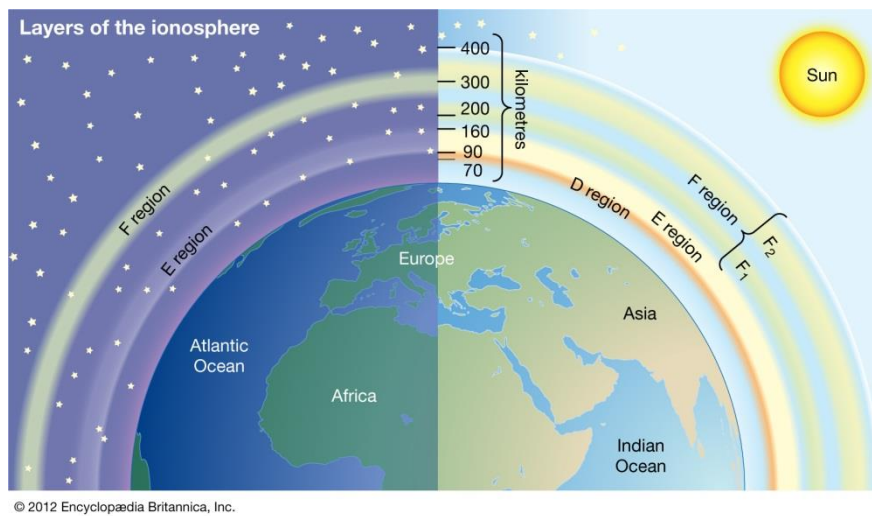


Figure 2.1: Layers of the Ionosphere [7]

The D region is the lowest region in the ionosphere which is located at altitudes between 65km and 95km. In this region, free electrons disappear at night due to the recombination of oxygen ions which causes the formation of electrically neutral oxygen molecules. [2] Ionization is one of the most important components in this region because it is the main cause of the absorption of the high frequency radio waves reflected by the higher layers [6]. High frequency radio waves will be

## BACKGROUND

explained in more detail later in this chapter. The E region, also known as the Kennelly-Heaveside layer, is located in the altitudes between 95km and 150km. Unlike the D region, the E region remains the same at night. It is this layer that is responsible for radio waves reflections [2]. Lastly, the F region has the greatest composition of free electrons [2]. During the day, both  $F_1$  and  $F_2$  layers disappear due to the variation of electron concentrations within the F layer varying between day and night. The  $F_2$  layer is the highest and most ionized reflective layer; it is the most important region for radio-wave systems. This region can reflect radio frequencies up to 35MHz [8].

Photoionization and diffusion are two principal processes that occur in the ionosphere. The electrical activity in the ionosphere is caused by photoionization, which is the ionization caused by light emissions. The term ionization is where electrically neutral atoms are converted to electrically charged atoms [9]. Ions and electrons produced at high altitude are free to diffuse downward, guided by Earth's magnetic field [2]. The ionosphere is largely influenced by the magnetic field because the Earth acts as a giant magnet. Due to the large amount of electrons, there are disturbances in the ionosphere. Some of the disturbances are X-rays, polar caps absorption, geomagnetic effects, and lightning. For the purpose of this project, auroras and airglow which contribute to geomagnetic effects will be described in more detail.

### ***2.1.1 Aurora***

Aurora is a luminous phenomenon that occurs within the ionosphere. This phenomenon occurs both in the northern and southern hemisphere. In the Northern hemisphere, it is known as aurora borealis, aurora polaris, or the northern lights. In the Southern hemisphere auroras are known as aurora australis or the southern lights. Figure 2.2, shown below, displays the Earth's full North Polar auroral oval, which is an image taken in ultraviolet light by the U.S. Polar spacecraft over northern Canada, April 6, 1996. The color-coded image below shows dayside and nightside simultaneously.

## BACKGROUND

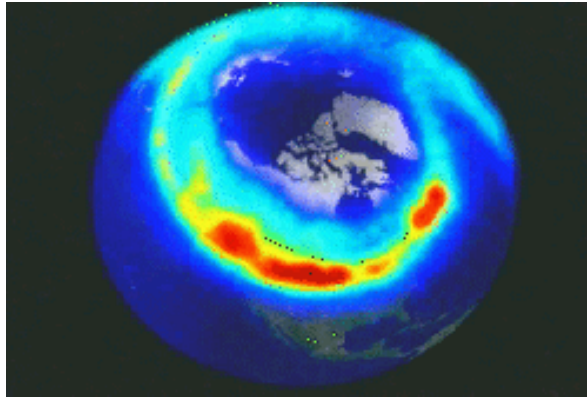


Figure 2.2: Northern hemisphere aurora [10]

Auroras are a manifestation of violent space weather. The sun, located about 149.60 million kilometers away from the Earth, sends plasma into space. When the plasma passes by the Earth, it causes disruption in the magnetic field. The solar energy coming into Earth's magnetic atmosphere causes the magnetic field to reconfigure. Auroras are caused by the interaction of energetic particles (electrons and protons) of the solar wind with atoms of the upper atmosphere [1]. Auroras have the forms of luminous curtains, arcs, bands, and patches. Auroras receive their energy from charged particles traveling between the Sun and the Earth along bundled, ropelike magnetic fields [1]. These particles collide with oxygen and nitrogen atoms causing the formation of free electrons and ions.

Free ions emit light radiation with different wavelengths and this is the reason one can observe different colors in the aurora as shown in Figure 2.3. Oxygen ions cause red and yellow light to radiate. Nitrogen ions radiate red, blue and violet light. In regions of the atmosphere where both oxygen and nitrogen are present, a green color is seen [11]. These colors can be observed at different altitudes. For example, the color blue and violet are usually seen at altitudes less than 120km in the atmosphere. The color green can be seen between altitudes of 120km and 180km in the atmosphere. Lastly, the color red is seen at an altitude 180km.

## BACKGROUND



Figure 2.3: Aurora [12]

NASA launched a mission in 2007 called THEMIS to study the auroras. “Substorm processes are fundamental to our understanding of space weather and how it affects satellites and humans in the magnetosphere,” said Vassilis Angelopoulos, THEMIS principal investigator at the University of California's Berkeley Space Sciences Laboratory, in Berkeley, California [11]. Auroras are part of substorms. During substorms, the solar wind overloads the magnetosphere with too much energy and the stretched magnetic field lines snap back like an enormous slingshot, energizing and flinging electrically charged particles towards Earth [13]. The electrically charged particles bring unwanted electrical discharges. The electrons cause currents that can disturb power grids, satellites, air travel, and GPS signals. These unwanted charges can even cause blackouts by overloading equipment and causing short circuits. The disruptions the auroras cause will be described in more detail later on in this chapter.

### ***2.1.2 Airglow***

Airglow is another phenomenon that occurs in the ionosphere. Airglow is caused by the solar radiation which energizes the atmospheric atoms and molecules during the day. Just like in the case of auroras, the sun provides the energy required to excite the atoms to produce emissions at a certain wavelength for airglow. Airglow is caused chemically when a nitrogen atom combines with the oxygen atom it forms a molecule called nitric oxide (NO). When this occurs, a photon is emitted at different wavelengths. The radiation that is emitted from the atoms is what causes the visible part of the electromagnetic spectrum. Usually the color that is visible is shade of blue color as shown below in Figure 2.4.

## BACKGROUND



Figure 2.4: Earth's airglow [14]

Observations of the airglow and aurora allow scientists to better understand the spatial distribution of charged particles in the ionosphere and to characterize their dynamics.

### 2.2 Ionospheric Effects on Radio Communication Systems

Solar radiation ionized the region of the atmosphere called ionosphere. During the ionization process electrons are removed from neutral atoms or molecules to leave positively charged ions and free electrons. The electrons released during ionization in the ionosphere are responsible for the refraction of high frequency (HF) radio waves, as well as their reflection back to earth [15]. As the density of the electrons increases in the ionosphere, more frequencies are reflected back to earth. The ionosphere's ability to reflect radio waves has been exploited as a medium to reflect radio signals around the curve of the Earth for communications between ships and the shore, trans-oceanic aircraft communication, and military surveillance systems. Figure 2.5 shows the wave propagation between a communication transporter and an aircraft. This type of communication is done via the ionosphere.

## BACKGROUND



Figure 2.5: Aircraft Communication via Ionosphere [16]

Several techniques have been used to investigate the ionosphere, and the most prominent instrument used is the ionosonde. The application of an ionosonde can be seen in Figure 2.6. An ionosonde is a high-frequency radar which is used to send short pulses of radio energy vertically into the ionosphere. The delay time between the transmission and the reception of the pulses is recorded via the ionosonde, over a spectrum of frequencies. The different layers of the ionosphere affect radio wave frequencies in different manners. The E, F<sub>1</sub>, and F<sub>2</sub> regions tend to refract HF waves and reflect LF waves, while the D region attenuates or absorbs them [15]. In high frequency radio wave propagation, the F<sub>2</sub> region appears to be the most important layer of the F section of the ionosphere because in contrast to the F<sub>1</sub> region, which is only present at certain times during the solar cycle, F<sub>2</sub> is present 24 hours of the day; and its property of reflecting the highest frequencies in the HF range can be exploited [15].

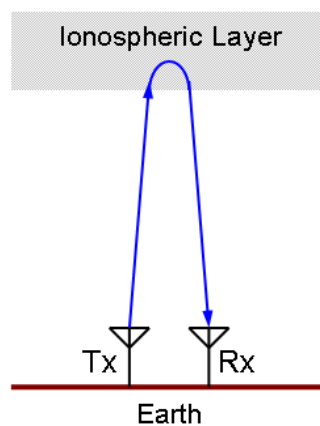


Figure 2.6: Ionosonde [17]

## BACKGROUND

### 2.2.1 Ionospheric Variations

Ionospheric variations have serious effects on satellite communications. Therefore, it is necessary to look at each variation and understand its effect on radio communication systems. Ionosphere variations include: solar cycle variations, seasonal variations, latitude variations, and daily variations.

Solar cycle variations are caused by the periodic rise and fall activities of the Sun, which affect high frequency communication devices. As more radiation is emitted from the Sun during a normal period of greatest solar activity in the 11-year solar cycle, also known as solar maximum, more electrons are produced. This increase in free electrons in the ionosphere causes higher frequencies within the high frequency band to successfully propagate. However, during the period of least solar activity in the 11-year solar cycle, solar minimum, lower frequencies are reflected back to Earth. When solar radiation is at its maximum, there is a greater chance for solar flares. During solar flares, which are caused by important explosions on the Sun, billions of tons of superhot gas holding charged particles are released toward the Earth (as seen in Figure 2.7) and this causes the D region of the ionosphere to ionize. This ionized D region is then responsible for the absorption and attenuation of high frequency waves [15].

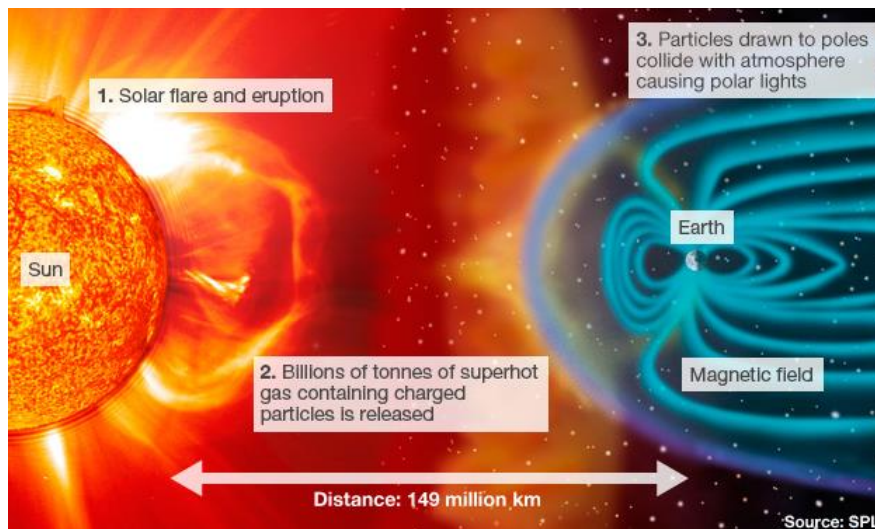


Figure 2.7: Solar storm sends charged particles toward Earth's atmosphere [18]

The cause of seasonal variations is simple to understand as it is related to seasonal change between summer and winter. During summertime, the E region has the ability to reflect more frequencies as compared to wintertime. This happens because solar radiation is greater during summer. However, the seasonal behavior of the F region of the ionosphere is complex and this complexity is known as the seasonal anomaly. This is due to the fact that at solar minimum cycle, the range of frequencies that the F region can reflect or refract during summertime is greater than those of the



## BACKGROUND

wintertime. On the other hand, during solar maximum frequencies the opposite occurs and the range of frequencies are greater in wintertime compared to those of the summertime. [15].

Latitude variations are related to site's specific latitude; and latitude location plays an important role in determining the angle of contact between the solar radiation and the Earth. As the latitude increases during the day, solar radiation strikes the atmosphere in a more inclined position. Therefore, the solar radiation has less impact on the ionosphere as the latitude increases [15].

Finally, daily ionospheric variations are due to the changes in solar radiations. During the day, the ionosphere gets more ionized as solar radiations energized its constituents. This attribute is more prominent when the Sun is at its highest elevation at noon. Thus, higher frequencies are more likely to be affected during the day while lower frequencies are more affected at night [15].

### **2.2.2 Radio Waves and the Ionosphere**

In the previous sections, ionospheric variations were discussed with respect to the different regions of the atmosphere. As the ionosphere interacts with solar radiation, its constituents get ionized; this characteristic is exploited in radio waves communication systems by using the atmosphere as a transmission medium. However, ionospheric variations cause radio waves to behave differently when they interact with the ionosphere. Common behaviors are absorption and reflection [19].

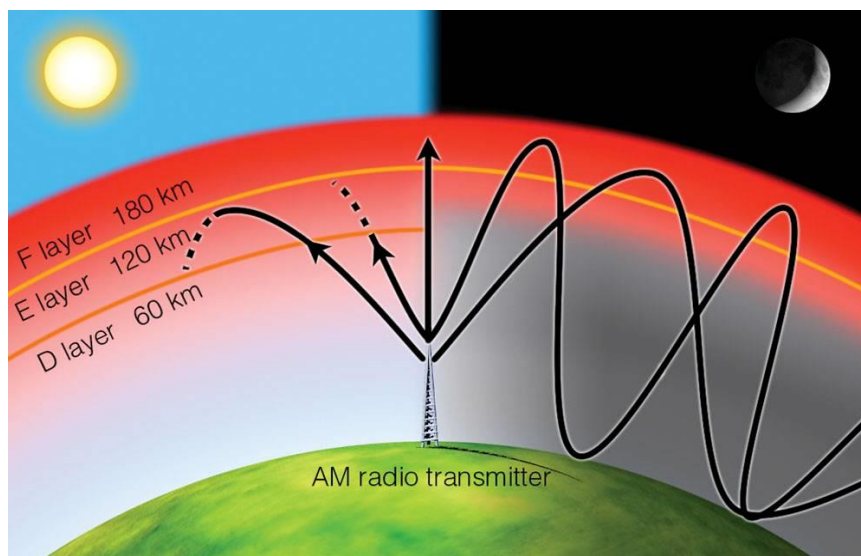


Figure 2.8: Radio waves reflection and absorption [20]

#### *2.2.2.1 Wave absorption*

Radio wave absorption occurs mostly in the D region of the ionosphere (the lower level as shown in Figure 2.8). The reason for this phenomenon to arise is because when a radio wave enters the

## BACKGROUND

atmosphere, it first encounters the D layer. As the wave enters the region, it comes across an important concentration of free electrons in the ionosphere. This causes some wave energy to be transferred to the electrons. Absorption happens when the energy transferred to the electrons attenuates as the result of the free electrons colliding with neutral particles in the upper atmosphere. However, the wave does not undergo any losses if there are no atoms or molecules in the upper atmosphere [19].

### *2.2.2.2 Reflection and Refraction*

As previously discussed, radio waves either reflect or refract in the ionosphere. This is due to the fact that given a certain frequency, the refractive index decreases as we move from a region of lower electron density to a region higher electron density. Therefore, a wave travelling upward can be refracted in the downward direction [19].

The ionosphere's properties play an important role in radio wave communication systems. The reflection property has enabled it to be used as a medium for transferring data from one point to another. However, disturbances within the ionosphere can cause serious communication problems. Observations of the airglow and aurora enable scientists not only to better understand the spatial distribution of the charged particles in the ionosphere; they also allow the understanding of communication disruptions that occur because of variance in the atmosphere.

## **2.3 Optics and Imaging Techniques**

As discussed above, there are many effects of airglows and auroras on satellite communication systems. To study airglows and auroras, scientists use imaging technologies to gather data in the form of photographs. When images are captured, airglow emissions exhibit considerable spatial and temporal fluctuations over large geographic regions. Thus, one requirement of atmospheric imaging is to capture large areas in each image of high quality.

### *2.3.1 Factors Affecting Choice of Optics*

To capture such images, two different kinds of optics can be used – narrow-angle optics and wide-angle optics. There are differences in the two technologies and as a result, both have their pros and cons with respect to auroral imaging. The primary difference in the two is the angle of view and optical distortion. In this subsection, we discuss the physical meanings of angle of view and optical distortion and their relevance to auroral imaging.

## BACKGROUND

### 2.3.1.1 *Angle of View*

In the fields of imaging and photography, angle of view describes the angular extent of a given scene that is imaged by a camera. The wider the angle, more angle of the scene is captured and the opposite the holds true. The angle of view of a lens is a function of the focal length of the lens and is inversely proportional to its tangent in normal lenses. Thus, longer the focal length, narrower will be the angle of view. This can be seen in Figure 2.9.

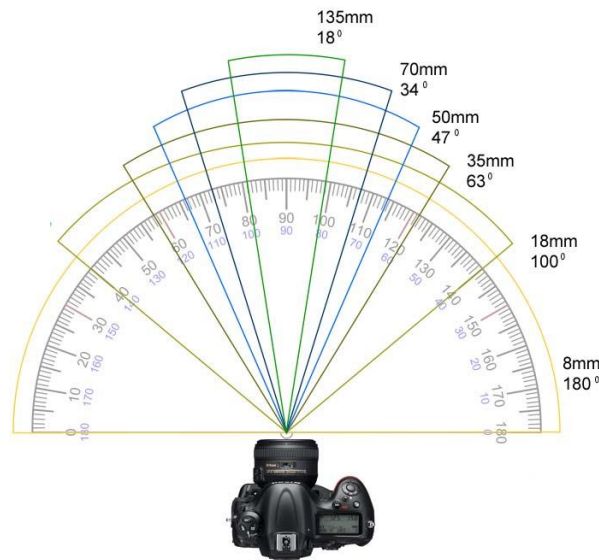


Figure 2.9: Angle of View by Focal Length for Full-Frame Sensors

To this point, it might seem that lenses with wider-angle views are better since they capture more information. While this is true, there are consequences of using wide-angle lenses. Wide-angle lenses are prone to optical distortion.

### 2.3.1.2 *Optical Distortion*

In the field of Optics, distortion is an optical aberration where straight lines in a scene do not appear as straight lines in the image. Majority of camera lenses produce images in line with the law of central perspective. This means that relative to the observer, all the converging lines lead towards a single vanishing point at the center of the image. This kind of projection of three-dimensional space onto a two-dimensional image surface is called rectilinear projection or gnomonic projection. The reason this rule is not obeyed, especially in wide-angle optics, is that the image scale is not constant throughout the entire image field. This means that the focal length of a lens showing distortion changes with the distance of an image point from the optical axis [21]. The effect of distortion when using wide-angle optics can be seen in Figure 2.10.

## BACKGROUND

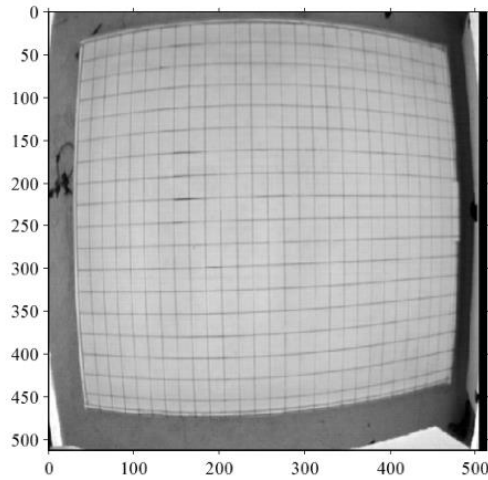


Figure 2.10: An Image with Noticeable Barrel Distortion [22]

### *2.3.2 Use of Fisheye Lens in Auroral Imaging*

Because of effects of optical distortion on the wave structure in the mesospheric airglow emissions, most airglow imaging in the past used relatively narrow-angle optics (typically  $30^\circ$ – $40^\circ$ ) [23]. Although narrow-angle optics had the advantage of limited optical distortion, they provided information about a very limited part of the gravity wave field. As solid-state (CCD) imagers improved with technology, new systems have started employing wide-angle optics which can view emissions over large geographic areas. Fisheye lenses are ultra-wide-angle lenses that achieve extremely wide angles of view by losing the straight lines of perspective as seen in rectilinear images. Figure 2.11 shows a picture taken using a fisheye lens. It is worth noting how the image is different from the conventional rectangular image taken from a rectilinear lens, which appears more natural to a human observer.



Figure 2.11: ESO's VLT image taken with a circular fisheye lens [24]

Due to the optical distortion because of fisheye lenses and the effect of undesirable characteristics of all-sky images, post-processing of these images is required. The need for this and the generally applied techniques are discussed in more detail in the following section.

### **2.4 Image Processing**

The processing of images using signal processing techniques is known as image processing. When we apply such techniques to digital images, the term used to describe such manipulations is digital image processing. Digital image processing techniques can be implementations of algorithms for a wide variety of purposes. Some examples of digital imaging processing techniques are color-to-grayscale conversion, rotation, projection, and pattern recognition. These techniques can be applied to images defined using geometric primitives based on mathematical expressions (such as circles, lines, curves, etc.), known as vector graphics, as well as to images stored in dot matrix data structures, known as raster graphics. Thus, based on the characteristics of the recorded image and the required output, the required image processing techniques can be determined.

#### ***2.4.1 Characteristics of All-Sky Images***

Cameras used for all-sky imaging record rasterized images with a finite number of pixels (picture elements). This means that an image acquired using these cameras have a finite number of dots defined in the horizontal as well as the vertical axis. Since the images are stored as files with pixel-wise information, no information about the geographical coordinates is present in the image. Furthermore, since the images are taken using a fisheye lens, they are not rectilinear. The aforementioned characteristics of the images are a consequence of the imaging technique employed. Apart from these, other effects may be noticeable because of the content of the actual scene. Other systematic errors may also exist in the image which would then need correction. For instance, CCDs are prone to having read-out noise especially when they have faint background levels, which is the noise associated with reading each on-chip amplifier [25]. Figure 2.12 shows how a preprocessed image may appear.

## BACKGROUND

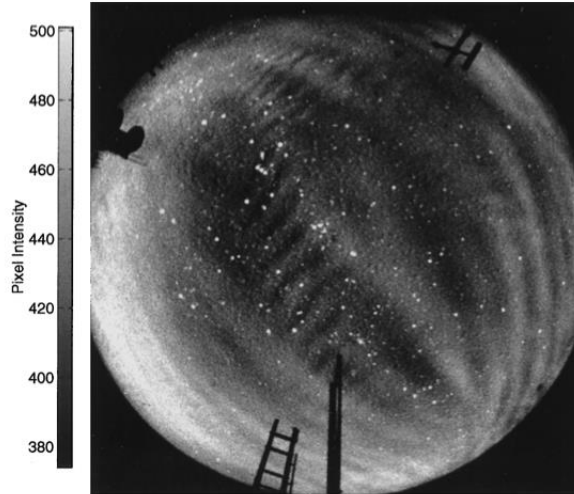


Figure 2.12: All-sky image before any post-processing [23]

### 2.4.2 Read-out Noise and Data Binning

As discussed earlier, CCDs that have faint background levels are prone to read-out noise. Data binning is a pre-processing method that can limit the read-out noise thereby improving the signal-to-noise ratio (SNR).

Typically, each on-chip amplifier measurement would contribute to one pixel in the final image, *i.e.*  $\text{Pixel}_{ij} = \text{Measurement}_{ij} + \text{Noise}_{ij}$ . If data binning is incorporated, a cluster of pixels would be read only once as one super-pixel thereby reducing the read-out noise. Doing so comes at the cost of resolution. For example, if an image originally read as  $1024 \times 1024$  pixels was instead binned  $2 \times 2$  before measurement, each cluster of  $2 \times 2$  pixels would be read as one pixel reducing the image resolution to  $512 \times 512$  pixels where each pixel would have an aggregate value of  $2 \times 2$  pixels, *i.e.*

$$\text{Pixel}_{ij} = \mu_{(2i,2j), (2i+1,2j), (2i,2j+1), (2i+1,2j+1)} + \text{Noise}_{\text{superpixel}}$$

### 2.4.3 Van Rhijn Effect and Atmospheric Extinction

Instruments used for astronomical photography introduce multiple types of noises such as electronic noises and optical noises. Readout noises from the charge-coupled devices (CCDs) are electronic noises are one instance of electronic noise; these noises are reduced by setting a convenient binning factor for the CCD's pixels. On the other hand, optical noises are also significant on all-sky images and the most important ones experienced in auroral and airglow imaging are known as the Van Rhijn effect and atmospheric extinction. Garcia et al. describe the Van Rhijn effect as a line-of-sight enhancement of the airglow signals at low elevation angles [23]. When using a fisheye lens for astronomical photography, it is important to note noises and

## BACKGROUND

distortions introduced by the physical characteristic of the fisheye lens for astronomical objects located at significant distances away from the optical axis of the lens. As described in Figure 2.13, objects that are located far away from the optical axis are projected at the edges of the image frame (their projections are illustrated in green). This projection is typical for fisheye lenses which have a near-180° field of view. Image projections at the edges of the frame are victim to Van Rhijn enhancement and atmospheric extinction. The Van Rhijn effect occurs when emissions from the objects at low elevations and away from the optical axis are concentrated onto few of the pixels that are located at the edges of the image frame. This concentration causes the pixels to be brighter than the rest of the pixels that are positioned closer to the optical axis. Therefore, the illumination of the pixels is not evenly balanced. The uneven balance of the brightness among the pixels of the CCD is described by the following equation [26]:

$$I(\theta) = \left[ 1 - \left( \frac{R_E}{R_E + h_{ag}} \right)^2 \sin^2(\theta) \right]^{-1/2} \cdot I(0)$$

Where:

- $I(\theta)$  is the intensity at inclination angle  $\theta$
- $R_E$  is the radius of the earth
- $h_{ag}$  is the height of the airglow
- $I(0)$  is the intensity at zenith

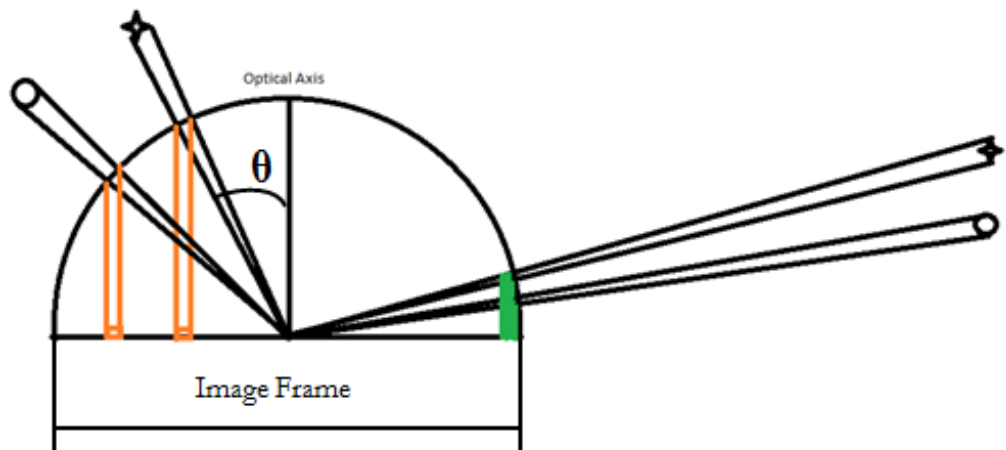


Figure 2.13: Van Rhijn Effect

Van Rhijn effect and atmospheric extinction oppose the effects of each other. While Van Rhijn effect causes CCD's outer pixels to be brighter, atmospheric extinction makes them darker or reduces their true illumination. In other words, in atmospheric extinction the intensity of the emissions coming from distant astronomical objects is reduced. This is due to the fact that the light

## BACKGROUND

coming from object positioned at low elevations is attenuated by the air's composition of dust and gases. This is illustrated in Figure 2.14. Object B is located at zenith and so the light coming from this object travels less distance compared to the emissions coming from objects A and C. Because object A has very low elevation, its light travels a longer distance to reach the observation point O and the intensity of this emission undergoes more attenuation by the composition the air. Therefore, the intensity received by the pixels located at the edges of the image is not the natural or real intensity emanating from the objects at low elevations. This causes the outer pixels to be darker than the rest of the pixels situated closer to the optical axis. This is described by the following equation:

$$I(\theta) = I_{true}(\theta) * 10^{-0.4aF(\theta)} \quad [26]$$

Where: 
$$F(\theta) = \left[ \cos\theta + 0.15 \left( 93.885 - \theta \frac{180}{\pi} \right)^{-1.253} \right]^{-1}$$

$a$  is the atmospheric coefficient

$I(\theta)$  is the intensity at inclination angle  $\theta$  received by the pixels

$I_{true}(\theta)$  is the intensity from the astronomical object

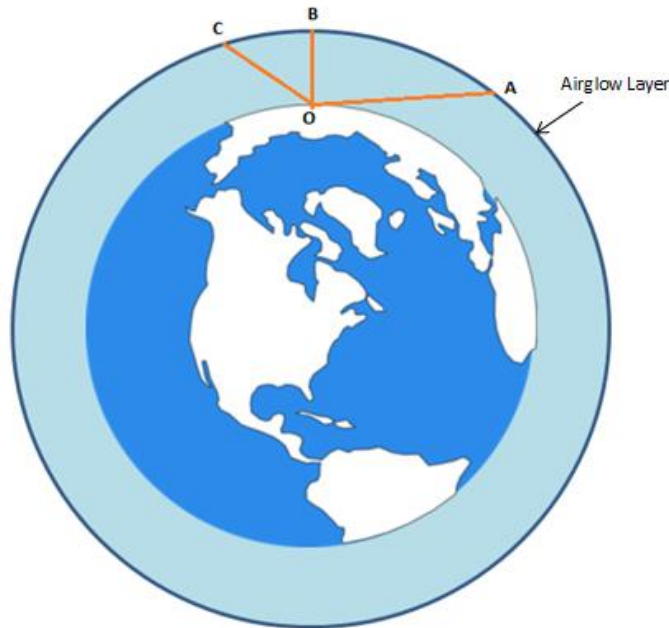


Figure 2.14: Atmospheric Extinction because of Increased Air Mass

### 2.4.4 Required Image Processing

Coordinate Mapping is a technique required to relate distances between pixels in the recorded image to physical distances in the airglow layer. Since the axes may not be oriented along a



## BACKGROUND

geographic orientation, a transformation is required from the original image coordinates to standard coordinates. The reference point for such a spatial calibration can be achieved by using the stars in each image as known reference points in the sky. This step is known as All-Sky Spatial Calibration. After this step, sufficient information about the distances between pixels is available but because of barrel optical distortion due to the fish-eye lens, the distances are not rectilinear. Figure 2.15 shows an example of star identification for all-sky calibration.

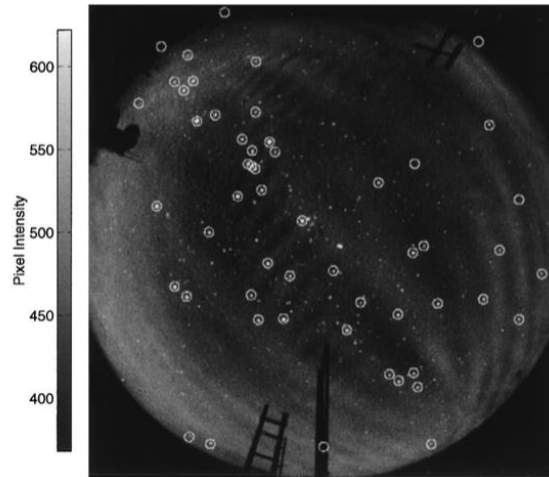


Figure 2.15: All-sky image with star calibration [23]

Thus, a transformation is required to project the hemispherical view onto a uniformly spaced geographical grid. This step would involve a movement of pixels method known as unwarping. The presence of stars in images can cause streaking because of unwarping; thus, star-removal is required in the image processing prior to the unwarping. Figure 2.16 shows the result of star removal.

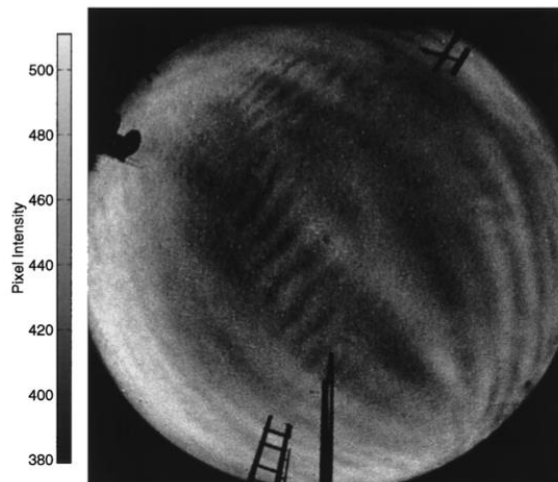


Figure 2.16: All-sky image after star removal [23]

## BACKGROUND

At this stage, the image must be projected onto a geographical grid. The image after being unwarped and projected onto a geographical grid can be seen in Figure 2.17.

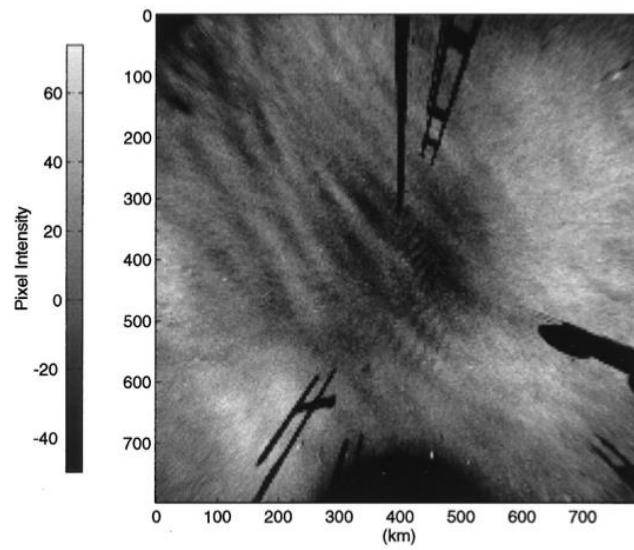


Figure 2.17: All-sky image after being projected onto a geographic grid [23]

Background information about ionospheres, ionospheric effects on communication systems, image acquisition and processing techniques are decisive in the understanding of ionospheric effects such as auroras and airglow. Thus, a design can be proposed to apply these techniques for SRI International's MANGO project.

## 3 Methodology

This chapter covers the study of methods that can and have been utilized to meet the three design objectives of the project. The three objectives are data acquisition, image processing, and visualization of astronomical information gathered using a fisheye lens.

### 3.1 Data Acquisition

Every all-sky imaging system needs to acquire images as a primary goal, but the remote system may also need to monitor other metrics to maintain smooth operation of the automatic imaging system without interruption. The following subsections discuss how all-sky images are acquired and how other system health metrics are monitored.

#### 3.1.1 *Image Acquisition*

The first step in the design involves the acquisition of raw image data from a camera source for further image processing. Depending on the system in place that interfaces with the camera, different methods can be employed to capture images from a live video stream.

Writing appropriate scripts in a higher-level programming language such as Python or C++ can automate the task of acquiring images from a video source. Many software packages also allow automation of such an image capture technique using a graphical user interface (GUI), e.g. Mathworks and LabView (developed by The Mathworks and National Instruments respectively). OpenCV (Open Source Computer Vision) is another open-source alternative with the required functionality. For example, the functions `CaptureFromCAM(...)` or `CaptureFromFile(...)` documented in the OpenCV API can serve our purpose here.

An advantage to using OpenCV instead of NI LabView or a homebrew script is that OpenCV also has many commonly used image processing methods in place to automate simpler tasks such as contrast and brightness adjustment, and image stitching. To acquire images in our system, it is necessary to consider hardware's constraints.

Our imager systems use CCD cameras with driver and software support only for Windows-based platforms. The manufacturer has provided its clientele with a software development kit (SDK) that allows developers to build custom software to operate the camera CCD. In our implementation,

## METHODOLOGY

Python's ctypes module was imported to interact with the camera's driver dynamic linked libraries (DLLs). While acquiring the image, certain settings need to be in place for the camera to work as desired. Some of the general settings and their importance are discussed below.

### *3.1.1.1 Optical Filter Usage*

An optical filter is a device that allows transmission of certain bandwidths of light through it, rejecting the rest. Since we wish to observe only 630nm emissions – the wavelength that corresponds to the F-region emissions from a 250km altitude [27] – a 630nm bandpass filter is placed in the optical path of our imager rejecting light of other wavelengths. Although an optical filter is not a part of the CCD, it is crucial to the all-sky-imager.

### *3.1.1.2 CCD Cooling*

Since the Signal-to-Noise Ratio (SNR) determines the quality of any measurement, we try to minimize noise from all possible sources. In CCD imaging, one of the major sources of noise is dark noise, a noise caused by dark currents generated from thermally generated electrons within the silicon structure of the CCD [28]. Cooling the CCD reduces the dark current dramatically, and thus, to maintain high quality and performance, our CCDs are cooled to  $-5^{\circ}\text{C}$  using a thermoelectric cooling device. Since our CCD camera manufacturer provides us with thermoelectric cooling hardware installed, a Python function was created in our camera software which facilitates CCD cooling to a target temperature.

### *3.1.1.3 Data Binning Implementation*

Another source of noise in CCD cameras is read-out noise – a type of noise generated in the process of converting CCD charge carriers into voltage signals for quantification and the following analog-to-digital (A/D) conversion. In our camera software, we have facilitated the user to be able set a variable binning to reduce read-out noise. The default binning in our imagers is set to  $2 \times 2$  since this reduces read-noise significantly while maintaining the resolution required for us to post-process the image without losing interesting features of the all-sky image.

### *3.1.1.4 Exposure Time Considerations*

Since our recorded images will be a result of only 630nm wavelength light, the flux reaching the CCD is a fraction of the total illumination. This means that a “snapshot” image where the camera shutter is open for only a fraction of a second would result in an image with minimal information. Thus, our camera software allows the user to set a variable exposure time with a default value of 300 seconds. An empirically determined exposure time of 300 seconds allowed sufficient information to be captured via the CCD without causing much blur since auroras and airglows are

slow-changing phenomena. A long exposure time can increase dark noise which makes CCD cooling even more important.

Once these settings are in place, all-sky imagers are configured to take pictures. Our implementation of this software also saves images in the FITS (Flexible Image Transport System) file format. This file format allows lossless storage of images along with a customizable primary header that is used to store the site latitude/longitude, CCD temperature, binning dimensions, and exposure time.

### ***3.1.2 System Information Acquisition***

Since our remote imaging system must be robust enough to run for arbitrary lengths of time without any intervention, we need to monitor metrics of the computer that controls the camera. This way, pre-emptive actions can be taken to prevent downtime. A nonresponsive computer would result in a defunct imaging system, and so we monitor 3 critical metrics of the imager-controlling computer: (1) Power Status, (2) Disk Space, and (3) Ambient Temperature.

Disk space can be at risk during long network interruptions where all the images gathered are cached on a local disk, and not backed-up on a server computer. Power outages can also happen erratically. Disk space and power status are monitored from a Python script by accessing Kernel32.DLL (a Windows DLL which exposes most Win32 base APIs to applications). If the system starts to overheat, hardware damage could also occur and thus the ambient temperature is also monitored. Since not all computers currently come with ambient temperature sensors, an external USB sensor is used and our software parses the data that the USB sensor saves periodically to a CSV file.

In this manner, the image data and system information are acquired. SRI International's proprietary Python-based data transport layer is also in place on the remote all-sky imaging system which transports data from the remote computer to the server computer over Ethernet.

### ***3.1.3 Integrating Image and System Information Acquisition on Remote Site***

As discussed in the previous sections, data acquisition is done on a remote system. Due to this requirement, it is necessary to automate data acquisition for both, the images and system information. Automating data acquisition on the remote system requires a schedule that can dictate when image capturing should begin and end. The program designed by us such that information about the ambient temperature and the solar zenith angle (the angle the sun makes with the

## METHODOLOGY

camera's optical axis) are monitored to avoid any damage to the CCD camera. If the conditions for both the ambient temperature and the solar zenith angle are not met, the system prevents the CCD from any exposure.

The flow chart below describes the automated method for simultaneous image and system information acquisition on the remote site. The technique has two parts. The first part, the initialization part, is executed when the program runs for the first time. Thus, when the program starts, it initializes the variable paths to the schedule file, the calibration information file, and the ambient temperature file. These files contain information such as the length of exposure, the binning factors, the target CCD temperature, the ambient temperature, the site name and the longitude and latitude of the site. After the program extracts the information from the CSV files, it sets the values of the important variables, which are necessary for the proper functioning of the camera, and creates a capture schedule. Capture schedule is created using calculations for the solar zenith angle. The calculations from the zenith angle are used to determine the position of the Sun at certain times of the day given a location. This is important because camera's exposures are done at night, and daytime exposures can be harmful to the camera CCD. Solar zenith angle is specific to each site; therefore, site's longitude and latitude are used in the calculations. However, longitude  $\lambda$  is the longitude of Earth from the vernal equinox. Its calculation includes determining the time from the previous vernal equinox  $t_v$ , with respect to the current given day, and the length of the year  $T$ .

$$\cos\theta_o = \sin\phi\sin\delta + \cos\phi\cos\delta\cos h \text{ (Solar zenith angle equation)}$$

Where,  $\phi$  is the imager latitude

$h$  is the hour angle

$\lambda = 360^\circ \frac{t_v}{T}$  (longitude)

$\epsilon = 23.5^\circ$  (Earth's tilt)

$$\sin\delta = \sin\epsilon \cdot \sin\lambda$$

After a capture schedule has been created, the camera is turned on when the web-power switch is switched on. The second section of the program is designed as a never-ending routine, which loops through subsections of the program and checks its dependencies. The results of these dependencies are used to determine which steps to take next in the progression of the program. As the first section of the program has already initialized the variables, the never-ending loop continuously checks for the updates on the schedule CSV file. This is done once a day, preferably in the afternoon because the camera is off. The binning factors, exposure time, and precooling lengths are updated accordingly. The program also constantly checks the current time to determine the precooling time; and when it is time to precool, the camera is turned on; the camera

## METHODOLOGY

is connected and it starts precooling to the desired target CCD temperature. Precooling session before exposures is important because lower CCD temperatures reduce the occurrence of hot pixels. When the solar zenith angle becomes greater than the threshold zenith angle, the camera starts taking images and saves them to a predetermined directory. However, if during the exposure the ambient temperature inside the enclosure increases above the threshold temperature, the web-power switch is switched off and so the camera is turned off. As the ambient temperature drops below the threshold, the camera is turned on again and the exposure resumes if the zenith angle is still greater than the threshold.

The camera control program monitors multiple sub-processes which are all executed at different time intervals. This was done using Python's multithreading module. Several threading instances are used to execute the sub-processes; these sub-processes are then monitored to ensure that they all run as intended: system information file updates, precooling begins, and images are captured as per schedule.

## METHODOLOGY

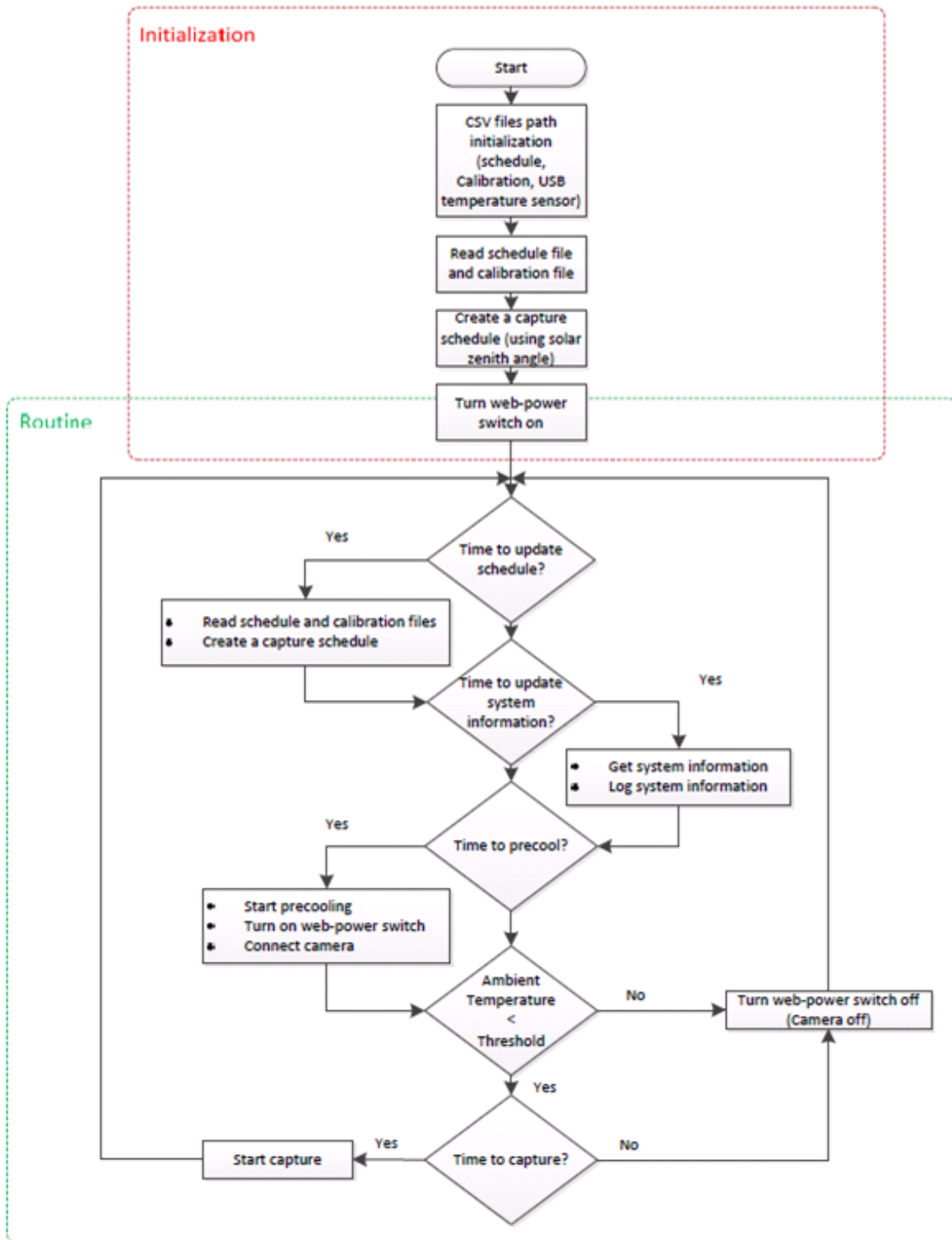


Figure 3.1: Camera Control Flow of Control

### 3.2 Image Processing

The previous section discussed how an all-sky imager is set up and how camera and system data are acquired. To make sense out of the images to analyze airglow data, some image processing



## METHODOLOGY

techniques need to be applied on the raw images. Such processing must transform the spherical image into an understandable image with geographic coordinates. Figure 3.2 shows the starting and ending points of such a transformation.

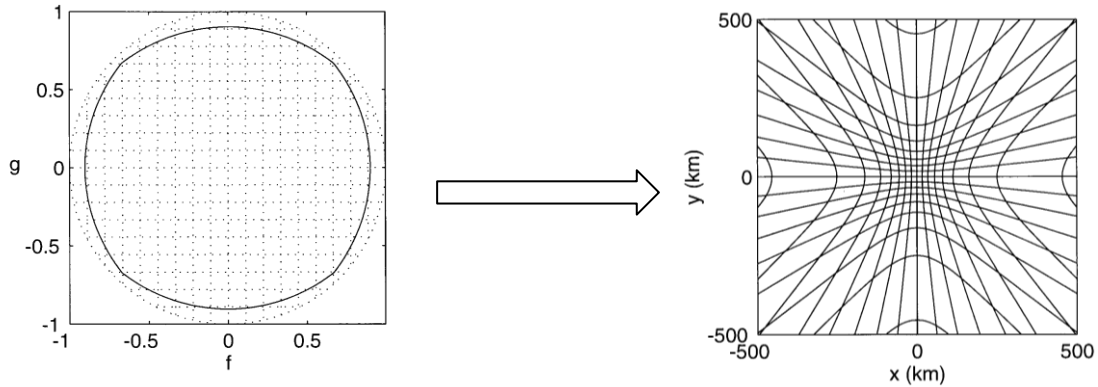


Figure 3.2: Projecting an all-sky image onto a geographic coordinate system [23]

Since the raw image has some aspects that need to be preserved such as auroras and airglows and others that need to be eliminated such as stars in the background, the image processing procedure consists of multiple steps, each with a distinct purpose. Fundamentally, these steps are transfer functions that operate on digital images in a sequential manner.

### 3.2.1 All-Sky Spatial Calibration

To calibrate raw acquired images, the method described in the Garcia, Hapgood, and Taylor study [23] was applied. Since calibration depends on matching star locations between a catalog and the all-sky image, stars identified in the image were compared to a star catalog. TheSky6 catalog software, which contains stars information at a certain time, date, and location in the world, was used as a reference. Hapgood and Taylor recommend using 8 stars; however, a total of 29 stars were identified to assure the accuracy of our calibration algorithm. Using the image, stars' coordinate positions  $(i, j)$  were identified where  $i$  is the column location and  $j$  is the row location of the pixel. Then using the star catalog, the azimuth and elevation of each star was retrieved. Obtaining the azimuth and elevation allowed us to find the azimuth and elevation in any point of the picture.

#### 3.2.1.1 Lens Function

Once we have identified stars in our image by comparing it with a star catalog, a scatter plot of distance of stars from zenith vs. angle of stars from zenith (known as the lens function) is created. Zenith angle is the complementary angle of elevation ( $90^\circ - \text{Elevation}$ ). The scatter plot typically

## METHODOLOGY

has a near linear relationship for ideal fisheye lenses however, in reality, the curvature changes at higher zenith angles. The following figure is an example of what such a scatter plot looks like:

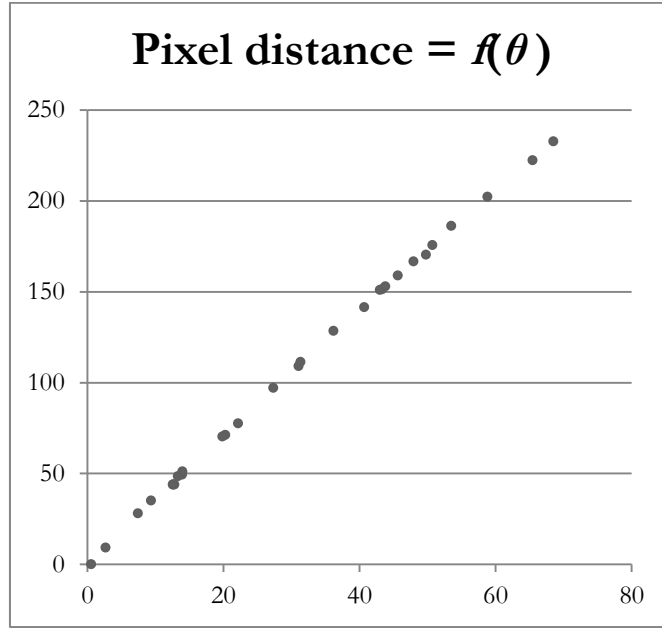


Figure 3.3: Lens Function Generated from Star Data

Thus, a lens function is crucial for calibration since it allows us to estimate the distance of a pixel based on its angle from the zenith (Lens function:  $d = f(\theta)$ ) and vice versa. Since we use a discrete number of stars to correlate zenith distance and angle, a least squares fit of third order is used to estimate a continuous lens function of the following form:

$$d = k_0 + k_1\theta + k_2\theta^2 + k_3\theta^3$$

The coefficients of such a fit  $k_{0-4}$  can be estimated using a least squares fit. To do so, first, a matrix  $\Theta$  is constructed in the following manner:

$$\Theta = \begin{bmatrix} 1 & \theta_1 & \theta_1^2 & \theta_1^3 \\ 1 & \theta_2 & \theta_2^2 & \theta_2^3 \\ \dots & \dots & \dots & \dots \\ 1 & \theta_n & \theta_n^2 & \theta_n^3 \end{bmatrix}$$

Where  $\theta_n$  is the angle from zenith of the  $n^{\text{th}}$  identified star

Then, the Moore-Penrose pseudoinverse is calculated for the matrix  $\Theta$ , which equals  $\Theta^+ = (\Theta^T \Theta)^{-1} \Theta^T$ . The coefficients,  $k_{0-4}$ , are then equal to  $\Theta^+ D$  where  $D$  is a vector of all the distances of the identified stars from the zenith.

## METHODOLOGY

After this, standard coordinates  $f$  and  $g$  are calculated for all the identified stars using the following relationship:

$$\begin{bmatrix} f \\ g \end{bmatrix} = G(\theta) \begin{bmatrix} \sin(az) \\ \cos(az) \end{bmatrix}$$

Where,  $G(\theta) = f(\theta)/f(90^\circ)$  [Normalized lens function]

$az$  is the azimuth of the identified stars.

We understand that the standard coordinates of stars are a transformation of the original coordinates after accounting for shift in origin, rotation of the fisheye circle to align true north and scaling the image so that the lens is uniform in both axes. Thus, this transformation can be represented as [29]:

$$\begin{bmatrix} f \\ g \end{bmatrix} = \begin{bmatrix} a_0 \\ b_0 \end{bmatrix} + \begin{bmatrix} \cos \phi & \sin \phi \\ -\sin \phi & \cos \phi \end{bmatrix} \begin{bmatrix} s_i & 0 \\ 0 & s_j \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix}$$

This transformation can be further simplified into the following form [23]:

$$\begin{bmatrix} f \\ g \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 \\ b_0 & b_1 & b_2 \end{bmatrix} \begin{bmatrix} 1 \\ i \\ j \end{bmatrix}$$

Once we have all star positions in standard coordinates, the constants  $a_{0-2}$  and  $b_{0-2}$  can be determined using a least squares fit similar to how it has been described above. Henceforth, each pixel can be transformed to a standard coordinate system using the  $a_{0-2}$  and  $b_{0-2}$  coefficients. In our implementation, to minimize runtime for image processing,  $\phi$  was derived from the  $a$  and  $b$  coefficients. Subsequently, inbuilt libraries are used to rotate the image by  $\phi$  using the following equivalence:

$$\phi = \tan^{-1} \left( -\frac{b_1}{a_1} \right)$$

or

$$\phi = \tan^{-1} \left( \frac{a_2}{b_2} \right)$$

Scaling by  $s_i$  and  $s_j$  was also reduced to scaling only the  $j$  axis by calculating  $s_j/s_i$  using the following relationships:

## METHODOLOGY

$$s_{ji} = \frac{a_2}{a_1 \tan \phi}$$

or

$$s_{ji} = \frac{b_2 \tan \phi}{b_1}$$

Once these operations were implemented, our calibration was efficient and quick. It is important to note that star identification for spatial calibration needs to be done only once per site as long as the camera orientation and position does not change.

### 3.2.2 Star Removal

In the section above we discussed the transformation process that transforms the arbitrary coordinates to standard coordinates. Before embarking in the transformation that will transform the standard coordinates to geographic coordinates, it is important to remove the stars from fisheye images. The presence of the stars in the image could cause streaking when the image is projected gnomonically. Star removal is an image processing concept that identifies and removes stars from an image. This can be done using various methods and commercial image-processing software packages such as Adobe Photoshop have the ability to perform such operations. For the purpose of this project, we developed a simple algorithm that uses statistical variations in intensity of pixels to identify stars and hot pixels.

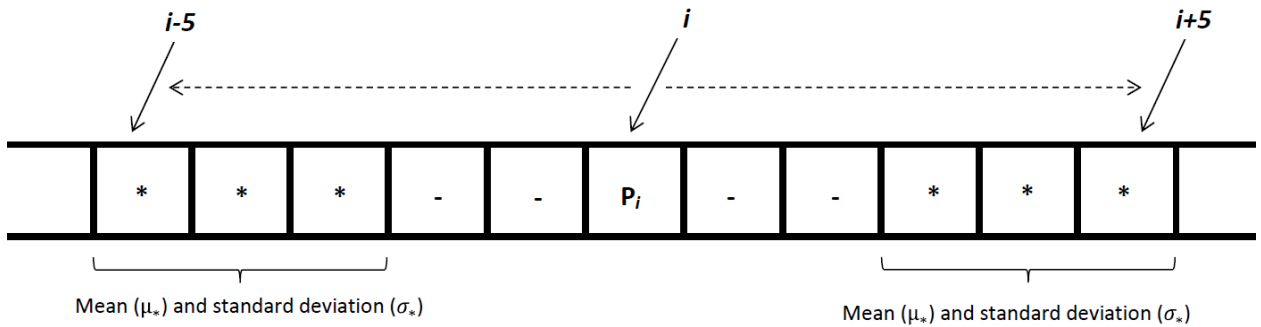


Figure 3.4: Star removal method

The star removal method used is described using the figure above. The algorithm runs through each pixel of the image, row-wise. As the loop goes through each pixel of the image,  $P_i$ , the mean ( $\mu_*$ ) and the standard deviation ( $\sigma_*$ ) of the pixels from the left and the right of each pixel are calculated. To account for the fact that a star may not fit in just one pixel, two pixels are skipped on both sides of the pixel. In other words, the pixels with the symbol (-) located at indices ( $i-1$ ), ( $i-2$ ),

## METHODOLOGY

$(i+1)$ , and  $(i+2)$  are ignored in each iteration. The value of the pixel ( $P_i$ ) is then compared to the sum of the mean and two standard deviations of the neighboring pixels ( $\mu_* + 2\sigma_*$ ); two standard deviations from the mean is used as a threshold to ensure that the value of pixel  $P_i$  is sufficiently greater than the neighboring pixel values for the following step. If the value of ( $P_i$ ) is greater than ( $\mu_* + 2\sigma_*$ ), the value of the pixel is set to -1. Otherwise, the value of pixel ( $P_i$ ) is left as-is and index  $i$  moves to the next pixel and the process is repeated until all the pixels are covered.

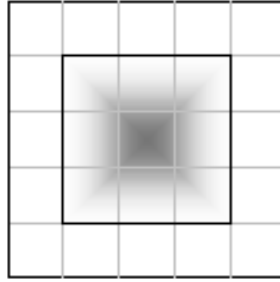


Figure 3.5: Representation of a star in pixels [32]

At the end of this, all pixels with a value of -1 are identified and interpolated over by using a standard bicubic interpolation routine thereby eliminating all stars. The star removal algorithm does not only remove the stars, but it also removes high-frequency disturbances such as hot pixels in the image. Such noise reduction is important for enhancing the SNR and getting rid of the image imperfections.

### ***3.2.3 Flat Field Correction: Countering Van Rhijn Effect and Atmospheric Extinction***

All-sky images even without stars are prone to noise because of variations in pixel-to-pixel sensitivity and distortions in the optical path. A good flat field correction technique is required to account for such variations. These variations can rise from a variety of sources such as hot pixels, vignetting, the Van Rhijn effect and atmospheric extinction. Electronic noise such as hot pixels can be attenuated by taking a master dark frame (an average of multiple images taken with the shutter on the camera) and subtracting this frame from other images. Although this method works well to remove hot pixels, our star removal algorithm does the same since it eliminates intensity deviations similar to those exhibited by hot pixels. Noise from sky sources such as aerosol scattering introduces uneven brightness on pixels of objects especially at low elevations. A typical method to counter this is to obtain a master flat field frame (an average of multiple images of an illuminated white Lambertian surface) and to divide the original image intensity values by the flat field intensity

## METHODOLOGY

values. This method, however, does not account for the Van Rhijn effect and thus, a mathematical approach was used to get rid of such noises produced by sky sources in our design.

To account for such variations, the equations below are used to correct the image after star removal. The equations describe what the observed effects of these phenomena are.

$$(1) \quad I_{VR}(\theta) = \left[ 1 - \left( \frac{R_E}{R_E + h_{ag}} \right)^2 \sin^2(\theta) \right]^{-1/2}$$

$$(2) \quad I_{AE}(\theta) = 10^{-0.4af(\theta)}$$

$$\text{Where,} \quad f(\theta) = [\cos\theta + 0.15 \cdot (93.885 - \theta)^{-1.253}]^{-1}$$

Examining the Van Rhijn effect equation as shown in (1), we see that the intensities of the pixels are not only dependent on the elevation angles but also the radius of the Earth ( $R_E$ ) and the height of the airglow ( $h_{ag}$ ). For our design purposes, the height of the airglow was assumed to be 250km, and the radius of the Earth was 6378km. By using these values, the Van Rhijn correction factor is calculated for each of pixel by calculating the pixel's elevation angle. Each pixel's elevation angle is calculated using the lens function derived above.

Atmospheric extinction causes the intensities of the emissions coming from astronomical sources located at lower elevation angles to not be the true intensities of the objects in question. The intensities are attenuated because of the increased air mass present at lower elevation angles and the lossy nature of air. As discussed before, the atmospheric extinction coefficient  $a$  is dependent on atmospheric conditions at the time of observation. A typical atmospheric value of 0.4 is often used to correct for this phenomenon however, the corrected image must be observed to decide whether the correction is acceptable. For our project we did this by trying several coefficient values in the atmospheric extinction equation for several images that were taken on a clear night, and observing which coefficient gave the most evenly balanced intensity across pixels. During this observation it was determined that an atmospheric coefficient of  $a=0.2$  was most appropriate for correction for our remote site. After determining this, the original image pixel intensities are divided by  $I_{VR} \cdot I_{AE}$  to get a flat-field corrected image.

### ***3.2.4 All-Sky Projection onto a Uniformly Spaced Grid***

Once our hemispherical image has been calibrated and the stars and hot pixels have been filtered from it, the next step involves shifting from the orthographic projection (hemispherical) to a mapping projection so that each pixel from the original image can be positioned on a map of the

## METHODOLOGY

Earth accurately. This section describes two solutions considered to carry this process out efficiently.

### 3.2.4.1 Hemispherical to Rectilinear Projection

To project a hemispherical image onto a rectilinear plane, it is important to observe how a fisheye image is first recorded. The first key difference between fisheye lenses and rectilinear lenses is that fisheye lenses record angular view per unit recorded area where as rectilinear lenses record unit distance per unit recorded area. *i.e.* in orthographic fisheye lenses  $r = f \cdot \sin(\theta)$ , where  $r$  is the distance of the object image from the centre of the image and  $\theta$  is the angle the object forms with the optical axis.

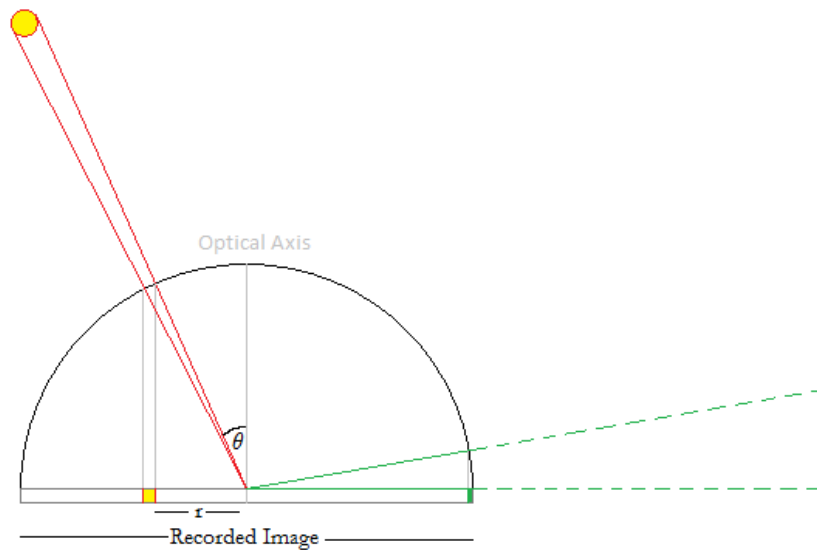


Figure 3.6: Fisheye Projection of 2D objects on to a 1D recorded image

Figure 3.6 shows how a 2D 180° scene is mapped from the real world onto a 1D image array. This same method is applied to record 2D images of the 3D world we live in. Given that the image contains only 630nm exposure predominantly from an altitude of 250km, we map the circular fisheye image back to a rectilinear plane in the following manner:

A pixel ' $p$ ' located at coordinates  $(i, j)$  in a fisheye image with center  $(i_c, j_c)$  will undergo the following transformation in order to be projected gnomonically:

Distance of  $p$  from center,  $r_{(i,j)} = \sqrt{(i - i_c)^2 + (j - j_c)^2}$

$R$  is the Radius of fisheye circle

## METHODOLOGY

From Figure 3.7, we can see that  $\sin \theta = r_{(i,j)}/R$

This yields,  $\theta = \sin^{-1}(r_{(i,j)}/R)$

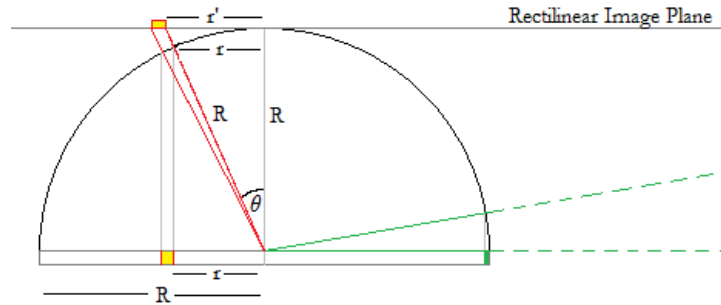


Figure 3.7: Projecting a fisheye image orthographically

We can now identify the new location by calculating it's new distance from the center by applying,

$$\tan \theta = r'/R$$

Applying the above, we get  $\tan(\sin^{-1}(r_{(i,j)}/R)) = r'/R$

Thus,  $r' = R \cdot \tan(\sin^{-1}(r_{(i,j)}/R))$

The individual coordinates  $(i_{new}, j_{new})$  can be calculated by scaling them by the amount the  $r_{(i,j)}$  is scaled to produce  $r'$ .

Thus,  $i_{new} = i \cdot r'/r$

$j_{new} = j \cdot r'/r$

Often lenses are not precise and thus a pixel's distance from the center of the CCD may not be proportional to the sine of the angle (or the angle itself in cases of  $f \cdot \theta$  equidistant lenses) the source makes with the optical axis. In such real cases, the lens function (empirically calculated earlier using star positions) is used to relate pixel distance from the zenith to the angle projected.

### 3.2.4.2 Hemispherical to Mercator Projection

Although the above described method works well for maps that scale distance linearly in both, the North-South direction as well as the East-West direction, web-based maps (such as Google Maps, Leaflet Maps, Bing Maps, etc.) use the Mercator projection. The advantage of using Mercator projection to describe the Earth is that the shapes of landmasses are quite accurate, however a drawback is that sizes farther away from the equator are highly distorted (they appear larger than they really are). A classical instance that exemplifies this distortion can be seen in Figure 3.8. In this image Greenland is projected to be nearly the same size as Africa and much larger than Australia, however, in reality Greenland has an area of 2,166,000 km<sup>2</sup> where as Australia has an area of



## METHODOLOGY

7,692,000 km<sup>2</sup> (approximately 3.5 times that of Greenland's). This distortion in Mercator projection occurs because in reality, towards the poles longitudes converge – however, the Mercator projection aims to keep longitudes equidistant irrespective of latitude. Thus at higher latitudes, countries appear stretched east west. To maintain the aspect ratio of landmasses, the latitudes are also projected farther away than reality.



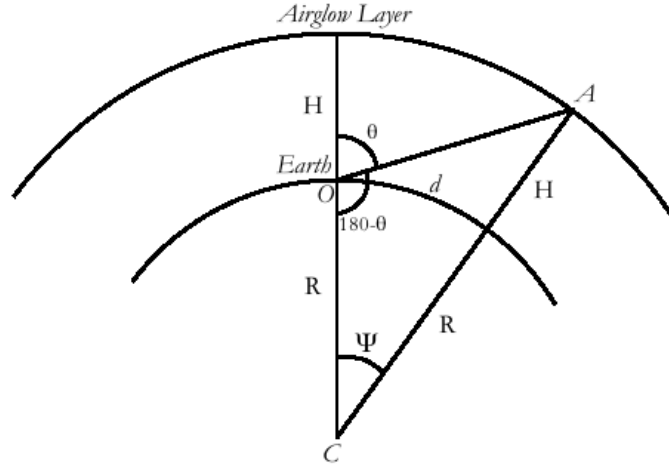
**Figure 3.8: Mercator Projection World Map**

Since images of airglow in our design are also supposed to be projected on web-based Mercator maps, this section describes the equations that govern Mercator projection and how they apply to all-sky images.

For any given calibrated image, any pixel's corresponding final position can be calculated in terms of latitude and longitude in the following manner:

- 1) Given the zenith pixel location and the query pixel location, we can calculate the pixel distance between the two points.
- 2) Using the previously determined lens function, the zenith angle  $\theta$  can be determined using the pixel distance calculated in step 1.
- 3) Using the geometry described in Figure 3.9, the great circle arc distance  $d$  can be calculated in the following steps:

## METHODOLOGY



**Figure 3.9: Airglow Layer Geometry**

- a. Given  $\theta$ ,  $180^\circ - \theta$  is known.
  - b. Then, side  $AO$  can be calculated by applying the Law of Cosines in  $AOC$ .
    - i.  $AC^2 = OC^2 + AO^2 - 2 \cdot OC \cdot AO \cdot \cos(180 - \theta)$
    - ii.  $AO^2 = AC^2 - OC^2 + 2 \cdot OC \cdot AO \cdot \cos(180 - \theta)$
    - iii.  $AO = [AC^2 - OC^2 + 2 \cdot OC \cdot AO \cdot \cos(180 - \theta)]^{0.5}$
    - iv. Of course,  $AO$  would mathematically yield 2 roots but one of the roots can be eliminated analytically since it will be negative.
  - c. Once  $AO$  is known,  $\Psi$  can be calculated using the relationship:
    - i.  $AO / \Psi = AC / (180 - \theta)$
    - ii.  $\Psi = AO \cdot (180 - \theta) / AC$
  - d. Great Circle Arc  $d = R \cdot \Psi$
- 4) Given the distance  $d$ , the final latitude and longitude can be calculated using the following equations (known as the Haversine Equations):

$$\varphi_2 = \sin^{-1} \left( \sin \varphi_1 \cdot \cos \frac{d}{R} + \cos \varphi_1 \cdot \sin \frac{d}{R} \cdot \cos \theta \right)$$

$$\lambda_2 = \lambda_1 + \text{atan2} \left( \cos \varphi_1 \cdot \sin \frac{d}{R} \cdot \sin \theta, \cos \frac{d}{R} - \sin \varphi_1 \cdot \sin \varphi_2 \right)$$

$$\theta = \frac{\pi}{2} + \text{atan2} \frac{\Delta j}{\Delta i} \quad (\text{often known as bearing})$$

Where,  $\Delta j$  is the pixel distance from the zenith pixel in the  $y$ -axis  
 $\Delta i$  is the pixel distance from zenith pixel in the  $x$ -axis  
 $\varphi_1$  is the zenith latitude and  $\lambda_1$  is the zenith longitude

Once the latitude and longitude of every pixel is determined, a Mercator projection image can be created. Given that a Mercator plane image has constant longitudes parallel to the  $y$ -axis and constant latitudes parallel to the  $x$ -axis, a Mercator image would appear as the following:

## METHODOLOGY



Figure 3.10: Latitudes and Longitudes on a Mercator Image

- The top edge of the image is constant latitude  $\varphi_{max} = \max(\varphi_{all})$
- The bottom edge of the image is constant latitude  $\varphi_{min} = \min(\varphi_{all})$
- The right edge of the image is constant longitude  $\lambda_{max} = \max(\lambda_{all})$
- The left edge of the image is constant longitude  $\lambda_{min} = \min(\lambda_{all})$

Although longitudes scale linearly on a Mercator map, latitudes distances scale increasingly at higher latitudes. For example, the shortest ruler-measured distance between the 65° meridian (longitude) and 75° meridian will be the same as that between 75° meridian and 85° meridian. However, the shortest ruler-measured distance between the 65° parallel (latitude) and 75° parallel is much less than that between the 75° parallel and the 85° parallel. Figure 3.11 illustrates how latitudes scale.

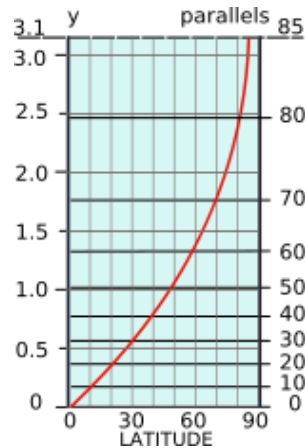


Figure 3.11: Parallel Scaling in Mercator Projection

Mathematically, y-axis scaling can be calculated given a latitude  $\varphi$  using the following equation [30]:

$$y = \ln \left[ \tan \left( \frac{\pi}{4} + \frac{\varphi}{2} \right) \right]$$

Thus,

- The top edge of the image is constant scaling  $y_{max} = \ln \left[ \tan \left( \frac{\pi}{4} + \frac{\varphi_{max}}{2} \right) \right]$

## METHODOLOGY

- The bottom edge of the image is constant scaling  $y_{min} = \ln \left[ \tan \left( \frac{\pi}{4} + \frac{\varphi_{min}}{2} \right) \right]$

Once the image bounds are available in terms of linearly scaling metrics, each pixel located at  $(i, j)$  associated with latitude and longitude  $(\varphi, \lambda)$  can be placed in a Mercator projection image at  $(i_{New}, j_{New})$  using the following equations:

$$i_{New} = \chi \cdot \left( \frac{\lambda - \lambda_{min}}{\lambda_{max} - \lambda_{min}} \right)$$

$$y = \ln \left[ \tan \left( \frac{\pi}{4} + \frac{\varphi}{2} \right) \right]$$

$$j_{New} = \tau - \tau \cdot \left( \frac{y - y_{min}}{y_{max} - y_{min}} \right)$$

Where,  $\chi$  is the image width  
 $\tau$  is the image height

### 3.2.4.3 Interpolation

Since our transformation forward maps initial pixel positions to a final image, not every pixel in the final image maps back to the original image. Pixels from the initial image that should map over multiple pixels in the final image require interpolation techniques to be applied in order to maintain accuracy. One key step in hemispherical to rectilinear/Mercator projection is to use interpolation to find the position of sampled points that do not lie in the center of the pixels. The requirement can be demonstrated as shown in Figure 3.12.

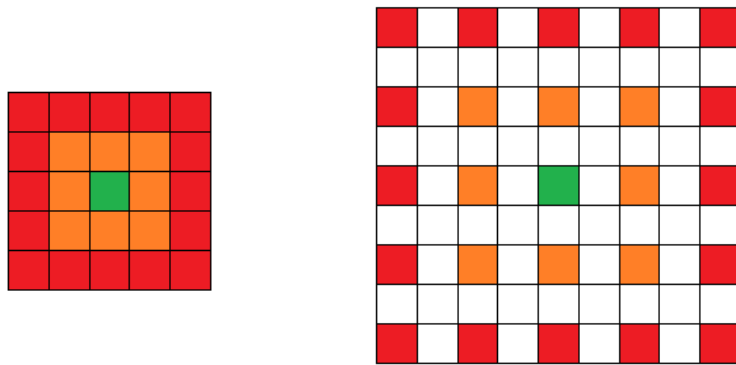


Figure 3.12: Interpolation requirement to estimate white (undefined) pixels

Bicubic interpolation is used to determine the value of data points that do not have an assigned value. As per this method, a third degree polynomial specified in Hermite form fits the given data values in order to obtain smooth continuous values for the unknown values in 2 dimensions.

### 3.3 Real-time Image Processing

The previous section discussed data acquisition for both the images and system information. These acquisitions are done on the remote sites of the all-sky imager system. The acquired images are then processed on the server to correct for the distortions and the noises introduced by the instruments used in all-sky imager architectures.

Before performing any image processing, the server needs to be alert for the availability of new images. Therefore, another program was written to ensure that each newly taken image is processed in a timely manner. The eventual outcome of the overall MANGO system is to acquire real-time observations of space phenomena. Hence, it is important for the design of the image processing program to account for real time processing.

When the CCD camera captures images, using a predetermined exposure time, images arrive at the desired directory at a certain frequency; and these images are to be processed immediately. Thus, it is important to implement a routine that will incessantly check the directory to determine if new images are available for processing. The program implemented here has a routine that also continuously watches the current time to regulate the updates of the list of sites. Thus, when it is time to update the list, the program opens the addsite CSV file and extracts sites list, and sites' statuses (whether or not a site is included in the updated list within the program). The program then regulates each of the included sites' directories for new images, as they arrive from the remote site. If there are new images located, the program obtains their locations and stores them in a waiting queue where they will be located until they are sent for processing by the processing queue. Thus, the processing queue is loaded with images to be processed. The processing queue has a maximum size (of 10 images) which signifies that no more than 10 images will get processed simultaneously. As soon as either the processing queue is full or the waiting queue is empty, the program starts processing each image in the processing queue. Whether or not there are images to process, the program has a never-ending routine that uninterruptedly updates the list of the sites on due time, checks each site for new images and processes the images.

Processing several images at once may take more time than available, and this is not acceptable in the design since the goal of the overall project is to achieve real-time observations of upper atmospheric phenomena. The initial design of the image processing guide program used a multithreaded execution method to process the images. However, the process was slow and did not meet the requirements. In an attempt to improve the processing time, the final design uses a

## METHODOLOGY

multiprocessing execution method. Although both methods parallel execution of code, threads run in a shared memory space. Processing multiple images on shared memory is not efficient because the time of execution increases almost linearly as the number of the images increases. On the other hand, processes run in separate memory spaces. This feature makes the use of multiprocessing efficient since processing multiple images splits the executions across several memory spaces and processor cores; hence making the use of multiprocessing time and space efficient. Our technique for handling this requirement is shown in Figure 3.13.

## METHODOLOGY

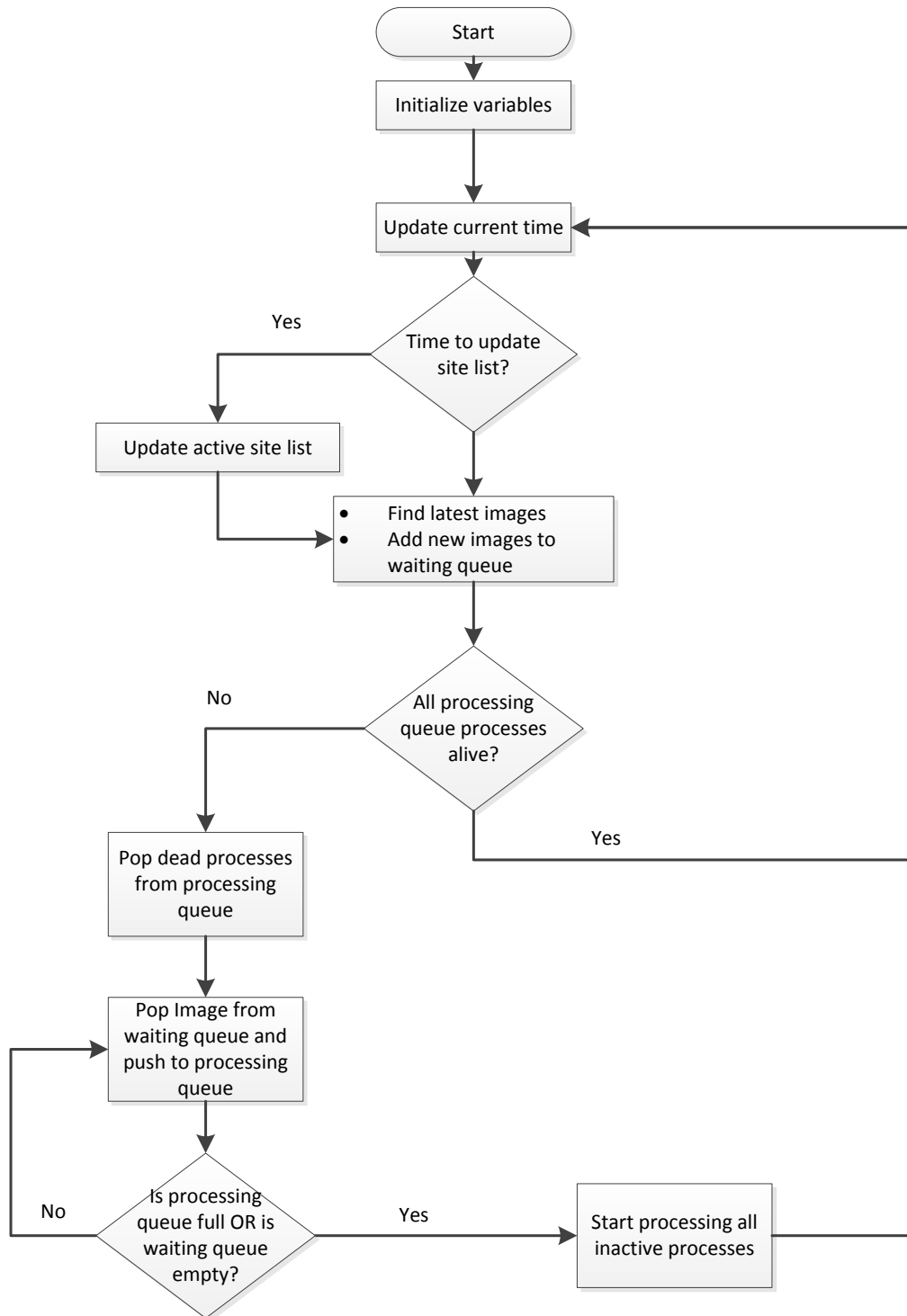


Figure 3.13: Processing multiple images in real-time efficiently

### 3.4 Display and Visualization

The third objective of the project is to display the processed images and the acquired system information. To meet this objective, we designed a website based on SRI International's

## METHODOLOGY

requirements. Their requirements include allowing the public, students, and scientists to be able to observe space phenomena in real time. The administrators, in this case SRI International, also require the ability to control the system through the web. To meet their requirements, a website was developed which is composed of a public and administrator web interface as shown in Figure 3.14. The public webpage displays the processed images in real-time while the administrator webpage displays all the system information obtained through the data acquisition program.

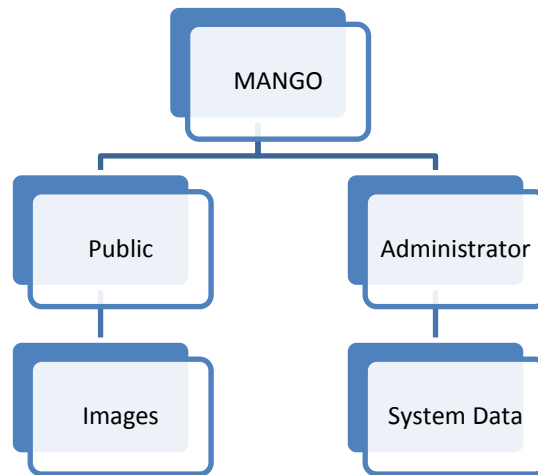


Figure 3.14: MANGO Website Structure

### 3.4.1 *Setting up the Website*

SRI International provided a domain name and a server for hosting the webpage. To create the webpage, a web development framework for Python known as Flask was used. Flask allows the usage of different languages such as Hyper Text Mark-up Language (commonly known as HTML), JavaScript, and Python for coding. It also permits constant interaction between the Python code, HTML, and JavaScript. Flask is a “push” and “pull” framework. This means that Flask uses actions to perform the required process, and then “push” the data to the view layer to render the results. The view layer, in this case is the HTML which is viewed in the web-browser. The “push” and “pull” are methods known as “GET” and “POST”. This means that both the Python code from Flask and the HTML and JavaScript from the view layer are able to “POST” information and “GET” information from each other as shown in Figure 3.15.



## METHODOLOGY

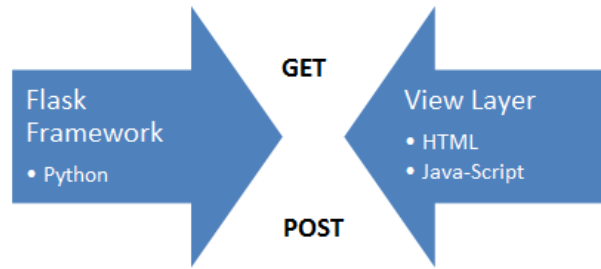


Figure 3.15: Push-Pull Flask Framework

For the styling of the page, Twitter Bootstrap, an open source library, was used for the front-end framework of the web development. Bootstrap is built on responsive 12-column grids, layouts, and components which allow for page resize for any web browser that the user may be utilizing. Bootstrap provided us with all the necessary base template examples for HTML, CSS, and Java-Script files for setting up the display.

For displaying charts and graphs on the website, Google Developers Chart Tools were used. In the API, Google provided the initial HTML and JavaScript examples for setting up the charts.

### 3.4.2 Public Webpage

For combining the research and educational goal, the images obtained through our data acquisition system are processed through our image-processing engine and then displayed on the public webpage. The hierarchy of the pages for the public website is shown in Figure 3.16.

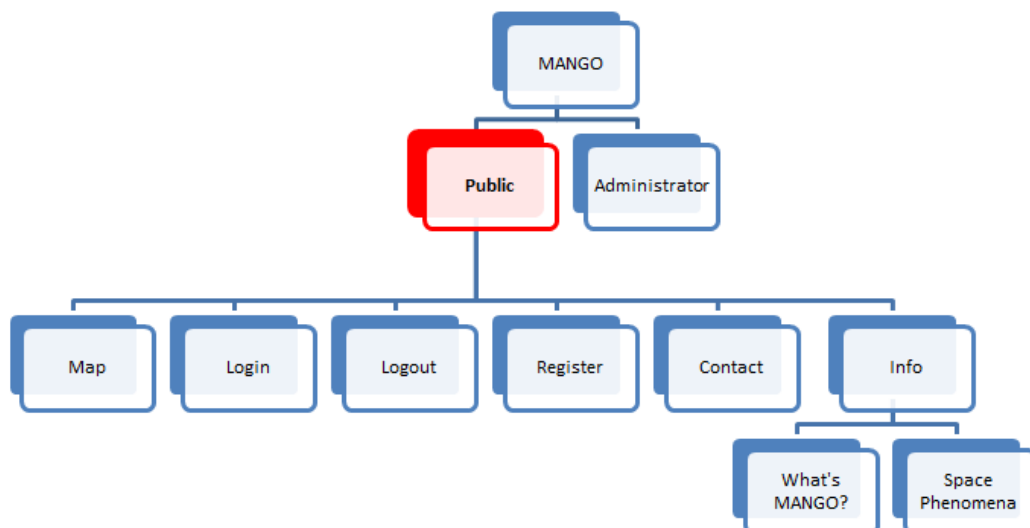


Figure 3.16: Public Section Website Structure

## METHODOLOGY

As shown in the figure, the public website is composed of a map, login, logout, register, contact, and information pages.

### *3.4.2.1 Image Overlay*

The main purpose of the public website is to display in real time space phenomena. The images must be displayed on top of a United States map accurately depicting where the airglows and auroras are observed. To create the map Leaflet, an open-source Java-script library for interactive maps, was employed. Using Leaflet, a Mercator projection map was created of the United States where the processed images are overlaid. The map also includes the state borderlines for each of the states; this is done using the GeoJSON layer which parses data and adds corresponding layers above the map. To set up an accurate state borderline, a multipolygon and multypolyline layer was applied with GeoJSON. These layers require the coordinates of each state's border. Accurate coordinates for each state have to be coded for the line to be overlaid properly on top of the map.

For an image to be overlaid on top of the map Leaflet requires for the southwest and northeast coordinates of the Mercator image to be passed to JavaScript. These coordinates are calculated after the user inserts data for image calibration. After the coordinates are calculated for each site, they are stored in the imageCoord.csv file. Through a Python script the CSV file is read and the data is extracted. Then the information is sent to the HTML's JavaScript. After setting up the map, the images are displayed over the map by creating a new layer.

### *3.4.2.2 User Control*

One of the requirements was the ability for students and other public users to be able to interact with scientists from SRI International. To meet this requirement, a database was developed to keep track of all registered users. The database was created by using SQLAlchemy, an open-source object-relational database management system that utilizes a client/server model, specifically for Python. The server manages the database files, accepts connections to the database from the website, and performs actions that the website will apply. SQLAlchemy (which uses SQLite 3) was chosen over MySQL or PostgreSQL because it is open-source and an easy-to-use database. In the database created, the following information from the user is stored: first name, last name, username, password, and affiliation, and role. The role is never asked to be input by the user. However, the python script assigns the role to the user. Since the user is registering through the public page, an arbitrary variable is assigned to the user during registration and this number is used to identify the registered user as a public user. The username will allow users to publicly interact with scientists without using their full name. The form asks the user to input affiliation. Affiliation is referenced as the school which the user attends or the current location of the user if the user

## METHODOLOGY

does not attend one of the schools in the network. This is asked, because the researchers at SRI International would like to have knowledge from which part of the United States is the user interacting from. The database will aid with future applications SRI International would like to implement of the website. One of the applications which SRI International will be implementing in the future is a blog where students and scientists can comment on the space phenomena which is observed on the map

### 3.4.3 Administrator Webpage

The private webpage displays all the system information obtained from remote sites through the data acquisition software. The hierarchy of the pages for the public website is shown in Figure 3.17.

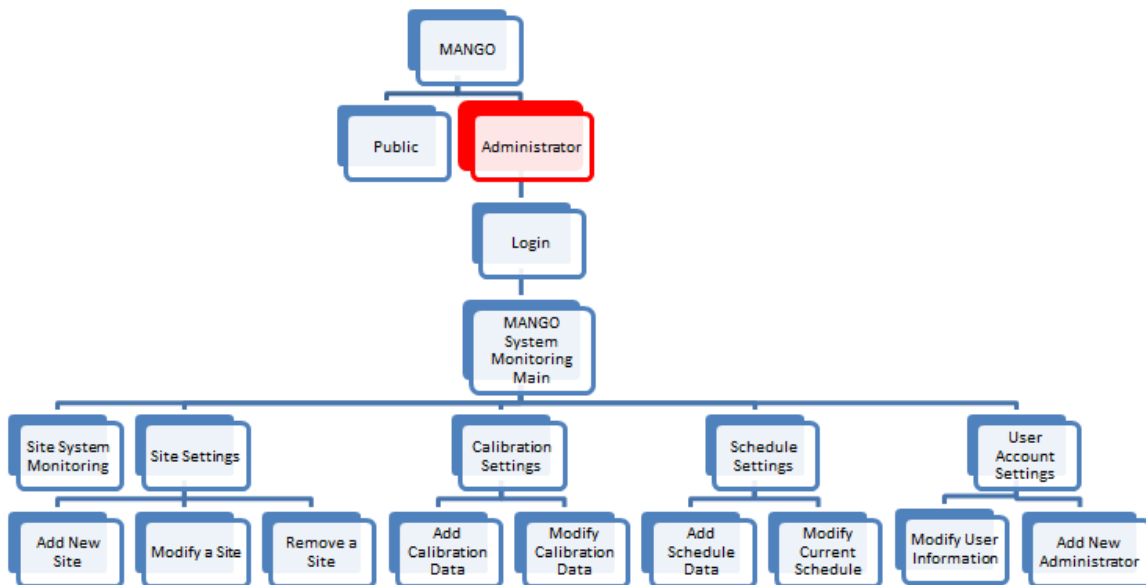


Figure 3.17: Administrator Section Website Structure

As shown in Figure 3.17, the administrator website is composed of a main, login, specific site system information, site settings, calibration settings, schedule settings, and account settings page.

#### 3.4.3.1 User Control

Administrators' user information is stored in the same database in which the public users' information is kept. To differentiate between public users and administrators, the arbitrary number assigned to the role changes. The following information from the administrator is stored: first name, last name, username, password, and affiliation, and role. The role is never asked to be input by the administrator but is assigned to them by the system. Another administrator through the

## METHODOLOGY

MANGO System Administrator site can only add an administrator to the system. Since the administrator is being registered through the administrator page, the arbitrary number assigned identifies the user as an administrator. The primary reason of the role is for the administrator to be able to access the MANGO System Administrator page.

### *3.4.3.2 System Monitoring*

The data acquisition program acquires site-specific information which includes the ambient temperature of the enclosure of the system, CCD camera temperature, disk space of where information is being stored on the remote site, and the current notebook's battery level. The administrators need to have the ability to view this information. During the data acquisition, the information acquired is sent to an archive using CSV files for each set of data. On the server, each individual file is opened and read through Flask using a Python script. The Python script opens the file and extracts the necessary data from the data set. The information has to be formatted accordingly so it can be outputted to the HTML and JavaScript.

From the HTML, Flask obtains the amount of data from the data set that needs to be extracted from the CSV files. The website gives the user the option to view certain amount of data acquired from the system. The user may view the data of the last 24 hours, last 5 days, last month, six months, and a year. Once the user selects the period of data to be viewed from the HTML, the HTML posts this request to the Python script. Then the Python script extracts the necessary data from the data set. The data is then sent to the HTML. Using JavaScript, Google graph is set up using the variables sent from Python. Once JavaScript obtains the information, it generates the charts. Data that will be visible on the page, on the charts, are ambient temperature versus time, CCD camera temperature versus time, free disk space versus time, and battery level versus time.

### *3.4.3.3 System Control*

The system control of the website constitutes of giving the administrator user the ability to control settings for each site on the network. This includes modifying site settings, calibration settings, and schedule settings.

#### Site Settings

The administrator has to have the ability to be able to add a new site to the network, modify current site information, and remove sites from the network. A CSV file (addsite.csv) was created which functions as a configuration file for the sites within the network. The CSV file includes the following columns: ID, site name, site latitude, site longitude, site elevation, and deleted. There is a site setting page which gives the user different options such as adding a new site, modifying

## METHODOLOGY

existing site, and removing the site from the network. When the user selects the option the user wishes to perform, they are redirected to the appropriate page to perform the task.

### Adding a New Site

Once the user is redirected to the “Add New Site” page the user will have the ability to add a new site. To add a new site, the user is prompted to form that asks the user to input information which includes site name, latitude, longitude, and elevation. Once the administrator has entered the necessary data in the fields of the form the user can submit the information. By clicking the “Submit” button, the Python code request the information from the form and writes the information to the addsite.csv file. The Python script reads the CSV file and looks for the last entry of the CSV file. The Python script assigns the new entry an ID. The DELETED column is originally set to ‘0’. The ‘0’ means that the site is active and available but it does not necessarily mean that the site is currently online. When a site is added it creates the necessary directories for site settings information to be stored.

### Modifying a Site

To modify existing information for a site, the user is redirected to the “Modify Site” page and is prompted to form that already includes the stored information such as site name, latitude, longitude, and elevation in the form fields. The information is displayed in the necessary fields through Python script from the Flask. Once the page is rendered, the code opens the CSV file where the data is stored and extracts the necessary data. Then it posts to the HTML which then is displayed in the form fields. This allows the user to view the information that is currently stored for the site. Once the administrator has modified the necessary data in the fields of the form the user can submit the information. By clicking the “Submit” button, the Python code requests the information from the form and modifies the information to the addsite.csv file. The Python script keeps the same “ID” variable and “DELETED” for the site.

### Removing a Site

The last site setting the user must be able to control, is the ability to remove a site from the network. To remove an existing site from the network, the user is redirected to the “Remove Site” page and is prompted the information of the site they wish to remove. This is done by when the user selects the site to be removed from the “Site Settings” page it stores the site the user has chosen to remove. Then through the Python script uses the site name to access the site information. Once the user confirms that the site information displayed on the screen is the site

## METHODOLOGY

the user wishes to remove, the user must click “Submit” button on the page. Once the button is clicked, the Python script on the Flask side opens the CSV file and modifies the “DELETED” column from the site and changes it to 1. This keeps the information stored but makes the page inactive. The site will be removed from the active sites keeping the stored information on the CSV file and the disk.

### Calibrating Sites

For every site, certain calibration data has to be input, which is necessary for the image processing to be performed. A page was created specifically for calibration. Within the “Calibration Settings” page it gives the administrator certain options such as “Add New Calibration Data” or “Modify the current Calibration Data”. To accomplish this, the user will select the site they wish to modify from a drop down menu. The information listed in the dropdown has been acquired from the addsite.csv file which contains the active remote sites. If the administrator decides to add new calibration data, the webpage will redirect the user to appropriate page. However, if the user wishes to modify current site calibration data, the webpage will redirect the user to appropriate page.

### Adding New Calibration Data

The “Add Calibration Data” page is composed of two forms in which the user must input the necessary fields. The administrator will fill out the form using the User Guide found in Appendix D. The first form is for storing the image information that user uses for acquiring the calibration data. When the user clicks “Submit” for the first form the information is requested using Python script which stores the information requested from the form to a CSV file called CalibrationInfo.csv.

The second form in the page is the star data information which is necessary to perform the calibration task in the image processing. The form prompts the user to input information regarding the raw image captured through the data acquisition while referencing the star catalog information and other tools. The form asks the user to input the image zenith information and 11 other stars identified in the image referencing. The data that is required for the form input for both the zenith and the stars is the following: star name, azimuth, and elevation,  $i$  coordinate, and  $j$  coordinate. Once the user has completely filled the form, the user can check the data input by clicking the “Check Data” button. If the “Check Data” button is clicked, the Python script from the Flask will request the information from the star data form and will run the lens function. As described in the Image Processing section, the lens function gives a plot of the distance from zenith vs. the angle from zenith. For the equations of the lens function, please refer to Lens Function section in the

## METHODOLOGY

Image Processing. This data is then displayed on the website using a figure which was created by a Python script. In the same webpage, a table under the plot lists the star names, the calculated distance from zenith, the calculated residual error when compared to a third order least-squares polynomial fit. The residual error is the observable estimate or the statistical error. By calculating the residual error for each star, the user can check the data that has been entered before finalizing the submission. Once the user is satisfied with the data entered, the user can click “Submit”. When the “Submit” button is clicked, the Python script generates the necessary files in the back end. The first file created is the Calibration.csv which extracts the zenith and star information and stores it in the CSV file. Then two other files are created which is the newI.csv file and the newJ.csv file which stores the new calculated i and j indices for the image. For the mathematical calculations of the indices, the equations can be found in the Hemispherical to Mercator projection in the Image Processing section.

The last file that that is created is called the imageCoordinates.csv file where the southwest and northeast coordinates for the image corners are calculated. It is important to calculate the southwest and northeast coordinate for each site so the image can be overlaid the site on the map (a Leaflet Maps requirement). For the mathematical calculations of the southwest and northeast coordinates equations can be found in the Hemispherical to Mercator projection in the Image Processing section.

### Modifying Calibration Data

To modify existing information for calibration data, the user is redirected to the “Modify Site” page from the “Site Settings” page by the website. The user is prompted to the same forms from the “Add Calibration Data” page. The difference between the pages is that the data stored in the CSV files is extracted using Python script from Flask.. Once the page is rendered, the script opens the necessary CSV files where the data set is stored and extracts the necessary data. After, it posts to the HTML which then is displayed in the form’s fields. This allows the user to view the information that is currently stored for the site. Just like the “Add Calibration Data” page the data entered can be checked by clicking on the “Check Data” button. If the user is satisfied with the data entered, the user can click “Submit” and the form requests the information from the form. The Python script rewrites the CSV files with the information modified by the user. Then, the Python script recalculates and recreates newI.csv, newJ.csv, and the imageCoord.csv files with the new data.

## METHODOLOGY

### Creating a Schedule

For every site the administrator also has to have the ability to add or modify the schedule in which the system is running on. The group has set up a schedule for each site by using the solar zenith angle. However, sometimes the administrator would like to modify the schedule and change some settings. The available settings are exposure time, rest period, image binning, and cooling period.

### Adding New Schedule

To add information for a new schedule, the user is redirected to the “Add New Schedule” page from the “Schedule Settings” page by the website. The user is prompted a form which is composed of two parts. The first part asks the user to input administrator information for record keeping. This will help the administrators to keep track of who has modified the schedule last. The second part of the form asks the user to input the exposure time, rest period between images, data binning, and the precooling duration for the data acquisition system. Once the user as input all the information, the user will click “Submit” and a Schedule.csv file is created through Python script. This schedule is then sent to the remote site using the proprietary Data Transport layer.

### Modifying Schedule

The “Modify Schedule” page contains the same functionalities of the “Add New Schedule” page. The difference between the pages is that the data stored in the CSV files is extracted using a Python script from Flask. Once the page is rendered, the script opens the necessary CSV file where the data set is stored and extracts the necessary data. After, it posts to the HTML which then is displayed in the form’s fields. This allows the user to view the information that is currently stored for the site. If the user is satisfied with the modified data the user has entered, the user can click “Submit” button. The Python script requests the data from the form and rewrites the Schedule.csv file with the new information. This schedule is then sent to the remote site.

## **3.5 Conclusion**

In summary, we first will be capturing the images of real time upper-atmospheric phenomena using specialized cameras that are currently being prototyped by SRI International. The images captured by the camera will be uploaded to a network. Once the images are downloaded onto a server, they can be processed by applying the specific image processing algorithms discussed earlier. After the images have undergone image processing, they ready to be visualized. In the last



## METHODOLOGY

step, we overlay these images onto a map of the United States of America. The design and results of such a system are discussed in the following chapter.

## 4 Results

Based on the background research conducted and the methods considered, an architecture was proposed (see Appendix A). The design and results of this architecture are discussed in this chapter.

### 4.1 Data Acquisition

The data acquisition process (Appendix C) designed by the group accomplishes the primary goal of the project which acquires the all-sky images and other system health information. As a result, an automated method to acquire real-time camera data and transfer to a computer for processing was developed. The following subsections discuss the results of the data acquisition program developed by the group.

#### 4.1.1 Image Acquisition

The image acquisition program (Appendix B) within the data acquisition software, acquires the image captured from the all-sky imager. As a result of executing this software, images are captured by the all-sky imager whenever the solar zenith angle is greater than  $100^\circ$  with the specified exposure period and binning. A sample output is shown in Figure 4.1.

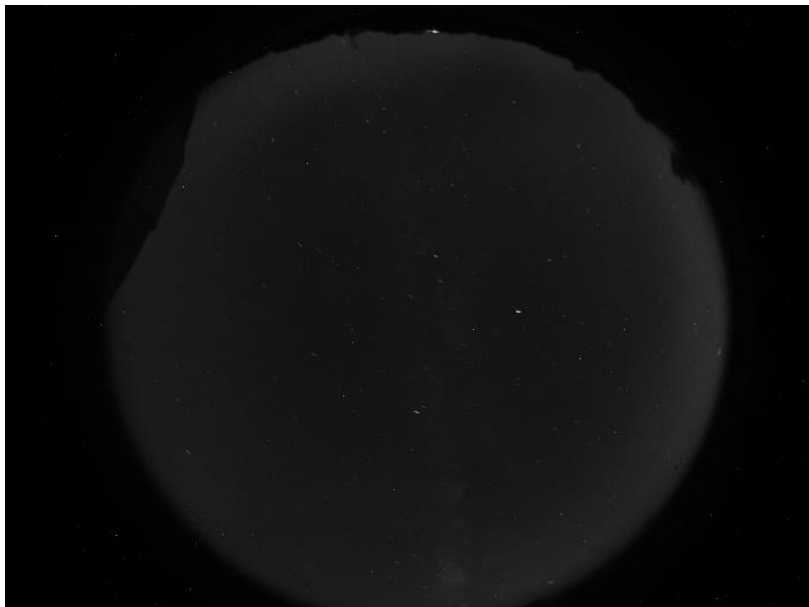


Figure 4.1: Image Acquisition Raw Image Result

## RESULTS

From the figure it can be observed, that the raw image acquired is orthographic and it cannot simply be overlaid on top of a map. As a result, the image must be transferred from the remote site to the server for processing.

### 4.1.2 System Information Acquisition

The system information acquisition protocol within the data acquisition software (Appendix C) acquires the system information for system maintenance and monitoring. This protocol acquires the system's ambient temperature, the camera CCD temperature, the disk space of the system where data is being stored on the remote site, and the current battery level of the imager-controlling computer. As a result, the information acquired through the system acquisition is stored in separate CSV files to be used during the visualization objective of the project. Sample outputs of the CSVs are shown in Table 1.

Timestamp (MM/DD/YYYY HH:MM:SS)	CCD Temperature (°C)	Ambient Temperature (°C)	Battery Level (%)	Free Disk Space (GB)
02/28/2013 20:29:45	2.29	26.38	100	369.6631
02/28/2013 20:30:15	-0.93	27	100	369.6631
02/28/2013 20:30:45	-2.14	26.5	100	369.663
02/28/2013 20:31:15	0.36	26.5	100	369.6625

Table 1: System Health Acquired Format

## 4.2 Image Processing

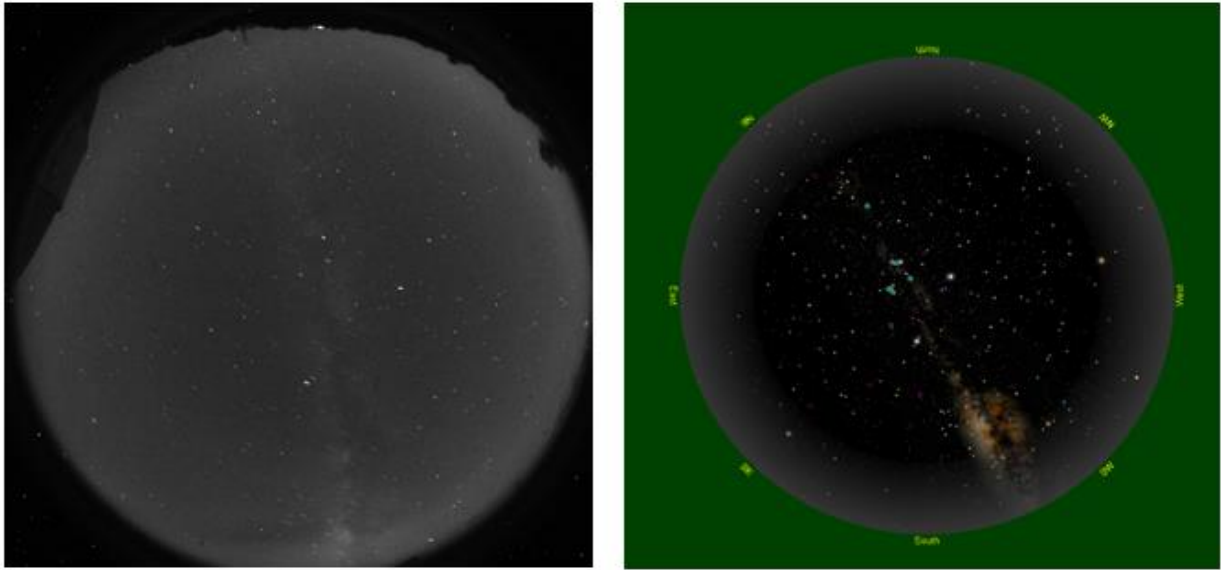
The raw images acquired through the data acquisition were obtained using a fish-eye lens. The second objective of the project was to map the pixels of the raw image to geographical coordinates for web display. The image processing program (Appendix D) transforms the spherical image into an understandable image with Mercator geographic coordinates. The astronomical image processing program created by the group was one of the first ones to be designed in Python. The following subsections discuss the results of the image processing program developed by the group.

### 4.2.1 All-Sky Spatial Calibration

The first task within image processing was all-sky spatial calibration. The purpose of performing calibration was to align true north with the top of the image, to project the image top-down, and

## RESULTS

to account for elliptical lens functions. Figure 4.2 displays the raw image acquired through the data acquisition from the all-sky image (left) and The Sky6 star catalog (right). The image displayed from the star catalog is matching the time and location of from where the image was captured from. The star catalog was used as reference to calibrate. As it can be observed, the original raw image displayed (left) is not aligned correctly with the true north and it is not a top-down projection of the sky on Earth. Instead the image is bottom-up as observing the space from Earth.



**Figure 4.2: Image Acquisition Raw Image Compared to TheSky6 Software**

As a result, the calibration task of the image processing routine produced a calibrated image. In Figure 4.3, the raw image (left) is compared to the calibrated image (right). The new calibrated image was rotated and scaled accordingly to take into account elliptical lens properties. The new image was created by transforming each pixel from the initial arbitrary coordinates to the standard coordinates as described in the previous chapter. From the calibrated image it can be noted that the image has been projected top-down to be able to be overlaid on top of a map for visualization.

## RESULTS

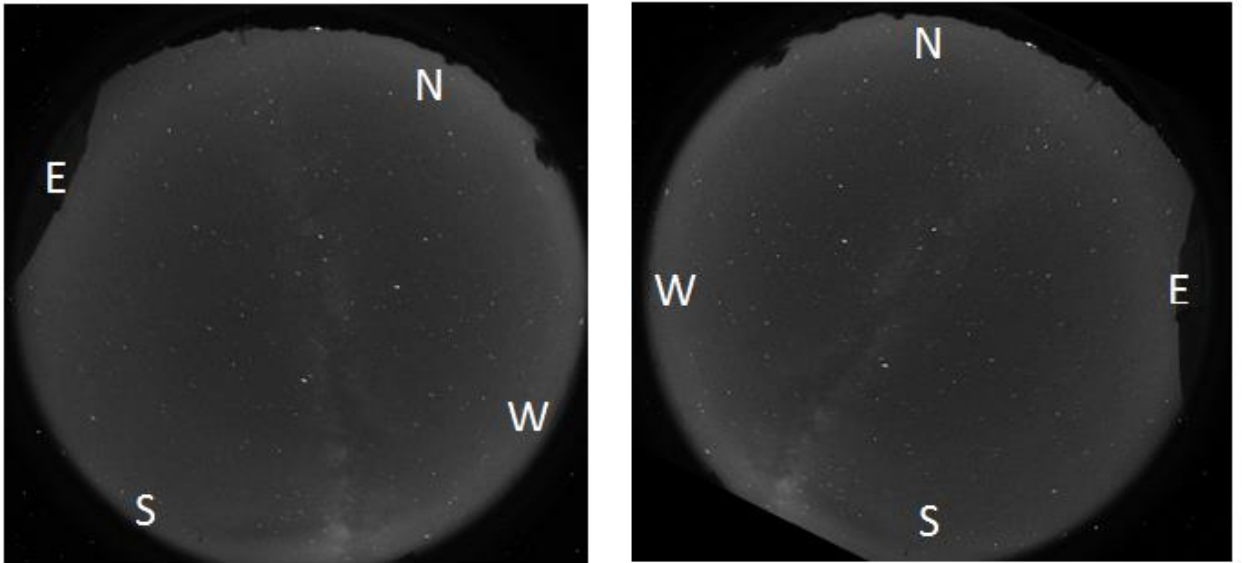


Figure 4.3: Raw Image Result versus Calibrated Image Result

### 4.2.2 Star Removal

The next step of the image processing is star removal. Star removal as mentioned in the previous chapter removes stars from calibrated images. The stars are removed by using statistical properties of pixels values to identify local peaks. Once the pixels are identified interpolation is performed over them. As a result, a new image is created without the stars. In Figure 4.4, the calibrated image (left) is displayed next to the image that has been processed through the star removal algorithm (right).

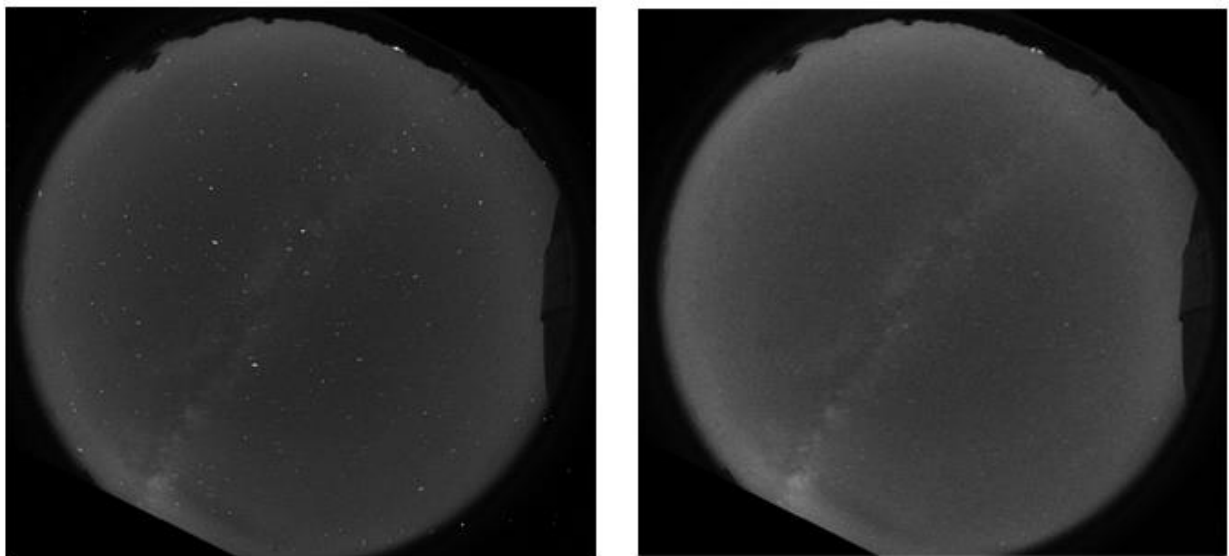


Figure 4.4: Star Removal Results

## RESULTS

The purpose of removing stars is that stars are not space phenomena but noise. If the stars are left in the image when the hemispherical to Mercator projection is performed, stars will cause streaking in the final image.

### ***4.2.3 Flat Field Correction: Countering Van Rhijn Effect and Atmospheric Extinction***

The subsequent step was to perform flat field correction. Flat field correction includes correcting variations such as vignetting, the Van Rhijn effect and atmospheric extinction. To correct these variations a correction filter was created by using the Van Rhijn effect equation and the atmospheric extinction equation. Figure 4.5 displays the star removal image (left) and the correction filter (right).

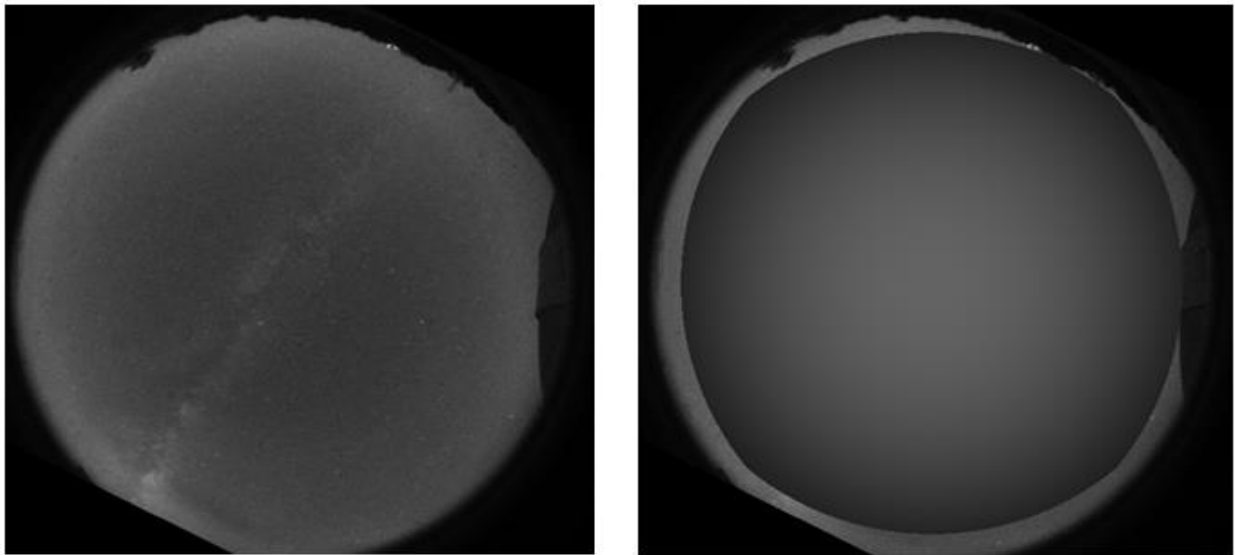


Figure 4.5: Overlaying the background correction filter on the all-sky image

Figure 4.6, displays the image created from star removal (left) next to the image produced after applying the correction filter (right).

## RESULTS

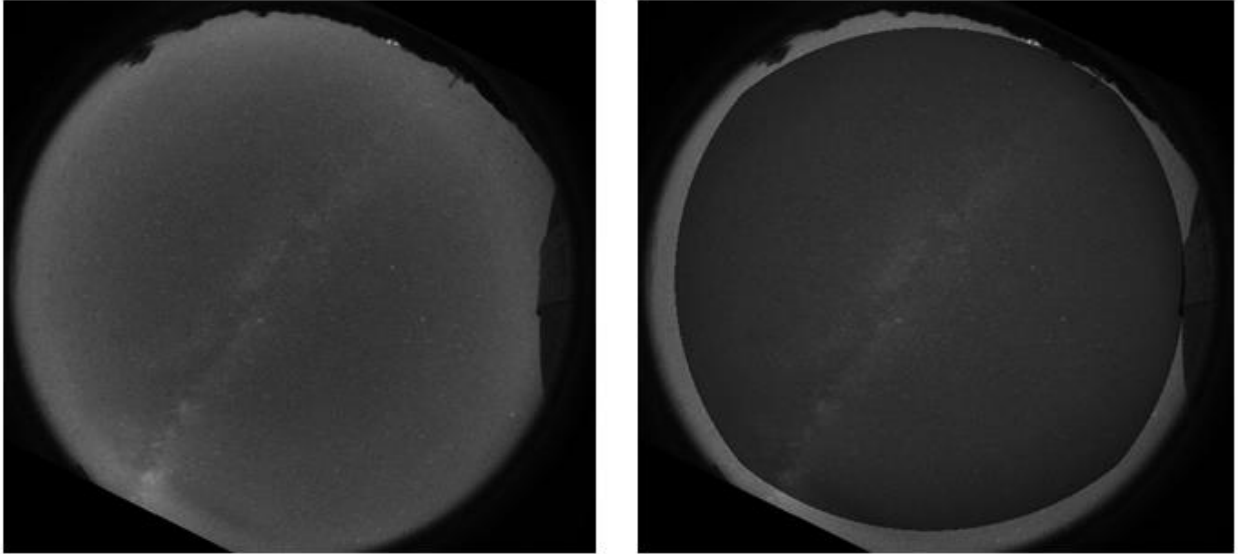


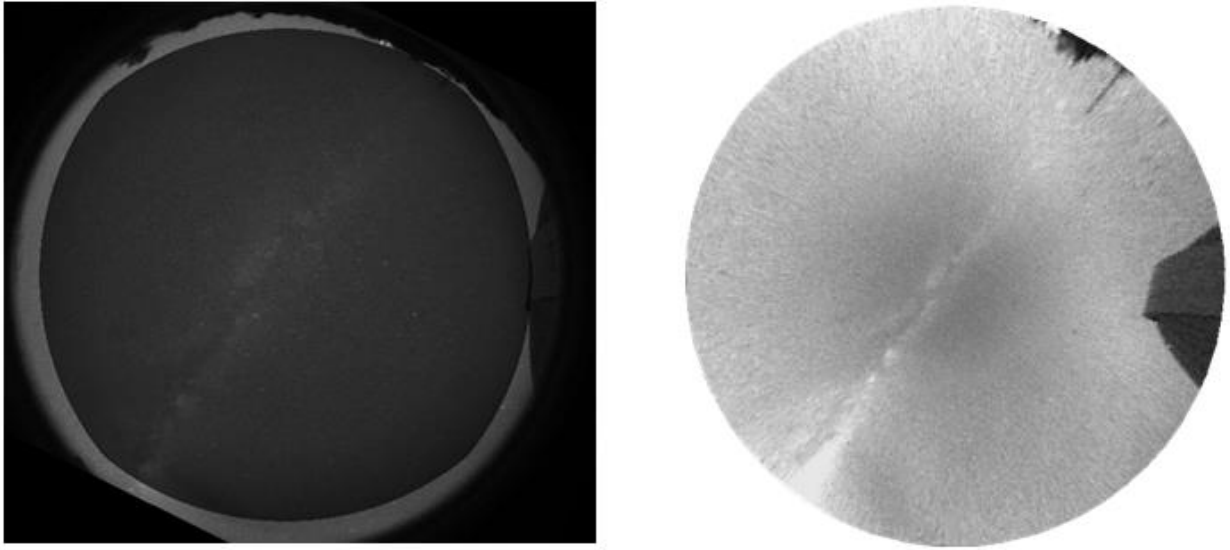
Figure 4.6: All-sky image after the background correction filter has been applied

As a result, the new image created by the flat fielding correction has balanced intensity across pixels through the whole image.

### ***4.2.4 Hemispherical to Mercator Projection***

The last step of the image processing routine involves shifting from the orthographic projection (hemispherical) to a mapping projection so that each pixel from the original image can be positioned on a map of the Earth accurately. Since images of airglow in our design are also supposed to be projected on web-based Mercator maps, the final image created is based on a Mercator projection. As a result, to obtain the Mercator projection image the latitude and longitude of every pixel was determined as explained the previous chapter. The final Mercator plane image has constant longitudes parallel to the  $y$ -axis and constant latitudes parallel to the  $x$ -axis. In Figure 4.7 the image obtained from the flat fielding correction (left) compared to the new image (right) created by the Mercator projection.

## RESULTS



**Figure 4.7: Hemispherical to Mercator Projection Result**

The new Mercator projection image has a constant viewing angle from zenith. The final image created in this task is the final image from the image processing. Now this image is ready to be overlaid on top of a map.

### **4.3 Visualization**

The third objective of our project was to display the processed images and the acquired system information from the remote sites. As a result, a website was designed to achieve the project goal. The website allows for the public, students, and scientists to be able to observe space phenomena in real time. The website also gives the administrators, in this case SRI International, the ability to control the system through the web. The website created is composed of a public and administrator web interface. The following subsections discuss the results of the website developed by the group.



## RESULTS

### ***4.3.1 Public Page***

The goal of the public page was to give the students and SRI International researchers a method to observe real time space phenomena. Through this website, the students and SRI International scientists will be able to interact with each other and learn from the data gathered by the all-sky imagers. As a final result, the public website is composed of information, main, login, logout, and a user register page which accomplishes the education goal of the project.

#### Base Template Components

The public website has a base template which includes all the components that are common in all the pages. The base template includes the menu bar that is located on the top of the page as shown in Figure 4.8.



**Figure 4.8: Public Page Top Menu Bar**

The top menu bar has different options such as Home, Info, Register, and Login. When a user chooses an option on the bar, he or she will be redirected to their page of choice.

#### ***4.3.1.1 Map Page***

The main page of the MANGO website is the “Map” page as shown in Figure 4.9. The purpose of the map page was to provide the public, students, and scientists the ability to observe the space phenomena. On a Mercator map of the United States, images processed through the image processing routine are overlaid.

## RESULTS

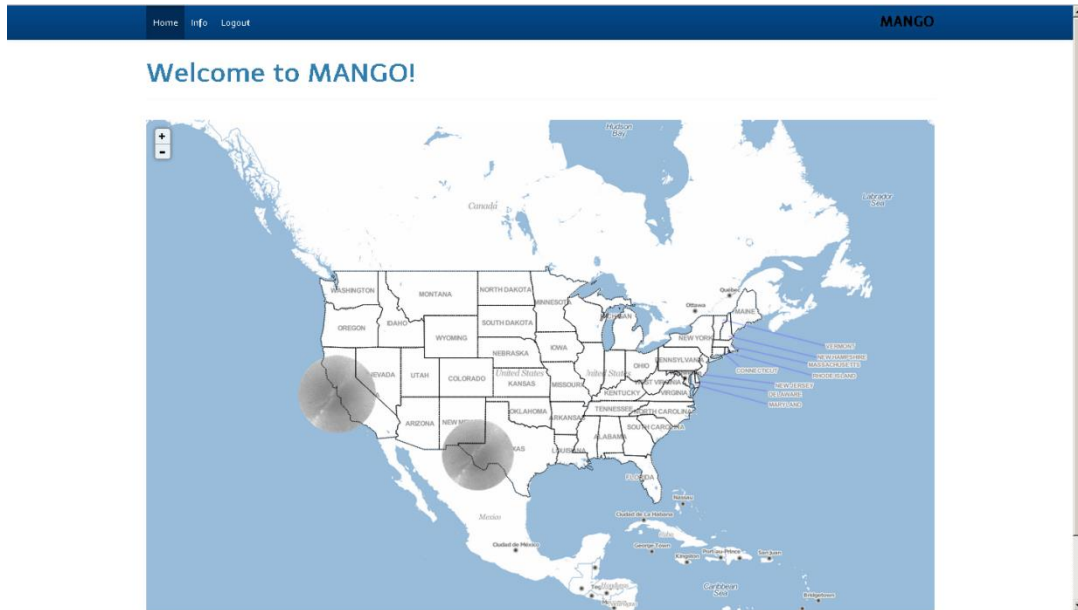


Figure 4.9: MANGO Public Main Page

As a result, depending on the activity of sites added to the network, images will be overlaid on the map.

### 4.3.1.2 Information Pages

The information pages which are located within the “Info” section of the website include information to acquaint the user with educational material about the site. The “Info” section of the website is composed of the following three pages:

- What’s MANGO?
- The Ionosphere
- Space Phenomena

The main information “What’s MANGO?” page explains the purpose of the MANGO project and the importance of the project as shown in Figure 4.10.

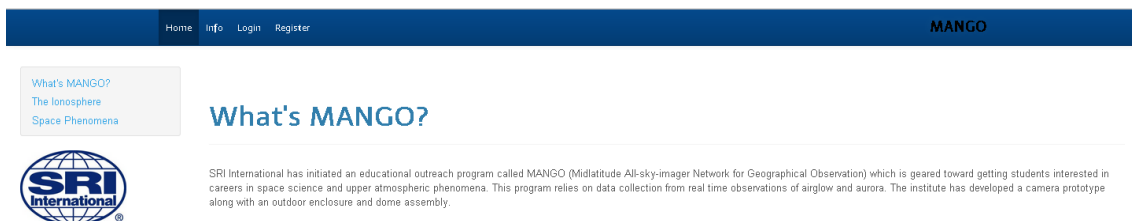


Figure 4.10: What's MANGO? Information page

## RESULTS

The second component of the information pages is the “Ionosphere” page. In this page, the importance and the components of the ionosphere are explained. The display of the page is shown in Figure 4.11.

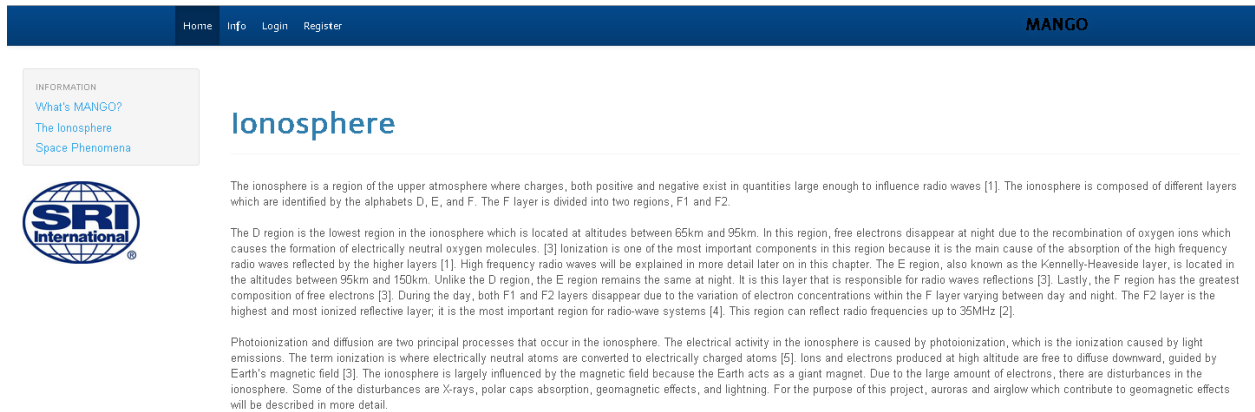


Figure 4.11: Ionosphere Information page

The “Space Phenomena” page explains the different types of space phenomena such as auroras and airglow. The layout of the page is shown in Figure 4.12. As a result, this page gives the user vital information of what is being observed on the map.

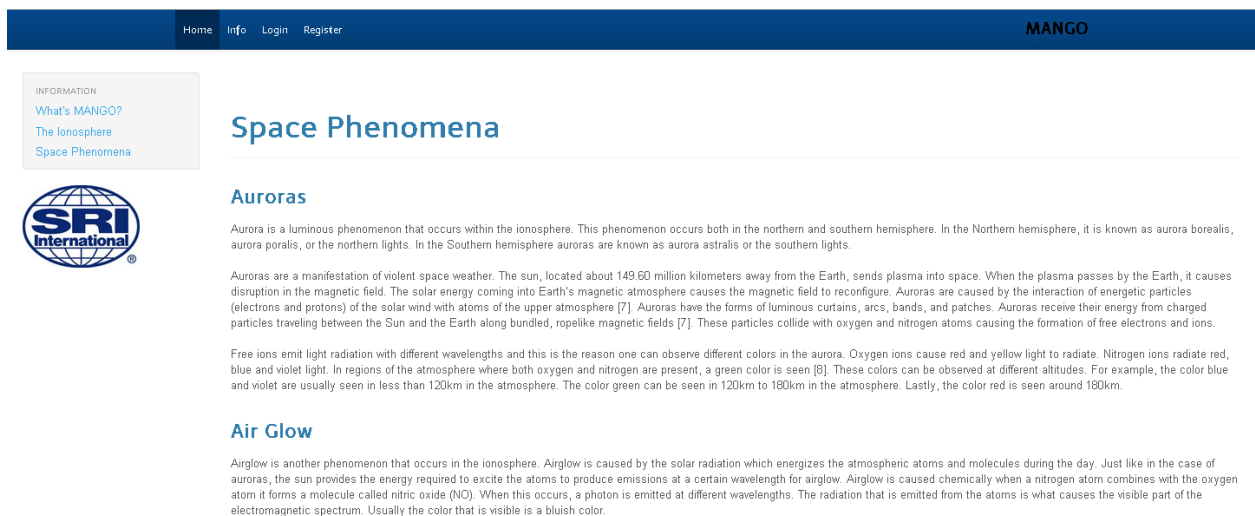


Figure 4.12: Space Phenomena Page

### 4.3.1.3 Register Page

SRI International had a requirement of the public having the ability to register. In the future, SRI International will be implementing a blog to the website where the public, students, and SRI International researchers are going to be able to leave comments and discuss the space phenomena observed on the map. The purpose of the register page is for the users to be able to register to the

## RESULTS

MANGO website and access the blog to interact with the scientists. Figure 4.13 displays the page that was created for user registration.

The screenshot shows the MANGO website's Register page. The header is dark blue with navigation links: Home, Info, Login, Register, and the MANGO logo. The main content area is white and titled "Register". It contains several input fields: First Name, Last Name, Username, Email address, Password, Repeat Password, and Affiliation. Below the fields is a checkbox for "I accept the TOS" and a blue "Register" button. A "Login" link is located at the bottom left of the form area.

**Figure 4.13: Register Page**

The information that is acquired by the user is their first name, last name, username, email, and password. All this information is then stored in an encrypted database.

### *4.3.1.4 Login Page*

The purpose of “Login” page is to login the user and allow the user to access the blog and some other applications that SRI International implements in the future. The layout of the login page is shown in Figure 4.14. For the user to login to MANGO, the login in page is composed of an email and password input. Once the information has been validated it keeps the user logged in for the session.

The screenshot shows the MANGO website's Login page. The header is dark blue with navigation links: Home, Info, Login, Register, and the MANGO logo. The main content area is white and titled "Login". It contains two input fields: Email address and Password. Below the fields is a blue "Login" button and a "Register" link.

**Figure 4.14: Login Page**

### *4.3.2 Administrator Page*

The goal of the administrator page was to provide SRI researchers the means to monitor the system. Through the administrator website, SRI researchers are able to set the settings of the

## RESULTS

system and monitor system specific information. As a result, the main administrator page links to the following pages: login, MANGO system, specific site system information, site settings, calibration settings, schedule settings, and account settings.

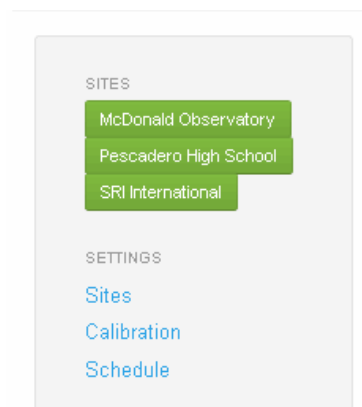
### Base Template Components

The administrator website has a base template which includes all the components that will be displayed in all the pages. The base template includes the menu bar, navigation side bar, and logos. The menu bar shown in Figure 4.15 is composed of home, map, account, and logout buttons. As a result, if one of the options is selected, it will redirect the user to the appropriate page.



**Figure 4.15: Administrator Top Navigation Bar**

The navigation bar located on the left side of the pages is composed of two main parts, “Sites” and “Settings”, as shown in Figure 4.16. The “Sites” includes all the sites that are active. The “Settings” are all the settings that can be applied to the network remote sites.



**Figure 4.16: Administrator Side Navigation Bar**

Under the navigation bar SRI International’s and Worcester Polytechnic’s logos are displayed as shown in Figure 4.17.

## RESULTS



Figure 4.17: Logos

### 4.3.2.1 MANGO Main page

The MANGO main page is a page that just has a title and a “Login” button as shown in Figure 4.18.

#### Welcome to MANGO

Login

© SRI International 2013

Figure 4.18: MANGO Main page

The simplicity of the page keeps the purpose of the page discreet. Once the administrator clicks “Login” the page will redirect to the “Login” page.

### 4.3.2.2 Login

The administrator login page is specific to the administrators. Publicly registered users will not have the required credentials to login here. The login page asks the administrator to input their email and password. Once the form input is validated, the administrator is able to access the page. The login page is shown in Figure 4.19.

#### Admin Login

Email address

Password

Login

Figure 4.19: Admin Login Page

## RESULTS

### 4.3.2.3 MANGO System Main page

The “MANGO System Management” page mainly functions as a welcome screen (accessible once the user has successfully logged in to the system) as shown in Figure 4.20.

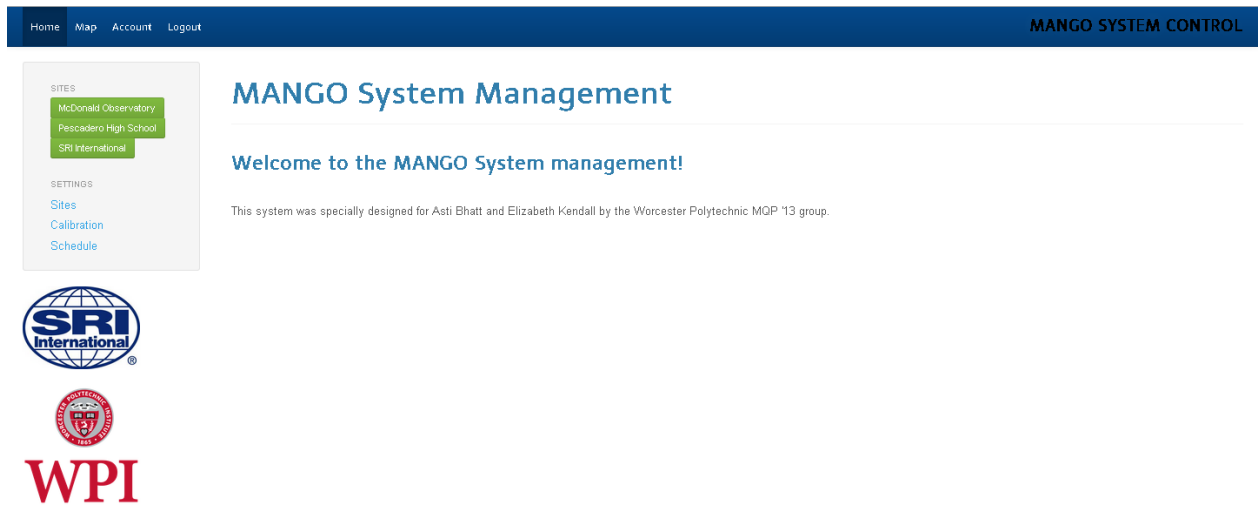


Figure 4.20: MANGO System Management Page

### 4.3.2.4 Specific Site System Information

The data acquisition program acquires site specific information which includes the following information:

- ambient temperature of the system
- camera CCD temperature
- free disk space of where information is being stored
- current battery level of the notebook

All this information can be accessed by the administrator. To view site specific information, a single “System Information” page was created. The user can access the page using the navigation bar, as shown in Figure 4.16, which displays the active on the network. For prototyping, only test sites are shown. Once the user selects the site which he or she wishes to monitor from the navigation bar, the user is routed to the “Site System Information” page as shown in Figure 4.21.

## RESULTS

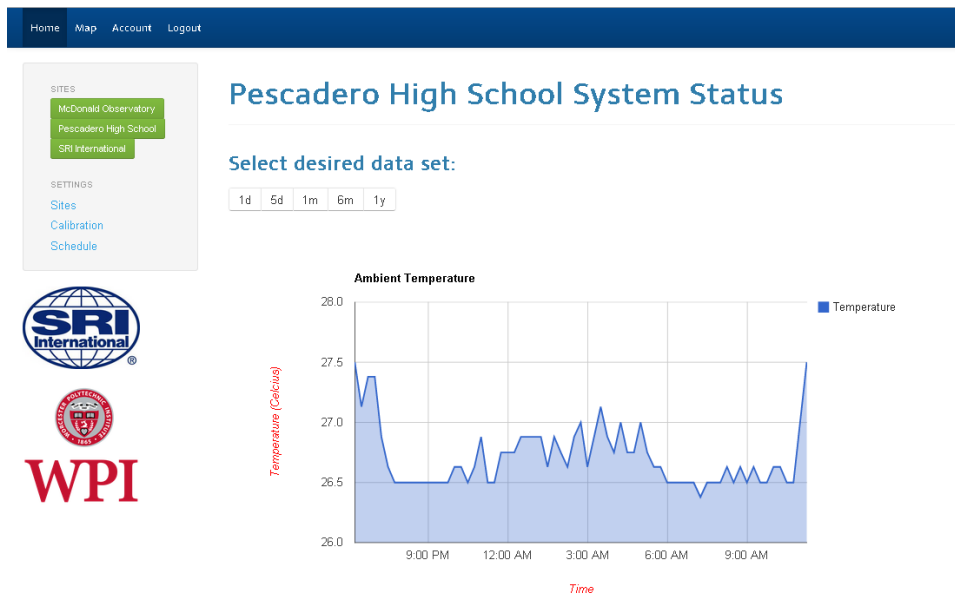


Figure 4.21: Site System Information Status Page

On this page, there are a total of four visible graphs. Under the heading of the page there is data set viewing button with different viewing options as shown in Figure 4.22.



Figure 4.22: Data Set Viewing Options Button

The user may view a data set of the last 24 hours, last 5 days, last month, six months, or a year. Once the user chooses the data set which he or she wishes to monitor, the graph will display the data set accordingly. The four graphs which display the system data information are the following:

### Ambient Temperature Graph

The ambient temperature graph displays the temperature versus time which has been acquired and stored through the data acquisition software. As shown in Figure 4.23, time is displayed in  $x$ -axis and the temperature in Celsius is displayed in the  $y$ -axis.



## RESULTS

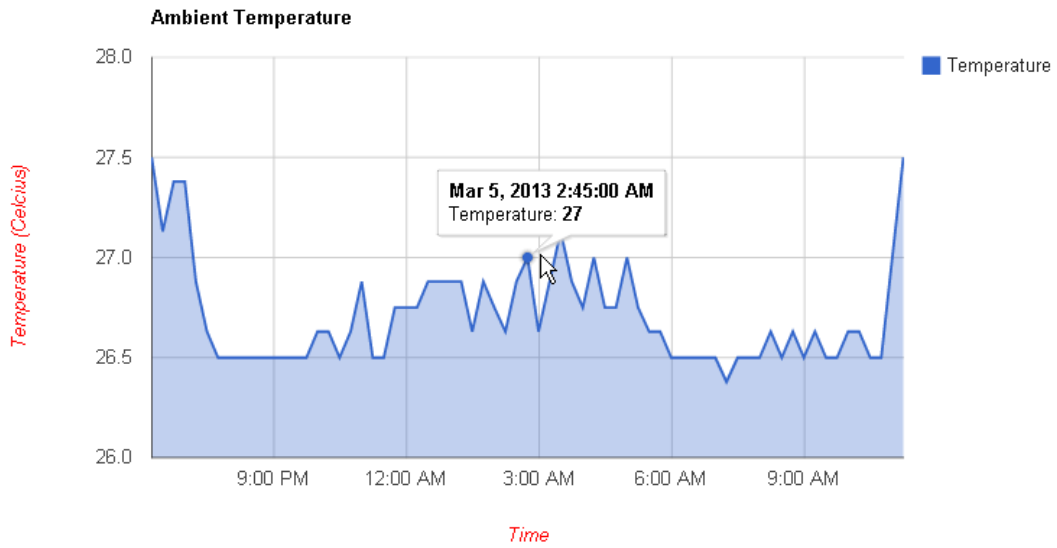


Figure 4.23: Ambient Temperature System Graph

### CCD Temperature Graph

To display the CCD temperature data gathered by the data acquisition software a temperature versus time graph has been generated on the “Site System Information” page as shown in Figure 4.24. Just like the previous graph, the  $x$ -axis displays the time and in the  $y$ -axis the temperature in Celsius.

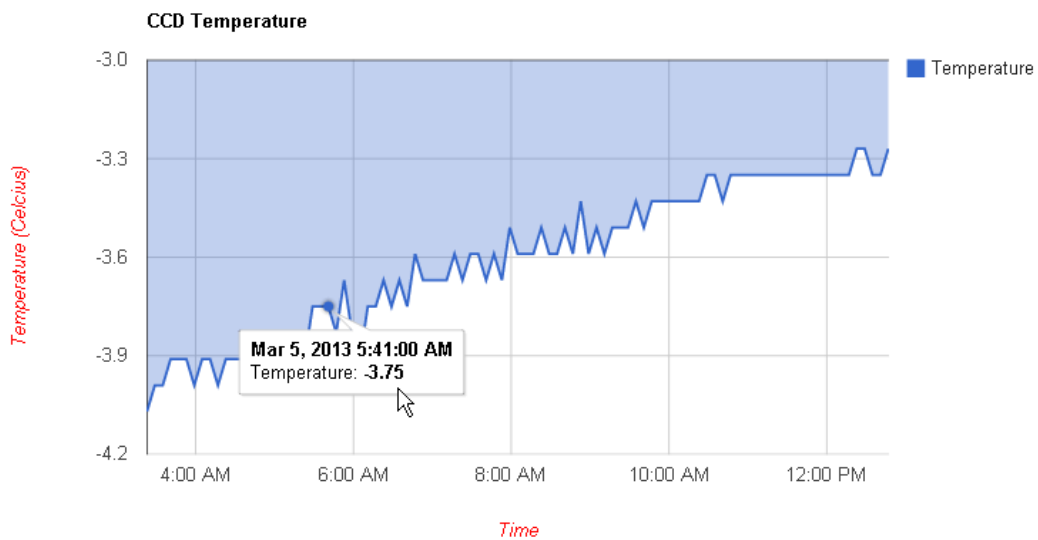


Figure 4.24: CCD Temperature System Graph

## RESULTS

### Disk Space Graph

To monitor the disk space available in the remote site computer a graph was created to monitor the amount of free disk space. The disk space graph shown in Figure 4.25 displays the disk space of the system versus time which has been acquired and stored through the data acquisition program. The  $y$ -axis displays the disk space in gigabytes.

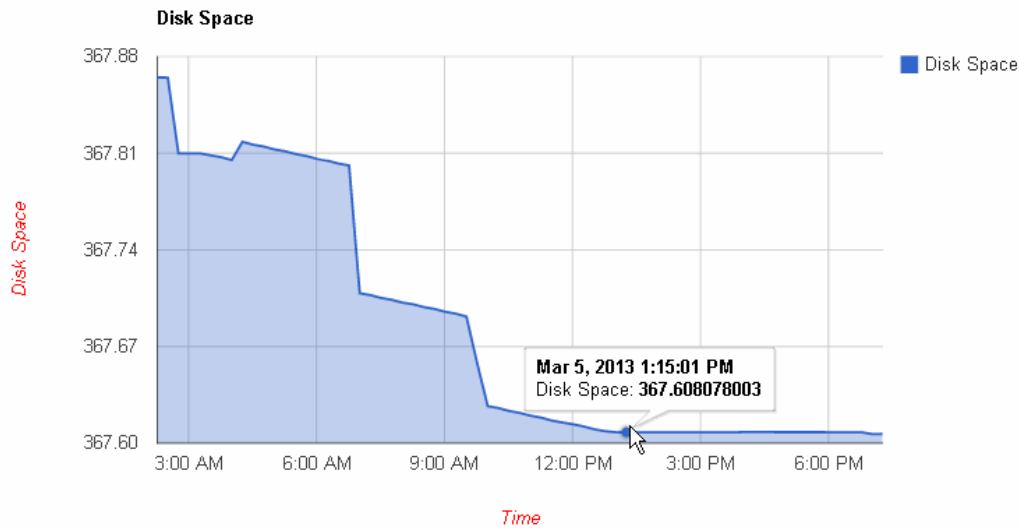


Figure 4.25: Disk Space System Graph

### Battery Level Graph

The battery level graph displays the battery percentage when the laptop is not running on AC power. As shown in Figure 4.26, time versus the battery level in percentage is displayed.

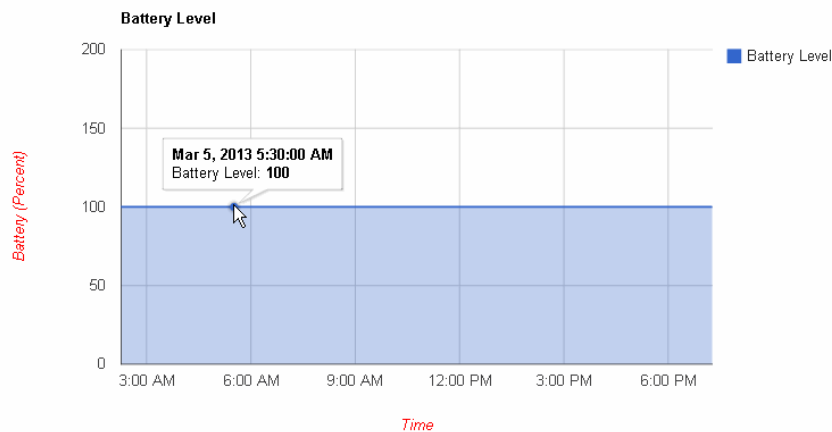


Figure 4.26: Battery Level System Graph

## RESULTS

### 4.3.2.5 Site Settings Pages

The administrator has to have the ability to modify remote site settings. The site settings page is composed of three parts:

- adding a new site to the network
- modifying existing site information
- removing an existing site from the network

As a result, once the user selects the task to be performed, the user is routed to the appropriate page to perform the task accordingly. Figure 4.27, displays the format of the page.

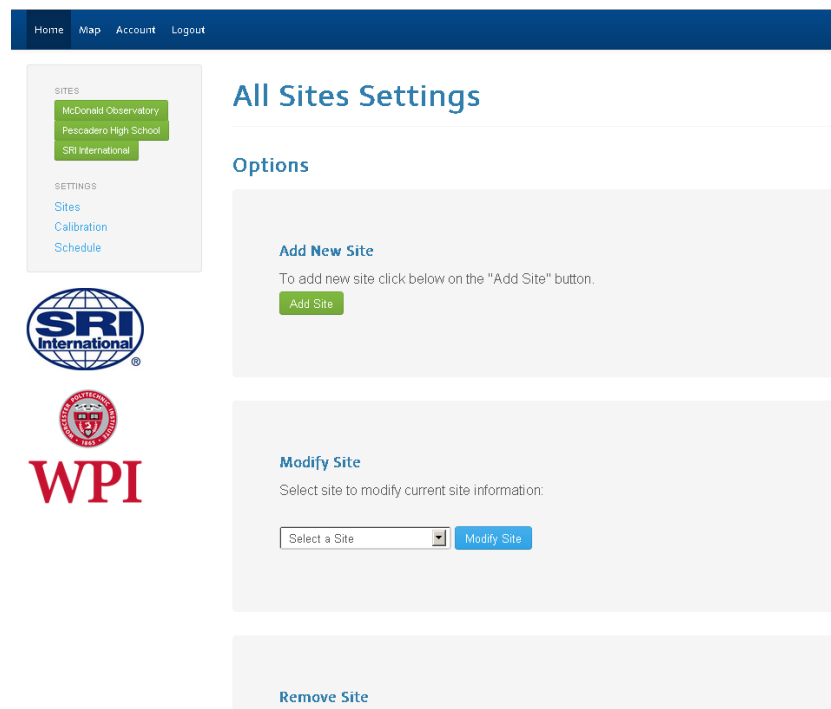
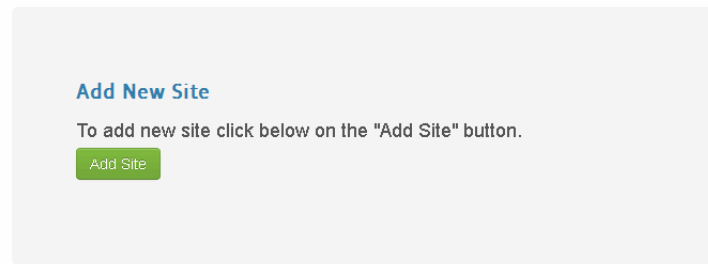


Figure 4.27: All Site Settings Page

To add a new site, the user must go to the “Add Site” section of the page as shown in Figure 4.28. The user must click the “Add Site” button from the “Site Settings Page”. This routes the user to the “Add Site” page.

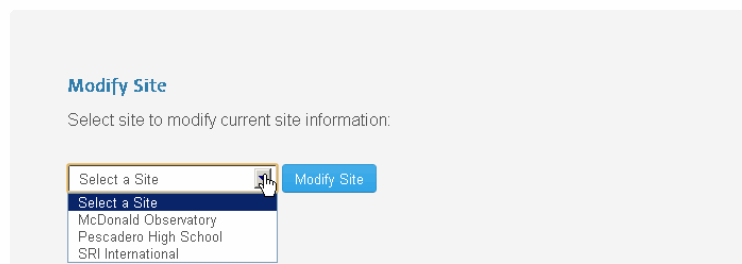
## RESULTS

### Options



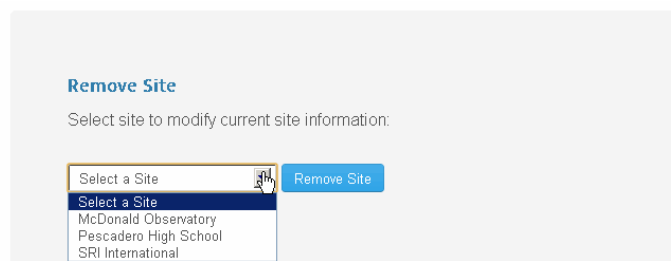
**Figure 4.28: Add a New Site Option**

If the user wishes to modify a site, the user must go to the modify section of the page as shown in Figure 4.29. Within the modify section, there is a dropdown menu which allows the user to select one of the existing sites. Once the user has selected the site to be modified the user may click "Submit". This routes the user to the "Modify Site Page".



**Figure 4.29: Modify a Site Option**

The last component of the page allows the user to remove a site as shown in Figure 4.30. Just as seen in the modify section, the user has a dropdown menu which allows the user to select the existing site which should be from the network. Once the user has selected the site to be removed, the user may click "Submit". This routes the user to the "Remove Site Page".



**Figure 4.30: Remove a Site Option**

## RESULTS

### Adding a Site

The add site page displays a form which the user must fill out to be able to add a new site as displayed in Figure 4.31.

Home Map Account Logout

SITES

- McDonald Observatory
- Pescadero High School
- SRI International

SETTINGS

- Sites
- Calibration
- Schedule

SRI International

WPI

### Add New Site

Input the following information:

Site Name

Latitude

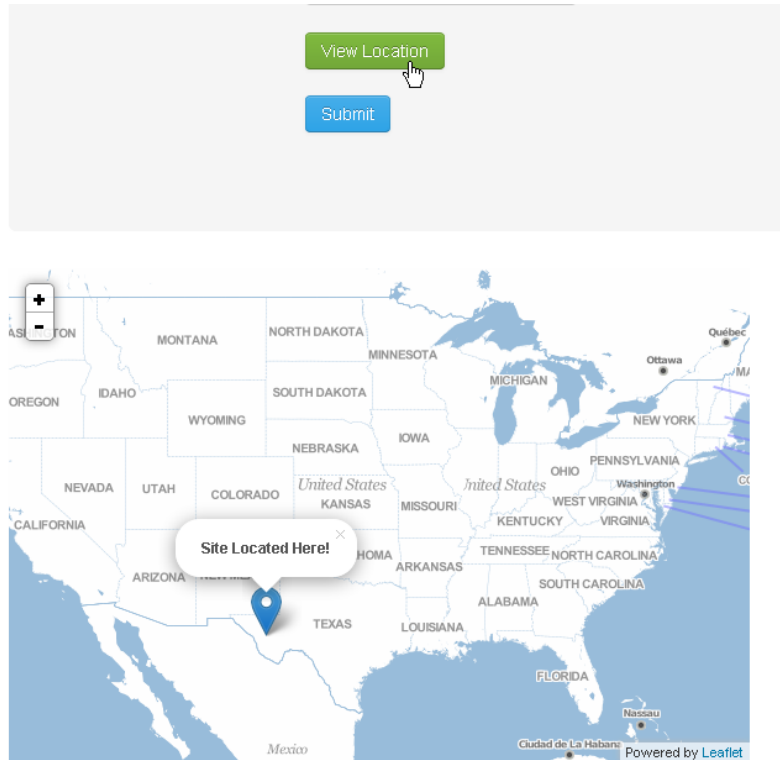
Longitude

Elevation

Figure 4.31: Add New Site Page

The form asks the user to input the site name, site latitude, site longitude, and site elevation. When all the information has been entered, the user may verify that the latitude and longitude of the site are correct by clicking the “View Location” green button. Once the button is clicked, this will cause a map to appear on the page for coordinate verification as shown in Figure 4.32.

## RESULTS



**Figure 4.32: Coordinate Verification**

After checking the location of the site, the user may submit the data. As a result, the information input by the user is then stored in a CSV file. Also, once the site has been created, server directories for the site are created accordingly to store all of the gathered data.

### Modifying a Site

The “Modify Site” page allows the user to modify current and existing information that has been previously stored for the site on the network that was previously chosen on the “Site Settings” page. The setup of the “Modify Site” page and its elements are shown in Figure 4.33.

## RESULTS

Home Map Account Logout

SITES  
McDonald Observatory  
Pescadero High School  
SRI International

SETTINGS  
Sites  
Calibration  
Schedule

SRI International

WPI

### Modify Site

Input the following information:

Site Name

Latitude

Longitude

Elevation

Figure 4.33: Modify Site Page

As a result, this page is just like the “Add Site” page with the same elements such as a form, check location button, and a submit button. The form fields contain the current information stored. Once the user makes the appropriate changes, the user may verify the location of the site just like they would in the “Add Site” page. When the user is satisfied with the information in the fields, the user can click the “Submit” button. The new information entered in the fields replaces the pre-existing information.

### Removing a Site

The “Remove Site” page allows the user to remove a site from the network. The setup of the page is shown in Figure 4.34.

Home Map Account Logout

SITES  
McDonald Observatory  
Pescadero High School  
SRI International

SETTINGS  
Sites  
Calibration  
Schedule

SRI International

WPI

### Remove Site

User will be removing the following site:

Site: SRI International  
Latitude: 37.457793  
Longitude: -122.176234  
Elevation: 6

To finalize remove click the "Remove" button:

Figure 4.34: Remove Site Page

## RESULTS

The “Remove Site” displays the information of the site the user wants to remove. Once the user views the information displayed on the page and is certain that the site information on the box is the correct site, he or she may click the “Remove” button to finalize removal. This makes the appropriate changes to the configuration file.

### 4.3.2.6 Calibration Setting Pages

The “Calibration Settings” page has two components which are:

- adding a new calibration data for a specific site
- modifying the existing calibration data for specific site

As a result, once the user selects the task to be performed, the user is routed to appropriate page to perform the task accordingly. The setup of the page and elements can be seen in Figure 4.35.

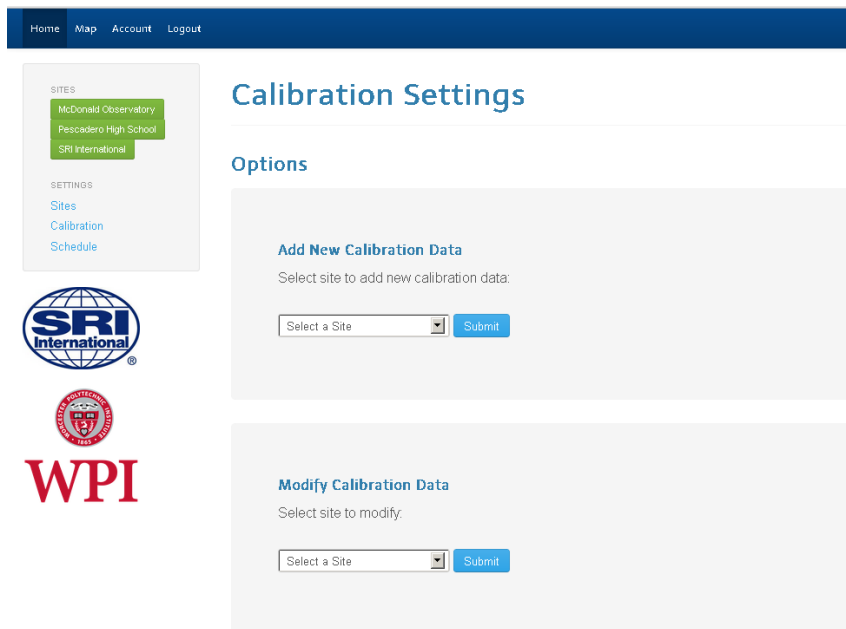


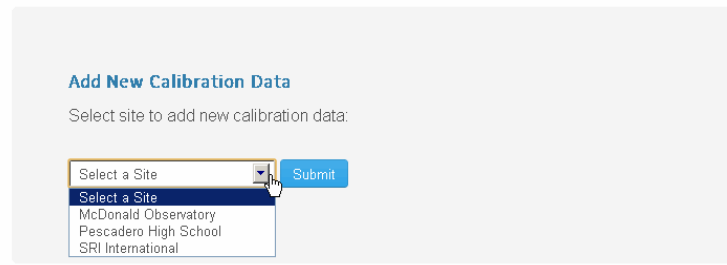
Figure 4.35: Calibration Settings Page

To add a new calibration data, the user must go to the “Add Calibration Data” section of the page as shown in Figure 4.36. Within the section there is a dropdown menu which allows the user to select the existing site to which the user wishes to add calibration data. The user must click the “Submit” button which routes the user to the “Add Calibration Data Page”.



## RESULTS

### Options



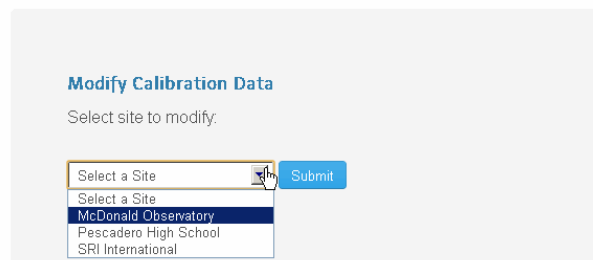
**Add New Calibration Data**  
Select site to add new calibration data:

Select a Site

- Select a Site
- McDonald Observatory
- Pescadero High School
- SRI International

**Figure 4.36: Add New Calibration Data**

If the user wishes to modify existing calibration data for a site, the user must go to the “Modify Calibration Data” section of the page as shown in Figure 4.37. Within the modify section there is a dropdown menu which allows the user to select the existing site the user wishes to modify. Once the user has selected the site to be modified the user must click “Submit”. As a result, this routes the user to the “Modify Calibration Page”.



**Modify Calibration Data**  
Select site to modify:

Select a Site

- Select a Site
- McDonald Observatory
- Pescadero High School
- SRI International

**Figure 4.37: Modify Calibration Data**

### Adding Calibration Data

The “Add Calibration Data” page displays a form which the user must fill out to be able to add new calibration data as displayed in Figure 4.38. Under the heading of the page, the site specific information is displayed so the user can note the site they previously selected in the main “Calibration Settings” page.

# RESULTS

Home Map Account Logout

SITES  
McDonald Observatory  
Pescadero High School  
SRI International

SETTINGS  
Sites  
Calibration  
Schedule

## Add New Calibration

Currently adding data for:  
Site: McDonald Observatory  
Latitude: 30.6714  
Longitude: -104.0225  
Elevation: 2070

User may modify the necessary fields below:

Image information:

Image Name

Date

UTC Time

Administrator's Name

Zenith data:

Name  Azimuth  Elevation  i Coordinate  j Coordinate

Star 1 data:

Name  Azimuth  Elevation  i Coordinate  j Coordinate

Star 2 data:

Name  Azimuth  Elevation  i Coordinate  j Coordinate

Star 3 data:

Name  Azimuth  Elevation  i Coordinate  j Coordinate

Figure 4.38: Add New Calibration Data

As shown in the above figure, the page is divided into two forms. The first form is the image information form which records the data used for calibration by the user. When the user has entered all the necessary information for the form, the user can click “Submit”. As a result the information is stored. A closer look of the form is shown in Figure 4.39.

Image information:

Image Name

Date

UTC Time

Administrator's Name

Figure 4.39: Image Information Form

The second form, which is shown in Figure 4.40, is the star information. This form is divided into two sections: the zenith and stars information.

## RESULTS

**Star Information:**

**Zenith data:**

Zenith  Azimuth  Elevation  i Coordinate  j Coordinate

**Star 1 data:**

Name  Azimuth  Elevation  i Coordinate  j Coordinate

**Star 2 data:**

Name  Azimuth  Elevation  i Coordinate  j Coordinate

**Star 3 data:**

Name  Azimuth  Elevation  i Coordinate  j Coordinate

Figure 4.40: Star Information Form

The information entered can be checked by clicking on the “Check Data” button. As a result, this renders the page and displays the lens function plot shown in Figure 4.41.

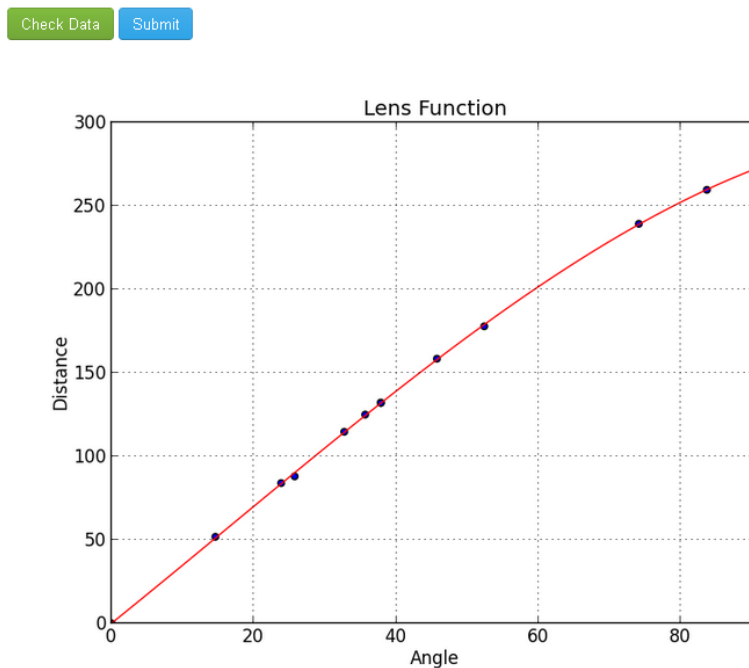


Figure 4.41: Lens Function Graph for Star Input Data Check

The plot displayed shows the angle from zenith versus distance from zenith. The user should make sure that the stars line up on the red line. To quantify the data in the plot, a table is also displayed when the “Check Data” button is clicked as shown in Figure 4.42.

## RESULTS

The residual sums of squares (error) is : 7.78559217805

Star Name	Angle from Zenith	Distance from Zenith	Error
Zenith	0.0	0.0	0.0187625021172
Altair	23.83	84.0238061504	0.184797189407
Deneb	14.585	51.8652099196	0.655042651142
Vega	25.731	88.0511215147	4.82507936833
Grumium	37.82862	132.230858728	0.145726437502
Caph	45.73084	158.492902049	0.309180295563
Algenib	52.3822	177.910089652	1.05274556224
Scheat	32.7045	114.738833879	0.101056258484
Markab	37.8564	132.018938035	0.00582236055392
Hamam	35.64584	125.035994817	0.323867712394
Mizar	74.116	239.267632579	0.139669950845
Arcturus	83.6772	259.624729177	0.0238418894768

Figure 4.42: Calculated Data for Star Input Data Check

The table displays the star name, the angle from zenith, distance from zenith, and the calculated error. Using the error the user can refer to the star that is incorrectly input and modify the field in the form. Once the user has checked the data and is content with the information, the user may click the “Submit” button. This will store the acquired information by the user and create the necessary calculated files used for the image processing on the server side.

## RESULTS

### Modifying Calibration Data

The “Modify Calibration Data” page contains the same functionalities of the “Add Data Calibration” page. As shown in Figure 4.43 the page is composed of the same elements.

Home Map Account Logout

SITES  
McDonald Observatory  
Pescadero High School  
SRI International

SETTINGS  
Sites  
Calibration  
Schedule

### Modify Site Calibration

Currently modifying data for:  
Site: McDonald Observatory  
Latitude: 39.6714  
Longitude: -104.0225  
Elevation: 2070

User may modify the necessary fields below:

**Image information:**

Image Name:

Date:

UTC Time:

Administrator's Name:

**Stars Information:**

**Zenith:**

Zenith:  Azimuth:  Elevation:  i Coordinate:  j Coordinate:

**Star 1:**

Name:  Azimuth:  Elevation:  i Coordinate:  j Coordinate:

**Star 2:**

Name:  Azimuth:  Elevation:  i Coordinate:  j Coordinate:

**Star 3:**

Figure 4.43: Modify Calibration Data Page

The difference between the “Modify Calibration Data” page and the “Add Data Calibration” page is that once the page is rendered, the “Modify Calibration Data” page displays the previously stored information in the system in their respective fields.

#### 4.3.2.7 Schedule Setting Pages

The “Schedule Settings” page is composed of two options which are:

- adding a new schedule for a specific site
- modifying the existing schedule data

As a result, once the user selects the task to be performed, the user is routed to appropriate page to perform the task accordingly. The setup of the page and elements can be seen in Figure 4.44.

# RESULTS

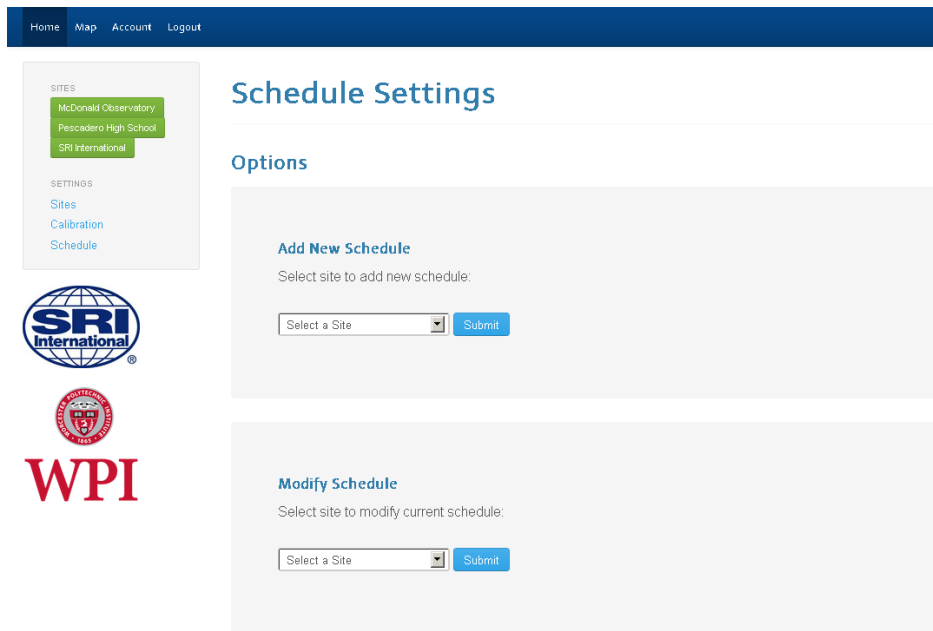


Figure 4.44: Schedule Settings Page

To add a new schedule for a site, the user must go to the “Add Schedule” section of the page as shown in Figure 4.45. Within the section there is a dropdown menu which allows the user to select the existing site which he or she wishes to add new schedule for. The user must click the “Submit” button which routes the user to the “Add Schedule Page”.

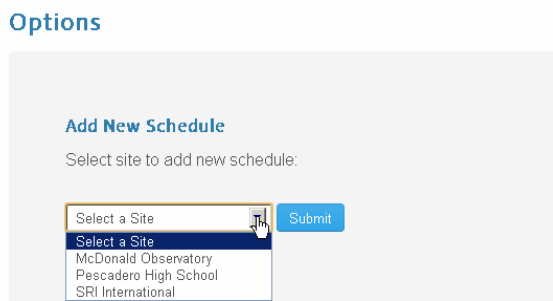


Figure 4.45: Add Schedule Option

If the user wishes to modify existing schedule data for a site, the user must go to the “Modify Schedule” section of the page as shown in Figure 4.46. Within the modify section there is a dropdown menu which allows the user to select the existing site the user wishes to modify. Once the user has selected the site to be modified the user must click “Submit”. As a result, this routes the user to the “Modify Schedule Page”.

## RESULTS

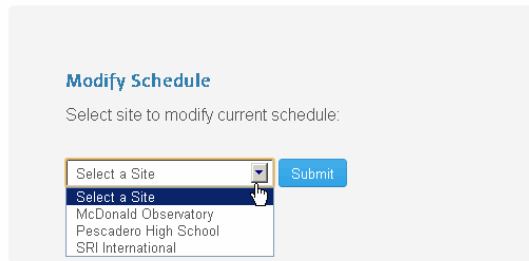


Figure 4.46: Modify Site Option

### Adding a Schedule

The “Add a Schedule” page displays a form where the user must input the necessary data to add a new schedule (as displayed in Figure 4.47). Under the heading of the page, the site specific information is displayed so the user can recall the remote site that he or she is adding a schedule for.

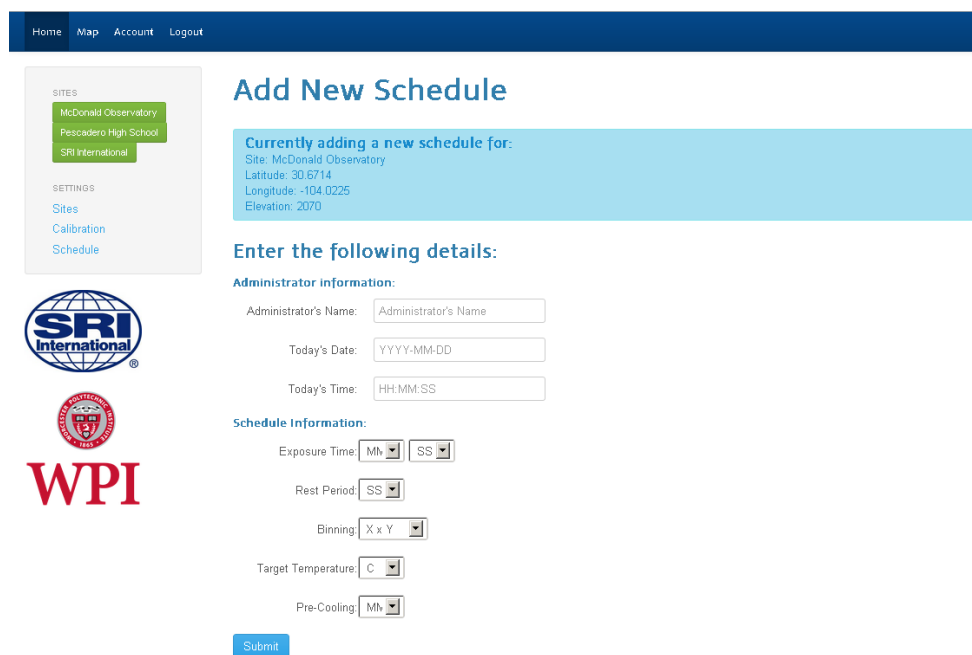


Figure 4.47: Add New Schedule Page

The page contains one form with two sections as shown Figure 4.48. The “Administrator Information” section asks the user to input information relating to the administrator entering the data. As a result, this information is stored for record keeping. The second section of the form is the “Schedule Information”. This section prompts the user to input the necessary information for the remote site system schedule.

## RESULTS

### Enter the following details:

#### Administrator information:

Administrator's Name:

Today's Date:

Today's Time:

#### Schedule Information:

Exposure Time:

Rest Period:

Binning:

Target Temperature:

Pre-Cooling:

**Figure 4.48: Add New Schedule Form**

Once the user has input the necessary fields and is content with the information, the user may click the “Submit” button. As a result, the information acquired is saved and transferred to the data acquisition program on the remote site. This will provide settings for the Data Acquisition program to run smoothly.

### Modifying a Schedule

The “Modify Schedule” page contains the same functionalities of the “Add New Schedule” page. As shown in Figure 4.49, the page comprises of the same elements. The difference between the “Modify Schedule” page and the “Add New Schedule” page is that the “Modify Schedule” page renders the details of the last edit (previously saved for record keeping). Also, once the page is rendered, it displays the pre-existing schedule information stored in the system in the respective fields.



## RESULTS

Home Map Account Logout

SITES  
McDonald Observatory  
Pescadero High School  
SRI International

SETTINGS  
Sites  
Calibration  
Schedule

### Modify Schedule

Currently modifying schedule for:  
Site: McDonald Observatory  
Latitude: 30.6714  
Longitude: -104.0225  
Elevation: 2070

This file was last edited:  
Administrator's Name: WPI  
Date Modified: 2013-02-28  
Time Modified: 11:09:53

#### Enter the following details:

**Administrator information:**  
Administrator's Name:   
Today's Date:   
Today's Time:

**Schedule information:**  
Exposure Time:    
Rest Period:   
Binning:   
Target Temperature:   
Pre-Cooling:

**SRI International**

**WPI**

Figure 4.49: Modify Schedule Page

As a result, having the information already in the fields allows easier schedule data modifications.

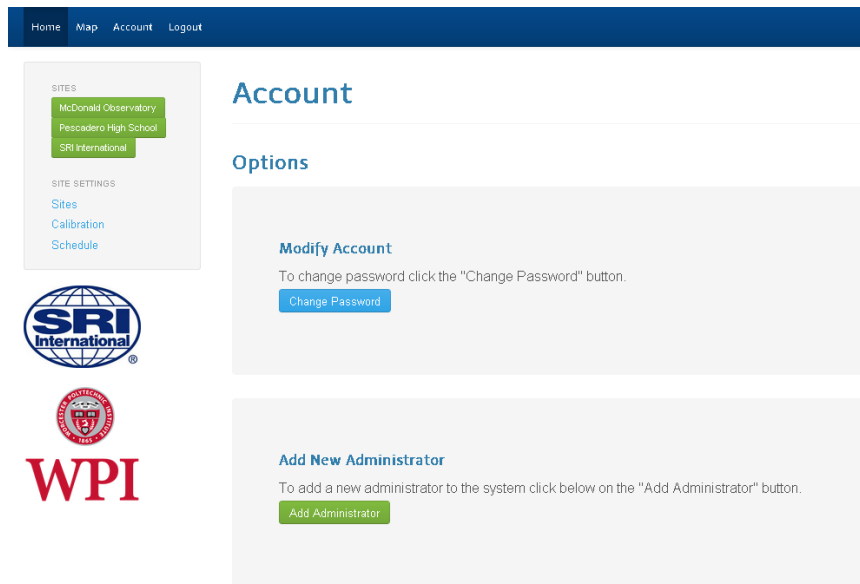
### 4.3.2.8 Account Pages

The “Account” page is composed of two options which are:

- change password
- add a new administrator to the MANGO system

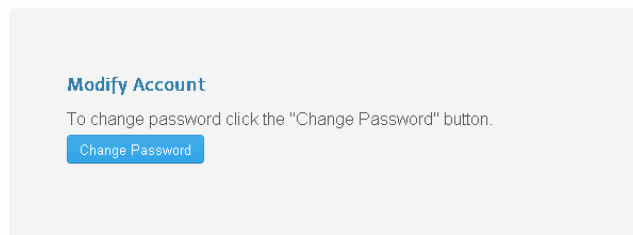
As a result, once the user selects one of the two tasks to be performed, he or she is routed to the appropriate page. Figure 4.50 displays the layout of the page.

# RESULTS



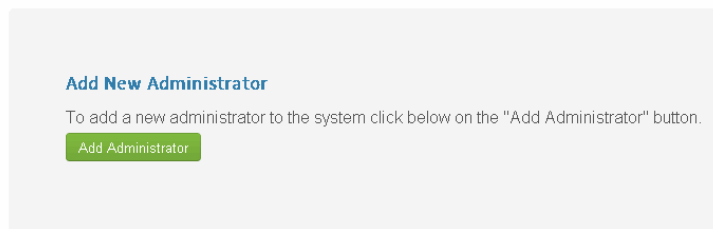
**Figure 4.50: Account page**

To change the password, the user must go to the “Modify Account” section of the page as shown in Figure 4.51. Clicking the “Change Password” button will redirect the user to the “Change Password” page.



**Figure 4.51: Modify Account Settings Option**

To add a new administrator to the system, the user can go to the “Add New Administrator” section on the page as shown in Figure 4.52.

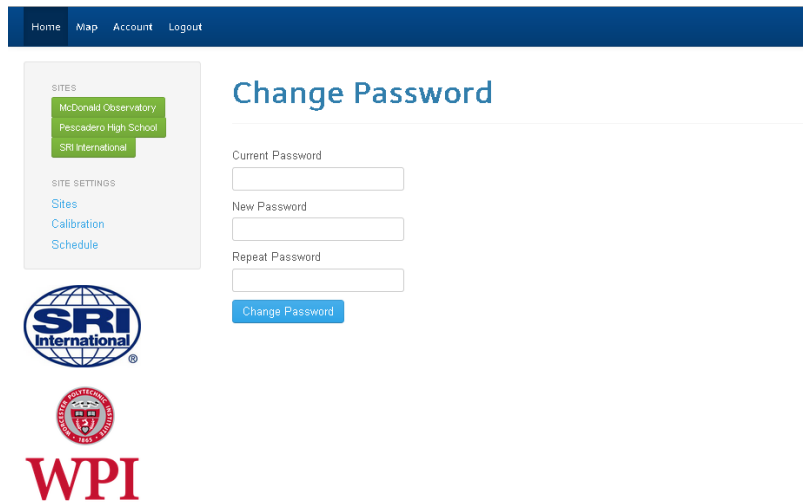


**Figure 4.52: Add New Administrator Option**

## RESULTS

### Changing Password

The “Change Password” page is composed of a form as shown in Figure 4.53. The form requests the user to input the current password, the new password, and to retype the new password. As a result, this page writes the new password to the internal encrypted database.

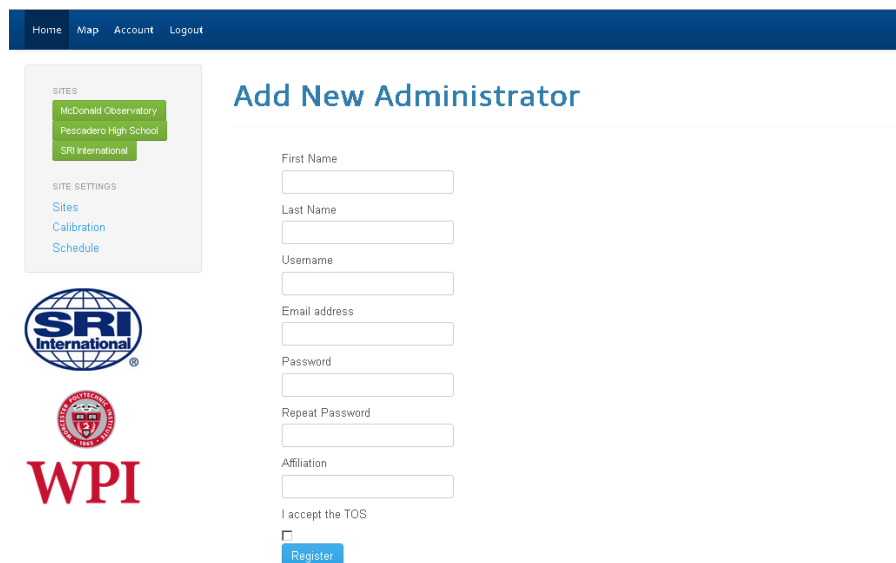


The screenshot shows a web interface for changing a password. At the top is a dark blue navigation bar with links for Home, Map, Account, and Logout. On the left is a sidebar menu with 'SITES' (McDonald Observatory, Pescadero High School, SRI International) and 'SITE SETTINGS' (Sites, Calibration, Schedule). The main content area is titled 'Change Password' and contains three input fields: 'Current Password', 'New Password', and 'Repeat Password'. A blue 'Change Password' button is located below the fields. At the bottom of the page are the SRI International and WPI logos.

Figure 4.53: Administrator Change Password Page

### Adding a New Administrator

The “Add a New Administrator” page allows an existing administrator to create another administrator account. The “Add a New Administrator” page is designed as a form as shown in Figure 4.54.



The screenshot shows a web interface for adding a new administrator. It features the same navigation bar and sidebar as Figure 4.53. The main content area is titled 'Add New Administrator' and contains a registration form with the following fields: 'First Name', 'Last Name', 'Username', 'Email address', 'Password', 'Repeat Password', and 'Affiliation'. Below the fields is a checkbox for 'I accept the TOS' and a blue 'Register' button. The SRI International and WPI logos are at the bottom.

Figure 4.54: Add New Administrator Page

## RESULTS

To add a new administrator, the administrator must fill-in the required fields. Once the administrator has completed the form, he or she must click “Register” to complete the process.

### *4.3.2.9 Logging out*

The administrator logout option is located in the menu bar of each page. The logout option will end the session and redirect the user to the MANGO main page.

With such features available in the website for system control and with our data acquisition and image processing programs in place, we successfully created a scalable system that can be used to study mid-latitude airglow and auroras.

## 5 Conclusion

Over the course of this project, the team learned about geophysical phenomena that affect contemporary telecommunication systems – an esoteric field of study within the broader discipline of electrical engineering. To meet SRI International’s need to study these geophysical phenomena, a problem statement was first reasoned and established. The identified problem was then broken down into simpler tasks and associated with a plan of action – a skill honed well by the Department of Electrical and Computer Engineering at WPI. In an attempt to meet all the requirements demonstrated by SRI International, the team efficiently applied its knowledge of image and signal processing algorithms, power systems, computer networks, and higher-level object-oriented design. Alongside, the team learned new topics from each other, from the scientists and engineers at SRI International and of course, from the Internet.

As a result of the comprehensive study of background topics and exploration of methodologies to design a robust real-time system for the study of mid-latitude airglows and auroras, our group designed one of the first astronomical image processing programs in Python thereby taking a detour from the more conventional IDL program designs. This was a feat of accomplishment resulting from the educated risk-taking and entrepreneurial culture fostered by the university. As a deliverable to our sponsor, we have developed a scalable system that can take synchronized all-sky-images from multiple sites across the United States and process them with minimal processing time by exploiting multicore processing. To visualize and monitor the data acquisition in real-time, we have also created a website which can display all the gathered information in the form of maps and plots for further scientific investigation.

### 5.1 Recommendations on Future Work

Although this project has successfully enabled SRI International’s goal of studying mid-latitude airglow and auroras, there are many scientific methods worth exploring to improve the MANGO project. The recommendations of such future work can be divided into three categories: firmware/hardware, back-end software, and front-end software.

## CONCLUSION

### ***5.1.1 Firmware/Hardware Recommendations***

In terms of future work in hardware development, the group first recommends creating Linux drivers for the <undisclosed astronomy grade> imager. The group was limited to using a Windows notebook to run the system because of the lack of availability of Linux drivers for the <undisclosed astronomy grade> CCD imager. Once Linux drivers are developed, the system can be transferred to a compact Linux computer such as Raspberry Pi. This development would have several benefits:

- Increased portability
- Free operating system
- Low cost hardware
- Low power consumption

### ***5.1.2 Back-end Software Recommendations***

Although the current image processing techniques are extremely robust and efficient, one of the steps in the multi-step procedure can be further improved. Currently in calibration, stars are identified manually in an image and compared with star-catalogue software. In theory, this method could be replaced with a random Monte-Carlo search which can plot possible values for camera tilt, pitch and yaw further increasing calibration precision. This could also result in a more accurate lens function enhancing the precision of the unwarping routine. Of course, Monte-Carlo searches can be time-consuming (in the order of hours at current computing speeds); however given that site calibration is not a procedure that needs to be repeated often, parameters saved from the first run can be re-used without affecting the real-time display requirements of this project.

### ***5.1.3 Front-end Software Recommendations***

Since SRI International intends to interact with MANGO-registered users via a comment wall or blog, it would be beneficial to use a database system more scalable than SQLite. SQLite is a single document database which was easy to setup and excellent for rapid development. However, SQLite is limited in size (an upper limit of 2 terabytes), does not allow robust user management and does not have many sophisticated features that some larger databases offer (such as concurrency). We recommend SRI International to consider larger databases such as PostgreSQL or MySQL. These options allow for better user and permissions management, simplified scaling, and larger amount of data storage.

## CONCLUSION

Since the MANGO map of airglow and auroras is a compilation of multiple images taken from all over the United States of America, multiple images may overlap the same geographic regions. To handle such situations, mapping APIs such as Leaflet Maps and Google Maps create a layer for each image whereby an upper layer will eclipse a lower layer. In our implementation, no specific image rank protocol has been developed to dictate which layer should eclipse the other. One metric that could be used to determine the layer rank of an image is the cloud cover in the region of the overlapping region. Near-real-time cloud cover databases are available free of charge online and these can be integrated with the MANGO system to determine which image should be displayed on top. This way, cloudy regions would be assigned a lower layer rank since they do not actually contain auroral and airglow information.

## References

- [1] Encyclopædia Britannica, “Atmosphere,” November 2012. [Online]. Available: <http://www.britannica.com/EBchecked/topic/41364/atmosphere>. [Accessed 18 November 2012].
- [2] Encyclopædia Britannica, “Ionosphere and Magnetosphere,” 2012. [Online]. Available: <http://www.britannica.com/EBchecked/topic/1369043/ionosphere-and-magnetosphere>. [Accessed 20 November 2012].
- [3] SRI International, “Specialized Facilities | SRI International,” 2012. [Online]. Available: <http://www.sri.com/engage/specialized-facilities>.
- [4] SRI International, “SRI Licenses and Ventures | SRI International,” 2012. [Online]. Available: <http://www.sri.com/blog/sri-licenses-and-ventures>.
- [5] SRI International, “Our Organization | SRI International,” 2012. [Online]. Available: <http://www.sri.com/about/organization>.
- [6] G. Vogt, *The Atmosphere: Planetary Heat Engine*, Minneapolis: Twenty-First Century Books, 2007.
- [7] Encyclopædia Britannica Online, “layers of Earth’s ionosphere,” 2012.
- [8] J. M. Goodman, *Space Weather & Telecommunications*, 1st Edition ed., New York: Springer, 2005.
- [9] Encyclopædia Britannica, “Ionization,” 2012. [Online]. Available: <http://www.britannica.com/EBchecked/topic/293007/ionization>. [Accessed 20 November 2012].
- [10] University of Iowa/NASA Scientific Visualization Studio, “Aurora Over the North Pole of Earth,” 2000.



## REFERENCES

- [11] C. Freudenrich, "How Auroras Work," 11 March 2008. [Online]. Available: <http://science.howstuffworks.com/nature/climate-weather/atmospheric/aurora1.htm>. [Accessed 20 November 2012].
- [12] Z. Kenwell, "Auroras invade the US," NASA, Edmonton, 2011.
- [13] National Aeronautics Space Administration, 17 01 2007. [Online]. Available: [http://www.nasa.gov/mission\\_pages/themis/news/Themis\\_intro.html](http://www.nasa.gov/mission_pages/themis/news/Themis_intro.html). [Accessed 20 11 2012].
- [14] NASA, "A quarter moon is visible in this oblique view of Earth's horizon and airglow," 2008.
- [15] IPS Radio and Space Services, "Introduction to HF Radio Propagation," 11 2012. [Online]. Available: <http://www.ips.gov.au/Category/Educational/Other%20Topics/Radio%20Communication/Intro%20to%20HF%20Radio.pdf>. [Accessed 20 11 2012].
- [16] International Space Environmental Service, "HF Radio for Trans-Polar Flights," 12 2010. [Online]. Available: [http://www.spaceweather.org/swo.php?target=hf\\_for\\_TPF&id=p3&include=hf\\_for\\_TPF&title=HF+Radio+for+Trans-Polar+Flights&lang=](http://www.spaceweather.org/swo.php?target=hf_for_TPF&id=p3&include=hf_for_TPF&title=HF+Radio+for+Trans-Polar+Flights&lang=). [Accessed 20 11 2012].
- [17] Exetel Pty Ltd., "Sporadic E Ionisation," 11 2012. [Online]. Available: <http://home.exetel.com.au/auriga/AR/Tech/es/Es.html>. [Accessed 11 2012].
- [18] SPL, 8 March 2012. [Online]. Available: <http://www.bbc.co.uk/news/science-environment-17295337>. [Accessed 20 November 2012].
- [19] R. S. Dabas, "Ionosphere and its Influence of Radio Communications," *Resonance*, vol. 5, no. 7, pp. 28-43, July 2000.
- [20] C. D. Ahrens, "Ionospheres and Radio Wave Propagation," in *Meteorology Today: An Introduction to Weather, Climate, and the Environment*, A. Berg, Ed., Thomson Higher Education, 2012, p. 17.
- [21] B. Hönlinger and H. H. Nasse, "Distortion," Carl Zeiss AG, October 2009. [Online]. Available: [http://www.zeiss.com/C12567A8003B8B6F/EmbedTitelIntern/CLN\\_33\\_Distortion\\_EN/](http://www.zeiss.com/C12567A8003B8B6F/EmbedTitelIntern/CLN_33_Distortion_EN/)

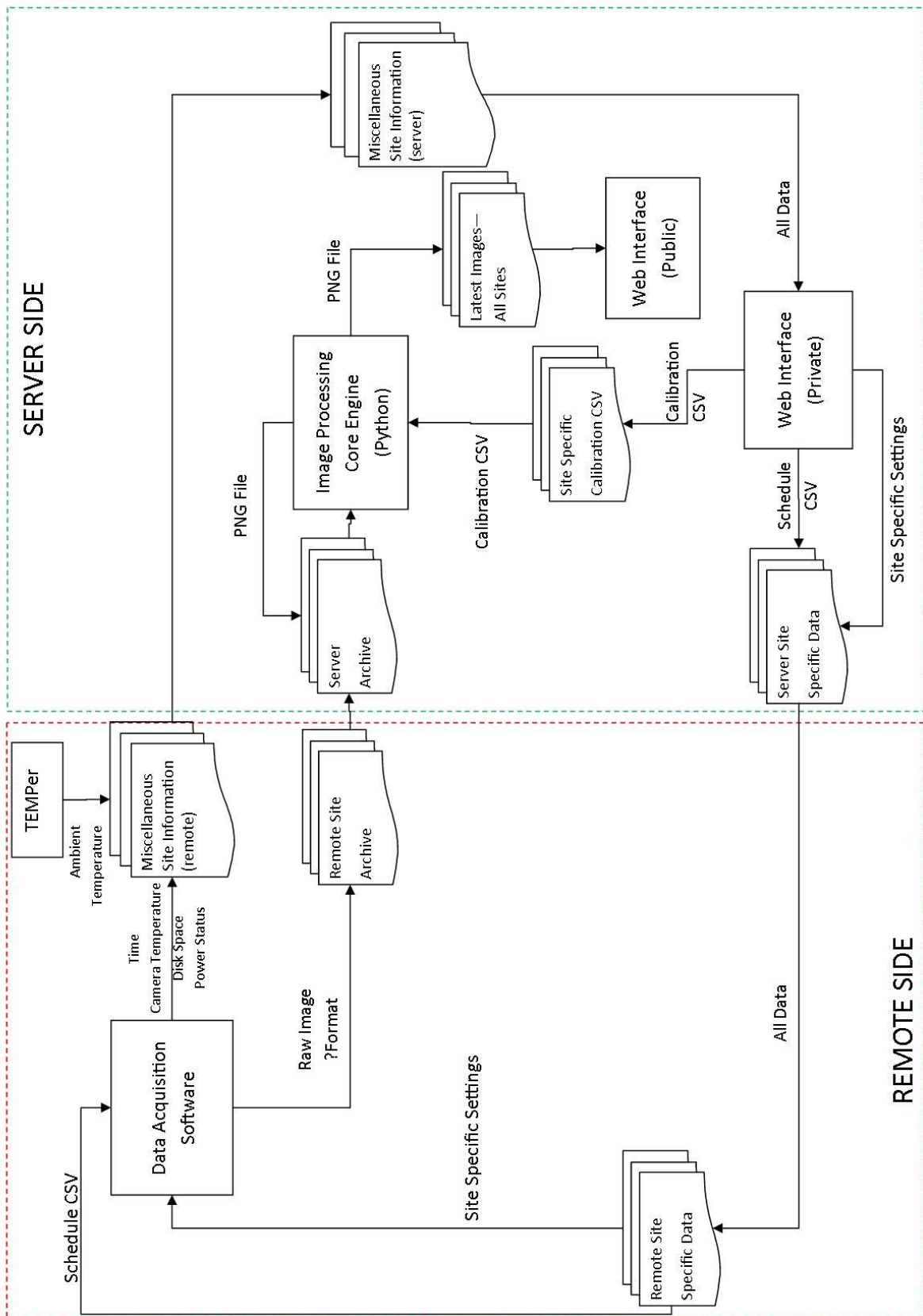
## REFERENCES

- \$File/CLN33\_Distortion\_Article.pdf. [Accessed 20 November 2012].
- [22] D. G. Bailey, "A new approach to lens distortion correction," in *Image and Vision Computing New Zealand*, Auckland, 2002.
- [23] F. J. Garcia, M. J. Taylor and M. C. Kelley, "Two-dimensional spectral analysis of mesospheric airglow image data," *Applied Optics*, vol. 36, no. 29, pp. 7374-7385, October 1997.
- [24] J. F. Salgado, "Imagine being a fly VLT," 2010.
- [25] Quantum Scientific Imaging Inc., "Understanding CCD Read Noise," [Online]. Available: [http://www.qsimaging.com/ccd\\_noise.html](http://www.qsimaging.com/ccd_noise.html). [Accessed 04 02 2013].
- [26] M. Kubota, H. Fukunishi and S. Okano, "Characteristics of medium- and large-scale TIDs over Japan derived from OI 630nm nightglow observation," *Earth Planets Space*, no. 53, pp. 741-751, 2001.
- [27] J. W. Chamberlain, *Physics of Aurora and Airglow*, J. V. Miegheem, Ed., London: Academic Press, 1961.
- [28] R. Widenhorn, M. M. Blouke, A. Weber, A. Rest and E. Bodegom, "Temperature dependence of dark current in a CCD," in *Proceedings of SPIE*, Bellingham, 2002.
- [29] M. J. Taylor and M. A. Hapgood, "Analysis of airglow image data," *Ann. Geophys.*, vol. 38, no. 6, pp. 805-813, 1982.
- [30] P. Osborne, "The Mercator Projections," Edinburgh, 2008.
- [31] J. L. Peterson, "Petri Nets," *ACM Computing Surveys*, vol. 9, no. 3, pp. 223-252, September 1977.
- [32] R. Fehling, "A concept of hierarchical Petri nets with building blocks," *Lecture Notes in Computer Science*, vol. 674, pp. 148-168, 1993.
- [33] R. Millner, *A Calculus of Communicating Systems*, Springer, 1982.
- [34] D. Taubner, "Finite representations of CCS and TCSP programs by automata and Petri nets,"

## REFERENCES

- Lecture Notes In Computer Science*, vol. 369, p. 168, 1989.
- [35] J. M. Fernandes, "VHDL generation from hierarchical petri net specifications of parallel controllers," *IEE Proceedings: Computers and Digital Techniques*, vol. 144, no. 2, pp. 127-137, March 1997.
- [36] M. J. Morley, Modelling British Rail's Interlocking Logic: Geographic Data Correctness, LFCS Report Series: ECS-LFCS-91-186, Dept. Computer Science, University of Edinburgh, 1991.
- [37] H. Sizun, Radio Wave Propagation for Telecommunication Applications, 1st Edition ed., Berlin: Springer, 2004.
- [38] Encyclopædia Britannica Online, "ozone layer: layers of the atmosphere," 2012.
- [39] D. Huddart, Earth Environments: Past, Present, and Future, 1st Edition ed., John Wiley & Sons, 2010.
- [40] T. Tsujii, T. Fujiwara and T. Kubota, "Novel navigation systems ensuring the safety of satellite navigation," Tokyo, 2012.
- [41] Wikimedia Commons - Tfr00, "Ra and dec on celestial sphere," 2012.
- [42] W. M. Smart, Textbook on spherical astronomy, 5th ed., Cambridge: Cambridge University Press, 1965.
- [43] M. P. Levesque and S. Buteau, Image processing technique for automatic detection of satellite streaks, 1st Edition ed., Vols. 2005-386, Valcartier: Defence Research and Development Canada, 2007.
- [44] J.-W. F. Zijffers, M. Janssen, J. Tramper and H. R. Wijffels, "Design Process of an Area-Efficient Photobioreactor," *Marine Biotechnology*, vol. 10, no. 4, pp. 404-415, July 2008.
- [45] W.-L. H. a. K.-Y. H. Ming-Shing Su, "Analysis of Multiresolution Mosaic Images," *IEE Transactions of Image Processing*, vol. 13, no. 7, pp. 952-959, 2004.
- [46] A. Levin, A. Zomet, S. Peleg and Y. Weiss, "Seamless Image Stitching in the Gradient Domain," *IEEE Transactions on Image Processing*, vol. 15, no. 4, pp. 969-977, 2006.

# Appendix A: MANGO System Architecture



# Appendix B: Image Acquisition Program

```
#####
#
# Camera Control Core Engine for MANGO
# (Midlatitude All-sky-imager Network for Geophysical Observation)
# 2013-02-01 Rohit Mundra/Fabrice Kengne
#           Initial implementation
#
# 2013-02-21 Fabrice Kengne
#           Added PyFITS support
#
#####

import os
import numpy
import re
import sys
import pyfits
import datetime
import optparse
import ephem
import csv

from ctypes          import *
from ctypes          import wintypes
from time            import sleep
from glob            import glob
from ntpath          import basename
from datetime        import datetime

#Specifying output files and folder locations
ccdFile      = os.path.dirname(os.getcwd()) + "\\system\\ccd.csv"
imageFolder  = os.path.dirname(os.getcwd()) + "\\images\\"

class atikCamera:
    def __init__(self, site = 'defaultName', exposureTime = 0.1, binX = 1, binY = 1, targetTemperature = -5
    , latitude = 0, longitude = 0):
        self.targetTemperature = targetTemperature
        self.binX = c_int(int(binX))
        self.binY = c_int(int(binY))
        self.driverDLL = windll.LoadLibrary("ArtemisHSC.dll")
        self.site = site
        self.exposureTime = exposureTime
        self.temperatureSensor = c_int(1) #Fixed Value
        self.cameraHandle = 0 #No Camera Connected yet
        self.connectFailureCount = -1 #Loop count variable
        self.disconnectFailureCount = -1 #Loop count variable
        self.currentCCDTemperature = c_int(-273) #Temperature not read yet
        self.dirPath = os.getcwd()
        self.latitude = latitude
        self.longitude = longitude

# Updates settings, captures image, saves to disk as a FITS image
# NOTE: Does not disconnect camera. This means that CCD will continue cooling.
    def process(self):
        self.connectCamera()
        self.setTargetTemperature()
        self.setBinning()
        self.capture()
        self.getCCDTemperature()
        self.getLastStartTime()
        self.getBinning()
        self.writeImage()
```

## APPENDIX B: Image Acquisition Program

```

# Connects the camera and in case of failure, alerts the user
def connectCamera(self):
    self.driverDLL.ArtemisDisconnectAll()
    while(self.cameraHandle == 0):
        self.connectFailureCount += 1
        self.driverDLL.ArtemisDisconnectAll()
        self.cameraHandle = self.driverDLL.ArtemisConnect(c_int(-1))
        if self.connectFailureCount == 25:
            print 'Cannot connect to camera after 25 attempts!'
        self.connectFailureCount = -1

# Takes exposure for the specified exposure time in seconds
def capture(self):
    error = self.driverDLL.ArtemisHighPriority(self.cameraHandle, c_bool(1))
    print error
    error = self.driverDLL.ArtemisStartExposure(self.cameraHandle, c_float(self.exposureTime))
    if error:
        print 'Cannot start exposure on camera! ERROR CODE = ', error, '(Check Artemis SDK for details)'
    else:
        while(self.driverDLL.ArtemisDownloadPercent(self.cameraHandle) != 100):
            sleep(2)
        x = c_int(0)
        y = c_int(0)
        w = c_int(0)
        h = c_int(0)
        self.driverDLL.ArtemisGetImageData(self.cameraHandle, byref(x), byref(y), byref(w), byref(h),
        byref(self.binX), byref(self.binY))
        imageSize = w.value*h.value
        pImage = c_void_p(None)
        print "CAMERA HANDLE : ", self.cameraHandle
        while (pImage.value == c_void_p(None).value) or (pImage.value == self.cameraHandle):
            pImage = c_void_p(self.driverDLL.ArtemisImageBuffer(self.cameraHandle))
            sleep(2)
        print "POINTER TO DATA : ", pImage
        imageBuffer = (c_ushort*imageSize)()
        bytesPerPixel = 2 # camera has a 16-bit ADC
        memmove(imageBuffer, pImage, imageSize*bytesPerPixel)
        self.imageArray = numpy.reshape(numpy.array(imageBuffer[0:imageSize]), [h.value, w.value]).
        astype(numpy.uint16)
        print "IMG DATA : ", self.imageArray

# Writes current Image Array in FITS format. The header is filled with information collected from the files
# in the code. The information
# contain: site name, binning factors, longitude, latitude, exposure time, etc.
def writeImage(self):
    hdu = pyfits.PrimaryHDU(numpy.flipud(self.imageArray))
    hdu.header['UTCTIM'] = (self.lastStartTime, 'UTC time at start of exposure')
    hdu.header['EXPTIM'] = (self.exposureTime, 'Length of exposure in seconds')
    hdu.header['FILTER'] = ('630nm', 'Filter wheel in use')
    hdu.header['TEMP'] = (self.currentCCDTemperature, 'CCD Temperature in degrees celsius')
    hdu.header['BINX'] = (self.binX.value, 'Binning X')
    hdu.header['BINY'] = (self.binY.value, 'Binning Y')
    hdu.header['SITE'] = (self.site, 'Site Name')
    hdu.header['LAT'] = (self.latitude, 'Site latitude')
    hdu.header['LON'] = (self.longitude, 'Site longitude')
    hdulist = pyfits.HDULIST([hdu])
    year, month, day, hour, minutes, seconds = re.split('-:| ', self.lastStartTime)
    year = int(year)
    day = int(day)
    month = int(month)
    #finds the number of days per month, and determine if a year is a leap year
    if year%4 == 0:
        febDays = 29
    else:
        febDays = 28
    if month == 1:

```

## APPENDIX B: Image Acquisition Program

```

        dayNumber = day
    elif month == 2:
        dayNumber = 31 + day
    elif month == 3:
        dayNumber = 31 + febDays + day
    elif month == 4:
        dayNumber = 31 + febDays + 31 + day
    elif month == 5:
        dayNumber = 31 + febDays + 31 + 30+ day
    elif month == 6:
        dayNumber = 31 + febDays + 31 + 30 + 31 + day
    elif month == 7:
        dayNumber = 31 + febDays + 31 + 30 + 31 + 30 + day
    elif month == 8:
        dayNumber = 31 + febDays + 31 + 30 + 31 + 30 + 31 + day
    elif month == 9:
        dayNumber = 31 + febDays + 31 + 30 + 31 + 30 + 31 + 31 + day
    elif month == 10:
        dayNumber = 31 + febDays + 31 + 30 + 31 + 30 + 31 + 31 + 30 + day
    elif month == 11:
        dayNumber = 31 + febDays + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 + day
    elif month == 12:
        dayNumber = 31 + febDays + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 + 30 + day
    if int(dayNumber/100) == 0:
        dayNumber = '0'+ str(dayNumber)
    else:
        dayNumber = str(dayNumber)
    year = str(year%100)
    time = hour + minutes + seconds
    filename = self.site + '_' + year + '_' + dayNumber + '_' + time + '_R' + '.fit'
    hdulist.writeto(imageFolder+filename)
    try:
        ccdFID = open(ccdFile, 'a+b')
    except:
        ccdFID = open(ccdFile, 'w+b')
    writer = csv.writer(ccdFID, dialect = 'excel')
    newCCDRow = [str(datetime.utcnow()), str(self.currentCCDTemperature)]
    writer.writerow(newCCDRow)
    ccdFID.close()

# Disconnect the camera. In case of failure, alerts the user
def disconnectCamera(self):
    disconnectSuccess = self.driverDLL.ArtemisDisconnectAll()
    while (disconnectSuccess==0):
        self.disconnectFailureCount += 1
        disconnectSuccess = self.driverDLL.ArtemisDisconnectAll()
        if disconnectFailureCount == 25:
            print 'Cannot disconnect camera after 25 attempts!'
    self.cameraHandle = 0
    self.disconnectFailureCount = -1

# Sets the binning factor. Default binning is X = 1 & Y = 1
def setBinning(self):
    error = self.driverDLL.ArtemisBin(self.cameraHandle, self.binX, self.binY)
    if error:
        print 'Cannot set binning on camera! ERROR CODE = ', error, '(Check Artemis SDK for details)'
        self.binX = -1
        self.binY = -1

# Sets the desired target temperature for the exposure in degrees celsius
def setTargetTemperature(self):
    error = self.driverDLL.ArtemisSetCooling(self.cameraHandle, c_int(int(100*self.targetTemperature)))
    if error:
        print 'Cannot set target temperature on camera! ERROR CODE = ', error, '(Check Artemis SDK for
details)'
        self.targetTemperature = -273

```

## APPENDIX B: Image Acquisition Program

```
# Gets the UTC time (camera time) of the start of the last exposure
def getLastStartTime(self):
    self.lastStartTime = c_char_p(self.driverDLL.ArtemisLastStartTime(self.cameraHandle)).value

# Gets the CCD temperature in degrees celsius at the time of capture.
# Alert the user in case the camera is not found
def getCCDTemperature(self):
    error = self.driverDLL.ArtemisTemperatureSensorInfo(self.cameraHandle, c_int(1), byref(self.
currentCCDTemperature ))
    if error:
        print 'Cannot read CCD temperature! ERROR CODE = ', error, '(Check Artemis SDK for details)'
    else:
        self.currentCCDTemperature = float(self.currentCCDTemperature .value)/100

# Gets the binning factor sets for the CCD camera
def getBinning(self):
    error = self.driverDLL.ArtemisGetBin(self.cameraHandle, byref(self.binX), byref(self.binY))
    if error:
        print 'Cannot get binning! ERROR CODE = ', error, '(Check Artemis SDK for details)'
```



# Appendix C: Data Acquisition Program

```
#####
#
# Schedule Manager
# (Midlatitude All-sky-imager Network for Geophysical Observation)
# 2013-02-12 Rohit Mundra
#           Initial implementation
#
# 2013-02-25 Rohit Mundra/Fabrice Kengne
#           Added Solar Zenith Angle Decisions
#           Added Precooling
#           Added Web Power Switch Simulation
#           Added System Information Logging
#
# 2013-02-26 Rohit Mundra/Fabrice Kengne
#           Completed Web Power Switch Implementation
#
#####

from datetime      import datetime
from time          import *
from atikCamera    import atikCamera
from ntpath        import basename
from glob          import glob
from ctypes        import wintypes
from ctypes        import *
from numpy         import sin, cos, tan, arcsin, arccos, radians, degrees
import numpy       as np
import math
import sys
import os
import threading
import ephem
import csv
import urllib2
import base64

#Some settings that can be changed

#Powerswitch variables - extremely critical for functioning!
powerswitchIPAddress = "192.168.0.100" #Format "255.255.255.255" NO SLASHES
outlet               = "4"             #Format "4" THIS VAR IS NOT AN INTEGER; IT IS A STRING
username             = "admin"        #default
password             = "1234"        #default

#Path to critical files for input: Schedule, Sensor data, and Calibration information
scheduleFilename     = os.path.dirname(os.getcwd()) + "\\fromserver\\Schedule.csv"
pcSensorDataPath     = os.path.dirname(os.getcwd()) + "\\temper\\*.csv*"
calibrationInfoFile  = os.path.dirname(os.getcwd()) + "\\fromserver\\CalibrationInfo.csv"

#Path for system information output files: battery life, disk spacem and collected ambient temperature
batteryFile          = os.path.dirname(os.getcwd()) + "\\system\\battery.csv"
diskFile             = os.path.dirname(os.getcwd()) + "\\system\\disk.csv"
ambientFile          = os.path.dirname(os.getcwd()) + "\\system\\ambient.csv"

#Solar zenith angle in degrees
solarZenithAngleThreshold = 100 #default

#seconds after 0000hrs, daily schedule update at 2200hrs UTC
timeToUpdateSchedule = 72000 #default

#degrees C (Above this temp, camera shuts off)
thresholdForPowerSwitch = 40.0 #default

#seconds - determines the frequency at which Sys. Info. should be acquired
```

## APPENDIX C: Data Acquisition Program

```

perImageTime
    self.imageTimesTotal = np.append(self.imageTimesTotal, imageTime)
    # REMOVING TIMES WHEN SUN IS PRESENT
    allZenithAngles = []
    for each in self.imageTimesTotal:
        allZenithAngles.append(self.getSolarAngle(each))
    for i in range(len(allZenithAngles)):
        if allZenithAngles[i] < solarZenithAngleThreshold:
            self.imageTimesTotal[i] = float('NaN')
    self.imageTimesTotal = [x for x in self.imageTimesTotal if not(math.isnan(x))]
    # END OF REMOVAL

# This function sets the web-power switch on. It uses admin and password information to login to
# the web page to get web control of the power-switch.
def setPowerSwitchOn(self):
    requestPage = "http://"+powerswitchIPAddress+"/outlet?"+outlet+"=ON"
    request = urllib2.Request(requestPage)
    base64string = base64.encodestring('%s:%s' % (username,password)).replace('\n', '')
    request.add_header("Authorization", "Basic %s" % base64string)
    info = urllib2.urlopen(request).read()
    print 'Web-PS: On'

# This function sets the web-power switch off. It uses admin and password information to login to
# the web page to get web control of the power-switch.
def setPowerSwitchOff(self):
    requestPage = "http://"+powerswitchIPAddress+"/outlet?"+outlet+"=OFF"
    request = urllib2.Request(requestPage)
    base64string = base64.encodestring('%s:%s' % (username,password)).replace('\n', '')
    request.add_header("Authorization", "Basic %s" % base64string)
    info = urllib2.urlopen(request).read()
    print 'Web-PS: Off'

# This function calculates the solar zenith angle using current local UTC time and last zenith equinox
# occurrence.
def getSolarAngle(self, secondsSinceDay):
    localTime = datetime.now()
    localTimeUTC = datetime.utcnow()
    lastEquinox = ephem.previous_equinox(localTime).datetime()
    timeSince = localTimeUTC-lastEquinox
    days = timeSince.days
    latitude = radians(self.latitude) #Get Latitude of site in rads
    longitude = radians(self.longitude) #Get Longitude of site in rads
    earthTilt = radians(23.5)
    myLambda = radians(360*days/365.24)
    localTimeOffset = (timezone if (localtime().tm_isdst == 0) else altzone)/(-3600)
    utcHours = secondsSinceDay/3600.0
    localTimeHours = utcHours + localTimeOffset
    hourAngle = radians(360*((localTimeHours+12)%24)/24)
    solarDeclination = arcsin(sin(earthTilt)*sin(myLambda))
    solarZenithAngle = arccos(sin(latitude)*sin(solarDeclination) + cos(latitude)*cos
(solarDeclination)*cos(hourAngle))
    return degrees(solarZenithAngle)

# This function checks for ambient temperature, solar zenith, current time and how much time left until
# next image capture, time until
# next CSV schedule update, time to check and write system data. If all the conditions are met, the program
# proceed to capturing
# images. When current less than the threshold zenith angle, the web-power switch is off. The switch turns
# on, wait for sometime and
# then start precooling. When it is time to start capture, the camera starts taking images.
def captureImages(self):
    camera = atikCamera(self.siteName, self.exposure, self.binX, self.binY,
self.targetTemperature, self.latitude, self.longitude)
    camera.connectCamera() #connect the camera to
the computer
    exposureThread = threading.Thread(target = camera.process) #gets the thread to run

```

## APPENDIX C: Data Acquisition Program

```

atik camera process function
powerStatus          = self.getPowerStatus()           #gets power status.
Battery life
totalDisk, usedDisk, freeDisk = self.getDiskUsage()    #gets the disk space
lastTemp, lastTempTime = self.getAmbientTemperature()  #gets last temperature
from the PCsensor file and the time the reading was recorded
while(1):
    sleepPeriod      = 0.5 #seconds
    currentTime      = datetime.utcnow()
    currentHour      = currentTime.hour
    currentMinute    = currentTime.minute
    currentSecond    = currentTime.second + currentTime.microsecond/1000000.0
    currentTimeTotal = currentHour*3600 + currentMinute*60 + currentSecond #seconds after 00:00:00
00
    if currentTimeTotal>=timeToUpdateSchedule-sleepPeriod and currentTimeTotal<timeToUpdateSchedule
:
        self.readScheduleCSV()
        self.readCalibrationInfoCSV()
        self.createCaptureSchedule()
        #check if it is time to read system information and update the CSV files
        if currentTimeTotal%periodToUpdateSystemInfo <= sleepPeriod:
            powerStatus = self.getPowerStatus()
            totalDisk, usedDisk, freeDisk = self.getDiskUsage()
            lastTemp, lastTempTime = self.getAmbientTemperature()
            try:
                batteryLifePercent = powerStatus.BatteryLifePercent
            except:
                batteryLifePercent = 0
            self.logSystemInformation(batteryLifePercent, freeDisk, lastTemp, lastTempTime)
            #find how much time is left till the next exposure
            timeDifference = [(x - currentTimeTotal + 86400)%86400 for x in self.imageTimesTotal]
            minTime = min(timeDifference) #time to begin exposure
            print 'Exp begins in:', minTime, 'seconds'
            #check if it is time to start precooling
            secondsBeforeImaging = 60*self.precoolingDuration + 30
            #check if it is also time to start exposures, also check the ambient temperature. If they all
succeed and precooling is set to 1 in the schedule file, checks the
            #time for precooling
            if minTime<secondsBeforeImaging:
                if (lastTemp<thresholdForPowerSwitch):
                    if self.precooling == 1:
                        timeTillBeginPrecool = minTime - 60*self.precoolingDuration - 5
                        print 'Pre-cooling begins in :', timeTillBeginPrecool, 'seconds'
                        #if it is time to precool and the ambient temperature is less than the threshold,
web power switch is switch on (almost time to start exposures), the
                        #camera is connected, and the cooling thread begins.
                        if timeTillBeginPrecool<sleepPeriod and timeTillBeginPrecool>-sleepPeriod:
                            self.setPowerSwitchOn()
                            camera = atikCamera(self.siteName, self.exposure, self.binX, self.
binY, self.targetTemperature, self.latitude, self.longitude)
                            camera.connectCamera()
                            coolingThread = threading.Thread(target = camera.setTargetTemperature)
                            if not(coolingThread.is_alive()):
                                coolingThread.start() #cooling thread starts
                                print 'Atik has started cooling'
                            self.setPowerSwitchOn()
                        if minTime<2.0:
                            camera = atikCamera(self.siteName, self.exposure, self.binX, self.
binY, self.targetTemperature, self.latitude, self.longitude)
                            if not(exposureThread.is_alive()):
                                exposureThread = threading.Thread(target = camera.process)
                                exposureThread.start() #exposure thread starts
                                print 'Atik has started capturing'
                    else:
                        self.setPowerSwitchOff() #if ambient temperature greater than threshold, the web
power switch is off

```

## APPENDIX C: Data Acquisition Program

```

        else:
            if not(exposureThread.is_alive()):
                self.setPowerSwitchOff() #if time to start expose greater than min time to start
exposure, web power switch is off
                sleep(sleepPeriod/2)

# This function reads the last ambient temperature from the PCsensor CSV file and returns the value as well
as date
# and time. If there is no value, it will assign one and record the temperature reading as 0.0 degrees,
date and
# time as 0/0/0000 0:00:00 AM. This will allow scheduler.py to run even if there is no PCsensor connected.
def getAmbientTemperature(self):
    path = self.pcSensorDataPath
    listOfFiles = glob(path)
    try:
        latestFile = max(listOfFiles, key=os.path.getmtime) #gets the
latest CSV file in the ambient temperature folder
        sensorData = np.genfromtxt(latestFile, dtype = float, delimiter=',') #gets sensor
data
        lastTempTime = np.genfromtxt(latestFile, dtype = str, delimiter=',')[-1, 2] #obtains last
temperature reading time
        lastTemperature = sensorData[-1, 1] #obtains last
temperature reading
    except:
        lastTemperature = 0.0 #if no reading is found, write 0.0 degrees
        lastTempTime = "0/0/0000 0:00:00 AM"

    return lastTemperature, lastTempTime

# This function creates log files for the battery, disk, and ambient temperature. It also update existing
log
# files. The log files contains their values as well as the time those values were recorded.
def logSystemInformation(self, batteryCharge, diskUsage, ambientTemperature, ambientTemperatureTime):
    currentTime = datetime.utcnow()
    try:
        batteryFID = open(batteryFile, 'a+b') #open file in binary mode for appending data
    except:
        batteryFID = open(batteryFile, 'w+b')
    writer = csv.writer(batteryFID, dialect = 'excel')
    newBatteryRow = [str(currentTime), str(batteryCharge)]
    writer.writerow(newBatteryRow)
    batteryFID.close()
    try:
        diskFID = open(diskFile, 'a+b')
    except:
        diskFID = open(diskFile, 'w+b')
    writer = csv.writer(diskFID, dialect = 'excel')
    newDiskRow = [str(currentTime), str(diskUsage)]
    writer.writerow(newDiskRow)
    diskFID.close()
    try:
        ambientFID = open(ambientFile, 'a+b')
    except:
        ambientFID = open(ambientFile, 'w+b')
    writer = csv.writer(ambientFID, dialect = 'excel')
    newAmbientRow = [str(ambientTemperatureTime), str(ambientTemperature)]
    writer.writerow(newAmbientRow)
    ambientFID.close()
    #Log input arguments to a CSV, which will be shared with server

# This function gets the battery status. It uses windows kernel.
def getPowerStatus(self):
    class SYSTEM_POWER_STATUS(Structure):
        _fields_ = [('ACLineStatus', wintypes.BYTE), ('BatteryFlag', wintypes.BYTE),
('BatteryLifePercent', wintypes.BYTE), ('Reserved1', wintypes.BYTE), ('BatteryLifeTime', wintypes.
DWORD), ('BatteryFullLifeTime', wintypes.DWORD), ]

```

## APPENDIX C: Data Acquisition Program

```
SYSTEM_POWER_STATUS_P          = POINTER(SYSTEM_POWER_STATUS)
GetSystemPowerStatus           = windll.kernel32.GetSystemPowerStatus
GetSystemPowerStatus.argtypes  = [SYSTEM_POWER_STATUS_P]
GetSystemPowerStatus.restype   = wintypes.BOOL
status                          = SYSTEM_POWER_STATUS()
if not GetSystemPowerStatus(pointer(status)):
    print 'Error acquiring power status of computer!'
return status

# This function gets the information about the disk. It uses windows kernel.
def getDiskUsage(self, path = 'C:\\'):
    _, total, free = c_ulonglong(), c_ulonglong(), c_ulonglong()
    if sys.version_info >=(3,) or isinstance(path, unicode):
        fun        = windll.kernel32.GetDiskFreeSpaceExW
    else:
        fun        = windll.kernel32.GetDiskFreeSpaceExA
    ret            = fun(path, byref(_), byref(total), byref(free))
    used          = total.value - free.value
    return total.value/(1024.**3), used/(1024.**3), free.value/(1024.**3)

if __name__ == '__main__':
    atikScheduler = scheduler()
    atikScheduler.process()
```

# Appendix D: Image Processing Core

```

#####
#
# Image Processing Core Engine for MANGO
# (Midlatitude All-sky-imager Network for Geophysical Observation)
# 2013-01-23 Rohit Mundra
#           Initial implementation
#
# 2013-02-05 Rohit Mundra
#           Robust Star Removal implemented
#           Histogram Equalization implemented
#
# 2013-02-22 Rohit Mundra
#           Changing unwarping to map to Normal Mercator Projection
#           Added alpha mask and PIL PNG saving
#
# 2013-02-22 Fabrice Kengne
#           Adding compatibility for FITS read
#
#####

from numpy          import *
from numpy          import genfromtxt
from numpy          import linalg
from math           import atan
from math           import acos
from math           import asin
from math           import atan2
from scipy          import misc
from scipy.interpolate import griddata

import PIL
import pyfits
import copy
import pylab
import string
import os
import sys
import logging
import matplotlib.pyplot as plt

class MANGOImage:
    def __init__(self, rawImageAddress):
        self.rawImageAddress = rawImageAddress
        self.loadFITS()

    def loadFITS(self):
        hdulist          = pyfits.open(self.rawImageAddress)
        self.imageData   = flipud(array((hdulist[0].data)))
        self.width       = int(hdulist[0].header['NAXIS1'])
        self.height      = int(hdulist[0].header['NAXIS2'])
        self.siteName    = hdulist[0].header['SITE']
        self.calibrationFlag = 0
        self.writeMode   = 'L'

    def loadCalibrationData(self):
        calibrationFilename = os.path.dirname(os.getcwd()) + "\\Sites\\" + self.siteName + "\\
calibration\\" + 'Calibration.csv'
        calibrationData     = genfromtxt(calibrationFilename, delimiter=',')
        self.azimuth        = array((calibrationData[1:,1]))
        self.elevation      = array((calibrationData[1:,2]))
        self.i              = array((calibrationData[1:,3]))
        self.j              = array((calibrationData[1:,4]))
        self.zenith         = array([self.i[0], self.j[0]])

```

## APPENDIX D: Image Processing Core

```

def loadNewIJ(self):
    newIFilename = os.path.dirname(os.getcwd()) + "\\Sites\\" + self.siteName + "\\
calibration\\" + 'newI.csv'
    newJFilename = os.path.dirname(os.getcwd()) + "\\Sites\\" + self.siteName + "\\
calibration\\" + 'newJ.csv'
    self.newIMatrix = genfromtxt(newIFilename, delimiter=',')
    self.newJMatrix = genfromtxt(newJFilename, delimiter=',')

def loadBackgroundCorrection(self):
    backgroundCorrectionFilename = os.path.dirname(os.getcwd()) + "\\Sites\\" + self.siteName + "\\
calibration\\" + 'backgroundCorrection.csv'
    self.backgroundCorrection = genfromtxt(backgroundCorrectionFilename, delimiter=',')

def process(self):
    self.loadCalibrationData()
    self.loadNewIJ()
    self.loadBackgroundCorrection()
    self.removeStars()
    self.setLensFunction()
    self.calibrate()
    self.mercatorUnwarp()
    self.equalizeHistogram()
    self.writePNG()

def removeStars(self):
    filteredData = copy.copy(self.imageData).astype(float)
    for i in range(self.width):
        leftPixels = self.imageData[:,max(i-5, 0):max(i-2,1)]
        rightPixels = self.imageData[:,min(i+3, self.width-2):min(i+6, self.width-1)]
        allPixels = append(leftPixels, rightPixels, 1)
        pixelData = self.imageData[:, i]
        stdVal = std(allPixels, 1)
        meanVal = mean(allPixels, 1)
        for j in range(self.height):
            if pixelData[j]>meanVal[j]+3*stdVal[j] :
                filteredData[j-1:j+2, i-1:i+2] = -1
    (iKnown, jKnown) = where(filteredData>=0)
    valuesKnown = extract(filteredData>=0,filteredData)
    (iAll, jAll) = where(filteredData>=-1)
    self.imageData = reshape(griddata((iKnown, jKnown), valuesKnown, (iAll, jAll), method='linear',
fill_value = 0), filteredData.shape)

def calibrate(self):
    #Spatial Calibration based on star data
    self.zenith = array([self.i[0], self.j[0]])
    G_el = 1.0 - [self.getPixelsFromAngle(angle) for angle in self.elevation]/self.fisheyeRadius
    self.f = G_el*sin(radians(self.azimuth))
    self.g = G_el*cos(radians(self.azimuth))
    firstColumn = array([1]*len(self.i))
    oneIJ = vstack((firstColumn, self.i, self.j)).transpose()

    intermediate0 = dot(oneIJ.transpose(), oneIJ)
    intermediate1 = linalg.pinv(intermediate0)
    intermediate2 = dot(intermediate1, oneIJ.transpose())

    aCoefficients = dot(intermediate2, self.f)
    bCoefficients = dot(intermediate2, self.g)

    self.a0 = aCoefficients[0]
    self.a1 = aCoefficients[1]
    self.a2 = aCoefficients[2]
    self.b0 = bCoefficients[0]
    self.b1 = bCoefficients[1]
    self.b2 = bCoefficients[2]

```

## APPENDIX D: Image Processing Core

```

rotationAngle_1 = degrees(atan(-self.b1/self.a1))
rotationAngle_2 = degrees(atan(self.a2/self.b2))
self.rotationAngle = .5*(rotationAngle_1 + rotationAngle_2)

self.imageData = flipplr(misc.imrotate(self.imageData, self.rotationAngle, interp = 'bicubic'))
self.imageData = self.imageData.astype(float)

#Rotating Zenith Counter-clockwise by rotation angle and flipping it left-right
zenithI = self.width - int(cos(radians(self.rotationAngle))*(self.zenith[0] - self.width/2) - sin
(radians(self.rotationAngle))*(self.zenith[1]-self.height/2) + self.width/2)
zenithJ = int(sin(radians(self.rotationAngle))*(self.zenith[0] - self.width/2) + cos(radians(self.
rotationAngle))*(self.zenith[1]-self.height/2) + self.height/2)
self.zenith = [zenithI, zenithJ]
self.setLensFunction()

def setLensFunction(self):
#Calculates lens function coefficients for the equation: Angle = a0 + a1.px + a2.px^2 + a3.px^3
xDiff = [self.i[0] - i for i in self.i]
yDiff = [self.j[0] - j for j in self.j]
distanceFromZenith = [sqrt(xDiff[k]**2 + yDiff[k]**2) for k in range(len(self.i))]
angleFromZenith = [self.elevation[0] - self.elevation[k] for k in range(len(self.i))]

firstColumn = array([[1]*len(distanceFromZenith)])
distanceFromZenithMat = vstack((firstColumn, distanceFromZenith, [x**2 for x in distanceFromZenith]
, [x**3 for x in distanceFromZenith])).transpose()
self.pixToAngleCoefficients = dot(linalg.pinv(distanceFromZenithMat), angleFromZenith)[::-1]

firstColumn = array([[1]*len(distanceFromZenith)])
angleFromZenithMat = vstack((firstColumn, angleFromZenith, [x**2 for x in angleFromZenith], [x**3
for x in angleFromZenith])).transpose()
self.angleToPixCoefficients = dot(linalg.pinv(angleFromZenithMat), distanceFromZenith)[::-1]

self.fisheyeRadius = self.getPixelsFromAngle(90)

def getPixelsFromAngle(self, angle):
#Input Angle in degrees
return polyval(self.angleToPixCoefficients, angle)

def mercatorUnwarp(self):
newImageWidth = 500
newImageHeight = 500
finalImage = ones([newImageHeight, newImageWidth])*-1

for j in range(self.imageData.shape[0]):
for i in range(self.imageData.shape[1]):
newI = self.newIMatrix[j][i]
newJ = self.newJMatrix[j][i]
if (not(isnan(newI)) and not(isnan(newJ)) and not(isnan(self.backgroundCorrection[j, i]))):
self.imageData[j, i] = self.imageData[j, i]*self.backgroundCorrection[j, i]
finalImage[newJ, newI] = self.imageData[j, i]

(iKnown, jKnown) = where(finalImage>=0)
valuesKnown = extract(finalImage>=0,finalImage)
(iAll, jAll) = where(finalImage>=-1)
interpolatedData = reshape(griddata((iKnown, jKnown), valuesKnown, (iAll, jAll), method='cubic',
fill_value = -1), [newImageWidth, newImageHeight])
alphaMask = ones([newImageHeight, newImageWidth])*255
for j in range(interpolatedData.shape[0]):
for i in range(interpolatedData.shape[1]):
if interpolatedData[j, i] == -1:
alphaMask[j, i] = 0

interpolatedData = (interpolatedData*255/(nanmax(interpolatedData))).astype('uint8')
alphaMask = alphaMask.astype('uint8')
self.imageData = dstack([interpolatedData, alphaMask])
self.writeMode = 'LA'

```



## APPENDIX D: Image Processing Core

```
def equalizeHistogram(self):
    #Histogram Equalization to adjust contrast [1%-99%]
    numberBins = 1000 # A good balance between time and space complexity, and well as precision
    flattenedImageData = self.imageData.flatten()
    imageHistogram, bins = histogram(flattenedImageData, numberBins, normed = True)
    cdf = cumsum(imageHistogram)
    maxIndex = min(range(len(cdf)), key=lambda i: abs(cdf[i]-0.99*max(cdf)))
    minIndex = min(range(len(cdf)), key=lambda i: abs(cdf[i]-0.01*max(cdf)))
    vmax = float(bins[maxIndex])
    vmin = float(bins[minIndex])
    lowValueIndices = flattenedImageData < vmin
    flattenedImageData[lowValueIndices] = vmin
    highValueIndices = flattenedImageData > vmax
    flattenedImageData[highValueIndices] = vmax
    self.imageData = flattenedImageData.reshape(self.imageData.shape)

def writePNG(self):
    #Writing for web display
    writeAddress = os.path.dirname(os.getcwd()) + "\\Sites\\All Sites Images\\" + self.siteName + ".png"
    "
    finalImage = PIL.Image.fromarray(self.imageData, self.writeMode)
    finalImage.save(writeAddress)

    #Writing for archiving to source folder
    writeAddress = self.rawImageAddress[:-3] + "png"
    finalImage.save(writeAddress)
```

# Appendix E: Image Processing Guide

This program locates new FITS images as they arrive from active remote sites and processes them in parallel using multiprocessing techniques. The maximum number of images it processes in parallel is 10.

```
#####
#
# Image Processing Guide for MANGO
# (Midlatitude All-sky-imager Network for Geophysical Observation)
# This tool searches for new FIT images, and sends them for
# processing as they reach the server
#
# 2013-02-27 Rohit Mundra
#           Initial implementation
#
# 2013-03-01 Rohit Mundra/Fabrice Kengne
#           Implemented multiprocessing
#
#####
from glob      import glob
from time      import sleep
from datetime  import datetime
from numpy     import *
from MANGOImage import MANGOImage

import os
import multiprocessing

currentSitesFile = "addsite.csv"
sitesPath       = os.path.dirname(os.getcwd()) + "\\Sites\\"
imagesDir       = "\\images\\*.fit*"

maxProcesses     = 10 #Number of images to process simultaneously: Max Process Count
timeToUpdateSitesList = 50400 #seconds after 0000hrs local time

class imageProcessingGuide:
    def __init__(self):
        self.waitingQueue = []
        self.processingQueue = []
        self.lastProcessedFiles = []
        self.updateCurrentTime()
        self.updateSitesList()
        self.startGuide()

    def updateSitesList(self):
        getData = array((genfromtxt(sitesPath+currentSitesFile, dtype = str, delimiter = ',')))
        self.sitesList = getData[1:, 1]
        getData = array((genfromtxt(sitesPath+currentSitesFile, dtype = int, delimiter = ',')))
        self.sitesStatus = getData[1:, 5]
        self.lastProcessedFiles = {}
        for i in range(len(self.sitesStatus)):
            self.lastProcessedFiles[self.sitesList[i]] = 'NoFile'

    def updateCurrentTime(self):
        currentTime = datetime.now()
        currentHour = currentTime.hour
        currentMinute = currentTime.minute
        currentSecond = currentTime.second + currentTime.microsecond/1000000.0
        self.currentTimeTotal = currentHour*3600 + currentMinute*60 + currentSecond #seconds after 00:00:00

    def startGuide(self):
        sleepPeriod = 1 #seconds
        while(1):
            self.updateCurrentTime()

            if (self.currentTimeTotal >= timeToUpdateSitesList - sleepPeriod and self.currentTimeTotal
                < timeToUpdateSitesList):
                self.updateSitesList()
```

## APPENDIX E: Image Processing Guide

```
for i in range(len(self.sitesList)):
    if self.sitesStatus[i] == 0:
        fileLocation = self.findNewImage(self.sitesList[i])
        if not(fileLocation == -1):
            self.waitingQueue.insert(0, fileLocation)

for i in range(len(self.processingQueue)):
    if not(self.processingQueue[i].is_alive()):
        self.processingQueue[i] = []

self.processingQueue = [x for x in self.processingQueue if x != []]

# Fill up the processing queue OR empty the waiting queue, whichever comes first
while (len(self.processingQueue) <= maxProcesses) :
    if (len(self.waitingQueue) == 0) or (len(self.processingQueue) >= maxProcesses):
        break
    else:
        nextAddress = self.waitingQueue.pop()
        image = MANGOImage(nextAddress)
        processHandle = multiprocessing.Process(target = image.process)
        self.processingQueue.insert(0, processHandle)

for i in range(len(self.processingQueue)):
    try:
        if not(self.processingQueue[i].is_alive()):
            self.processingQueue[i].start()
    except:
        self.processingQueue[i] = []
self.processingQueue = [x for x in self.processingQueue if x != []]


sleep(sleepPeriod)
print "Active : ", len(self.processingQueue)
print "Waiting: ", len(self.waitingQueue)

def findNewImage(self, siteName):
    imagesFolder = sitesPath+siteName+imagesDir
    try:
        listOfFiles = glob(imagesFolder)
        latestFile = max(listOfFiles, key=os.path.getmtime)
        if not(self.lastProcessedFiles[siteName] == latestFile):
            self.lastProcessedFiles[siteName] = latestFile
            return latestFile
        else:
            return -1
    except:
        return -1

if __name__ == '__main__':
    guide = imageProcessingGuide()
    imageProcessingGuide.startGuide()
```

## Appendix F: Web Interface Code

Since a calibration, background correction and unwarping stay fixed for a site unless a site is recalibrated, these functions are just executed when a user submits calibration information on the web interface in the following manner:

```
figureAddress, angle, distance, individualErrors, residualSumSquares, distanceCoefficients, zenith =  
    lensfunction(zenith, azimuth, elevation, iCoord, jCoord, siteName)   
generator = pixelsToLatLon(zenith, distanceCoefficients, siteLatitude, siteLongitude, siteName)
```

The functions called are described in Appendix G.

# Appendix G: Web-based Image Processing

```
#####
#
#   Written by: Maria Rangel
#   Started:   02-06-2013
#   Finished:  02-28-2013
#   Description: Contains all the functions used for the calibration pages
#
#####
from flask import *
from numpy import *
from StringIO import StringIO
from math import atan, acos, asin, atan2
import matplotlib as plt
import matplotlib.pyplot as plt
from pylab import *

mod = Blueprint('users', __name__, url_prefix='')

# SYSTEM VARIABLES
SITES_DIR = 'app/static/Sites/'
CAL_DIR = '/calibration/'
SYS_DIR = '/system/'
SCHEDULE_DIR = '/schedule/'
IMG_DIR = '/images/'
ALL_IMG_DIR = '/static/Sites/All Sites Images/'

# Data verification for site calibration
def lensfunction(zenith, azimuth, elevation, iCoord, jCoord, siteName):
    imageWidth = 695 # Atik 314L+ CCD at 2x2 binning
    imageHeight = 519 # Atik 314L+ CCD at 2x2 binning

    lenOfData = len(elevation)

    # Creating empty arrays
    distance = []
    angle = []
    individualErrors = []
    residualSumSquares=[]

    # Calculates the distance and angle from zenith
    for k in range(0,lenOfData ):
        temp = sqrt(((zenith[1]- iCoord[k])**2)+((zenith[2]- jCoord[k])**2)) # distance formula
        distance.append(temp)
        temp2 = (zenith[0] - elevation[k]) # angle from zenith = subtract the angle from the star from
        the zenith angle
        angle.append(temp2)

    # Performs Fitting
    firstColumn = array([[1]*len(angle)])
    angleMat = vstack((firstColumn, angle, [x**2 for x in angle], [x**3 for x in angle])).transpose()
    angleCoefficients = dot(linalg.pinv(angleMat), distance)
    angleCoefficients = angleCoefficients[:, :-1]

    firstColumn = array([[1]*len(angle)])
    distanceMat = vstack((firstColumn, distance, [x**2 for x in distance], [x**3 for x in distance])).
    transpose()
    distanceCoefficients = dot(linalg.pinv(distanceMat), angle)
    distanceCoefficients = distanceCoefficients[:, :-1]
    distances = range(260)
    testangles = polyval(distanceCoefficients, distances)

    anglefit = range(91)
    pixfit = polyval(angleCoefficients, anglefit)
```

## APPENDIX G: Web-based Image Processing

```

empirical = polyval(angleCoefficients, angle)
#observed = distance
varianceDistance = (std(distance)**2)

residualSumSquares = 0
individualErrors = []
for i in range(len(distance)):
    errorValue = ((distance[i]-empirical[i])**2) #residual sum of squares
    individualErrors = append(individualErrors, errorValue)
    residualSumSquares += errorValue

plot(anglefit, pixfit, 'r-')
scatter(angle, distance)
xlim((0,91))
ylim(0)
grid(True)
ylabel('Distance')
xlabel('Angle')
title('Lens Function')
savefig(SITES_DIR + siteName + CAL_DIR + 'lensfunction.png')
figureAddress = ('/static/Sites/' + siteName + CAL_DIR + 'lensfunction.png')
clf()

##### MODIFYING ZENITH #####
G_el = 1.0 - [polyval(angleCoefficients, value) for value in elevation]/polyval(angleCoefficients, 90)
f = G_el*sin(radians(azimuth))
g = G_el*cos(radians(azimuth))
firstColumn = array([[1]*len(iCoord)])
oneIJ = vstack((firstColumn, iCoord, jCoord)).transpose()
intermediate0 = dot(oneIJ.transpose(), oneIJ)
intermediate1 = linalg.pinv(intermediate0)
intermediate2 = dot(intermediate1, oneIJ.transpose())

aCoefficients = dot(intermediate2, f)
bCoefficients = dot(intermediate2, g)

a0 = aCoefficients[0]
a1 = aCoefficients[1]
a2 = aCoefficients[2]
b0 = bCoefficients[0]
b1 = bCoefficients[1]
b2 = bCoefficients[2]

rotationAngle_1 = degrees(atan(-b1/a1))
rotationAngle_2 = degrees(atan(a2/b2))
rotationAngle = .5*(rotationAngle_1 + rotationAngle_2)
flash(zenith)
zenithI = imageWidth - int(cos(radians(rotationAngle))*(zenith[1] - imageWidth/2) - sin(radians
(rotationAngle))*(zenith[2]-imageHeight/2) + imageWidth/2)
zenithJ = int(sin(radians(rotationAngle))*(zenith[1] - imageWidth/2) + cos(radians(rotationAngle))*
(zenith[2]-imageHeight/2) + imageHeight/2)
zenith = [zenith[0], zenithI, zenithJ]
flash(zenith)
##### DONE MODIFYING ZENITH #####

return figureAddress, angle, distance, individualErrors, residualSumSquares, distanceCoefficients,
zenith

# This function needs the zenith information (zenith elevation, zenith i coordinate, zenith j coordinate),
pixels to Angle Coefficients
# which were calculated in the lens function, the zenith longitude (site longitude) , zenith latitude (site
latitude), and the site name.
# The function calculates the latitude and longitudes for each pixel and stores them in a CSV file. The

```

## APPENDIX G: Web-based Image Processing

```

function also calculates the southwest
# and northeast coordinates
def pixelsToLatLon(zenith, pixToAngleCoefficients, zenithLatitude, zenithLongitude, siteName):
    viewingAngle = 63.3
    imageWidth = 695 # Atik 314L+ CCD at 2x2 binning
    imageHeight = 519 # Atik 314L+ CCD at 2x2 binning
    newImageWidth = 500 # Web display images will be 500 pixels wide
    newImageHeight = 500 # Web display images will be 500 pixels tall
    zenithLatitude = float(zenithLatitude) # converting the zenith latitude to a float
    zenithLongitude = float(zenithLongitude) # converting the zenith longitude to a float
    zenithLatitude = radians(zenithLatitude) # converting the zenith latitude from degrees to radians
    zenithLongitude = radians(zenithLongitude) # converting the zenith longitude from degrees to radians

    latitudes = empty([[imageHeight, imageWidth]]) # creating an empty array with the same
    dimensions as the raw image
    longitudes = empty([[imageHeight, imageWidth]]) # creating an empty array with the same
    dimensions as the raw image

    latitudes[:] = NAN # Setting the latitude array to NANs
    longitudes[:] = NAN # Setting the longitude array to NANs

    for j in range(imageHeight):
        for i in range(imageWidth):
            yDistance = j - zenith[2]
            xDistance = i - zenith[1]
            distanceFromZenith = sqrt(yDistance**2 + xDistance**2)
            angleFromZenith = polyval(pixToAngleCoefficients, distanceFromZenith)
            if angleFromZenith <= viewingAngle:
                earthDistance = earthDistanceFromZenithAngle(angleFromZenith)
                bearing = pi/2 + atan2(yDistance, xDistance)
                [latitudes[j, i], longitudes[j, i]] = greatCircleArc(bearing, earthDistance, zenithLatitude
, zenithLongitude)

    latBounds = [nanmin(latitudes), nanmax(latitudes)]
    latRange = latBounds[1]-latBounds[0]
    lonBounds = [nanmin(longitudes), nanmax(longitudes)]
    lonRange = lonBounds[1]-lonBounds[0]
    scaleBounds = log(tan(radians(latBounds)*0.5 + pi/4))
    scaleRange = scaleBounds[1]-scaleBounds[0]

    newImatrix = empty([[imageHeight, imageWidth]]) # creating an empty array with the same
    dimensions as the raw image
    newJmatrix = empty([[imageHeight, imageWidth]]) # creating an empty array with the same
    dimensions as the raw image
    newImatrix[:] = NAN # Setting the newI array to NANs
    newJmatrix[:] = NAN # Setting the newJ array to NANs

    for j in range(imageHeight):
        for i in range(imageWidth):
            pixLat = latitudes[j][i]
            pixLon = longitudes[j][i]
            if (not(isnan(pixLon)) and not(isnan(pixLat))):
                newI = int((newImageWidth-1)*(pixLon - lonBounds[0])/lonRange)
                scaleJ = log(tan(radians(pixLat/2.0) + pi/4))
                newJ = (newImageHeight-1) - int((newImageHeight-1)*(scaleJ- scaleBounds[0])/scaleRange)
                newImatrix[j, i] = newI
                newJmatrix[j, i] = newJ

    backgroundCorrection = empty([[imageHeight, imageWidth]]) # creating an empty array with the same
    dimensions as the raw image
    backgroundCorrection[:] = NAN # Setting the backgroundCorrection
    array to NANs

```

## APPENDIX G: Web-based Image Processing

```

for j in range(imageHeight):
    for i in range(imageWidth):
        yDistance = j - zenith[2]
        xDistance = i - zenith[1]
        distanceFromZenith = sqrt(yDistance**2 + xDistance**2)
        angleFromZenith = polyval(pixToAngleCoefficients, distanceFromZenith)
        backgroundCorrection[j, i] = getBackgroundCorrection(angleFromZenith)

# Stores the latitude and longitude path to file location
newIPath = SITES_DIR + siteName + CAL_DIR + 'newI.csv'
newJPath = SITES_DIR + siteName + CAL_DIR + 'newJ.csv'
backgroundCorrectionPath = SITES_DIR + siteName + CAL_DIR + 'backgroundCorrection.csv'

imageLatLonPath = SITES_DIR + siteName + CAL_DIR + 'imageLatLong.csv'

# calculates the southwest's latitude and longitude and northeast's latitude and longitude for image
# overlay for the map
southWestLatLon = [nanmin(latitudes), nanmin(longitudes)]
northEastLatLon = [nanmax(latitudes), nanmax(longitudes)]
imageLatitudesLongitudes = array((vstack((southWestLatLon, northEastLatLon)))) # creates an array
# stacked for writing to a CSV

# Creates CSV files for Latitudes and Longitudes data for each pixel
savetxt(backgroundCorrectionPath, backgroundCorrection, delimiter=",") # writes the
backgroundCorrection CSV file
savetxt(newJPath, newJmatrix, delimiter=",") # writes the newJ CSV file
savetxt(newIPath, newImatrix, delimiter=",") # writes the newI CSV file
savetxt(imageLatLonPath, imageLatitudesLongitudes, delimiter=",") # writes the image
coordinates (southwest and northeast) CSV file

def getBackgroundCorrection(angleFromZenith):
    #Effects corrected: Van Rhijn, Lens Vignetting, M. KUBOTA et al, 2001
    earthRadius = 6371.0 #kilometers
    airglowHeight = 250.0 #kilometers
    vanRhijnFactor = sqrt(1.0 - ((earthRadius/(earthRadius + airglowHeight))**2 * sin(radians
    (angleFromZenith))**2))
    #Effects corrected: Atmospheric Extinction
    a = 0.2 #atmospheric extinction coefficient, M. KUBOTA et al, 2001
    F = 1/(cos(radians(angleFromZenith)) + 0.15*(93.885 - angleFromZenith)**(-1.253))
    extinctionFactor = 10.0**(0.4*a*F)
    return vanRhijnFactor*extinctionFactor

def greatCircleArc(bearing, earthDistance, zenithLatitude, zenithLongitude):
    earthRadius = 6371.0
    try:
        lat = asin(sin(zenithLatitude)*cos(earthDistance/earthRadius) +
            cos(zenithLatitude)*sin(earthDistance/earthRadius)*cos(bearing))
        lon = zenithLongitude + atan2(sin(bearing)*sin(earthDistance/earthRadius)*cos(zenithLatitude),
            cos(earthDistance/earthRadius)-sin(zenithLatitude)*sin(lat))
        lat = degrees(lat)
        lon = degrees(lon)
        return lat, lon
    except:
        flash('Failure in function greatCircleArc(...)!')

def earthDistanceFromZenithAngle(zenithAngle):
    try:
        earthRadius = 6371.0 #kilometers
        airglowHeight = 250.0 #kilometers
        a = 1
        b = -2*earthRadius*cos(radians(180 - zenithAngle))
        c = -(airglowHeight**2)-(2*earthRadius*airglowHeight)
        possibleDistances = roots([a, b, c])
        distance = max(possibleDistances)
        earthAngle = arcsin(distance*sin(radians(180 - zenithAngle))/(earthRadius+airglowHeight))
        earthDistance = earthRadius*earthAngle

        return earthDistance
    except:
        flash('Failure in function earthDistanceFromZenithAngle(...)!')

```



# Appendix H: MANGO User Guide

**SRI International**  
**MANGO SYSTEM CONTROL**  
**User Guide**



**Written by:**

Maria Alexandra Rangel

**Edited by:**

Fabrice Kengne

Rohit Mundra

**Table of Contents**

1 Introduction..... 5

    1.1 The Website..... 5

2 Monitoring Existing Sites on the Network..... 7

3 Site Settings..... 9

    3.1.1 Adding a Site to the Network..... 10

    3.1.2 Modifying a Site on the Network ..... 13

    3.1.3 Removing Site from the Network ..... 14

4 Site Calibration Settings..... 16

    4.1 Adding Calibration Data to Site..... 17

        4.1.1 The Information Form..... 18

        4.1.2 The Star Information Form..... 19

        4.1.3 Data Verification..... 26

    4.2 Modifying Calibration Data for a Site ..... 27

5 Schedule Settings ..... 31

    5.1 Adding a Schedule..... 32

    5.2 Modifying a Schedule Page..... 34

6 Account Settings ..... 37

    6.1 Changing Password..... 37

    6.2 Adding an Administrator..... 39

**Table of Figures**

Figure 1. MANGO Main page..... 5

Figure 2: “Administrator Login” Page..... 6

Figure 3: “MANGO System Management “ page..... 6

Figure 4: Navigation bar “Sites” available..... 7

Figure 5. Site Specific System Status Page..... 8

Figure 6: Specific site monitoring data set view button..... 8

Figure 7: Selecting “Sites” from the navigation bar ..... 9

Figure 8: All Sites Settings Page..... 9

Figure 9: Add a New Site Option..... 10

Figure 10: Add New Site Page..... 10

Figure 11: Add New Site Form ..... 11

Figure 12: Google Maps “Where’s Here?” option ..... 11

Figure 13: “What’s Here?” latitude and longitude display..... 12

Figure 14: Coordinate Verification..... 12

Figure 15: Addition of a new site to the network submission success message..... 13

Figure 16: "Modify a Site" option from the "Site Settings" page.....	13
Figure 17: "Modify Site" page.....	13
Figure 18: "Modify Site" form.....	14
Figure 19: Modify site success message.....	14
Figure 20: Remove site option from the "Site Settings" page.....	15
Figure 21: "Remove Site" page.....	15
Figure 22: Removing a site from the network success message.....	15
Figure 23: Selecting "Calibration" from the navigation bar.....	16
Figure 24: "Calibration Settings" page.....	17
Figure 25: Add New Calibration Data.....	17
Figure 26: "Add New Calibration" page.....	18
Figure 27: Information Form in the "Add New Calibration" page.....	18
Figure 28: Submission successful for Information form.....	19
Figure 29: TheSky6 program.....	20
Figure 30: Site Information in the "Add Calibration Data" page.....	20
Figure 31: TheSky6 Location Dialog box.....	21
Figure 32: TheSky6 Time Dialog box.....	21
Figure 33: TheSky6 Time Dialog box selecting the Time Options.....	21
Figure 34: TheSky6 Time Options Dialog box.....	22
Figure 35: TheSky6 Time Options Dialog box selecting UTC Time.....	22
Figure 36: TheSky6 Time Dialog box view after selecting UTC Time.....	22
Figure 37: Raw Image in Paint.....	23
Figure 38: Star Information Form.....	24
Figure 39: TheSky6 Object Information box.....	24
Figure 40: TheSky6 Object Information box with.....	25
Figure 41: Obtaining the <i>i</i> and <i>j</i> coordinates using Paint.....	25
Figure 42: "Check Data" button in the Star Information form.....	26
Figure 43: Data verification in the Star Information form.....	26
Figure 44: Error table for data verification in the Star Information form.....	27
Figure 45: Modify Calibration Data option from the "Calibration Settings" page.....	28
Figure 46: "Modify Calibration" page.....	28
Figure 47: Image Information form in the "Modify Calibration" page.....	29
Figure 48: Star Information form in the "Modify Calibration" page.....	30
Figure 49: Selecting "Schedule" from the navigation bar.....	31
Figure 50: "Schedule Settings" Page.....	32
Figure 51. Add Schedule Option.....	32
Figure 52. Add New Schedule Page.....	33
Figure 53: Current site information for "Add New Schedule" page.....	33
Figure 54. Add New Schedule Form.....	34
Figure 55: Modify schedule option in the "Schedule Settings" page.....	35
Figure 56: "Modify Schedule" page.....	35
Figure 57: Site information box (top) and modifications box (bottom).....	36
Figure 58: "Modify Schedule" page form.....	36
Figure 59: Success modifications done for schedule.....	36
Figure 60: Navigating to the "Account" page.....	37

## APPENDIX H: MANGO User Guide

Figure 61: "Account" page .....	37
Figure 62. Modify Account Settings Option .....	38
Figure 63. Administrator Change Password Page .....	38
Figure 64: Change password form .....	39
Figure 65: Add New Administrator Option.....	39
Figure 66. Add New Administrator Page.....	40

## 1 Introduction

This user guide manual serves as a guide to the MANGO Monitoring System that WPI has developed. This guide will walk the administrator through the navigation of the website the team has created. The website writes the necessary files that both our data acquisition software and image processing software need to run properly.

### 1.1 The Website

The website as mentioned before writes the necessary files to control the data acquisition and imaging software. The website functions as the front-end to control the back-end. The website allows the administrator to perform the following actions:

- System health monitoring for available sites
- Site settings – adding a site to the network, modifying a current site, and removing site from network
- Calibration settings- adding calibration data and modifying current calibration data
- Schedule Settings- adding a new schedule for a site and modifying current schedule

To access the MANGO System Monitoring website the administrator must first login to the system. The “MANGO” main page is a page that just has a title and a “Login” button as shown in Figure 1.



**Figure 1. MANGO Main page**

The simplicity of the page does not let anyone know what the purpose of the page is. This was done in case a public user or anyone comes across the page. To access the administrator page the administrator must click the “Login” button. Once the administrator clicks “Login” the page will redirect to the “Admin Login” page as shown in Figure 2.



Figure 2: “Administrator Login” Page

The login page asks the administrator to input their email and password. Once the administrator has input the information required, the form input is validated. Then the administrator is able to access the main “MANGO System Monitoring” page as shown in Figure 3.

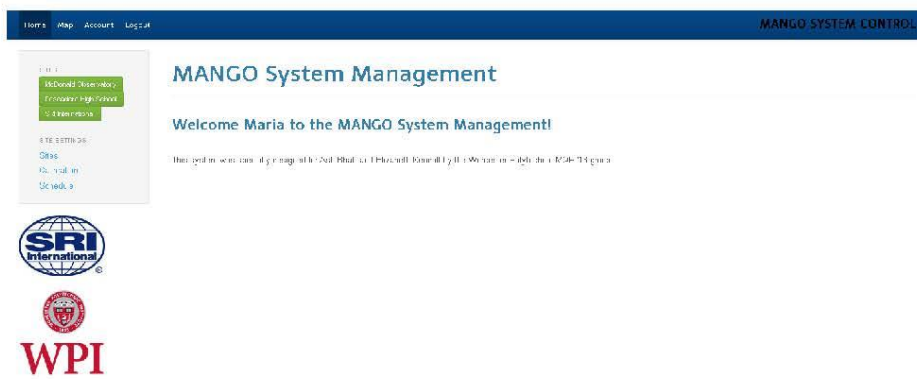


Figure 3: “MANGO System Management “ page

Once the administrator is logged in the administrator can begin monitoring and making modifications to the system.

PAGES 7-40 OF THE MANGO USER GUIDE HAVE BEEN  
UNDISCLOSED