

EBEX: A Balloon-Borne Telescope for Measuring  
Cosmic Microwave Background Polarization

Daniel Chapman

Submitted in partial fulfillment of the  
requirements for the degree of  
Doctor of Philosophy  
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2015

© 2015

Daniel Chapman

All rights reserved

## ABSTRACT

### **EBEX: A Balloon-Borne Telescope for Measuring Cosmic Microwave Background Polarization**

by

Daniel Chapman

EBEX is a long-duration balloon-borne (LDB) telescope designed to probe polarization signals in the cosmic microwave background (CMB). It is designed to measure or place an upper limit on the inflationary B-mode signal, a signal predicted by inflationary theories to be imprinted on the CMB by gravitational waves, to detect the effects of gravitational lensing on the polarization of the CMB, and to characterize polarized Galactic foreground emission.

The payload consists of a pointed gondola that houses the optics, polarimetry, detectors and detector readout systems, as well as the pointing sensors, control motors, telemetry systems, and data acquisition and flight control computers. Polarimetry is achieved with a rotating half-wave plate and wire grid polarizer. The detectors are sensitive to frequency bands centered on 150, 250, and 410 GHz. EBEX was flown in 2009 from New Mexico as a full system test, and then flown again in December 2012 / January 2013 over Antarctica in a long-duration flight to collect scientific data.

In the instrumentation part of this thesis we discuss the pointing sensors and attitude determination algorithms. We also describe the real-time map making software, “Quick-Look”, that was custom-designed for EBEX. We devote special attention to the design and construction of the primary pointing sensors, the star cameras, and their custom-designed flight software package, “STARS” (the *Star Tracking Attitude Reconstruction Software*).

In the analysis part of this thesis we describe the current status of the post-flight analysis procedure. We discuss the data structures used in analysis and the pipeline stages related

to attitude determination and map making. We also discuss a custom-designed software framework called “LEAP” (the *LDB EBEX Analysis Pipeline*) that supports most of the analysis pipeline stages.

# Contents

<b>List of Figures</b>	<b>v</b>
<b>List of Tables</b>	<b>xxiv</b>
<b>1 CMB Polarization Science</b>	<b>1</b>
1.1 $\Lambda$ CDM - The Standard Cosmological Model . . . . .	1
1.2 Cosmological Inflation . . . . .	1
1.3 The Cosmic Microwave Background . . . . .	2
1.3.1 Temperature Anisotropies . . . . .	3
1.3.2 Polarization Anisotropies . . . . .	6
<b>2 EBEX Overview</b>	<b>10</b>
2.1 Science Goals . . . . .	10
2.2 Observation Strategy . . . . .	10
2.3 Instrument . . . . .	12
2.3.1 Gondola and Attitude Control . . . . .	12
2.3.2 Optics and Receiver . . . . .	14
2.3.3 Polarimetry . . . . .	16
2.3.4 Detectors and Readout . . . . .	17
2.3.5 Telemetry . . . . .	18
2.3.6 Power . . . . .	19
2.3.7 Thermal . . . . .	19
<b>3 Attitude Control System</b>	<b>21</b>
3.1 Introduction . . . . .	21
3.2 Coordinate Systems . . . . .	22
3.2.1 Celestial Coordinate Systems . . . . .	22
3.2.2 Roll Angle . . . . .	25
3.2.3 Fair Measure Coordinates . . . . .	26
3.3 Pointing Requirements . . . . .	27
3.3.1 Real-Time . . . . .	27
3.3.2 Post-Flight . . . . .	28
3.4 Sensors . . . . .	28
3.4.1 Gyroscopes . . . . .	30

3.4.2	Star Cameras . . . . .	35
3.4.3	Coarse Sensors . . . . .	36
3.5	Control Algorithms . . . . .	39
3.5.1	Attitude Determination Loop . . . . .	39
3.5.2	Scan Control Loop . . . . .	41
3.5.3	Low Level Control Loops . . . . .	45
<b>4</b>	<b>Star Cameras</b>	<b>46</b>
4.1	Introduction . . . . .	46
4.2	Star Camera 1 . . . . .	52
4.2.1	Design . . . . .	52
4.2.2	Design Principles . . . . .	56
4.2.3	Construction . . . . .	58
4.2.4	Thermal Consideration . . . . .	58
4.3	Optical Baffles . . . . .	63
4.4	Pre-Flight Tests and Predictions . . . . .	68
4.4.1	Sensitivity . . . . .	68
4.4.2	Solution Uncertainty . . . . .	74
4.4.3	Vignetting . . . . .	77
4.4.4	Disk Space . . . . .	77
4.4.5	Pointing Offsets . . . . .	80
4.5	EBEX 2012 Performance . . . . .	82
<b>5</b>	<b>Star Camera Software, “STARS”</b>	<b>84</b>
5.1	STARS Design Requirements . . . . .	84
5.2	STARS Design Principles and Architecture . . . . .	85
5.2.1	Architecture . . . . .	86
5.2.2	Standard Operation . . . . .	87
5.2.3	Shared Memory . . . . .	87
5.2.4	Settings Files . . . . .	89
5.2.5	Testing . . . . .	89
5.3	STARS Components . . . . .	90
5.3.1	Solving . . . . .	90
5.3.2	Solving - Statistics . . . . .	91
5.3.3	Solving - Source Finding . . . . .	92
5.3.4	Solving - Pattern Matching . . . . .	96
5.3.5	Star Catalog . . . . .	98
5.3.6	Displaying . . . . .	104
5.3.7	Imaging . . . . .	106
5.3.8	Networking . . . . .	111
5.4	STARS - Successful In-Flight Performance . . . . .	112

<b>6</b>	<b>Real-Time Sky Maps with “QuickLook”</b>	<b>114</b>
6.1	Description . . . . .	114
6.2	Back-End Server with Naive Map Maker . . . . .	115
6.3	Quicklook User Interface with Google Maps . . . . .	116
6.3.1	Map Display . . . . .	117
6.3.2	User Options . . . . .	118
6.4	Testing . . . . .	120
6.5	In-Flight Performance . . . . .	121
<b>7</b>	<b>2012 Antarctic Science Flight</b>	<b>122</b>
7.1	Flight Details . . . . .	122
7.2	Data Extraction . . . . .	125
<b>8</b>	<b>Data Structures for Post-Flight Analysis</b>	<b>127</b>
8.1	Introduction . . . . .	127
8.2	Merging Data from Redundant Flight Computers . . . . .	130
8.2.1	Aligning Dirfiles . . . . .	130
8.2.2	Resolving Conflicts . . . . .	133
8.3	Base Data Structures . . . . .	135
8.4	Derived Data Structures . . . . .	140
<b>9</b>	<b>The “LEAP” Software Framework</b>	<b>142</b>
9.1	Terminology . . . . .	142
9.2	Overview . . . . .	143
9.3	Libraries . . . . .	145
9.3.1	Parent App . . . . .	145
9.3.2	IO Management . . . . .	147
9.3.3	Other Libraries . . . . .	150
9.4	Resources . . . . .	151
<b>10</b>	<b>Data Analysis</b>	<b>152</b>
10.1	Overview . . . . .	152
10.2	Selected Pipeline Stages . . . . .	154
10.2.1	Flight Base Creation . . . . .	154
10.2.2	Star Camera Solving . . . . .	154
10.2.3	Pointing Reconstruction . . . . .	159
10.2.4	Map Making . . . . .	166
10.3	Preliminary Results and Next Steps . . . . .	171
<b>11</b>	<b>Polar Mesospheric Clouds</b>	<b>176</b>
11.1	Data Set Characterization . . . . .	178
11.2	Feature Tracking and Characterization . . . . .	180
11.2.1	Projection Onto the Cloud Layer . . . . .	181
11.2.2	Preliminary Results . . . . .	185

<b>Bibliography</b>	<b>186</b>
<b>Appendix A Gyro Orthogonalization Results</b>	<b>197</b>
<b>Appendix B Star Camera Assembly Procedure</b>	<b>199</b>
<b>Appendix C Star Camera 1 Electrical Documentation</b>	<b>201</b>
<b>Appendix D Leap IO Management Loading Parameters</b>	<b>206</b>



# List of Figures

1.1	Model of the temperature, E-mode, and B-mode power spectra using the best-fit Planck $\Lambda$ CDM model and assuming a tensor-to-scalar ratio of 0.1. The B-mode curve is separated into the lensing contribution and the inflationary gravitational wave contribution. The blue shaded area represents possibilities for ratios $< 0.1$ . Figure adapted from [8]. . . . .	4
1.2	Measurements of the temperature, E-mode, and B-mode power spectra as well as best-fit models and a model assuming a tensor-to-scalar ratio ( $r$ ) of 0.1. The top figure shows the Planck CMB temperature (TT) power spectrum. Figure from [9]. The middle and bottom figures show E-mode (EE) and B-mode (BB) measured data points or upper limits. The B-mode model shown in the bottom plot also shows the lensing contribution and the inflationary gravitational wave contribution. For the gravitational wave contribution, a tensor-to-scalar ratio ( $r$ ) of 0.1 is used. Middle and bottom figures from [8] in which it was adapted from [10] and [11]. . . . .	5
1.3	Net polarization resulting from Thomson scattering of photons originating from a quadrupolar pattern. Cooler photons (red) originating from above can only scatter towards the observer with horizontal polarization. Likewise, warmer photons (blue) originating from the left can only scatter towards the observer with vertical polarization. Figure from [17]. . . . .	7

2.1	A model of the gondola and the telescope components that it houses. Some of the critical components from the attitude control, power, optics, and detector systems are labeled. . . . .	13
2.2	A model of the warm optics. Light from the sky reflects off the primary mirror onto the secondary mirror and then into the cryostat. Figure courtesy of Huan Tran. . . . .	14
2.3	A model of the receiver, which contains the cold optics. The window is the entry point for light into the cryostat. Filters help protect subsequent stages from thermal load. A half-wave plate and polarizing grid are used for polarimetry. Lenses re-image the primary mirror onto an aperture stop and the two focal planes. . . . .	15
2.4	(left) An example focal plane. Each focal plane has four 150 GHz wafers, two 250 GHz wafers, and one 410 GHz wafer. Detectors within $3^\circ$ of the center have a Strehl ratio greater than 0.9, as shown by the black circle. (center) An example wafer. (right) An example TES bolometric detector. . . . .	16
2.5	(a) Conceptual diagram of a bolometer. An absorptive element with temperature $T$ and heat capacity $C$ absorbs incoming power $P$ , which is then conducted to a thermal bath at temperature $T_{\text{bath}}$ through a weak thermal link with conductance $G$ . Diagram courtesy of Johannes Hubmayr. (b) Superconducting transition shows a steep change in electrical resistance as a function of temperature. . . . .	18

3.1	Overview of the attitude control system. Filled blue boxes represent software/firmware loops. Black bordered boxes represent physical components. The physical components can be grouped into three categories represented by red lines: sensors, control algorithms, and actuators. There are many physical connections between components, but the arrows here represent the relevant flow of data. Sensor data is used by the flight computers to compute a real-time attitude solution (see Section 3.5.1) and run a scan control loop (Section 3.5.2) to output a requested azimuth velocity and elevation position for the low level control loops. The gyros and elevation encoder are also used directly by the low level control loops, along with the requested azimuth velocity and elevation position, to output requested power for the motor controllers (see Section 3.5.3). . . . .	23
-----	---	----

- 3.2 The performance in cross-elevation (left column) and elevation (right column) of each absolute pointing sensor during the EBEX 2012 Antarctic flight. Each plot shows a histogram of the differences between the “true” pointing and the pointing measured in flight. The true pointing is taken to be the post-flight reconstructed pointing stream. The measured pointing is the individual pointing stream from each active axis of each absolute pointing sensor that was calculated during flight by the flight control program. However, the calibration angles to the microwave boresight used here may differ from those used in flight. The angles used here are those determined before flight on the ground, and used during flight up until the coarse sensors were calibrated to the star cameras. This gives a fair evaluation of how the sensors performed individually, without the help of the star cameras. Note that not all sensors had valid pointing streams throughout the flight, as indicated by the percent coverage shown in each plot. Most notably the sun sensors had very limited coverage because they did not cover the full azimuth range, and the ebex\_dgps had no coverage at float for unknown reasons. . . . . 31
- 3.3 Example data from a gyro orthogonalization test. The box is placed on the outer face that gy1 is orthogonal to, and then rotated at various velocities for one minute. The angular velocities measured by gy2 and gy1 are plotted against one another for every timestep. The slope of the resulting best-fit line is used as an element in the inverse orthogonalization matrix. The results from the final orthogonalization run before the 2012 flight are shown in Appendix A. 34

3.4	Plot created during a clinometer calibration run in 2009. The clinometer exhibits a significant ( $\sim 1.5^\circ$ ) systematic error. The blue data points represent the difference in elevation angles measured by the clinometer and the elevation encoder, as a function of elevation angle (as measured by the encoder). The deviation from a line with slope zero is due to a systematic error in the clinometer. This curve was fitted with a 5th order polynomial and corrected for in FCP. After this correction was implemented, the test was repeated, and the resulting data points in green show that the systematic error was removed to less than $0.1^\circ$ . . . . .	38
3.5	A flow chart of the general EBEX scan. The scan can be in one of three states: “snap” (the gondola remains stationary so that the star cameras can capture images), “throw” (the gondola slews in azimuth at a fixed elevation), and “step” (the gondola steps to the next elevation). The 3 different realizations of the scan (“cmb”, “calibrator”, and “horizontal”) perform slightly different actions in the “step” state. . . . .	42
3.6	Conceptual drawings of the coverage obtained by the horizontal scan (left), cmb scan (center), and calibrator scan (right). The horizontal scan is fundamentally horizontal based (e.g. in azimuth and elevation). The cmb scan is equatorial based, though the throws are still performed at a fixed elevation, resulting in a tilt. The calibrator scan is horizontal based except that it follows a fixed equatorial location. . . . .	43

3.7	Simulated coverage plots for the cmb scan. Subfigure (a) shows the resulting coverage in equatorial coordinates from 1 day of flight, with red representing more coverage. The cross-linking that results from performing multiple scans throughout the day can be seen, as well as the resulting rectangular shape. Subfigure (b) shows the resulting coverage from 11 days of flight for all 150 GHz detectors, with blue representing more coverage. Over the course of an entire flight the coverage becomes more even. . . . .	44
4.1	Subfigure (a) shows a 3-D plot of a small region of an image centered on a star captured by Star Camera 0. The image was captured on the ground, with a near optimal focus position. As a star is effectively a point source, this is a measure of the point spread function (PSF). We measure the width of a star by finding the $\sigma$ of a best-fit Gaussian. Subfigure (b) shows the width of this star as a function of the camera's focus position. A $\sigma$ of $\sim 0.75$ px is found at the optimal focus position, and corresponds to an angular resolution of $\sim 20''$ by the Rayleigh criterion. . . . .	48
4.2	The quantum efficiency of the CCD and the transmission function of the red filter used in the star cameras. Figure courtesy of Yury Vinokurov. . . . .	50
4.3	Star Camera 1 components and internal structure. . . . .	53
4.4	Side by side comparison of the actual and modeled XSC1 internal components, structure, and back flange. . . . .	54
4.5	The front of XSC1 without the front flange. The front weld flange jets inward about an inch to form four perpendicular edges, allowing the four rods of the internal structure to be fastened directly to the front weld flange. . . . .	54

4.6	(a) Top, (b) side, and (c) bottom view photographs of the Star Camera 1 internals. In the bottom view, the wires converging near the bottom are connected to four terminal blocks that are mounted on an aluminum shelf that is mounted to the underside of the camera controller. A diagram of the terminal blocks labeling the wires is shown in Appendix C. . . . .	57
4.7	Section view of Solidworks model of Star Camera 1 with optical baffle. . . .	59
4.8	Holes in the internal support structure's rings enable heat convection from the back of the star camera to the front of the star camera. . . . .	61
4.9	The star camera temperatures during the 2012 Antarctic flight. Each star camera has four temperature sensors placed in different locations inside the pressure vessel. The sensors are mounted near the computer (comp), lens (lens), DC-to-DC converters (dcdc), front flange near the window (flange), XSC1 electronics plate (plate), and/or the front of the vessel near the window (vessel). Each star camera has four of these possibilities. Shown are the temperatures for the entire flight (top) and a zoom of the temperatures during ascent (bottom). . . . .	62
4.10	(a) Cross section of a 3-D model of the star camera baffle and (b) optical drawing with specifications. In the drawing the black lines represent the baffle itself, which is mounted to the front of the star camera vessel, flush with the vessel window. The star camera lens sits a couple inches behind the window. The position of the lens differs in the two subfigures because subfigure (a) shows xsc1, while the baffle is designed for the more restrictive lens position which is the lens position in xsc0. Linear distances are in inches. In the 3-D model the vanes have finite thickness and are beveled at the inner edge. . . .	64
4.11	Diagram showing how the positions and heights of the vanes are defined to prevent light from reflecting off the tube directly into the vessel window. The procedure is described in detail in Section 4.3. . . . .	65

4.12	A 3-D model (a) of the star camera baffle compared with an actual constructed baffle before painting (b). . . . .	67
4.13	Subfigure (a) shows how many stars were identified in 240 ms exposures in an environment with 700 kepsa of sky brightness, as a function of apparent magnitude. It shows the total number of stars identified along with the total number of stars in the catalog (top), and then shows the fraction of catalog stars identified (bottom). Subfigure (b) uses the results from (a) to estimate how many stars will be identified in an arbitrarily selected star camera field from the Antarctic science patch. It shows a cumulative histogram showing the fraction of simulated fields of view (FOVs) that had at least N identified stars, as a function of N. . . . .	73
4.14	Results of a simulation testing the error reported by the star camera software. Each plots shows a comparison between the true errors (blue histogram) and the reported errors from the software (green histogram). The green histogram should be centered on the standard deviation of the blue histogram. There are three plots, one for each attitude coordinate. . . . .	75
4.15	Subfigure (a) shows a 3d plot representing all the pixels in an image and identifies those that are used to calculate the vignetting metric. The vignetting metric is the mean value of some representative edge pixels divided by the mean value of some representative center pixels. The raised pixels that in the subfigure that are far from the center are the representative edge pixels, and the raised pixels at the center of the image are the representative center pixels. Subfigure (b) shows the result of this metric for multiple images plotted as a function of aperture setting. Images taken with aperture settings 0 through 4 show vignetting. The test was done with and without the optical baffle installed, which has no noticeable effect. . . . .	76



4.16	Histograms of the star camera solution uncertainties, as estimated by STARS during post-flight analysis of the images. The histograms contain 15050 Star Camera 0 solutions and 17175 Star Camera 1 solutions, which amount to >90% of the solvable images from flight. The uncertainties are 1.5" in cross-declination, 1.5" in declination, and 48" in roll. These uncertainties are smaller than the predicted uncertainties due to the fact that there are roughly 8 stars per image instead of 4. This was possible because the brightness at float was not as high as the conservative case for which the camera performance was modeled. . . . .	83
5.1	Block diagrams of STARS running on a star camera computer. The shaded boxes show the threads that belong to the STARS process. Arrows with dotted lines indicate the flow of data. Data sent between threads is done safely and efficiently with circular buffers. The top panel shows the standard operation: image data and/or pointing solutions make their way from the camera controller to the flight computers. The middle panel shows the path that lens requests and results follow from the flight computers to the lens controller and back. The bottom panel shows the paths of various objects for display purposes. . . . .	88

5.2	The algorithm for determining the regional level around a pixel. This image represents a $44 \times 44$ px region of an image, where each pixel is shown as a small square with a gray outline. The STARS source finder compares a filtered version of the image to a leveled version of the image, where the leveled version is calculated by assigning the mean value of the 448 pixels shaded in light gray to the 16 pixels in the center. The light gray pixels are chosen because they are close enough to the dark gray pixels to be representative of the local level, but are far enough away that a potential source centered on a dark gray pixel itself does not bias the level. As an optimization, the operation is performed on a coarse version of the image. The coarse version of the image is a factor of $4 \times 4$ smaller. In this example the coarse pixels are represented by black outlines. . . . .	93
5.3	Demonstration of the selective masking utility. This html/javascript utility allows users to toggle blocks on or off using the mouse and provides them with a command to send to STARS during flight. STARS then ignores the disabled blocks during source finding. In this example the utility is configured to block out the left $\frac{2}{12}$ of the image, and the STARS screenshot shows that this area is shaded out of the image, indicating that the source finder will not search the left $\frac{2}{12}$ of the image. . . . .	95

5.4	The process of determining the flux (and by extension magnitude) of a given star. Subfigure (a) shows the responsivity of the CCD (in blue) and the transfer function of the EBEX star camera (in red). The star camera's transfer function is the CCD responsivity function multiplied by the red filter's transfer function. Subfigure (b) shows the flux model of an example star in green, the EBEX transfer function in red, and the multiplication of the two in cyan. The integral of the cyan curve is the resulting flux of the star in the EBEX star camera band. The flux model comes from fitting a blackbody to the flux data points (shown as blue circles) at available band frequencies. Subfigure (c) shows the measured flux vs catalog flux for the stars in a single image before this procedure is applied (using the V-band), while subfigure (d) shows the same thing for the computed flux of the same stars in the EBEX band. The linear relationship in Subfigure (d) suggests an improvement for the stars in this image. . . . .	100
5.5	The catalog stars before (a) and after (b) reduction. The sky is shown as an equatorial Mollweide projection, and the blue dots are individual stars. It is apparent in Subfigure (a) that the Galaxy presents an inhomogeneity in the density of stars. Subfigure (b) plots an example list of stars that has been reduced down to only include stars with global rank 1. Note the homogeneity in this equal-area projection. . . . .	101
5.6	A zoom of part of the sky showing four adjacent circular catalog regions. Overlaid on the plot is a black rectangle representing an example EBEX star camera field of view. Given the size of the circular regions, and their spacing, any possible placement and rotation of the field of view will always be completely contained within the nearest region. . . . .	103

5.7	All the regions in the STARS catalog. Each region is a circle of radius $10^\circ$ , and the regions are spaced on a grid separated by $10^\circ$ in angular distance in both right ascension and declination. Given these parameters, the regions overlap. There is additional overlap at right ascension zero (an imaginary vertical center line in the plot) due to the fact that the distance around the sphere is not an integer multiple of the grid spacing at every declination. . .	103
5.8	A screenshot of the STARS display, running post-flight on a ground computer on an image from the EBEX Antarctic flight. . . . .	104
5.9	The probability that an image is solvable given the level of sky brightness. 12 images were captured on a dark night with 120 ms exposure times. Different levels of sky brightness were added to each image, using the brightness simulator discussed in Section 5.2.5, to test whether it was still solvable at that level. Shown in this plot is the fraction of the 12 images that were able to solve at that level of sky brightness (blue circles). The test was then repeated with effective 240 ms exposures, which were actually sets of two co-added 120 ms exposures (green circles). The longer effective exposure time produces better results because of increased signal-to-noise. A 240 ms exposure would normally saturate at 417 kepsa, but with two separate co-added exposures the saturation limit doubles to 833 kepsa, as indicated by the dotted blue vertical line. . . . .	107

5.10	Capturing an image with multiple (triple) exposures while the gondola is in motion. The top plot shows a strip chart of the Star Camera 0 trigger line, which is the electronic line that drives the mechanical shutter in the camera. When the line is 1 the shutter is open, and when it is 0 the shutter is closed. Below the plot is the image captured. Each star shows the characteristic set of three streaks with gaps in between that appear for each star due to the shutter opening and closing three times. This may occur on smaller scales during the flight, which is why the Motion PSF source finding method is equipped to handle it. Note that the source finder was in motion PSF mode for this run, and as a result only the top end of the 3 streaks is identified for each source, even though the combined signal from all three streaks is used so that dimmer sources can be identified than with a single exposure. . . . .	108
6.1	Example screenshot of the quicklook user interface. The map display is featured prominently in the screenshot. The screenshot was captured during ground tests, when the gondola was performing a science scan and the detector data was simulated to create a non-uniform map. . . . .	117
6.2	The binning options as displayed on the Quicklook user interface. . . . .	119
6.3	The displaying options as displayed on the Quicklook user interface. . . . .	120
7.1	Altitude profile for the first 11 days of the EBEX 2012 Antarctic flight. . . . .	123
7.2	Geographic profile of EBEX in its 2012 Antarctic flight. The Antarctic continent is shown in white overlaid with a geographic grid with the lines of longitude labeled. The EBEX flight path for the first 11 days, during which time scientific data was collected, is shown in solid red. The flight path for the remaining 14 days is shown in dotted red. . . . .	123

7.3	Sky coverage from the 2012 Antarctic flight, shown in galactic coordinates. The patch is circular, centered on the equatorial South pole, covers 5735 square degrees, and has a width of roughly 27° due to the telescope’s latitude being 10° from the South pole and the focal plane being 7° wide. The calibrator is contained within this patch. . . . .	124
8.1	Diagram representing the union of ACS data from both flight computers. In this diagram time increases to the right, and each filled box represents a dirfile that contains multiple timestreams that are continuous for the length of the box. The top two lines of data are drawn to demonstrate a situation in which the flight computers stored some overlapping data and some unique data. The bottom line represents the desired result, which is the union of the top two lines. . . . .	130
8.2	Example data demonstrating the merging of FC1 and FC2 data. The figure shows 12 hours of (ACS) altitude data. Gray lines represent data that is stored redundantly by both flight computers, blue lines represent data points that only existed on FC1, and red lines represent data that only existed on FC2. Green circles represent the start of continuous sections of data, and are at a y-value of 0 because the system has not yet initialized after powering up. The union of FC1 and FC2 data results in longer continuous timestreams than from a single flight computer. . . . .	132
8.3	The ACS timing channel, in Unix time, for the entire flight. The two subplots show the same data at different y-ranges, calling out data that is valid (top) and data that is invalid (bottom). The colors follow the same scheme as in Figure 8.2. . . . .	134

8.4	Example of conflict resolution in an ACS data stream called “raw_gy1” using the “By Edge” method. In the top subplot gray data points are identical on both flight computers, blue data points are unique to FC1, and red data points are unique to FC2. Conflicts exist when the two flight computers have two different data points at the same index. The bottom subplot shows the resulting data stream after the conflict has been resolved in gray. The conflict occurs at the end of an FC2 dirfile, as evidenced by the transition in the top subplot from both red and blue data points to only blue data points. In this case FC1 is preferred, and the resulting data stream in the bottom subplot matches the blue data points. . . . .	136
8.5	Example of conflict resolution in an ACS data stream called “raw_gy1” using the “By Expectation” method. The plot layout is identical to that of Figure 8.4. In this case the conflicts are resolved by expectation, and it is evident that the resolved time stream in the bottom subplot does not contain spikes that would be due to the red outliers in the top subplot. . . . .	137
8.6	Diagram representing the union of ACS data from both flight computers, as in Figure 8.1, but with the one of the segments and its subsegments labeled.	138
8.7	Examples of “derived” datasets (“pointing”, “acs etime”, and “hwp template removed bolo”) and the base dirfiles with which they are aligned (“acs base” and “bolo base”), in diagram form similar to that of Figure 8.6. Note that each derived dataset has subsegments that are aligned with, synchronous with, and of the same frame length as their respective base datasets. . . . .	140
8.8	Table of the four base datasets and some of their potential derived datasets.	141

10.1	Data flow diagram of the analysis pipeline from raw flight data to maps. The rest of the pipeline (after maps) is not shown. Green ovals represent individual programs (apps), and blue boxes represent data (input and output for the apps). Every app shown here is part of the LEAP framework. The pointing and signal calibration results are iteratively improved using the temperature maps and therefore exist in a feedback loop with the map maker. . . . .	153
10.2	The fcp_counters (top) and stars_counters (bottom) for both star cameras (left and right) for the entire flight. For each star camera, both FCP and STARS keep a copy of both counters. The blue data points show FCP's copy of the counter from the numerical timestreams, and the red data points show STARS's copy of the counter from the image headers. When there are only blue points, the flight computers were running and sending triggers, but the star cameras were off. Most notably around December 31 the gondola operators were debugging the pivot motor controller issue instead of collecting scientific data. Each time the flight computers were powered on counter_fcp reset to zero. Each time the star cameras were powered on counter_stars reset to zero. . . . .	156



10.3	These plots show, for each star camera, a data point for each image that has been aligned to the dirfiles. The data points represent the difference between the timestamp associated with the dirfile at that index (which comes from the flight computer system clock) and the timestamp associated with the image (which comes from the star camera system clock), plotted against time. A non-zero slope represents a drift between the two machines' system clocks. A relative drift of this magnitude is not a problem for analysis because we primarily use the "tigger line" timestream and a series of counters to align the star camera images to the ACS timestreams. The fact that none of the differences exceed 40 s indicates that the trigger alignment app did not make any alignments that mistakenly cross resets. . . . .	157
10.4	The declination results from running the unscented Kalman filter forwards and backwards on simulated data, as explained in detail in Section 10.2.3 and in [60]. . . . .	162
10.5	The result from estimating the Gyro 2 bias in a simulation [60]. The black curve shows the true simulated bias of Gyro 2. The red and orange curves show the estimated bias from the forward and backward runs of the unscented Kalman filter. . . . .	163
10.6	Binning the average filter error as a function of time since the last star camera solution. . . . .	165
10.7	A preliminary temperature map of the EBEX calibration source, RCW 38, created from EBEX 2012 Antarctic flight data using the LEAP map maker. The map includes 2 hours of data from 91 150 GHz detectors. . . . .	172

10.8 Preliminary maps in $I$ , $Q$ , and $U$ of part of the galactic plane, created from EBEX 2012 Antarctic flight data using the LEAP map maker. The map includes data from 91 250 GHz detectors from a 17 minute section of flight. The $I$ map is a map of total intensity. The $Q$ and $U$ maps represent different alignments of polarization on the sky. . . . .	173
11.1 Example images containing polar mesospheric clouds. . . . .	177
11.2 Coverage plot of the star camera images over Antarctica. Each rectangle represents the location of an image, and the color of the rectangle represents the MAD value. Purple and dark blue represent little to no cloud activity, light blue represents moderate cloud activity, and green, yellow, and red represent significant cloud activity. The left plot shows all the images, and the right plot shows a zoomed region that has significant cloud activity. . . . .	179
11.3 Visual explanation of the vectors involved in projecting the point of interest (POI) onto the cloud layer (shown in white). The EBEX star camera, shown in red, is suspended $\sim 35$ km above the Earth. It observes polar mesospheric clouds that are at an altitude of $\sim 82$ km. Figure courtesy of Michael D'Anvers.	180
11.4 Example plot showing the projection of dozens of star camera images onto the cloud layer. Cloud features that span multiple images can be seen. . . .	183
11.5 Example of feature tracking and evolution observation. Four plots are shown, each plot containing four images taken in the same one minute window. The four plots span 6 minutes, and a unique feature, identified by the green arrow, can be seen to move downward over time. . . . .	184

A.1	The results of the gyro orthogonalization procedure discussed in Section 3.4.1 for gyro boxes A and B, shown respectively in Subfigures (a) and (b). The slopes of the best fit lines provide the six off-diagonal elements of the inverse orthogonalization matrix. In the plots, the center bulges are a result of accidental rotations when changing directions. We therefore grouped the data into center points (green) and edge points (blue), and only used the edge points to find the slope. . . . .	198
C.1	Diagram documenting Star Camera 1 terminal blocks wiring. The top left terminal block labeling is deprecated, while the terminal block labeling shown in the “Heater Wiring Diagram”, also in the appendix, is up to date. . . . .	202
C.2	XSC1 connector pinout. . . . .	203
C.3	Block wiring diagram of XSC1. . . . .	204
C.4	XSC1 Heater Wiring Diagram. (Left) Block diagram of the components in the heating system. There are six 10 W heaters: two on the vessel window, and four on the lens. They are activated by either software control (positive line A) or bang-bang control (positive line C), implemented in parallel so as to act as a logical OR. (Center) Wiring configuration on the heater terminal block. (Right) Pinout of the two power lines in the 12-pin cable. . . . .	205

# List of Tables

3.1	Summary of the three coordinate systems used in this document. A more in depth discussion of celestial coordinate systems can be found in [42]. . . . .	22
3.2	Table of absolute sensors. The different types of sensors are listed, as well as their short names (used elsewhere in this document) and how many of each type are flown (Num). In most cases EBEX flies two of each type of sensor for redundancy. This table includes which axes the sensors are capable ( <b>C</b> ) or not capable ( <b>N</b> ) of measuring, and - if an axis is capable of being measured - whether or not a real-time pointing stream is implemented ( <b>I</b> ) for that axis in the flight computers. Not all capable axes are implemented due to limited benefits and limited manpower. The table also lists each sensor's measurement precision, predicted sensor accuracy, and data acquisition rate. . . . .	30
4.1	Three methods of predicting the sky brightness for the EBEX Antarctic flight. The second method predicts two numbers as a consequence of not knowing the value of a parameter that could have been in two states during the BLASTPol 2010 flight. . . . .	71

4.2	Comparison of lossless compression methods considered for the star camera images. The compression algorithms were tested on noisy images from the 2009 test flight. The compression ratio is the uncompressed size divided by the compressed size. Also listed is whether the compressed file is directly compatible with the <i>GNU Image Manipulation Program</i> (GIMP) or the <i>cfitsio</i> C library, which is also used by the <i>pyfits</i> library for python. Compatibility with each of these is highly convenient and increases productivity during star camera development and testing. Not shown is the “ezip” compression (“EBEX zip” - developed specifically for EBEX), which was comparable in compression ratio, but unfortunately was 2.5 to 7.7 times slower than the standard options.	78
4.3	Disk usage calculations.	79
4.4	The angular offsets between each star camera and the EBEX telescope bore-sight, as determined before flight and after flight. The offsets are added to the star camera pointing to obtain the microwave telescope pointing.	82
5.1	List of key functionalities provided by STARS.	86
8.1	List of framefile streams. The timestreams within a framefile stream are synchronous with each other, but each framefile stream is asynchronous with the framefile streams. There are 59 total, as listed here. Some contain timestreams stored at different samples per frame (SPF) rates, so the existing SPF rates are listed. The Bolo and HWP frame rates are more precisely defined as $25.0 \times 10^6 \text{ Hz}/2^{18}$ .	129

# Acknowledgements

First and foremost I would like to thank my advisor, Amber Miller, for her guidance and support on this work. She has been everything one could hope for in an advisor, and I consider myself fortunate to have been her student.

I would also like to acknowledge two others who have played a mentoring role for me in the lab: Michele Limon and Britt Reichborn-Kjennerud. They have both been so generous with their time and I've learned so much from them. I am thankful to have had the opportunity to work on a hardware project with Michele. I have also been fortunate to work with and learn from many other members of the Miller lab, including Joy Didier, Will Grainger, and Seth Hillbrand.

I also thank all the members of the EBEX collaboration, many of whom I worked closely with over the years and who have made my time on this experiment rewarding. I would especially like to thank our PI, Shaul Hanany.

I thank my mom for always being so incredibly kind, helpful, and supportive. I thank my sister, Dara, for also being supportive and generous. My parents always tried to push me to do my best, and I appreciate that. Thanks Dad, I like to think you would be so happy with this accomplishment.

Finally I would like to thank my girlfriend, Joy Didier. I would be a very different person without her. To make the telescope flight-ready we suffered through endless cold nights in North America and endless cold days at the bottom of the world, and her emotional support is what kept me sane.

# Chapter 1

## CMB Polarization Science

### 1.1 $\Lambda$ CDM - The Standard Cosmological Model

The Big Bang theory describes a Universe expanding from an initial hot, compact phase to the state in which we observe it today. The “standard” model of Big Bang cosmology is a parameterization known as the  $\Lambda$ CDM model. In this model the Universe is spatially flat, expanding, and is predominantly comprised of dark energy in the form of a cosmological constant ( $\Lambda$ ) and cold dark matter (CDM). The current parameters hold that the Universe is 13.8 billion years old and that the make-up of the Universe is roughly 68% dark energy, 27% dark matter, 5% ordinary matter, and a small fraction of radiation and neutrinos. This model has been successful in describing a wide range of cosmological data [1] from a host of experiments, however it does not include a well-defined description of the first fraction of a second of the Universe.

### 1.2 Cosmological Inflation

Inflation is an extension to the standard cosmological model, and describes a Universe that underwent rapid exponential expansion in the first fraction of a second after the Big Bang [2].

Such an expansion would stretch quantum fluctuations to classical scales, seeding the inhomogeneities that evolved into the large scale cosmological structures that exist today. Inflation solves three problems in the current cosmological model [3][4]:

- **Horizon Problem** - The observable Universe is found to be statistically homogeneous and isotropic on large scales despite the fact that in the standard Big Bang scenario, currently observable parts of the Universe would not have been in causal contact, and thus able to equilibrate, given the age of the Universe. An exponential expansion solves this problem by bringing the observable Universe together in the first fraction of a second after the Big Bang.
- **Flatness Problem** - The Universe is measured to be spatially flat, or have a spatial curvature of zero, to less than 1%. This implies that in the past the spatial curvature must have been many orders of magnitude closer to zero, which presents a fine-tuning problem. Inflationary expansion would serve to dilute the curvature of space, removing the sensitive dependence of the flatness of the Universe on its initial conditions.
- **Missing Relics** - Grand unification theories predict that at high temperatures in the very early Universe symmetry breaking would produce a number of relic particles that have not been observed to date, including magnetic monopoles. If these particles did exist, inflationary expansion could serve to dilute them below the densities at which experiments have placed limits.

### **1.3 The Cosmic Microwave Background**

The cosmic microwave background (CMB) is radiation that was emitted 380,000 years after the Big Bang. Prior to this event, the Universe contained an opaque plasma in which photons continually ionized any neutral hydrogen that formed into constituent protons and electrons. In this plasma, photons could not travel far without Thomson scattering off of free electrons. As the Universe cooled the number of photons at or above the ionization energy of Hydrogen



decreased and at one point electrons became able to bind to protons [5]. Shortly thereafter the mean free path of photons exceeded that of the horizon scale at the time, resulting in a transparent Universe and a number of last-scattered photons that we can observe today as the cosmic microwave background. The expansion of the Universe has redshifted the CMB photons to a temperature of 2.73 K today.

The CMB was first discovered by Penzias and Wilson in 1965 [6], and has since been measured ever more precisely. The CMB is found to be a blackbody that is isotropic to a level of  $10^{-5}$  when accounting for the motion of the observer relative to the rest frame of the CMB [7].

In the following subsections we will discuss the anisotropies that exist in the CMB in both temperature and polarization. Figures 1.1 and 1.2 respectively show models and measurements of these anisotropies, and in the following sections we will refer back to these figures and explain them.

### 1.3.1 Temperature Anisotropies

The temperature of the CMB is anisotropic to roughly 1 part in  $10^5$  as a consequence of the inhomogeneity of matter when the CMB was emitted. As mentioned in Section 1.2, according to inflationary theories these inhomogeneities were the result of quantum fluctuations being driven to macroscopic scales by inflation. Prior to the decoupling of photons and charged particles, perturbations in the plasma density evolved over time. Gravity acted to compress dense regions, while photon pressure provided a restoring force, resulting in acoustic oscillations of the photon-baryon fluid until decoupling. At the time of decoupling photons leaving hot, dense regions of space had higher energy, though the energy lost in escaping a deeper gravitational well overcompensated for this and resulted in cooler photons leaving overdense regions, while warmer photons left underdense regions. The pattern of density perturbations in the plasma left signatures in the CMB that reflect the dynamics of

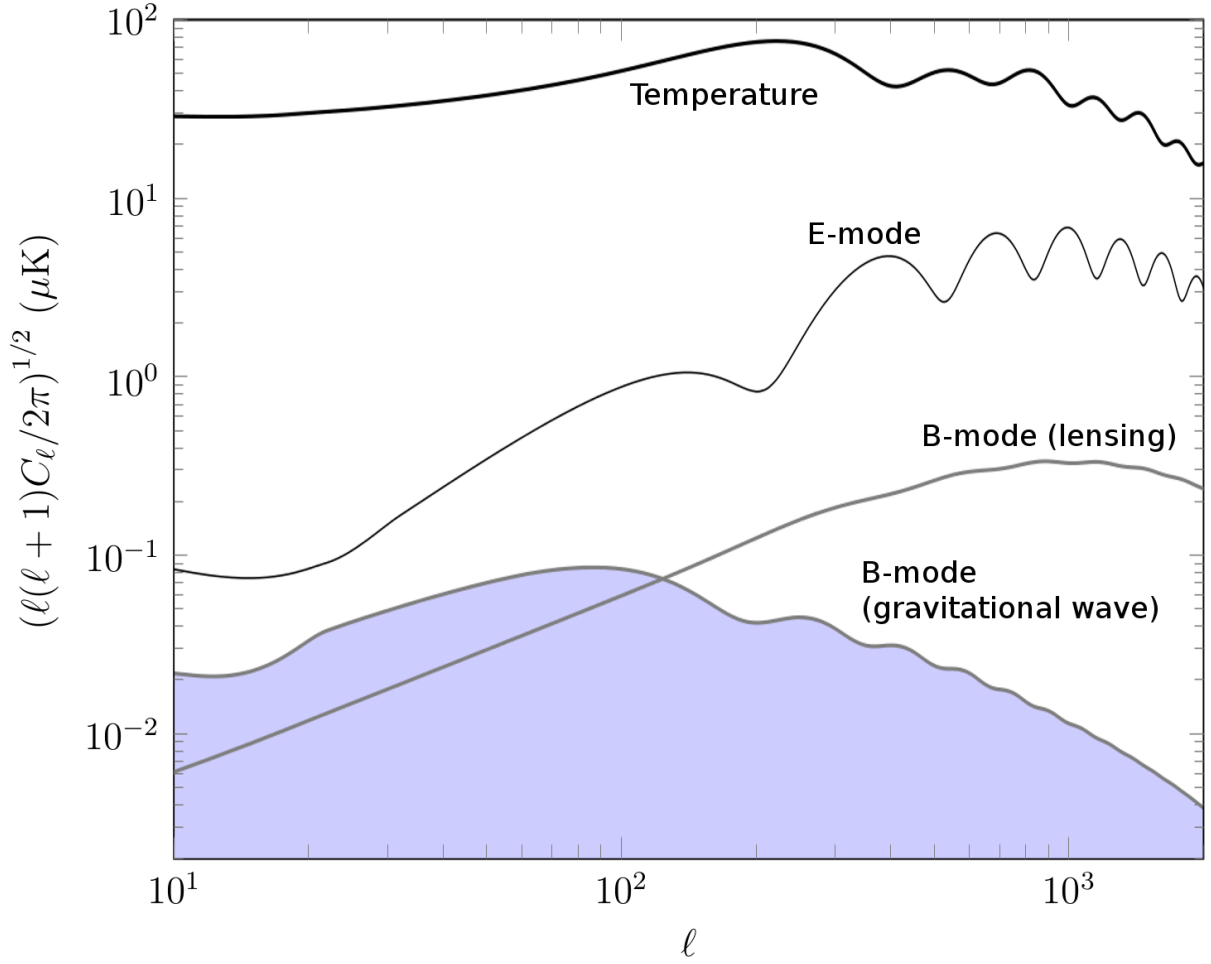


Figure 1.1: Model of the temperature, E-mode, and B-mode power spectra using the best-fit Planck  $\Lambda$ CDM model and assuming a tensor-to-scalar ratio of 0.1. The B-mode curve is separated into the lensing contribution and the inflationary gravitational wave contribution. The blue shaded area represents possibilities for ratios  $< 0.1$ . Figure adapted from [8].

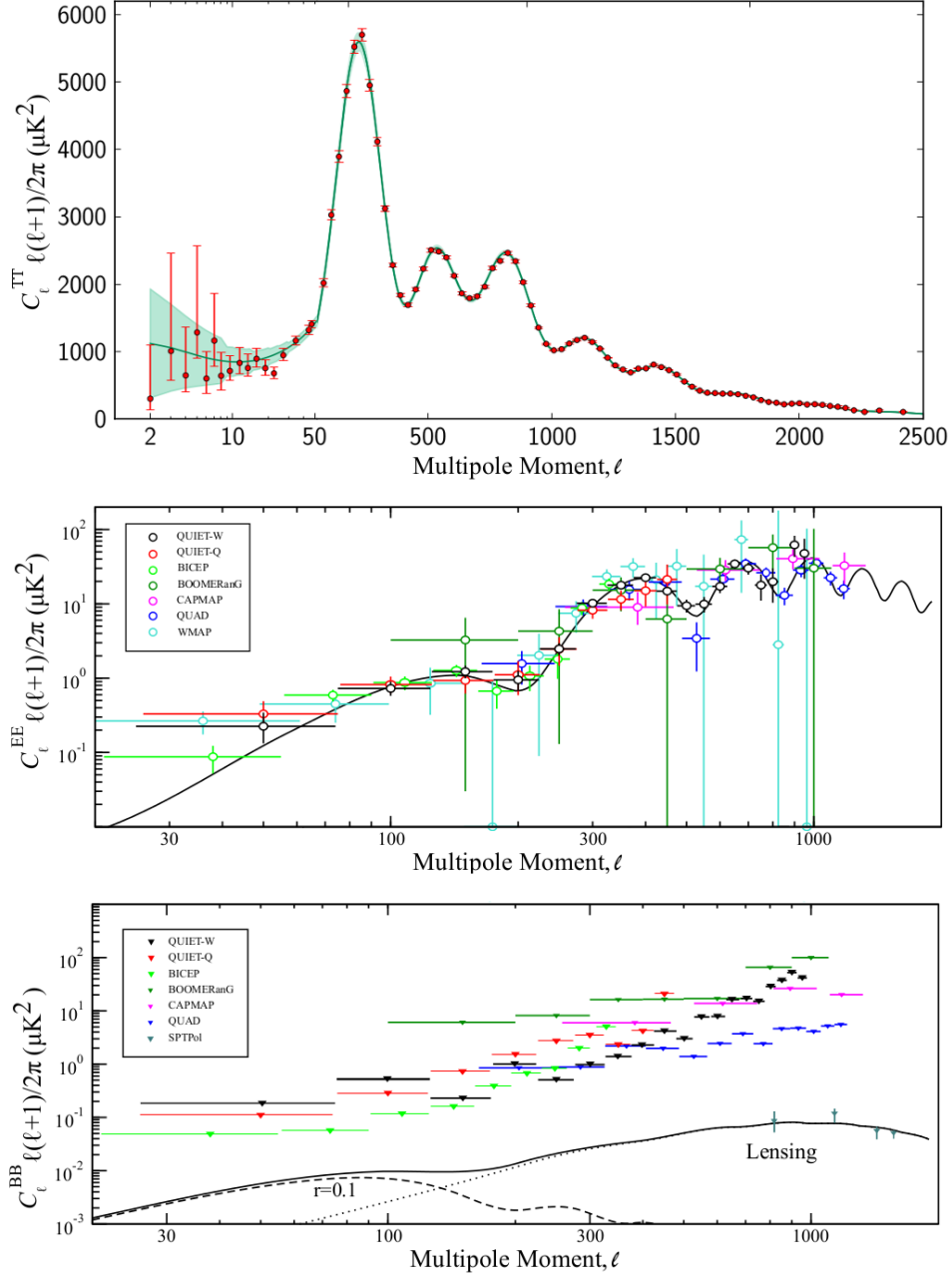


Figure 1.2: Measurements of the temperature, E-mode, and B-mode power spectra as well as best-fit models and a model assuming a tensor-to-scalar ratio ( $r$ ) of 0.1. The top figure shows the Planck CMB temperature (TT) power spectrum. Figure from [9]. The middle and bottom figures show E-mode (EE) and B-mode (BB) measured data points or upper limits. The B-mode model shown in the bottom plot also shows the lensing contribution and the inflationary gravitational wave contribution. For the gravitational wave contribution, a tensor-to-scalar ratio ( $r$ ) of 0.1 is used. Middle and bottom figures from [8] in which it was adapted from [10] and [11].

these inhomogeneities [12].

To quantify the anisotropy at different length scales, it is useful to decompose the temperature pattern into spherical harmonics:

$$\frac{\Delta T}{T} = \sum_{l,m} a_{lm} Y_{lm}(\theta, \phi)$$

where measurements of the coefficients  $a_{lm}$  yields a sampling of their distribution. Under the assumption that the Universe is isotropic we can average together the modes at a given angular frequency  $l$ . We can then quantify the variance, given by  $C_l$ , at an angular frequency  $l$  with:

$$C_l = \frac{1}{2l+1} \sum_m \langle a_{lm}^* a_{lm} \rangle$$

Plotting  $C_l$  as a function of  $l$  reveals a power spectrum like that of the temperature line in Figure 1.1 and the top plot of Figure 1.2. The shape of the power spectrum encodes the 6 parameters of the  $\Lambda$ CDM model. Temperature anisotropies in the CMB were first discovered by COBE in 1992, and subsequent experiments have better characterized the shape of the power spectrum [13][14][15][16].

### 1.3.2 Polarization Anisotropies

Polarization anisotropies in the CMB are imprinted at the surface of last scattering as a result of quadrupole anisotropies in the plasma density. As shown in Figure 1.3, hot and cold photons originating from a quadrupolar pattern create a net polarization after being Thomson scattered toward the observer. Three physical effects may have created quadrupolar anisotropies in the surface of last scattering [18]:

- **Density Fluctuations** - Density fluctuations in the plasma form a **scalar** perturbation that contains quadrupoles.

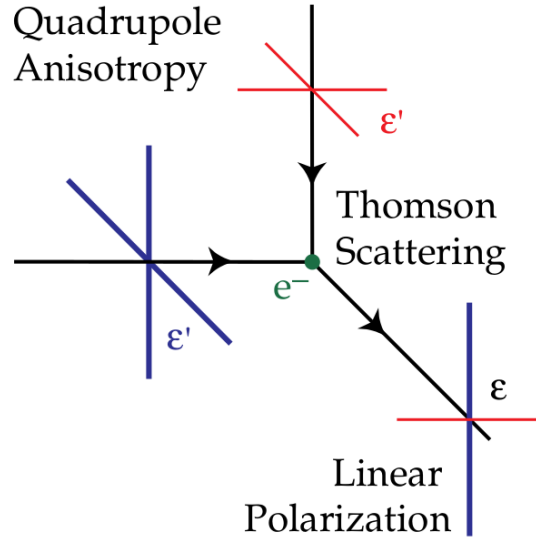


Figure 1.3: Net polarization resulting from Thomson scattering of photons originating from a quadrupolar pattern. Cooler photons (red) originating from above can only scatter towards the observer with horizontal polarization. Likewise, warmer photons (blue) originating from the left can only scatter towards the observer with vertical polarization. Figure from [17].

- **Vorticity** - Vorticity in the plasma can lead to **vector** perturbations that result in quadrupoles. In most inflationary models these modes are expected to be negligible when compared with the scalar and tensor perturbations resulting from the other two effects.
- **Gravitational Waves** - Gravitational waves remnant from inflation would be **tensor** perturbations that create quadrupole variations.

These three types of perturbations result in different polarization patterns in the CMB [17]. To distinguish between the sources of polarization it is useful to decompose the polarization field in the CMB into curl-free (“E-mode”) and divergence-free (“B-mode”) components. Due to the symmetries involved in each type of perturbation, only scalar and tensor perturbations contribute to E-mode patterns, and only vector and tensor perturbations contribute to B-mode patterns. Given the expected magnitude of the effects, scalar perturbations are expected to dominate the E-modes and tensor perturbations are expected to dominate the

B-modes at the surface of last scattering. This makes the primordial B-mode, if it exists, a significant piece of evidence for the theory of inflation. Furthermore, in the simplest inflationary models, the ratio of the amplitudes of the tensor and scalar modes, often referred to as the “tensor-to-scalar ratio” or just “ $r$ ”, would allow us to constrain the energy scale at which inflation took place by [19]:

$$V^{1/4} \propto 10^{16} r^{1/4} \tag{1.1}$$

Although the B-modes at the surface of last scattering can originate from gravitational waves, there are two significant additional sources of B-mode patterns that the CMB collects on its journey from the surface of last scattering to our telescopes:

- Gravitational lensing - Gravitational lensing of the CMB by intervening galaxies mixes E-modes and B-modes into each other [16]. The B-mode signal is expected to be much smaller than the E-mode signal, therefore this mixing contributes relatively little to the E-mode pattern but contributes largely to the B-mode pattern. The B-modes introduced by gravitational lensing will occur at smaller angular scales than those of the primordial B-modes, which allows us to distinguish between the two as shown in Figure 1.1. In addition, the magnitude of the lensing signal can be estimated using current measurements of the matter density in the Universe to integrate the gravitational potential along the line of sight.
- Galactic dust - Polarized dust emission from our Galaxy contributes B-modes at the same angular scales as the primordial B-modes [20]. However, B-modes from Galactic foregrounds will have a different frequency response than those from gravitational waves. We can therefore subtract foreground contaminants by characterizing them at different frequencies.

Polarization anisotropies were first detected by DASI in 2002 [21]. To date the E-mode spectrum has been characterized by many experiments [22][23][24][10][25][26], and an indirect

measurement of lensing B-modes was made by SPTPol in 2013 [11]. The data points from many of these experiments are shown in the middle and bottom plots of Figure 1.2. Furthermore, in 2014 BICEP2 announced a discovery of the inflationary B-modes [27], though subsequent papers have cast doubt on the origin of BICEP2's B-mode measurements, including recent dust measurements by Planck [28].

\*

\* \*

In this thesis we describe the E and B Experiment (EBEX), a balloon-borne CMB polarimeter. EBEX is designed to measure or place an upper limit on the inflationary gravitational wave B-modes and to detect the lensing B-modes. In addition to detectors sensitive to radiation at 150 GHz and 250 GHz, EBEX employs detectors sensitive to the relatively high frequency of 410 GHz. This enables EBEX to characterize the polarized dust foregrounds so that they can be removed.

# Chapter 2

## EBEX Overview

### 2.1 Science Goals

EBEX is a balloon-borne telescope designed to achieve the following three science goals by measuring the polarization of the CMB between  $20 \lesssim l \lesssim 1000$ :

- To detect or place an unprecedented upper limit on the primordial gravitational wave B-mode signal [29].
- To characterize the polarized dust emission and to determine its angular power spectra in both E-mode and B-mode polarizations.
- To achieve a high signal-to-noise detection of both the lensing induced B-modes and the power spectrum of the lensing deflection angle.

EBEX was flown in 2009 as a full system test, and then flown again in December 2012 / January 2013 to collect scientific data (which we will refer to as simply the 2012 flight).

### 2.2 Observation Strategy

EBEX is designed to climb into the stratosphere above Antarctica on a large helium balloon and float at about 120 000 ft (37 km) for two weeks while pointing at specific patches of the



southern celestial sky.

The thin atmosphere at float altitude allows EBEX to measure the CMB with high signal-to-noise when compared with ground based experiments, especially at higher frequencies, due to a number of reasons:

- Lower fluctuations in atmospheric emission result in a smaller noise contribution.
- Lower absolute level of atmospheric emission results in lower optical loading which allows for higher instantaneous detector sensitivity.
- Lower atmospheric absorption at 410 GHz allows more CMB photons to reach the telescope (this effect is less significant at 150 GHz and 250 GHz).

EBEX is supported as a long duration balloon project by NASA’s Columbia Scientific Balloon Facility (CSBF) and so it is designed to be launched from Willy Field near McMurdo Station in Antarctica and travel westward around the continent with the polar vortex winds. The altitude that the payload reaches is a function of, among other things, the weight of the payload, and in 2012 EBEX flew between 110 500 ft and 120 500 ft. The duration that the payload can remain at float is generally longer than two weeks, and so is not the bottleneck for how long EBEX can collect scientific data. Instead the flight is limited by the capacity of the cryogenic system that keeps the detectors cold. During the 2012 Antarctic flight EBEX collected scientific data for  $\sim 11$  days.

EBEX is designed to scan a 400 square degree patch of the sky centered on right ascension 4.8 h and declination  $-45.5^\circ$  to extract scientific parameters from the CMB. We call this the “science” patch. The location of this patch was chosen for its low foreground contribution, and the size was chosen to be maximally sensitive to IGB B-modes. In addition, EBEX is designed to scan the embedded star cluster RCW 38 for pointing and signal calibration, in what we call a “calibrator” scan patch. It is also designed to take occasional elevation dips to calibrate for atmospheric loading.

## 2.3 Instrument

### 2.3.1 Gondola and Attitude Control

The components that constitute the EBEX payload are mounted to an aluminum structure called a gondola. The gondola is designed in such a way that it can support all of the necessary components while having the freedom to actuate the desired attitude control, all the while keeping the entire telescope within the geometric constraints required for launch and under the weight limit for flight. A model of the gondola is shown in Figure 2.1.

The gondola is composed of two large sections, an outer frame and an inner frame. The optics and receiver are mounted to the inner frame, which can move in elevation (up and down) relative to the outer frame. A linear actuator attached to both frames controls this motion. The outer frame hangs from a triangular support structure, which effectively hangs from the balloon. In between the support structure and the connection to the balloon system is a motor (referred to as the pivot motor) that controls the azimuth motion of the gondola. In addition, a reaction wheel, driven by another motor, is mounted to the outer frame and contributes to azimuth control.

The attitude control system (ACS) collects pointing information and controls the gondola during flight. Pointing information is used in post-flight analysis to place detector samples into the correct pixels on a sky map, and is used during flight for real-time attitude control. Real-time attitude control is necessary to realize the desired scan patterns, which have been designed to maximize the scientific output from the flight, and to support subsystems that have special pointing requirements. These subsystems include the star cameras, which require stationary pointing, the power system, which requires sun coverage on the solar panels, and various other subsystems that have sun shielding requirements for thermal reasons.

The ACS's sensors and control loops will be discussed in more detail in Chapter 3. The ACS's actuators, along with the gondola in general, are discussed in more detail elsewhere [30][31].

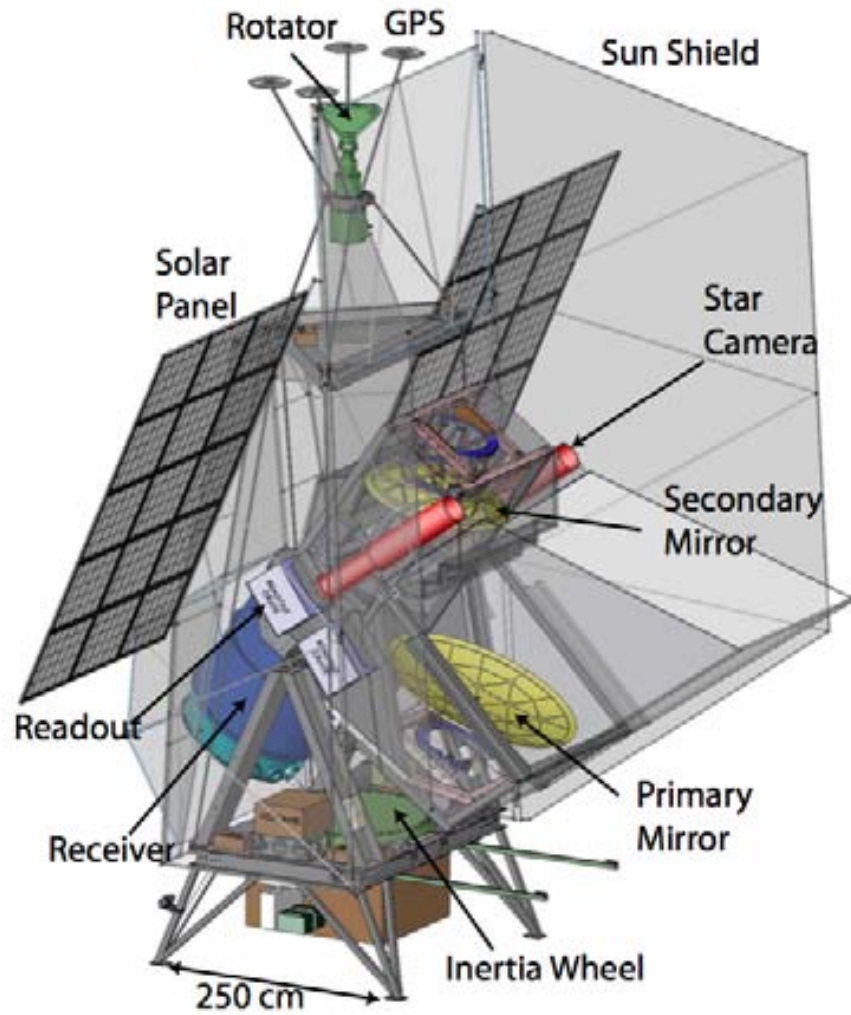


Figure 2.1: A model of the gondola and the telescope components that it houses. Some of the critical components from the attitude control, power, optics, and detector systems are labeled.

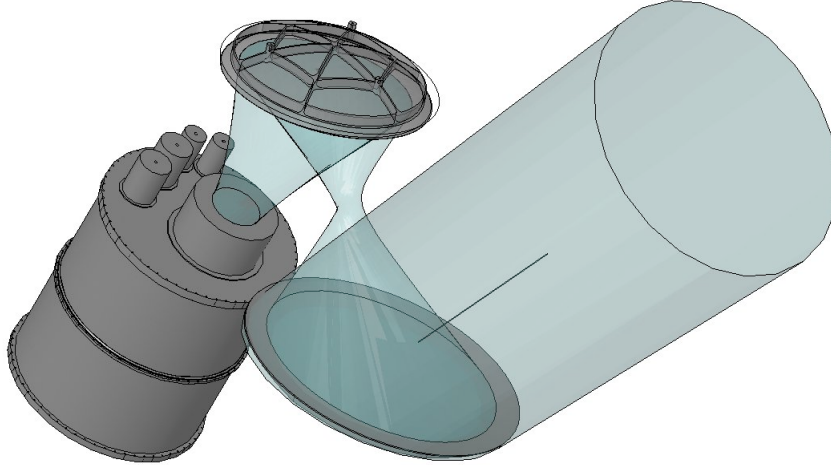


Figure 2.2: A model of the warm optics. Light from the sky reflects off the primary mirror onto the secondary mirror and then into the cryostat. Figure courtesy of Huan Tran.

### 2.3.2 Optics and Receiver

The EBEX receiver is located inside a cryostat. As a result, it is convenient to group the optical components of the telescope into two categories: warm and cold, which respectively sit outside and inside the cryostat.

The warm optics consist of two mirrors that sit in an off-axis Gregorian Mizuguchi-Dragone configuration as shown in Figure 2.2. Light from the sky first encounters the primary mirror, which is a 1.5 m section of a paraboloid, then reflects off the secondary mirror, which is a 110 cm by 98 cm section of an ellipsoid, before making its way into the cryostat. The Gregorian Mizuguchi-Dragone configuration is chosen to minimize polarized systematics [32].

After reflecting off the warm mirrors, radiation from the sky enters the cryostat through a 0.02 in thick 30 cm diameter vacuum sealing window. Until the payload reaches float an additional 0.5 in thick window helps maintain the more significant pressure differential between the cryostat and the atmosphere. The cryostat contains five progressively cooler stages: 300 (ambient), 77, 4, 1, and finally 0.25 K where the detectors are located. Low-pass

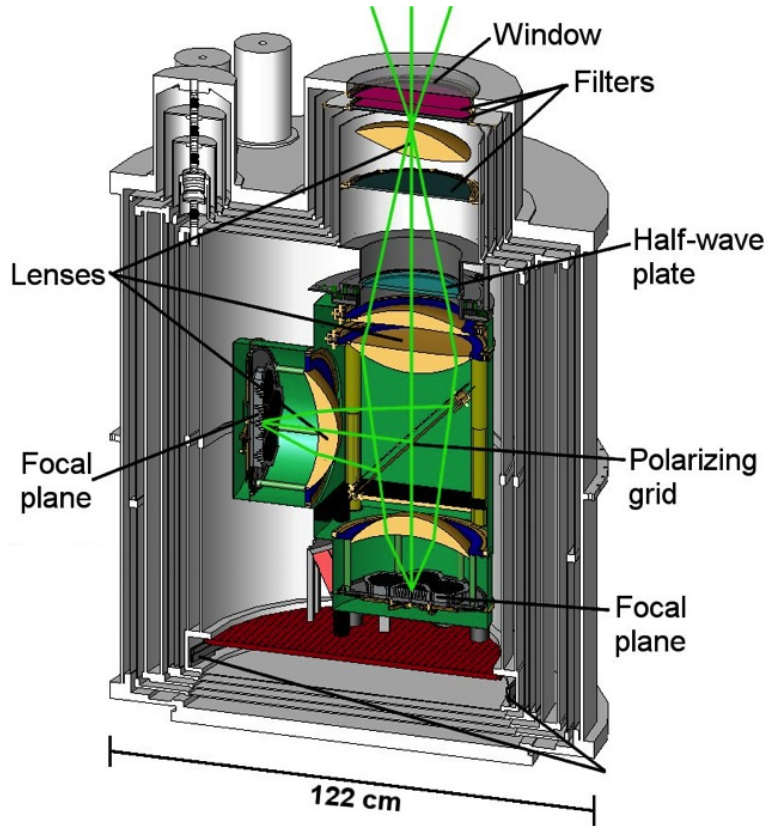


Figure 2.3: A model of the receiver, which contains the cold optics. The window is the entry point for light into the cryostat. Filters help protect subsequent stages from thermal load. A half-wave plate and polarizing grid are used for polarimetry. Lenses re-image the primary mirror onto an aperture stop and the two focal planes.

filters are installed at each of the first three stages to reduce thermal loading on subsequent stages [33]. Lenses re-image the primary mirror onto a 22 cm (cold) aperture stop and then again onto the focal planes. An achromatic half-wave plate (HWP) is located at the cold aperture stop, which modulates the polarization of the light before it encounters a wire grid polarizer. The polarizing grid is mounted at  $45^\circ$  to the optical path and splits the light into two polarizations, sending each polarization to one of two separate focal planes. On each focal plane conical feed horns couple the light to the detectors. The frequency bands for the detectors are defined by high-passing wave guides and low-passing metal mesh filters. The receiver is discussed in more detail elsewhere [34][35].

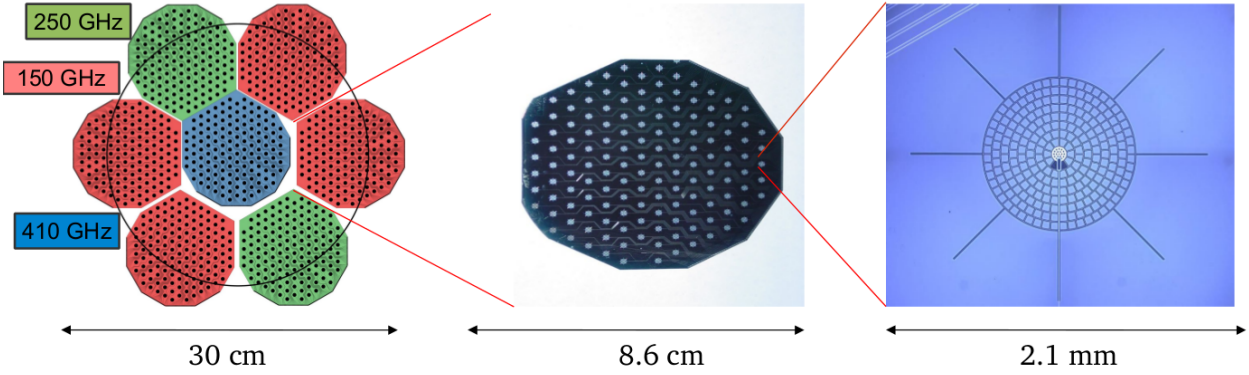


Figure 2.4: (left) An example focal plane. Each focal plane has four 150 GHz wafers, two 250 GHz wafers, and one 410 GHz wafer. Detectors within  $3^\circ$  of the center have a Strehl ratio greater than 0.9, as shown by the black circle. (center) An example wafer. (right) An example TES bolometric detector.

The EBEX optical design results in an  $8'$  beam and Strehl ratio greater than 0.9 across  $6^\circ$  of field of view for all three frequency bands (see Figure 2.4).

### 2.3.3 Polarimetry

The half-wave plate rotates continuously in order to rotate the polarization of the incoming light at twice the HWP rotation rate ( $f$ ). This modulates the signal at  $2f$  before the wire grid polarizer separates it into horizontal and vertical states and sends each state to one of the two focal planes. Since the detectors are phase insensitive, they measure a given polarization signal at  $4f$ . Combined with the non-zero scan speed of the telescope, this puts the sky signal into sidebands of  $4f$ , which greatly mitigates errors for two reasons. The first is that it moves the signal away from low frequency noise. The second is that it distinguishes polarized sky signal from polarized emissions originating from the telescope itself. For example:

- constant polarized emission originating from telescope components behind the HWP are found at 0 Hz (i.e. unmodulated)

- constant polarized emission originating from telescope components in front of the HWP are found at exactly  $4f$
- polarized signal from the sky is found in the sidebands of  $4f$  because it is additionally modulated by the scanning motion of the telescope

The HWP rotates on a superconducting magnetic bearing in order to reduce friction, which would otherwise be a significant source of thermal noise [36]. A tensioned kevlar belt drives the HWP. Slots in the HWP mechanism, combined with an LED and a cryogenic photodiode detector, allow us to measure the angular position of the HWP as an optical encoder. The EBEX polarimetry system is discussed in more detail elsewhere [37][35].

### 2.3.4 Detectors and Readout

The EBEX detectors are transition edge sensor (TES) bolometers. A bolometer is a device that allows one to measure the power of incident light on it by detecting a change in its electrical resistance. It consists of an absorber connected to a thermal bath with a weak thermal link so that it warms due to incident light, and a thermometer to measure the temperature of the absorber (see Figure 2.5a) TES bolometers are superconductors that are highly sensitive to changes in temperature because of their narrow superconducting transition, as shown in Figure 2.5b. The EBEX bolometer signals are amplified with superconducting quantum interference devices (SQUIDs) [38].

The bolometers are fabricated onto silicon wafers using thin film deposition and optical lithography. Each wafer holds 139 bolometers, and each focal plane holds 7 wafers. On each focal plane there are four 150 GHz wafers, two 250 GHz wafers, and one 410 GHz wafer, as shown in Figure 2.4. Although these numbers suggest that there could be 1946 detectors in operation, for various reasons (primarily limitations on fabrication yield) there were 1107 in operation during the 2012 Antarctic flight [39].

Voltage biasing and read out is controlled by 28 electronics boards with preprogrammed

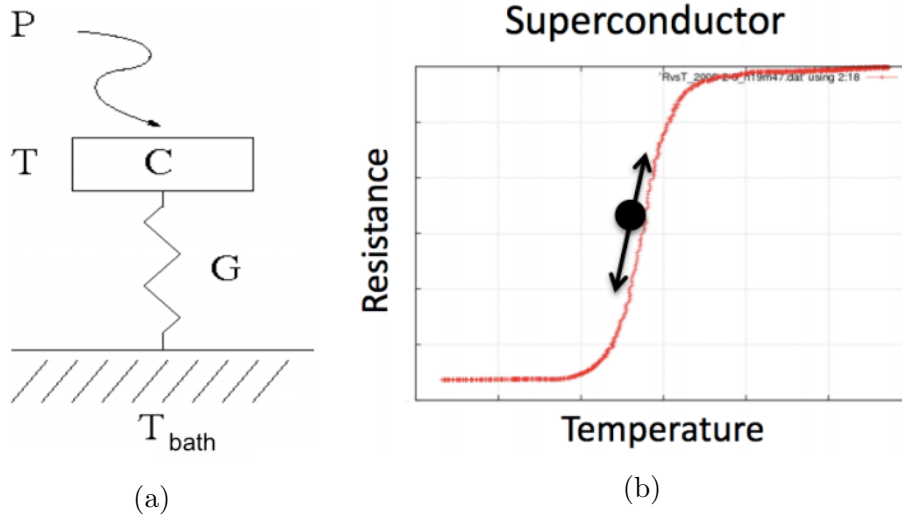


Figure 2.5: (a) Conceptual diagram of a bolometer. An absorptive element with temperature  $T$  and heat capacity  $C$  absorbs incoming power  $P$ , which is then conducted to a thermal bath at temperature  $T_{\text{bath}}$  through a weak thermal link with conductance  $G$ . Diagram courtesy of Johannes Hubmayr. (b) Superconducting transition shows a steep change in electrical resistance as a function of temperature.

field-programmable gate array (FPGA) processors that we call digital frequency domain multiplexing (DfMUX) boards. The DfMUX boards have digital-to-analog converters and analog-to-digital converters to generate analog carrier signals and read analog bolometer signals. To reduce thermal load on the cryostat by limiting the number of wires between stages, the readout system frequency multiplexes the signals by a factor of 16 [38][39].

### 2.3.5 Telemetry

CSBF supports the payload with telemetry to allow ground operators to communicate with the telescope during flight. During the first day of flight, the payload is within line-of-sight and can use high rate communication. After that we rely on the TDRSS and Iridium satellite systems for communication so data rates and delays worsen.

The ground operators do not require the ability to send large commands (in terms of data size), so during satellite communication the commanding delay is the primary concern



for uplink. The primary concern for downlink is data rate, as the ground operators require significant amounts of data to monitor the telescope and perform diagnostics. During line-of-sight communication the downlink data rate is 1 Mbps. Outside of line-of-sight the data rate depends on whether the pointed high gain antenna is operational at 70 kbps, or if the system has to fall back on the omnidirectional antenna at 6 kbps. Data prioritization and compression schemes are employed to make the most of the limited bandwidth [40].

### **2.3.6 Power**

The gondola is powered by lithium ion batteries that store enough energy to support the experiment for several hours. The batteries are recharged at float by solar panels. The solar panels face backwards, since the EBEX scan strategy involves pointing roughly anti-sun, and are tilted upward to maximize exposure to the sun. The power system is discussed in more detail elsewhere [31].

### **2.3.7 Thermal**

Most of the payload components are shielded from solar radiation by a large baffle structure called the sun shield. Another baffle structure, called the ground shield, protects the mirrors and surrounding area from ground radiation. These baffle structures are composed of foam pieces lined with aluminized mylar and mounted to an aluminum skeleton.

Being shielded from the sun, most of the electronics do not overheat as their excess power is transferred to the gondola or directly radiated away. The 28 DfMUX boards discussed in Section 2.3.4, however, consume approximately 600 W of power combined. They are distributed across four crates mounted on the sides of the gondola, which do not have enough surface area to dissipate this heat through conduction to the gondola and radiation to space. To solve this problem, EBEX has a liquid cooling system that pumps Dynalene HC-40 to transfer heat from the readout crates to large aluminum radiator panels located at the front

of the gondola. The thermal design is discussed in more detail elsewhere [31].

# Chapter 3

## Attitude Control System

### 3.1 Introduction

The attitude control system (ACS) can be broken down conceptually into three categories:

1. **Sensors** - Data from seven types of pointing sensors, discussed in Section 3.4, are stored on disk for post-flight processing and fed into the flight computers for real-time processing.
2. **Control Algorithms** - Real-time attitude determination algorithms on the flight computers calculate a telescope boresight attitude solution at every timestep. Flight computer and digital signal processor (DSP) control loops combine this attitude solution with desired scan patterns to output signals for the motor controllers.
3. **Actuators** - Three motors drive the telescope in azimuth and elevation.

Two redundant flight computers run high level control algorithms, and are located inside a crate known as the flight computer crate. DSPs run low level control loops and are located on custom electronics boards that we refer to as “ACS cards”, which are located inside a crate known as the ACS crate. There are four ACS cards inside the ACS crate, and in addition to the low level control loops they also handle input/output (I/O) for the sensors and actuators. The ACS crate also routes power and signal lines to and from ACS components. In addition

to the two flight computers, the flight computer crate also houses two redundant network switches and custom boards that handle timing and watchdogging. Two pressure vessels protect the hard disks from the space-like environment.

The ACS data flow is shown in more detail in Figure 3.1. Much of the attitude control system is adopted and expanded upon from the Balloon-borne Large Aperture Submillimeter Telescope (BLAST) [41].

In this chapter we begin by reviewing coordinate systems and discussing the pointing requirements. We then discuss the sensors and control algorithms in detail.

## 3.2 Coordinate Systems

### 3.2.1 Celestial Coordinate Systems

Table 3.1: Summary of the three coordinate systems used in this document. A more in depth discussion of celestial coordinate systems can be found in [42].

Coordinate System	Fundamental Plane	Reference Point	Azimuth Coordinate	Elevation Coordinate
horizontal	horizon	true north	azimuth	elevation
equatorial	celestial equator	vernal equinox	right ascension	declination
galactic	galactic plane	galactic center	galactic longitude	galactic latitude

There are three coordinate systems that we will use in this document to define the direction that the telescope is pointing at an instant in time: the horizontal, equatorial, and galactic systems. In each system there is an azimuth coordinate and an elevation coordinate, which are the equivalent of longitude and latitude in the geographic coordinate system used to describe locations on Earth. These coordinate systems describe a direction in space, as if the telescope is pointing at a celestial object, but do not include a distance coordinate (which would be akin to altitude in geographic coordinates), and so the two coordinates can be thought of as defining a point on a unit sphere that is sometimes called the

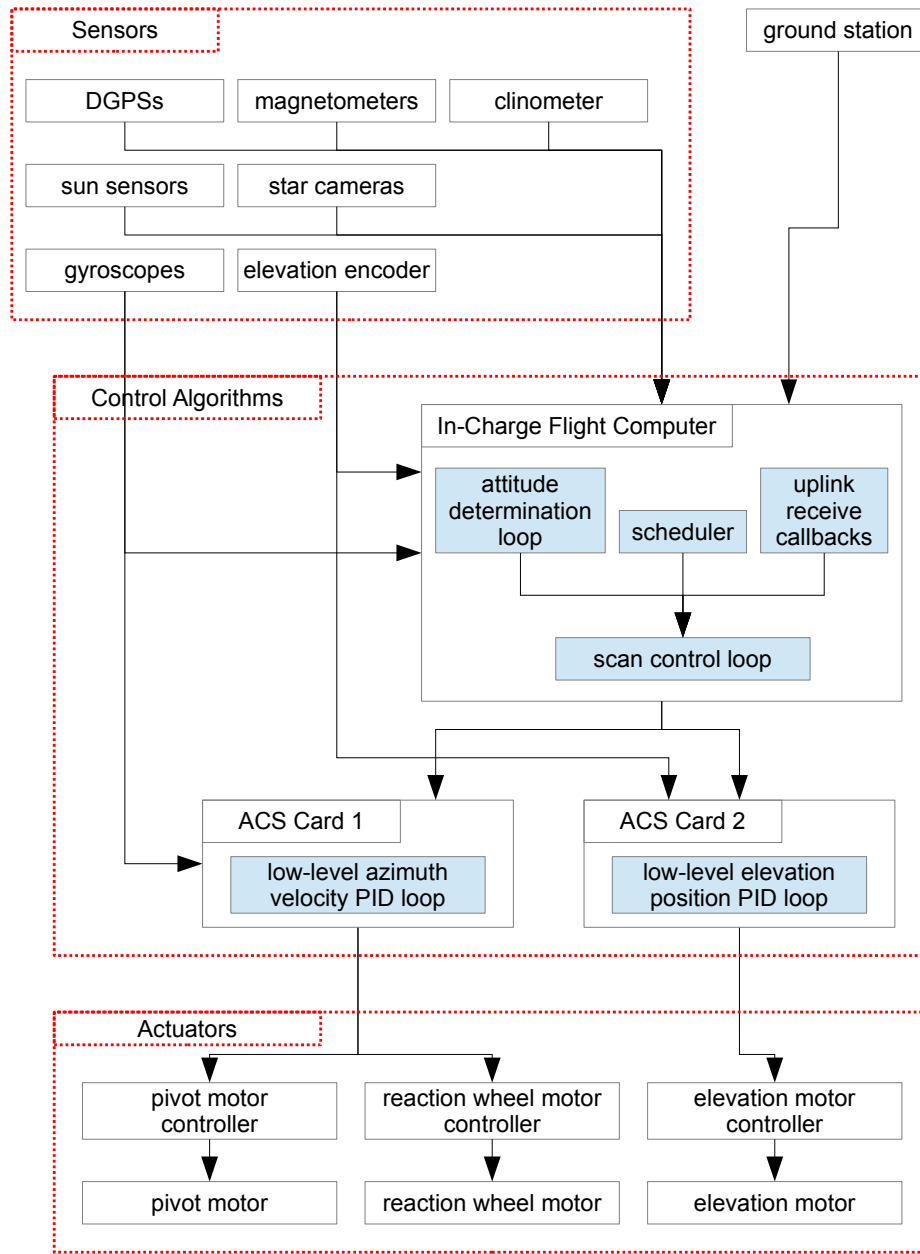


Figure 3.1: Overview of the attitude control system. Filled blue boxes represent software/firmware loops. Black bordered boxes represent physical components. The physical components can be grouped into three categories represented by red lines: sensors, control algorithms, and actuators. There are many physical connections between components, but the arrows here represent the relevant flow of data. Sensor data is used by the flight computers to compute a real-time attitude solution (see Section 3.5.1) and run a scan control loop (Section 3.5.2) to output a requested azimuth velocity and elevation position for the low level control loops. The gyros and elevation encoder are also used directly by the low level control loops, along with the requested azimuth velocity and elevation position, to output requested power for the motor controllers (see Section 3.5.3).

spacecraft-centered celestial sphere [42]. The two coordinates must be defined in relation to a fundamental plane, which is the equivalent of the equator in geographic coordinates, as well as a reference point or reference meridian to define the zero-point of the azimuth coordinate, which is the equivalent of the Prime meridian that passes through Greenwich in geographic coordinates. The two coordinates also take on different names in each coordinate systems. We will describe the three coordinate systems below and summarize their definitions in Table 3.1.

- **Horizontal Coordinate System** - The fundamental plane of the horizontal coordinate system is the horizon as viewed from the telescope's local vantage point on Earth, and the reference point is true North. A more strict definition of the fundamental plane is that it divides the celestial sphere in half and is normal to the line that runs through the telescope and the center of the Earth. This more strict definition accomodates the fact that the actual horizon cannot be used as the fundamental plane on high altitude payloads, as the horizon at those altitudes is approximately  $6^\circ$  lower in elevation than the plane that divides the celestial sphere in half.

In this system the azimuth and elevation coordinates are simply named **azimuth** and **elevation**.

We use this coordinate system for **real-time control of the telescope**, because its axes are conveniently aligned with the motors of the telescope. This fact is evident when you consider that the line passing through the center of the Earth, which defines the horizontal coordinate system, is aligned with the gravity vector that keeps the telescope upright when hanging from a balloon. The pivot and reaction wheel motors which rotate the gondola left and right are in fact controlling azimuth, while the motor that drives the inner frame up and down is controlling elevation. FCP performs its real-time attitude determination in this coordinate system.

- **Equatorial Coordinate System** - The fundamental plane in this system is the ce-

lestial equator, which is the Earth’s equator projected onto the celestial sphere, and the reference point is the vernal equinox. The two relevant angles are called **right ascension** and **declination**. This system is fixed with respect to the stars<sup>1</sup>. We can convert between this system and the horizontal system if the geographic location of the telescope and the time of the observation are known.

This is the system that the star cameras fundamentally operate in, since they find pointing solutions based on catalog star positions that are defined in this coordinate system (see Chapter 4). In real-time, however, the star camera solutions are converted to angles in the horizontal coordinate system for the flight computers. This system is also convenient for map making and scientific analysis, as the maps are independent of the telescope’s location and the times of the observations, so maps created by different observations should be similar. We use this system for **pointing reconstruction**.

- **Galactic Coordinate System** - In this system the fundamental plane is defined to be approximately aligned with the galactic plane, and the reference point is in the approximate direction of the center of our galaxy. The two relevant angles are called **galactic longitude** and **galactic latitude**. This system is useful for the same reasons that the equatorial system is useful, though the equatorial system can be more intuitive when thinking about scan patterns for an Earth-bound or suborbital telescope. However, the galactic coordinate system is more convenient for analysis that requires making special accommodations for the galaxy. We primarily use this coordinate system when **constructing maps** during post-flight analysis.

### 3.2.2 Roll Angle

When describing the orientation of the telescope, we are also concerned with a third angle. This angle is the rotation of the telescope about its pointing vector, or the “roll” angle. Note

---

<sup>1</sup>Whether this is entirely accurate depends on whether one includes the Earth’s precession and nutation in their definition.

that this is not identical to the third angle commonly used in spherical coordinates, e.g. altitude in geographic coordinates or  $\rho$  in a mathematical  $(\phi, \theta, \rho)$  system. Instead this third angle completes a set of angles that are known as Euler angles. Euler angles describe the rotation of one basis with respect to another. In our case, we often define the rotation of the gondola body frame, which is a basis that is fixed to the telescope optics, with respect to the equatorial reference frame or the galactic reference frame.

The roll must be included in the reconstructed pointing solution for two reasons. The first is that the pointing solution of the microwave boresight must be rotated into a pointing solution for each individual detector and this rotation depends on the roll of the boresight. The second reason is that the angle of the half-wave plate axis with respect to the celestial coordinate system must be preserved for polarization analysis.

### 3.2.3 Fair Measure Coordinates

When discussing pointing errors we will often use an angle called cross-elevation as a replacement for azimuth. The reason is that an azimuth differential does not represent a consistent angular distance<sup>2</sup> at different elevations due to the nature of spherical coordinates. As an example, consider two points separated by  $10^\circ$  in azimuth. If they are both located at  $0^\circ$  elevation then they are separated by an angular distance of  $10^\circ$ . However if they are located at  $60^\circ$  elevation, then they are only separated by an angular distance of  $5^\circ$  due to the fact that the meridian lines close in on each other as they approach the poles.

A cross-elevation differential is defined as an azimuth differential multiplied by  $\cos(\text{elevation})$ . We use it as the “x-coordinate” when discussing pointing errors because it provides a measure of angular distance under a flat sky approximation with small angles. Similarly, we use cross-declination instead of right ascension when discussing pointing errors in the equatorial coordinate system.

---

<sup>2</sup>When using the term “angular distance” we are referring to the arc length of a great circle arc connecting two points.



## 3.3 Pointing Requirements

The attitude control system determines the pointing of the telescope for two distinct purposes: as part of a feedback loop to control the telescope in real time, and as part of the post-flight data analysis procedure for map making. The real-time and post-flight pointing requirements are different, and are discussed here.

### 3.3.1 Real-Time

The real-time pointing requirement is roughly  $0.5^\circ$  in cross-elevation and in elevation, and is related to the size of the calibrator patch. The calibrator scan, discussed in Section 3.5.2, is a raster scan designed to be just large enough so that every pixel in the  $\sim 7^\circ$  focal plane scans past the relatively small calibration source RCW 38.

The consequence of a real-time pointing error is that the calibrator patch must be expanded to contain a buffer on each edge to ensure that every detector will still scan past the calibrator source. Increasing the calibrator scan size results in more time spent performing calibrator scans, and thus less time spent performing science scans. Buffers of more than  $\sim 10\%$  of the focal plane width, or  $\sim 0.7^\circ$ , consume a significant fraction of the flight time given the number of calibrator scans we must perform. We therefore set the requirement at  $0.5^\circ$ , safely under 10%.

The real-time pointing algorithms assume that the telescope has zero horizontal roll, i.e. it is balanced left-right. A left-right imbalance, multiplied by the lever arm from the telescope boresight to an off-axis detector, also results in a pointing error. However, given the pre-flight balancing procedure and the magnitude of roll pendulations at float, this effect is subdominant.

### 3.3.2 Post-Flight

In post-flight data analysis, the pointing at every timestep is used to place detector samples in the correct pixel on a map. A power spectrum of the map reveals scientific results, and pointing errors have the effect of biasing these results. Therefore, the post-flight pointing requirement is much more stringent than the real-time requirement, and it happens to be much more strict. The requirement is specified in the map domain, and requires that the average of all the pointing errors inside a given pixel of size  $2'$  be less than  $\sim 10''$  [43]. It is calculated by requiring that the most dominant effect caused by pointing errors, which is the mixing of E-modes into B-modes at small angular scales [44], be negligible.

## 3.4 Sensors

EBEX employs six different types of absolute sensors to measure angular position and high precision optical gyroscopes to measure angular velocity. The flight control program (FCP) acquires measurements from each sensor and records them to disk. FCP also holds a continuous real-time pointing stream for each active axis of each absolute pointing sensor. This means that at every timestep of the real-time attitude determination loop FCP has an estimate of the current azimuth and/or elevation of each pointing sensor that is active for those coordinates. Each of these pointing streams is the result of combining the absolute sensor data whenever it is available with gyroscope data, which is assumed to always be available at the rate of the attitude determination loop. The attitude determination loop, which runs at 100.16 Hz, runs synchronously with the ACS sensor data acquisition loop.

The different types of absolute pointing sensors have varying degrees of precision, accuracy, reliability, and rates. For example, the star cameras are more precise and more accurate than the magnetometers, but are more complex and therefore less reliable, and only provide new measurements every 40 seconds. For the absolute sensors it is important to distinguish between their measurement precision, sensor accuracy, and boresight accuracy:

- **Measurement Precision** - Each sensor has an inherent limitation on the precision with which it can measure its own orientation with respect to some physical reference. For example, the EBEX differential GPS (DGPS) can measure its azimuth with respect to the GPS satellite constellation with a precision of 24'.
- **Sensor Accuracy** - Most of the absolute sensors are not limited by their inherent precision, rather they are limited in the accuracy with which they can convert their orientations into an azimuth or elevation measurement that is fixed with respect to the inner frame of the gondola. For example, the DGPS antenna system is mounted to a structure that has some azimuthal torsion with respect to the gondola's inner and outer frames, limiting its accuracy to 1°.
- **Boresight Accuracy** - A sensor's accuracy is also limited by the accuracy of a calibration angle that aligns its measurement of azimuth or elevation with that of the microwave boresight of the telescope (i.e. the center of the focal plane). For example, the DGPS antenna system is installed in rough azimuthal alignment with the telescope boresight, but not exactly, so a correction is applied that is only accurate to 0.5°. Any time a calibration angle is determined for an individual sensor, it is used to calculate the calibration angles for all the absolute sensors, because the relative angles between all the absolute sensors are known. Therefore all the absolute sensors suffer from the exact same error in calibration angle.

All of the absolute pointing sensors are listed in Table 3.2, along with their measurement precision, predicted sensor accuracy, and data acquisition rate. The boresight accuracy is not included because it is the same for all sensors. The measured real-time performance of each sensor is shown in Figure 3.2, which includes the limitations on boresight calibration, and also notably includes the limitation on the star camera pointing streams that results from their low measurement rate (this limitation will be explained in Section 3.5.1).

The gyroscopes and star cameras form the core of the pointing system given their accu-

Table 3.2: Table of absolute sensors. The different types of sensors are listed, as well as their short names (used elsewhere in this document) and how many of each type are flown (Num). In most cases EBEX flies two of each type of sensor for redundancy. This table includes which axes the sensors are capable (**C**) or not capable (**N**) of measuring, and - if an axis is capable of being measured - whether or not a real-time pointing stream is implemented (**I**) for that axis in the flight computers. Not all capable axes are implemented due to limited benefits and limited manpower. The table also lists each sensor’s measurement precision, predicted sensor accuracy, and data acquisition rate.

Sensor Type	Short Name	Num	Az	El	Roll	Meas. Precision	Predicted Accuracy	Data Acq. Rate
star camera	xsc	2	<b>I</b>	<b>I</b>	<b>C</b>	1.8''	1.8''	0.025 Hz
sun sensor	pss	2	<b>I</b>	<b>C</b>	<b>N</b>	1'	1°	100.16 Hz
magnetometer	mag	2	<b>I</b>	<b>C</b>	<b>C</b>	12'	4°	100.16 Hz
dgps	dgps	2	<b>I</b>	<b>N</b>	<b>N</b>	24'	1°	5 Hz
elevation encoder	enc	1	<b>N</b>	<b>I</b>	<b>N</b>	20''	30'	100.16 Hz
clinometer	clin	1	<b>N</b>	<b>I</b>	<b>C</b>	3'	30'	100.16 Hz

racy, while the remaining absolute pointing sensors are considered coarse sensors due to their limited accuracy. We will discuss the gyroscopes and star cameras in detail in the following sections and chapters, and briefly review each of the coarse sensors.

### 3.4.1 Gyroscopes

EBEX uses six KVH DSP-3000 fiber optic gyroscopes (gyros). We mount three gyros orthogonally inside an aluminum box to create a 3-axis gyro box, and then fly two of these gyro boxes for redundancy.

These gyros have an angle random walk (noise) of  $4^\circ/\text{h}/\sqrt{\text{Hz}}$ , which as discussed below largely determines how well we can reconstruct the pointing post-flight. There are also two significant systematic errors that need to be accounted for in this gyro system:

- **Bias instability.** The bias drifts over time. This is accounted for in the real-time and post-flight pointing construction filters by comparing integrated gyro readings to differential measurements from the absolute sensors.

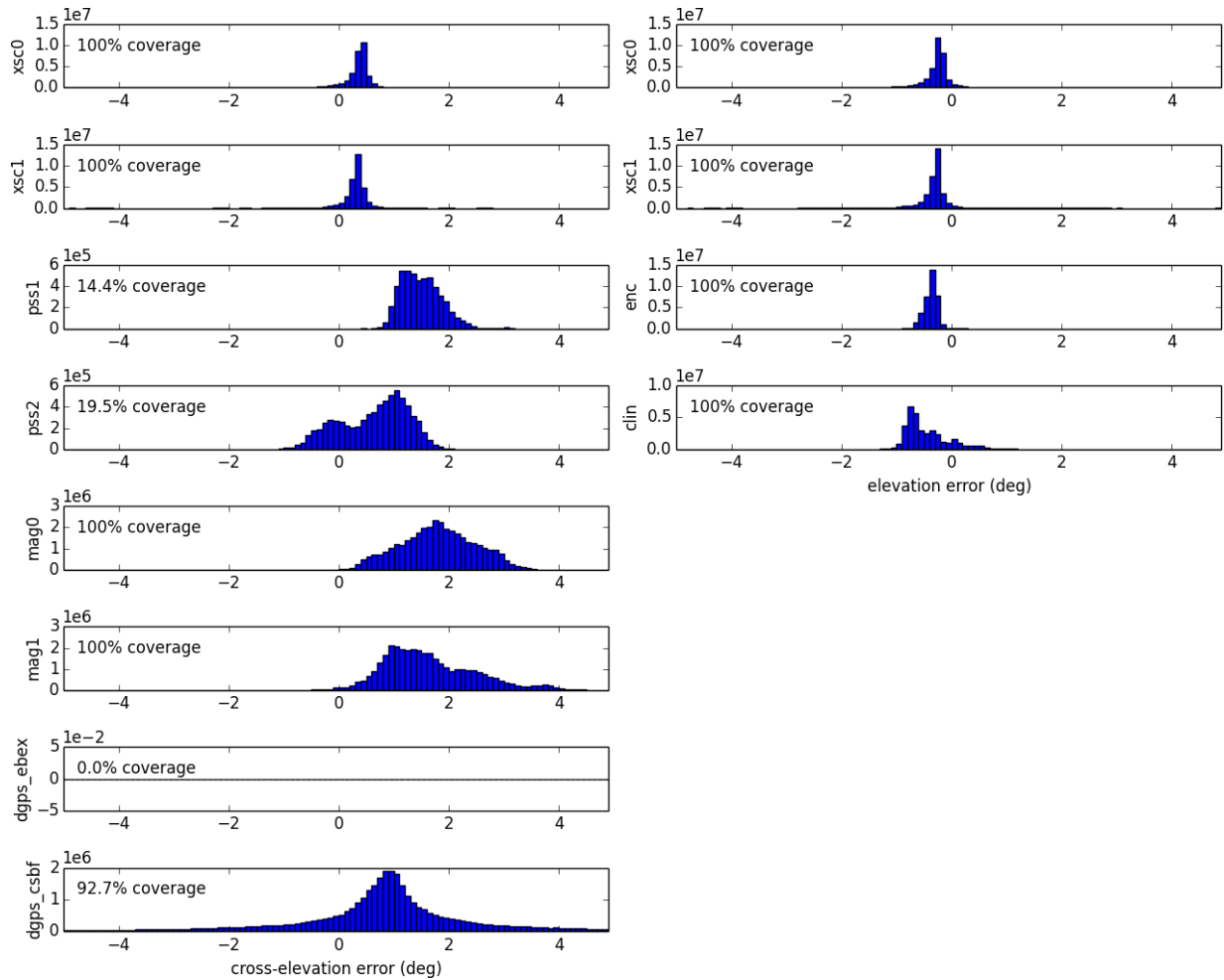


Figure 3.2: The performance in cross-elevation (left column) and elevation (right column) of each absolute pointing sensor during the EBEX 2012 Antarctic flight. Each plot shows a histogram of the differences between the “true” pointing and the pointing measured in flight. The true pointing is taken to be the post-flight reconstructed pointing stream. The measured pointing is the individual pointing stream from each active axis of each absolute pointing sensor that was calculated during flight by the flight control program. However, the calibration angles to the microwave boresight used here may differ from those used in flight. The angles used here are those determined before flight on the ground, and used during flight up until the coarse sensors were calibrated to the star cameras. This gives a fair evaluation of how the sensors performed individually, without the help of the star cameras. Note that not all sensors had valid pointing streams throughout the flight, as indicated by the percent coverage shown in each plot. Most notably the sun sensors had very limited coverage because they did not cover the full azimuth range, and the ebex.dgps had no coverage at float for unknown reasons.

- **Non-orthogonality.** The three gyros in each box are not mounted exactly orthogonal to each other. The magnitude of the misalignments can be determined by performing the procedure described below, which is used to calculate an “orthogonalization” matrix that can be applied to the gyro measurements to correct for these misalignments.

## Gyro Orthogonalization Procedure

The procedure for determining the orthogonalization matrix involves rotating the gyro box about each of three axes. To understand how it works we must first distinguish between two bases that are close to each other but not identical:

- The three axes defined by the orientation of the three physical gyros ( $\hat{1}$ ,  $\hat{2}$ ,  $\hat{3}$ )
- An ideal, imaginary orthogonal basis ( $\hat{x}$ ,  $\hat{y}$ ,  $\hat{z}$ ) that is close to the other basis

The matrix we are looking to create transforms angular velocity measurements along ( $\hat{1}$ ,  $\hat{2}$ ,  $\hat{3}$ ) into measurements along ( $\hat{x}$ ,  $\hat{y}$ ,  $\hat{z}$ ):

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = O \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}$$

An arbitrary rotation can be written as

$$\vec{\omega} = \omega_x \hat{x} + \omega_y \hat{y} + \omega_z \hat{z}$$

so the angular velocity that gyro1 (oriented along  $\hat{1}$ ) will pick up from an arbitrary rotation  $\vec{\omega}$  is

$$\omega_1 = \vec{\omega} \cdot \hat{1} = \omega_x \hat{x} \cdot \hat{1} + \omega_y \hat{y} \cdot \hat{1} + \omega_z \hat{z} \cdot \hat{1} = \begin{bmatrix} \hat{1} \cdot \hat{x} & \hat{1} \cdot \hat{y} & \hat{1} \cdot \hat{z} \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

Doing the same for  $\omega_2$  and  $\omega_3$  gives

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = \begin{bmatrix} \hat{1} \cdot \hat{x} & \hat{1} \cdot \hat{y} & \hat{1} \cdot \hat{z} \\ \hat{2} \cdot \hat{x} & \hat{2} \cdot \hat{y} & \hat{2} \cdot \hat{z} \\ \hat{3} \cdot \hat{x} & \hat{3} \cdot \hat{y} & \hat{3} \cdot \hat{z} \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \implies O^{-1} = \begin{bmatrix} \hat{1} \cdot \hat{x} & \hat{1} \cdot \hat{y} & \hat{1} \cdot \hat{z} \\ \hat{2} \cdot \hat{x} & \hat{2} \cdot \hat{y} & \hat{2} \cdot \hat{z} \\ \hat{3} \cdot \hat{x} & \hat{3} \cdot \hat{y} & \hat{3} \cdot \hat{z} \end{bmatrix}$$

which is simply the inverse of what we're looking for.

Rotating the gyro box about the ideal axes ( $\hat{x}$ ,  $\hat{y}$ ,  $\hat{z}$ ) allows us to measure the terms in  $O^{-1}$ . To see why, imagine rotating the gyro box about  $\hat{x}$ , recording the velocities measured by gyros 1 and 2, and dividing them:

$$\frac{\omega_2}{\omega_1} = \frac{\hat{2} \cdot (\omega_x \hat{x})}{\hat{1} \cdot (\omega_x \hat{x})} = \frac{\hat{2} \cdot \hat{x}}{\hat{1} \cdot \hat{x}}$$

The key to making use of this measurement is to recognize that under a small angle approximation to first order, we can approximate the denominator as being 1, thus giving us a direct measurement of one of the off-diagonal terms in the inverse orthogonalization matrix  $O^{-1}$ :

$$\frac{\hat{2} \cdot \hat{x}}{\hat{1} \cdot \hat{x}} = \frac{\cos \theta_{2x}}{\cos \theta_{1x}} = \frac{\sin \theta_{2y}}{\cos \theta_{1x}} \approx \frac{\theta_{2y}}{1 - \frac{1}{2}\theta_{1x}^2} \approx \frac{\theta_{2y}}{1} \approx \hat{2} \cdot \hat{x}$$

where  $\theta_{1x}$  is the angle between  $\hat{1}$  and  $\hat{x}$ ,  $\theta_{2x}$  is the angle between  $\hat{2}$  and  $\hat{x}$ , and so on. The small angle approximations can be made for angles between corresponding axes, namely  $\theta_{1x}$ ,  $\theta_{2y}$ , and  $\theta_{3z}$  (but not, for example,  $\theta_{2x}$ ). The conceptual interpretation of this approximation is that when rotating about  $\hat{x}$ , gyro1 can accurately (to second order) be used as a measure of the true angular velocity  $\omega$ , while the pickup in gyro2 is a measure of its own misalignment (to first order) from the ideal basis. By the same logic, the diagonal terms are approximated as unity.

When rotating about a given axis, continuous streams of data are taken, then the data for the two gyros are plotted against each other (as shown in Figure 3.3 and in Appendix A),

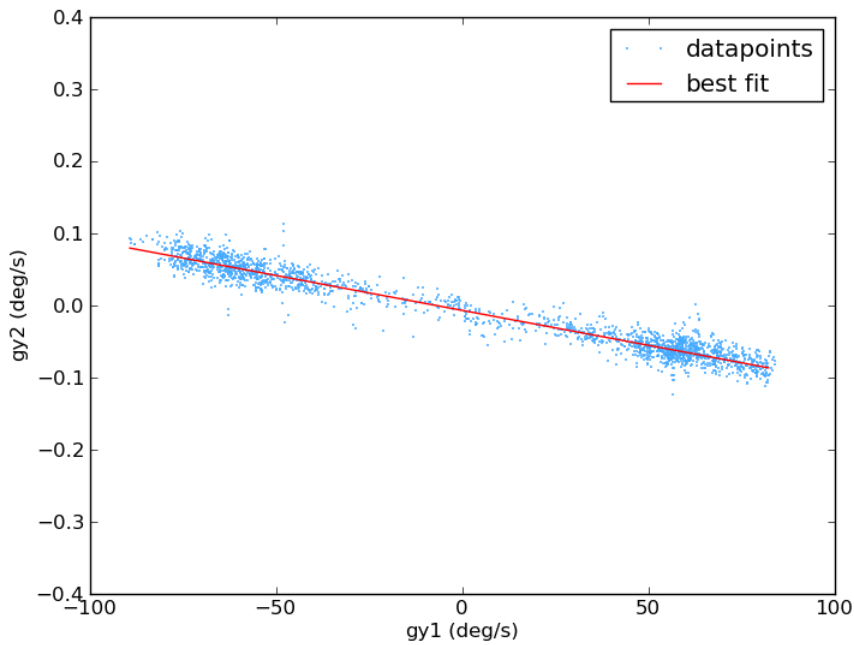


Figure 3.3: Example data from a gyro orthogonalization test. The box is placed on the outer face that gy1 is orthogonal to, and then rotated at various velocities for one minute. The angular velocities measured by gy2 and gy1 are plotted against one another for every timestep. The slope of the resulting best-fit line is used as an element in the inverse orthogonalization matrix. The results from the final orthogonalization run before the 2012 flight are shown in Appendix A.



and the slope of the best-fit line is used as the inverse orthogonalization matrix element. The reason for this is two-fold. First, including multiple data points results in a more precise measurement. Second, the slope of the best-fit line is not affected by any constant gyro biases, whereas a simple division would be. To ensure that the biases are roughly constant we only collect data for up to about one minute.

A final complication is that it is difficult to rotate the gyro box about the axes of an idealized orthogonal basis. Instead we placed three external faces of the gyro box onto a smooth (kapton tape lined) flat surface for the rotations. This method adds an error term to each element of the matrix that is limited by the machine tolerance of the box surfaces, which in our case is 5 mils and results in an error of  $\sim 4.9'$ . The final values for the inverse orthogonalization matrix are shown in Appendix A without this additional  $4.9'$  error.

Ultimately, the individual orthogonalization matrix values were not input into the post-flight pointing reconstruction pipeline and applied directly. Instead the filter was allowed to find its own angles through a least-squares optimization. However, the maximum orthogonalization angle of  $\sim 21'$  determined by the procedure described here was used as a necessary upper bound for the parameter search space. The results of the pointing reconstruction procedure will be discussed in Section 10.2.3.

### 3.4.2 Star Cameras

The star cameras are the most precise pointing sensors. They capture images of the sky, identify the stars in the images by comparing observed stellar patterns to those in an on-board star catalog, and use these identifications to determine the pointing coordinates of the images. These coordinates are then served to the flight computers. They also store the images to disk for post-flight processing. The solutions are precise to  $1.8''$  in cross-declination,  $1.8''$  in declination, and  $67''$  in roll for four-star patterns. If more than four stars are identified in an image the precision increases. However, the EBEX star cameras are

limited to only capturing images when the gondola is stationary (rotating less than  $0.03^\circ/\text{s}$ ), which happens roughly once every 40 s, in order to minimize motion blur during the  $\sim 300$  ms exposure time necessary to achieve the required sensitivity. The star cameras are discussed in detail in Chapters 4 and 5.

### 3.4.3 Coarse Sensors

The suite of coarse sensors are less precise and less accurate, but are used to obtain pointing information when the star cameras are not operating. In flight they are used to give a pointing estimate to the star cameras, before the star cameras have found a pointing solution in order to reduce the search parameter space. During ground tests, they are used to control the gondola when stars are not available, most commonly during the day or when indoors. In that case most often the magnetometers are used for azimuth and the elevation encoder is used for elevation. In addition, the elevation encoder is used directly by the low level elevation control loop due to its reliability. These coarse sensors are:

**Sun Sensors** - The sun sensors are custom built and employ Hamamatsu S5991-01 position sensitive diodes (PSD) as their sensors. Each PSD is contained in a housing with a single pinhole so that the Sun illuminates one spot on the PSD. The position of the incident light can be estimated by measuring the currents in four electrodes. From the mount angle of the sun sensors, the current location of the sun, and an assumed roll (of zero), the azimuth and elevation of the gondola can be determined. The sun sensors are calibrated before flight, though calibration on the ground is difficult because of reflections and because the Sun positions measured by the PSDs are biased by more sky brightness than the float conditions for which the sensors are designed. Although the sun sensor is capable of measuring the location of a bright spot to an accuracy of  $1'$ , the pre-flight calibration limitations were expected to degrade its in-flight accuracy to  $\sim 1^\circ$ .

Note that the sun sensor is technically also a star tracker. However, since it is only

sensitive to one star - the Sun - it can only measure two angles (x and y on the PSD) and thus it can only solve for two coordinates (azimuth and elevation, in our case) while having to assume the third (roll).

**Magnetometers** - The magnetometers are TFS100 three axis magnetometers from MEDA, inc. They measure the Earth's magnetic field and, when compared with a magnetic model of the Earth, can be used to determine gondola attitude. Before flight and during ascent we attempt to measure the gondola's contribution to the magnetic field so that it can be corrected for in flight. Although the magnetometer can measure its orientation inside a magnetic field to  $\sim 12'$ , limitations of the magnetic model near the Earth's magnetic poles and due to the gondola itself were expected to limit its in-flight accuracy to  $\sim 4^\circ$ .

**DGPSs** - The Differential GPS (DGPS) unit is an ADU5 from Thales Navigation. Four antennae are mounted in a fixed planar configuration, separated from each other by a few feet. The DGPS unit provides the mount's attitude solutions at 5 Hz. CSBF also flies their own DGPS unit to control the attitude of their telemetric system. In the EBEX 2012 flight, one of the flight computers had access to one DGPS unit, while the other flight computer had access to the other DGPS unit. The antenna systems are not mounted on the inner frame so they cannot be used to measure telescope elevation, though they are used for azimuth. They are mounted on top of a gondola structure that primarily exists to support the triangle at the top of the payload. The accuracy of the DGPS systems in predicting their own orientations, given our configurations in which the antennae are mounted 1 m apart, is  $24'$ . However, given that the antennae are mounted on a support structure that can twist by approximately 0.5 in at a lever arm of 4 ft, each system's ability to measure telescope azimuth was predicted to degrade to  $\sim 1^\circ$ .

**Elevation Encoder** - The elevation encoder is a Gurley Model A25S optical encoder that measures the angle between the gondola's inner and outer frames. This can be considered a measure of the elevation angle of the telescope where the sensor accuracy depends on the

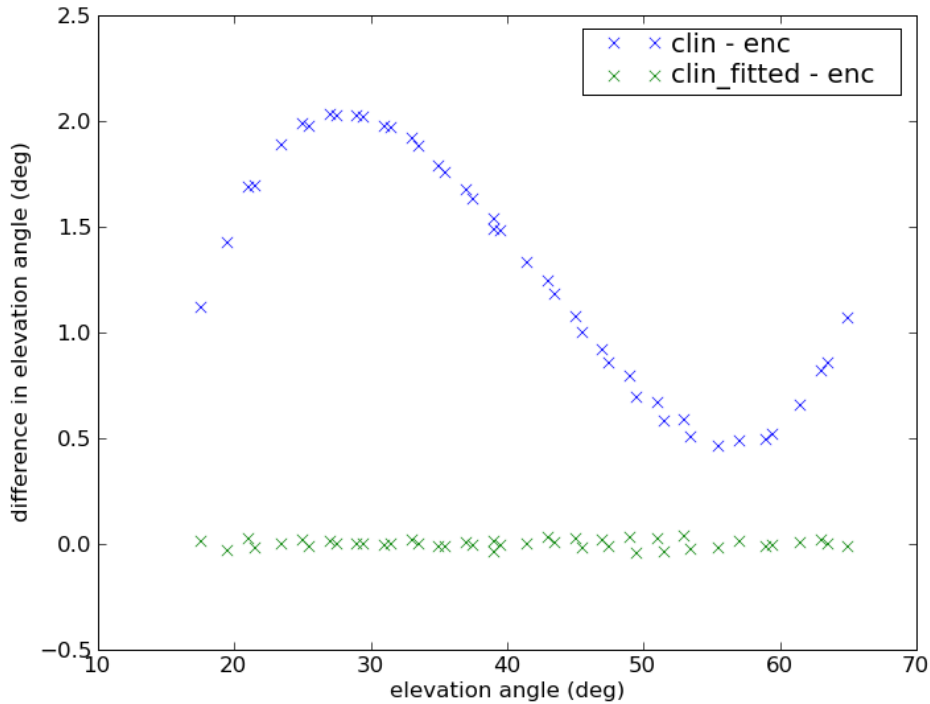


Figure 3.4: Plot created during a clinometer calibration run in 2009. The clinometer exhibits a significant ( $\sim 1.5^\circ$ ) systematic error. The blue data points represent the difference in elevation angles measured by the clinometer and the elevation encoder, as a function of elevation angle (as measured by the encoder). The deviation from a line with slope zero is due to a systematic error in the clinometer. This curve was fitted with a 5th order polynomial and corrected for in FCP. After this correction was implemented, the test was repeated, and the resulting data points in green show that the systematic error was removed to less than  $0.1^\circ$ .

magnitude of gondola pendulations. Although the precision of a stationary reading is  $20''$ , gondola pendulations in flight were expected to degrade the accuracy of the encoder as an absolute pointing sensor to  $\sim 30'$ .

**Clinometer** - The clinometer is a Model 904-T from Applied Geomechanics. It uses a liquid-filled electrolytic transducer to measure two tilt angles that can be used to determine elevation and roll. The clinometer has a significant systematic error that we measured and accounted for in FCP, as shown in Figure 3.4. The accuracy of the clinometer, with this

systematic error corrected for, is roughly 3' when the gondola is stationary. Sloshing of the electrolytic fluid under gondola acceleration was expected to degrade this accuracy to  $\sim 30'$  in flight.

## 3.5 Control Algorithms

### 3.5.1 Attitude Determination Loop

FCP (the *Flight Control Program*) maintains a running pointing solution for each selected axis of each absolute pointing sensor. A running pointing solution consists of an estimate of the current angle of the sensor  $\theta$  (either azimuth or elevation<sup>3</sup>) and an estimate of the variance  $\sigma^2$ . These individual pointing solutions are part of the larger attitude determination loop, and so they are also calculated at 100.16 Hz. An individual pointing solution is the output of a 1-D modified Kalman filter [45] that uses gyroscope data to interpolate between absolute sensor readings when they are available.

At every time step FCP has a gyro measurement,  $\Delta\theta_g$ , with uncertainty  $\sigma_g$ , of how much the gondola rotated since the last timestep:

$$\Delta\theta_g = \omega_g \Delta t$$

$$\sigma_g = 40'' \text{ s}^{-1} \Delta t$$

where  $\omega_g$  is the angular velocity of the gondola about the relevant axis,  $\Delta t$  is the time since the previous timestep, and the number  $40'' \text{ s}^{-1}$  comes from the specifications of the gyroscopes. We can then use this measurement along with the angle from the previous time step ( $\theta_{i-1}$ ) to predict the angle for the current time step ( $\theta_i$ ), but in doing so the error also increases:

---

<sup>3</sup>FCP does not maintain any roll solutions.

$$\theta_i \leftarrow \theta_{i-1} + \Delta\theta_g$$

$$\sigma_i^2 \leftarrow \sigma_{i-1}^2 + \sigma_g^2$$

This is known as the prediction step.

If the absolute sensor has a new reading for this time step then the pointing solution is corrected with the new reading  $\theta'$  (which has uncertainty  $\sigma'$ ) by taking the weighted average of the new reading and the existing angle from the prediction step. The weighted average is calculated as:

$$\theta \leftarrow \frac{\theta \frac{1}{\sigma^2} + \theta' \frac{1}{\sigma'^2}}{\frac{1}{\sigma^2} + \frac{1}{\sigma'^2}}$$

$$\frac{1}{\sigma^2} \leftarrow \frac{1}{\sigma^2} + \frac{1}{\sigma'^2}$$

The consequence of this algorithm is that when no absolute sensor data is available the uncertainty on the solution angle increases with time, and then the uncertainty drops back down when an absolute measurement becomes available. For most of the sensors the measurement rate is the same as the rate of this attitude determination loop, and so the uncertainty converges to a fixed value. In the star camera pointing stream, however, the measurements are only available every 40 seconds and so the uncertainty increases significantly between measurements. This is why the error histograms for the real-time star camera streams in Figure 3.2 are much wider than the star camera accuracy of 1.8'.

Each individual pointing stream also maintains an estimate of the corresponding gyro's bias. Between two timesteps when absolute sensor readings are available, the discrepancy is calculated between how much rotation the absolute sensor detected and how much rotation the gyro detected. This discrepancy is a single measurement of the gyro bias, and each individual pointing stream maintains a running estimate of the gyro bias by passing the individual measurements into 8 stage FIR filters. When the general pointing solution

is calculated, a general estimate of the gyro bias is also calculated. This bias is in turn accounted for when using gyro measurements for the prediction step of each individual pointing solution. In other words, it is accounted for in  $\Delta\theta_g$  above.

For each axis (azimuth and elevation), a general pointing solution is computed by taking the weighted average of every absolute sensor’s individual pointing solution if the sensor is enabled<sup>4</sup>. Before the averaging takes place a calibration angle for each sensor is included so that the general pointing solution represents the pointing of the microwave telescope boresight. The weight for each sensor is the inverse of its pointing stream’s variance.

### 3.5.2 Scan Control Loop

The scan control loop (also running at 100.16 Hz) takes the current attitude solution and a desired scan command as inputs, along with any parameters associated with the scan command. It outputs a requested azimuth velocity and elevation position for every timestep, which it stores in variables that get communicated to the low level control loops discussed in Section 3.5.3.

There are three primary scan commands that EBEX uses: the “cmb”, “calibrator”, and “horizontal” scans. These three scans are essentially different realizations of a single generalized raster scan, and are implemented as such in FCP. The flow of this algorithm is shown in Figure 3.5.

The general raster scan starts at one corner of a rectangle, slews back and forth in azimuth (at a fixed elevation), and then takes a step in elevation. It repeats this process until it reaches the opposite edge of the rectangle. At each end point of an azimuth slew, the gondola pauses long enough<sup>5</sup> for the star cameras to capture images that do not contain significant motion blur.

The general scan takes seven input parameters:

---

<sup>4</sup>During flight, gondola operators have the ability to manually disable sensors.

<sup>5</sup>The pause time is commandable, though a reasonable pause time is 1 s.

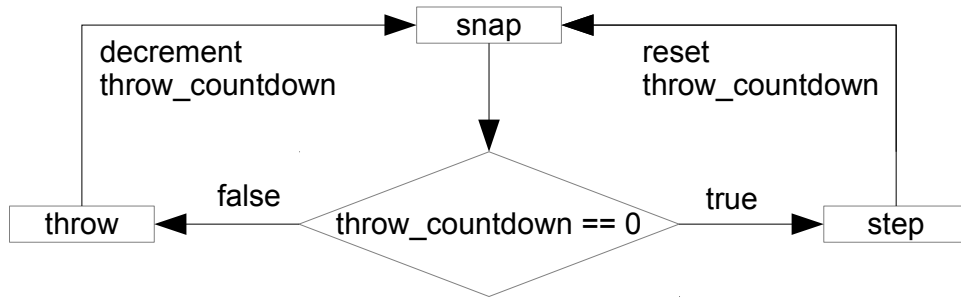


Figure 3.5: A flow chart of the general EBEX scan. The scan can be in one of three states: “snap” (the gondola remains stationary so that the star cameras can capture images), “throw” (the gondola slews in azimuth at a fixed elevation), and “step” (the gondola steps to the next elevation). The 3 different realizations of the scan (“cmb”, “calibrator”, and “horizontal”) perform slightly different actions in the “step” state.

- x-coordinate center (azimuth or right ascension)
- y-coordinate center (elevation or declination)
- azimuth scan speed
- azimuth throw width
- y-coordinate width
- number of elevation steps
- number of azimuth throws in between elevation steps

These seven parameters either come from commands sent from the ground, or from schedule files that are pre-loaded onto the flight computer disks. These are shown in Figure 3.1 as the “uplink receive callbacks” and the “scheduler” boxes, respectively. The scan control loop knows when it is time to move from one state to another based on the current pointing solution, which it receives from the attitude determination loop.

The three different realizations of the generalized scan do slightly different calculations with the input parameters in the “step” state of the scan. This is because the “step” state is where the gondola readjusts itself for the next series of azimuth throws. It is worth noting here that except for the “step” state, all the differences between the three realizations are attributed to the seven input parameters (e.g. “cmb” commands are called with a larger



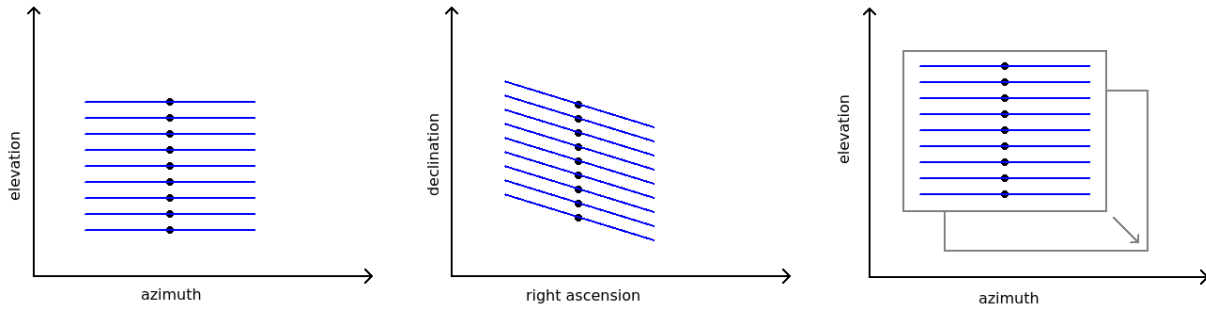


Figure 3.6: Conceptual drawings of the coverage obtained by the horizontal scan (left), cmb scan (center), and calibrator scan (right). The horizontal scan is fundamentally horizontal based (e.g. in azimuth and elevation). The cmb scan is equatorial based, though the throws are still performed at a fixed elevation, resulting in a tilt. The calibrator scan is horizontal based except that it follows a fixed equatorial location.

argument for the azimuth throw width parameter than in a “calibrator” scan call). The differences in the “step” state between the three realizations can be seen in the conceptual drawings of their coverages in Figure 3.6, and are explained in detail here:

- **horizontal scan** - This is the most straightforward scan, though it is not used in flight. It is typically used during ground testing. The steps are evenly spaced in elevation and centered on a fixed azimuth.
- **cmb scan** - This is the scan used to collect scientific data during flight. The steps are evenly spaced in declination and centered on a fixed right ascension. However, as in all scans, the throws are still azimuth based, so the resulting coverage is more similar to a parallelogram than a rectangle. During flight multiple cmb scans are used throughout the day, and at different times of day the horizontal frame crosses the equatorial frame at different angles. This results in approximately rectangular net coverage, and has the added benefit of creating cross-linking within the patch, as shown in Figure 3.7. The cross-linked coverage pattern is important in minimizing systematic errors.
- **calibrator scan** - The calibrator scan can functionally be thought of as a combination of the horizontal and cmb scans. The purpose of the calibrator scan is to scan across a

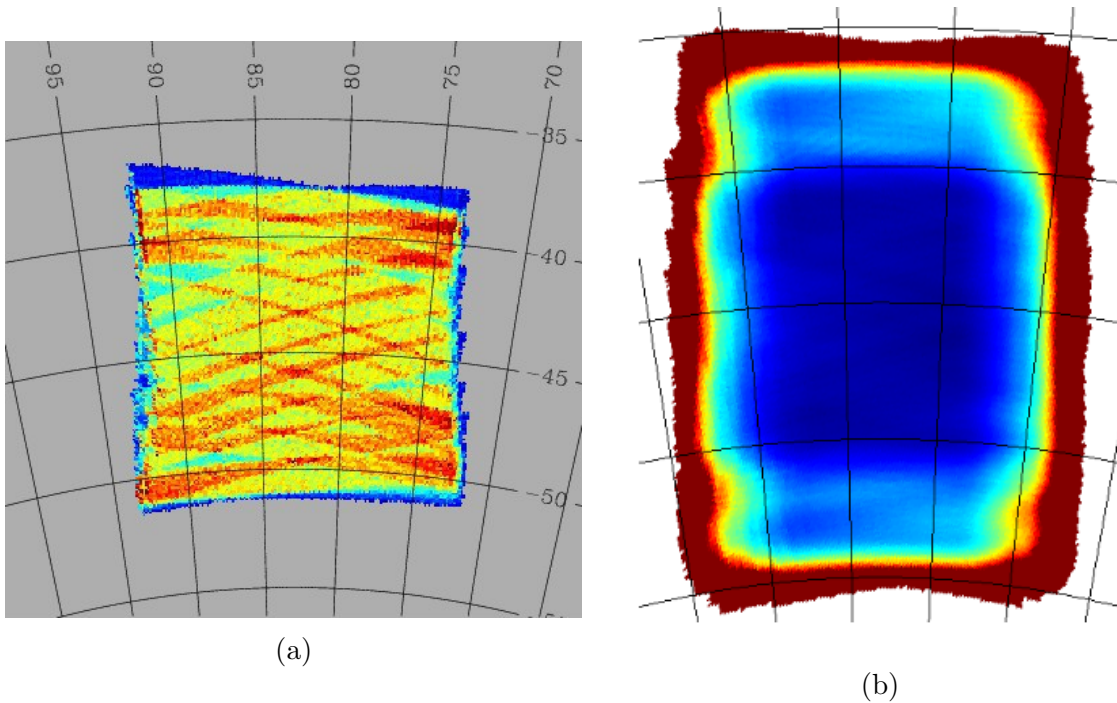


Figure 3.7: Simulated coverage plots for the cmb scan. Subfigure (a) shows the resulting coverage in equatorial coordinates from 1 day of flight, with red representing more coverage. The cross-linking that results from performing multiple scans throughout the day can be seen, as well as the resulting rectangular shape. Subfigure (b) shows the resulting coverage from 11 days of flight for all 150 GHz detectors, with blue representing more coverage. Over the course of an entire flight the coverage becomes more even.

point source in the sky (e.g. RCW 38) so that the entire EBEX focal plane is exposed to it. As a result the center of the scan must follow a fixed set of equatorial coordinates (right ascension and declination). However, the purpose is to raster scan a projection of the EBEX focal plane on the sky, which is ultimately horizontal based (cross-elevation and elevation) rather than equatorial based like the cmb scan. As a result, except for the fact that the center follows a fixed equatorial point, the steps would be evenly spaced in elevation and centered on a fixed azimuth.

### 3.5.3 Low Level Control Loops

Digital signal processors (DSPs) on the ACS cards translate the requested azimuth velocity and elevation position from the flight computer into power level requests for the motor controllers.

The DSP on ACS card 2 converts the requested elevation position, which it receives from FCP, into a requested power level for the elevation motor. This requested power level is sent as a pulse-width modulated (PWM) signal to the elevation motor's controller. The DSP uses a proportional-integral-derivative (PID) feedback loop to calculate the output signal, though the derivative term is not implemented. The error term in the PID loop is the difference between the requested position and the actual position reported by the elevation encoder, to which ACS card 2 has direct access, as shown in Figure 3.1.

The DSP on ACS card 1 is responsible for the reaction wheel and pivot motors. The error term for the reaction wheel motor is the difference between the requested azimuth velocity and the actual velocity as measured by the gyros. The error term for the pivot motor is the difference between the reaction wheel velocity and a set point, which is close to zero. In this manner the reaction wheel attempts to accelerate the gondola to the requested azimuth velocity through conservation of angular momentum, and the pivot prevents the reaction wheel from saturating by acting to keep it at the set point, which the pivot achieves by torquing against the flight train.

\*

\* \*

In the next two chapters we will dive into the details of the star cameras, the primary absolute pointing sensors, because I was responsible for ensuring their success.

# Chapter 4

## Star Cameras

### 4.1 Introduction

The star cameras operate by capturing images of the sky and identifying stellar patterns that can be associated with known stars in the field of view. We refer to the process of finding a pointing solution with the star cameras as “solving”. The precision of a star camera solution is primarily determined by the specifications of the physical camera. The EBEX star cameras are designed to have a precision of 1.8'' in cross-declination, 1.8'' in declination, and 67'' in roll<sup>1</sup>. The two star cameras are named Star Camera 0 and Star Camera 1, with short names XSC0 and XSC1.

Each star camera consists of a pressurized vessel that contains a digital camera, an embedded computer, a hard disk, and various supporting electronics. The front of the vessel contains a window, and an optical baffle is mounted to the vessel in front of the window to limit stray light, reducing image noise. The EBEX flight computers control the camera triggering so that the timing of each image can be precisely linked with other sensor data recorded by the flight computers. The *Star Tracking Attitude Reconstruction Software* (STARS) is a software package custom-designed for EBEX that runs on the embedded com-

---

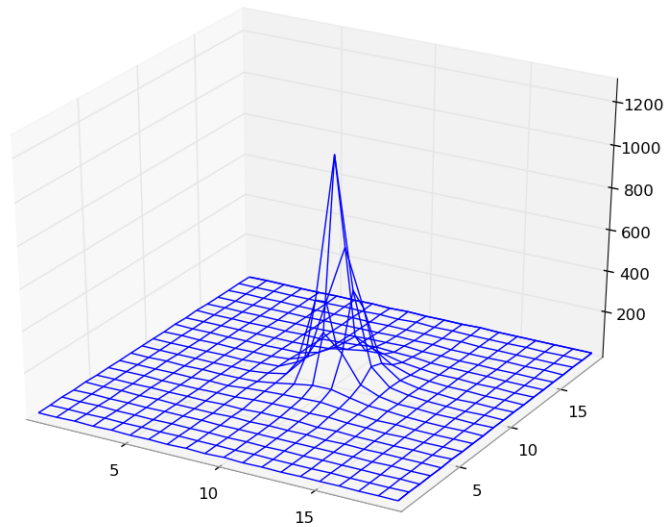
<sup>1</sup>This is the precision for solutions that contain 4 stars. Solutions that contain more than 4 stars will have higher precision.

puters. STARS downloads images from the camera controller when they become available and stores them to disk, then processes the images to find pointing solutions, and serves these solutions to the EBEX flight computers. STARS is described in more detail in Chapter 5.

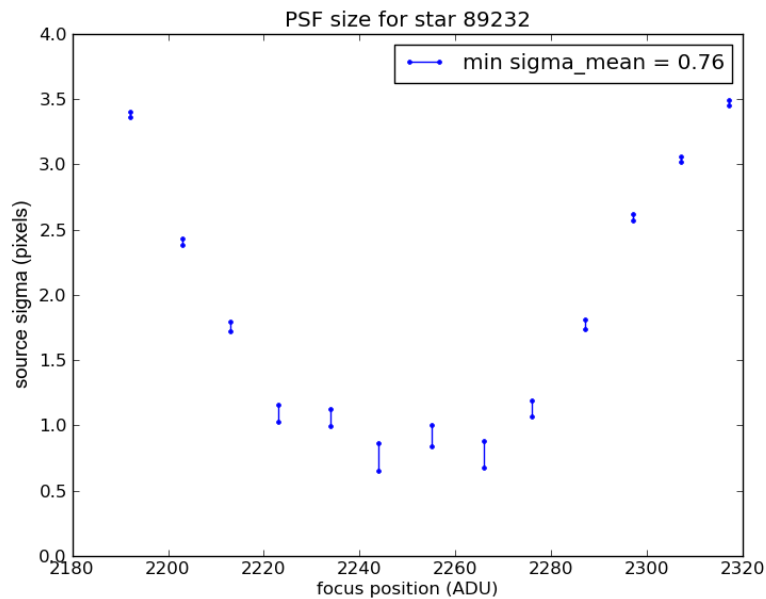
The star cameras use two forms of downlink telemetry available in flight so that ground operators can monitor their states and make corrections when appropriate. First, making use of line-of-sight communications available during the first day of flight, the star cameras transmit their computers' VGA (Video Graphics Array) outputs over NTSC (National Television System Committee) video downlinks. To make use of this capability, STARS is designed to display visual state information to the VGA display. Secondly, the star cameras share 44 status variables with the EBEX *Flight Control Program* (FCP), which FCP may then downlink with other numerical data streams. FCP decides whether to downlink each of these data streams at any given time throughout the flight depending on the bandwidth available and the priority of each data stream as defined by the ground operators.

The two star cameras share nearly identical hardware components:

- **Lens** - The lenses are Canon EF 200 mm F/1.8 L USM lenses, which have 11.1 cm diameter apertures. A larger aperture, though more expensive, allows the camera to collect more light, and thus obtain higher signal-to-noise sources (stars) for a given exposure time. This is used to decrease the exposure time and corresponding motion blur. A larger aperture can also produce images with higher angular resolution. The angular resolution of the EBEX star cameras have been measured to be  $\leq 20''$  (Rayleigh criterion) by observing stars from the ground, which is an upper limit as the test may have been limited by astronomical seeing (see Figure 4.1). The diffraction limit for these specifications is  $\sim 1.6''$ .
- **Camera** - The cameras are Redlake Megaplus II 1603 cameras which contain Kodak KAF-1603E CCD image sensors and support IEEE 1394 readout. The CCDs contain  $1536 \times 1024$  square pixels that are  $9 \mu\text{m}$  on a side. The pixels have well depths of



(a)



(b)

Figure 4.1: Subfigure (a) shows a 3-D plot of a small region of an image centered on a star captured by Star Camera 0. The image was captured on the ground, with a near optimal focus position. As a star is effectively a point source, this is a measure of the point spread function (PSF). We measure the width of a star by finding the  $\sigma$  of a best-fit Gaussian. Subfigure (b) shows the width of this star as a function of the camera's focus position. A  $\sigma$  of  $\sim 0.75$  px is found at the optimal focus position, and corresponds to an angular resolution of  $\sim 20''$  by the Rayleigh criterion.

100 000 e<sup>-</sup> and are read out with 12 bits of dynamic range. A deeper well, though more expensive, allows the camera to collect more light before saturating and thus obtain higher signal-to-noise sources. The pixel size coupled with the lens focal length results in a platescale of 1 px/9.5". The CCD dimensions coupled with the lens focal length results in a field of view of 4.05° × 2.70°. A larger field of view allows the camera to find more stars, which increases the probability of finding a solution and can decrease the uncertainties associated with the solution angles. The trade-off, however, is that with a larger field of view each pixel covers a larger area of the sky, and therefore reduces the precision with which the software can measure the locations of stars. The CCD is primarily sensitive to visible frequencies of light. The quantum efficiency of the CCD is shown in Figure 4.2. The cameras support four buffers that can be stored before being flushed through the readout, which allows the star cameras to capture multiple exposures in rapid succession. Each camera consists of two physical components connected by a tether cable: a camera head, which houses the CCD, and a controller, which stores the buffers for readout.

- **Red Filter** - A Hoya 25A red color filter is placed between the lens and the camera to block out light with wavelengths shorter than 600 nm. This blocks out a significant amount of the atmosphere (blue sky), which is the dominant source of noise in the images, while blocking out proportionally less star light, which tends to be redder than the atmosphere [46]. The transmission function of this red filter is also shown in Figure 4.2.
- **Lens Controller Interface** - The lenses use Birger<sup>2</sup> Canon EF 115.0 lens mounts to control the focus and aperture positions electronically with STARS via USB.
- **Computer** - The star cameras contain PC/104 computers from Advanced Digital Logic. Star Camera 1 contains a more recent model (ADL855PC-373C-G5) than Star Camera 0's (MSM855-C373). Both computers have single threaded 1 GHz processors,

---

<sup>2</sup>Birger Engineering is located in Boston, Massachusetts 02111.

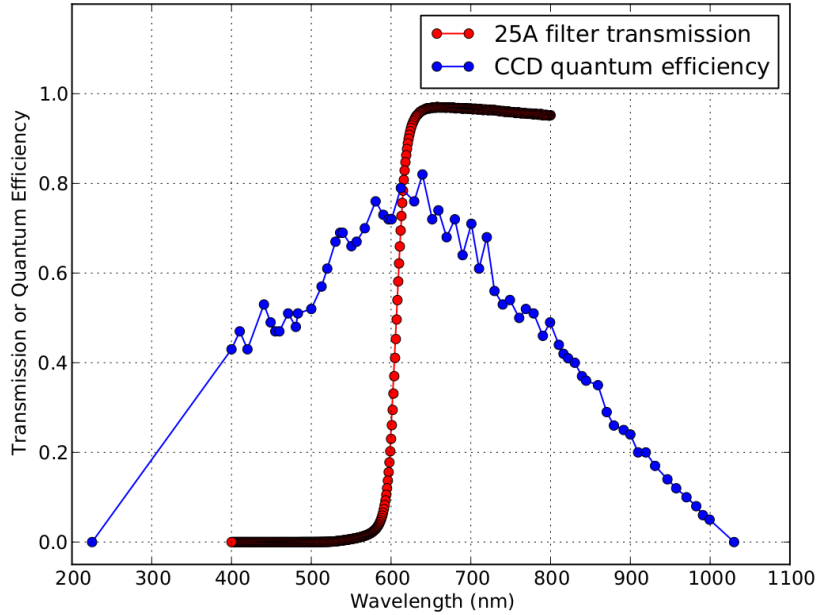


Figure 4.2: The quantum efficiency of the CCD and the transmission function of the red filter used in the star cameras. Figure courtesy of Yury Vinokurov.

1 GB of RAM, and two-piece heat sinks with fans, and are rated to 70°F. There is a trade-off between power consumption and processor speed, and we chose the most powerful processor that our thermal environment can tolerate.

- **IEEE 1394 Board** - The computers mate with MSMW104+ PC/104 firewire boards for camera readout.
- **Hard Drives** - The star cameras contain hard drives to store the operating system, flight software (including a 3.6 GB star catalog), and the images collected during flight. All the images captured in flight are stored to disk in case additional information needs to be extract from them after the flight. We predicted before flight that each star camera may require up to 418 GB of space for storing the images (see Section 4.4.4). Star Camera 0 flew a 500 GB Seagate ST9500325AS and Star Camera 1 flew a 750 GB Western Digital Scorpio Blue WD7500BPVT. Both star cameras contain ADSAIDE44 SATA-IDE converters to allow the computers to communicate with the modern high



volume SATA drives.

- **VGA to NTSC Converters** - The star cameras contain converter boards (QVGA2TV Videosecu PC to TV Converter) to convert the VGA output of the computers into NTSC signals for video downlink during line-of-sight communication. STARS displays the images it captures along with an abundance of debugging information to the screen at  $640 \times 480$  resolution so that it can be viewed clearly by the ground operators.
- **Environmental Sensors** - The star cameras contain AD590 temperature sensors, MPX4250A pressure sensors, and Measurement Computing USB-1208LS digital-to-analog converters in order to monitor the environment inside the star camera vessels.
- **DC-to-DC Converters** - The star cameras contain DC-to-DC converters to convert the +28 V input voltage, supplied from the gondola's power crate, into +5 V and +12 V output to meet the power needs of all the electrical components. The converters are Calnex 24S5.20HEW and 24S12.8HEW converters for 5 V and 12 V, respectively.

The mechanical shutters in the digital cameras are opened and closed by separate electronic trigger lines, which are controlled by the *Flight Control Program* (FCP) on the flight computers and fed directly into the star cameras. FCP records this trigger line in an ACS data stream, and STARS reads out new images from the camera controller buffers as soon as they become available. This arrangement allows the timing between the star camera images and the rest of the pointing sensor data to be determined precisely, assuming that the image recorded by the star camera software can be linked to a pulse in the trigger line. Linking an image to a trigger pulse is non-trivial because STARS runs asynchronously from FCP. Both FCP and STARS keep a running counter of how many images they have triggered/captured, and they each share their counter with the other system over the network so that each program has access to both counters. FCP tags each pulse with the two counters, and STARS tags each image with the two counters. If there is enough time for both systems to increment and share their counters after an exposure<sup>3</sup>, then for the following exposure the image tags

---

<sup>3</sup>This requires about 1 second.

will match the pulse tags.

The first star camera, Star Camera 0, was built at Brown University and flown in the EBEX 2009 test flight. Details about the design and construction of this star camera are discussed elsewhere [47]. The second star camera, Star Camera 1, was designed and built at Columbia University. Both star cameras were flown in the EBEX 2012 Antarctic flight. We discuss the design and construction of Star Camera 1 here.

## 4.2 Star Camera 1

Star Camera 1 uses the same components as Star Camera 0 and therefore has nearly identical specifications, but the vessel, internal structure, and internal layout were designed differently in an effort to improve upon the original design. In particular the redesign focused on robustness and on making the internal components more accessible upon partial disassembly. Photographs and 3-D models of XSC1 are shown in Figures 4.3, 4.4, 4.5, 4.6, and 4.7.

### 4.2.1 Design

Figures 4.3 and 4.4 show a model of the design along with photographs of the internal structure that supports all of the components inside the star camera. This structure consists of four G-10 rods connected by four circular flanges placed at various positions along the rods. The lens is mounted near the front of this structure, and the camera head is mounted to the lens and also to one of the flanges. An aluminum plate, named the “electronics plate”, is mounted to the back of the structure. Mounted to the top of the electronics plate are the computer, hard drive, DAQ, and housekeeping breakout board. Mounted underneath the plate is the camera controller, and attached to the bottom of the camera controller is an aluminum plate that houses four terminal blocks for distributing power to all the electrical components (see Figure 4.6c).

The internal structure fits inside of a pressure vessel. The pressure vessel essentially

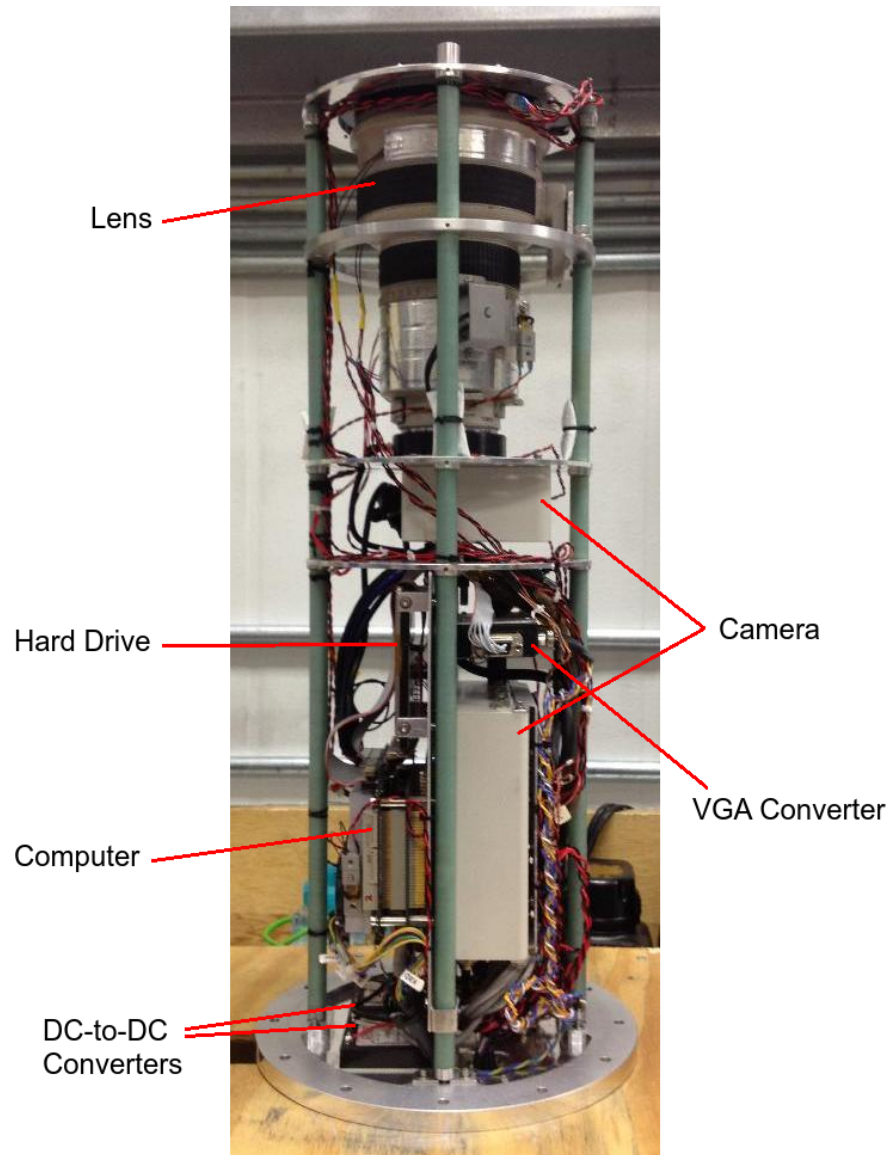


Figure 4.3: Star Camera 1 components and internal structure.

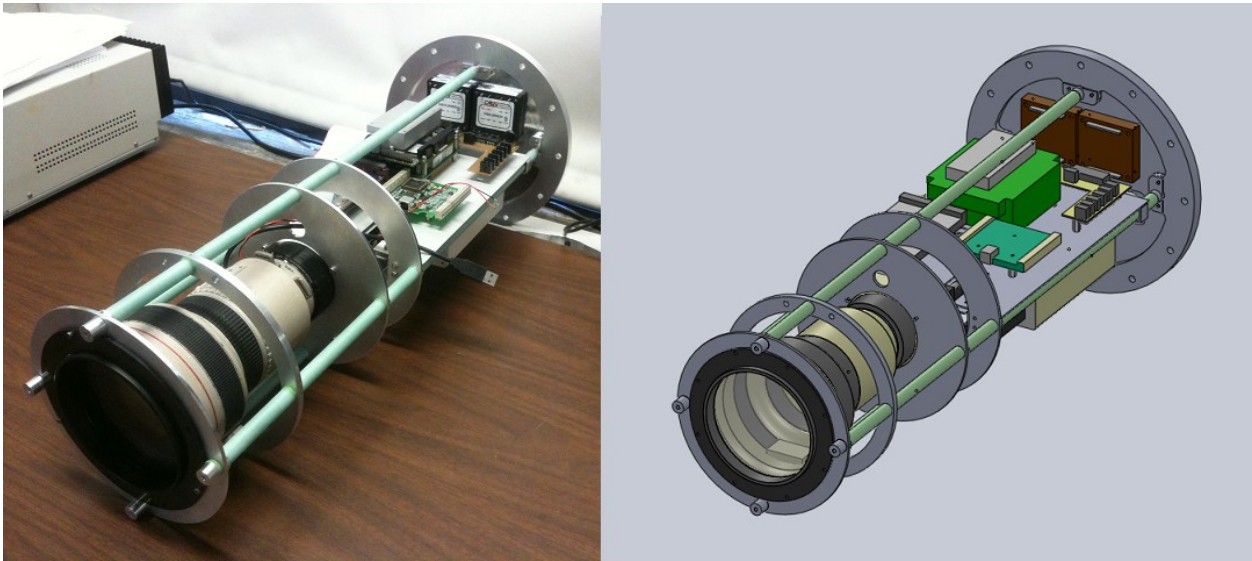


Figure 4.4: Side by side comparison of the actual and modeled XSC1 internal components, structure, and back flange.

Fascening locations for the internal structure

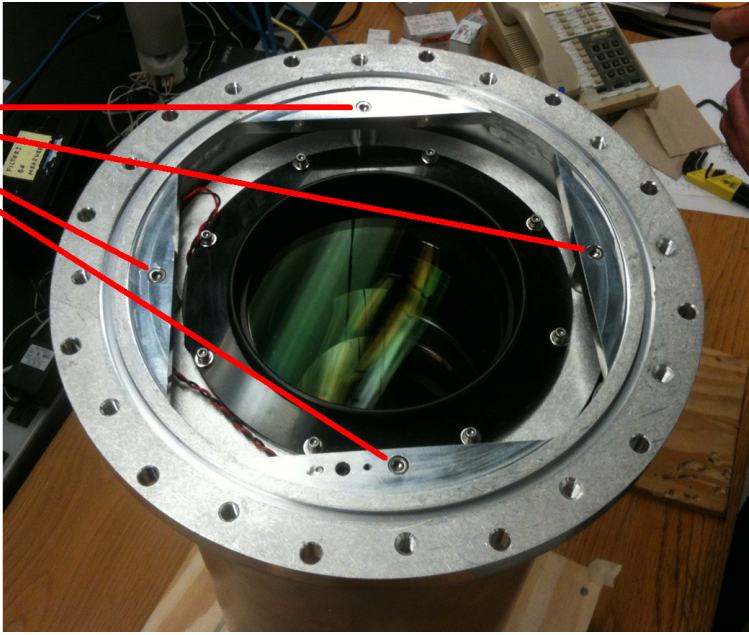


Figure 4.5: The front of XSC1 without the front flange. The front weld flange jets inward about an inch to form four perpendicular edges, allowing the four rods of the internal structure to be fastened directly to the front weld flange.

consist of a hollow aluminum tube with open flanges welded to each end, known as the “weld flanges”, and is secured closed with O-rings and a closed flange on each end: a “front flange” and “back flange”. The front flange contains the window. The internal support structures is placed inside the pressure vessel and fixed to it<sup>4</sup>. The internal structure and some of the components of Star Camera 1 are shown in Figure 4.3. The part of the internal structure to which the digital camera (lens and camera head) is mounted is screwed directly into the front of the pressure vessel. This design helps prevent the pointing angle of the digital camera from shifting with respect to the vessel during flight. The front of Star Camera 1 is shown in Figure 4.5. The vessel is pressurized to roughly 1 atm to protect the computer, lens, and hard drive from the low stratospheric pressure, and dry nitrogen is selected for this pressurization to help prevent condensation on optical components when traveling through the coldest regions in the tropopause.

The window in the front flange is partially inset on the inner side of the front flange. An O-ring sits in a groove in the front flange between the front flange and the window. An aluminum ring is screwed into the front flange around the window to press it onto the O-ring. Placing the window on the inside of the front flange is favorable given the direction of the pressure differential at float, and allows the ring that holds it in place to be relatively thin.

Two DC-to-DC converters are mounted to the back flange. The back flange contains three hermetic electrical connectors for communications between the star camera and the rest of the gondola. An MS3114H-14-12P 12 pin and an MS 3114H-14-19P 19 pin connector pass power for the DC-to-DC converters, power for the heaters, ethernet lines, the trigger line that drives the camera shutter, Video Graphics Array (VGA) lines, USB lines, and keyboard lines. A TNC connector passes the NTSC video signal. Documentation for the connector pinouts is found in Appendix C.

---

<sup>4</sup>The DC-to-DC converters are mounted directly to the back flange.

## 4.2.2 Design Principles

The principles that guided the design are:

- **Vessel** - The camera needs to be contained in a pressurized vessel to maintain atmospheric pressure while in a space-like environment. Maintaining pressure is necessary for the hard drives to work, for the computer to work (though that the computer would fail under vacuum or rapid depressurization was not clear to us until it was determined empirically), and may or may not be necessary for the lens to work correctly. Atmospheric pressure also allows for thermal convection, which carries heat from the electronic components to the vessel wall.
- **Single Internal Structure** - All the internal components should be mounted to a single fixed structure, as shown in Figure 4.6. The motivation for using a single structure design is that the dual design of XSC0 is more difficult to disassemble, primarily because it requires frequently disconnecting and reconnecting multiple hard-to-reach electrical cables, including the camera head tether cable which is particularly sensitive.
- **Fixed Camera** - A design requirement stemming directly from the pointing requirements is that the lens and camera head combination be fixed relative to the vessel so that the pointing angle of the camera does not vary by more than  $\sim 1''$ . In order to simultaneously meet this requirement and the single structure requirement, the front of the internal structure, the part closest to the lens and camera head, is designed to screw directly into the front of the vessel. To not overconstrain the design, the back of the internal structure sits freely inside of gaps in the back of the vessel. This is important given the different coefficients of thermal expansion between the G-10 rods and the aluminum vessel. A photograph of the front attachment is shown in Figure 4.5.
- **Identical Electrical Interface to XSC0** - The two star cameras are intended to be functionally equivalent in order to minimize confusion and minimize the number of spare parts required in the field. This includes having nearly identical electrical

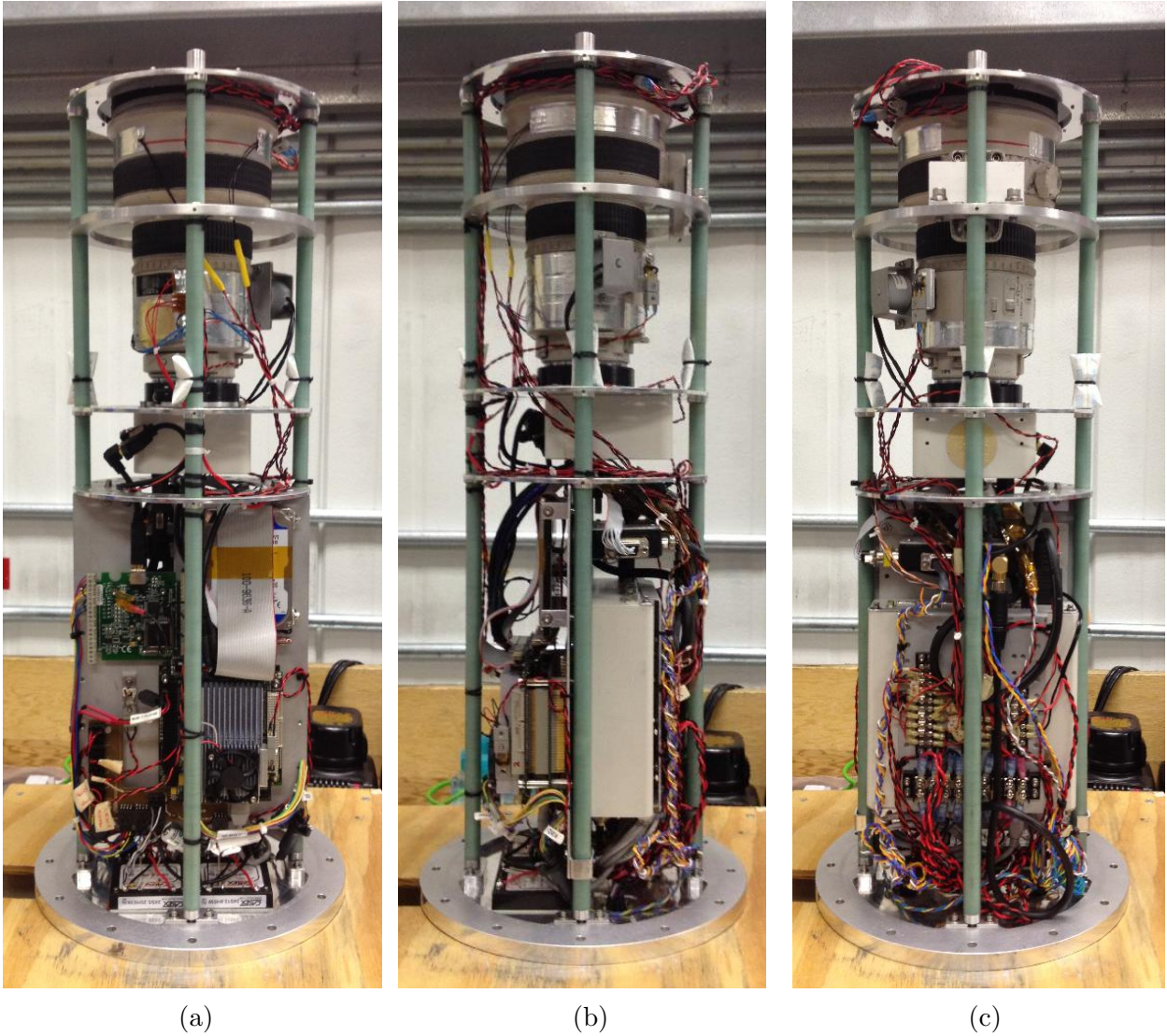


Figure 4.6: (a) Top, (b) side, and (c) bottom view photographs of the Star Camera 1 internals. In the bottom view, the wires converging near the bottom are connected to four terminal blocks that are mounted on an aluminum shelf that is mounted to the underside of the camera controller. A diagram of the terminal blocks labeling the wires is shown in Appendix C.

interfaces.

With the resulting design, XSC1 can be disassembled by unscrewing the screws on the back flange and front flange, and then sliding the vessel off the top. This leaves the internals standing upright on the back flange in a single piece. The full assembly procedures for both star cameras are listed in Appendix B.

### 4.2.3 Construction

XSC1 was constructed at Columbia University. Most metallic parts were machined elsewhere and shipped to Columbia. The G-10 rods were glued to the circular support flanges, and the electronics plate was screwed to the support flanges, completing the support structure. The electronic components were then mounted to it, completing the internal structure. Once the internal structure was assembled and attached to the back flange, all the internal wires were connected. A block wiring diagram is available in Appendix C. Special care is taken with the camera tether cable, which connects the camera head to the camera controller. To allow this cable to maintain a large turning radius, a section was designed to be cut out of the otherwise rectangular electronics plate. The vessel was then installed according to the assembly procedure.

Figure 4.7 shows a section view of a 3-D model of XSC1 in its entirety (save the wiring, which was not included in the 3-D model), including its optical baffle which will be discussed in the Section 4.3.

### 4.2.4 Thermal Consideration

On a high altitude balloon platform one is often concerned with both extremes of the thermal environment. In order to reach a float altitude of  $\sim 120\,000$  ft, the payload ascends through the troposphere and tropopause, and into the stratosphere. The tropopause presents the cold extreme, in which temperatures tend to be around  $-60^\circ\text{C}$  and the atmosphere is dense



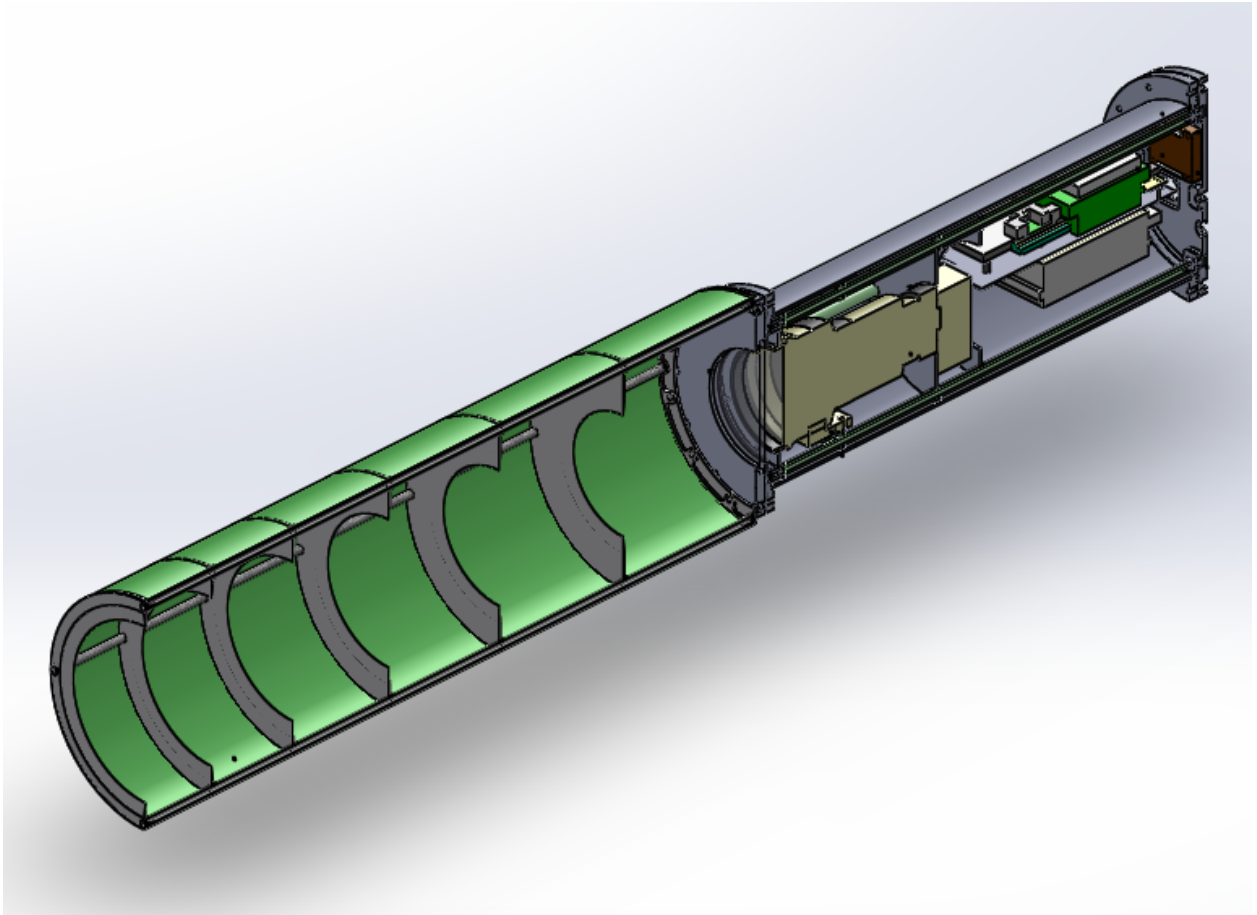


Figure 4.7: Section view of Solidworks model of Star Camera 1 with optical baffle.

enough to convect heat away efficiently. Once the payload reaches float altitude, however, the atmosphere is too thin for efficient convection, and electronic components that generate heat must dissipate their heat solely through radiation and conduction to the gondola, which presents a challenge in the hot extreme.

Following the conclusions of thermal analysis, we painted the star cameras white to prevent overheating at float altitudes, and outfitted each star camera with heaters capable of providing 60 W of additional power so they would remain sufficiently warm during ascent.

XSC1 contains six 10 W heaters: two on the front flange of the vessel (near the window), and four on the lens. Without the heaters, the lens and window are at risk of becoming too cold. The concern is that condensation may form on the window and that the mechanical lens components that drive the focus and aperture may freeze. In addition, these components are near the front of the vessel, whereas the electronic components that continuously generate heat are all located near the back.

The six heaters are powered directly from the power crates on the gondola, bypassing the DC-to-DC converters in the star cameras. As shown in the XSC1 Heater Wiring Diagram in Appendix C, there are two distinct 24 V lines provided to the star cameras for this purpose:

- One of the power lines can be switched on and off in software by FCP. We implemented two software modes to control this switch: manual and automatic. In manual mode, ground operators can manually command the heaters to turn on or off. In automatic mode, the heaters are activated on a commandable set point. FCP compares the set point to the actual star camera temperatures that are recorded by temperature sensors inside the vessels, low pass filtered by STARS, and communicated by STARS to FCP.
- The other power line is always on, however inside the star camera vessels it passes through a bang-bang controller that activates at  $-20^{\circ}\text{C}$  and deactivates at  $-10^{\circ}\text{C}$ . This serves as a parallel backup system to the software control.

The gas inside the vessel enables the redistribution of heat throughout the star camera,

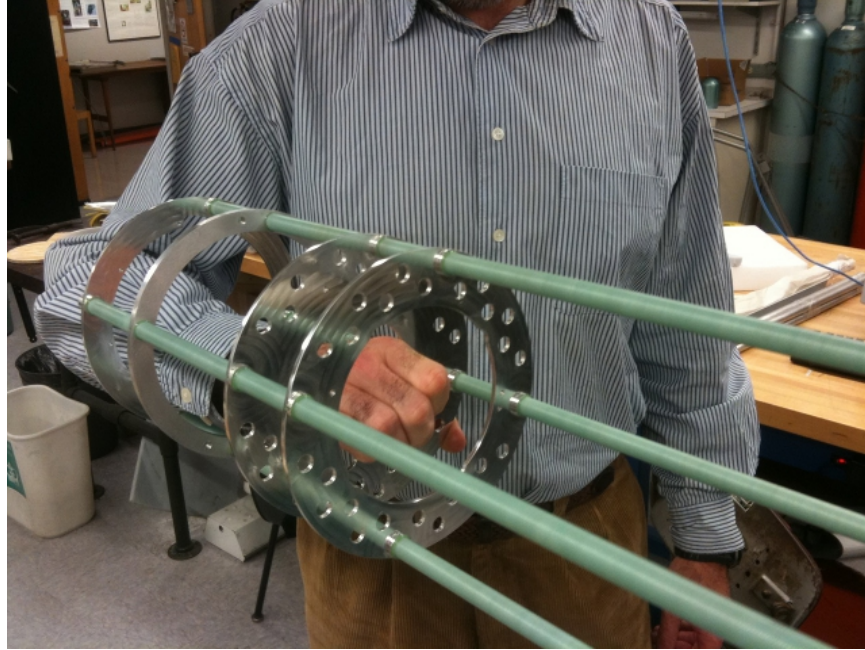


Figure 4.8: Holes in the internal support structure's rings enable heat convection from the back of the star camera to the front of the star camera.

and special care was given to the components most susceptible to temperature extremes. The lens is supported by G-10 rods in order to take advantage of G-10's low thermal conductivity<sup>5</sup> to insulate it from the vessel so that it would not freeze during ascent. A heat sink with a fan is installed on the computer to transfer heat away from the computer at float altitudes. In order to improve the convection of heat from the back of the star camera, where most of the electronics are located, to the front, where the lens and window are located, there are numerous holes in the aluminum rings that would otherwise impede this flow. These holes are shown in Figure 4.8.

Both star cameras were tested in a thermal vacuum chamber provided by CSBF to simulate ascent and float conditions. The star cameras were successful in the 2012 Antarctic flight, and did not underheat or overheat. The heaters were set to activate based on a commandable set point, and did so repeatedly during ascent. The star camera temperatures

---

<sup>5</sup>G-10's thermal conductivity is approximately  $1 \text{ W m}^{-1} \text{ K}^{-1}$  vs aluminum's  $237 \text{ W m}^{-1} \text{ K}^{-1}$ .

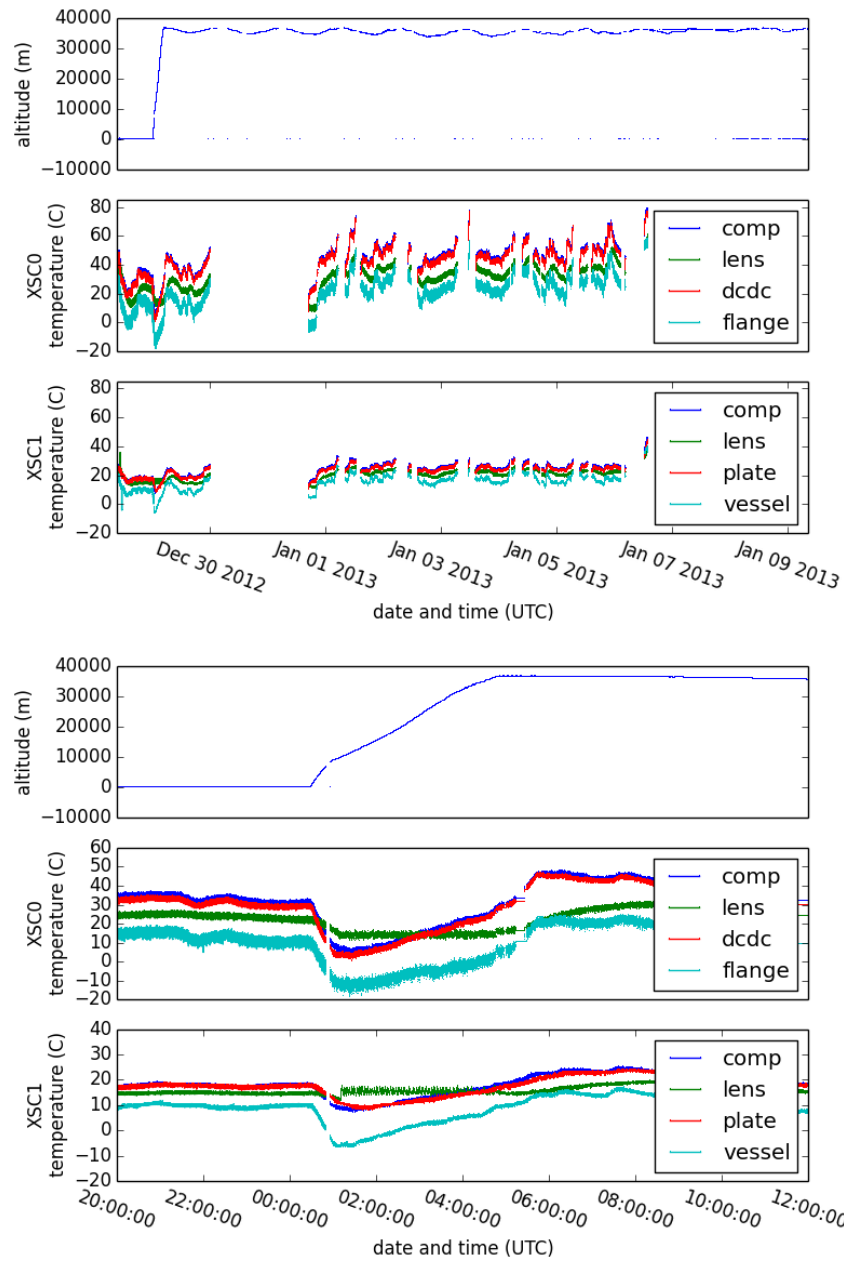


Figure 4.9: The star camera temperatures during the 2012 Antarctic flight. Each star camera has four temperature sensors placed in different locations inside the pressure vessel. The sensors are mounted near the computer (comp), lens (lens), DC-to-DC converters (dcdc), front flange near the window (flange), XSC1 electronics plate (plate), and/or the front of the vessel near the window (vessel). Each star camera has four of these possibilities. Shown are the temperatures for the entire flight (top) and a zoom of the temperatures during ascent (bottom).

during flight, including ascent, are shown in Figure 4.9.

### 4.3 Optical Baffles

In order to minimize the optical loading in the star camera images, an optical baffle is mounted to the front of each star camera to block out stray light. Each baffle is a hollow tube that is painted flat matte black on the inside to reduce reflections. It contains a series of vanes, which are knife-edge circular components that protrude inwards to help block reflections. The placement and inner radii of the vanes are chosen in such a way that no part of the vessel window can directly see the inner surface of the tube<sup>6</sup>.

The optical design drawing and a cross-section of the 3-D model of the EBEX baffle are shown in Figure 4.10. The optical design drawing labels the length of the tube and the position and heights of each vane, and also shows the edges of the star camera field of view (expanding at  $2.4^\circ$ ) and the baffle's rejection angle of  $11.7^\circ$ . The rejection angle is the largest angle by which light can enter directly into the vessel window, and is larger than the field angle (the angle that the field of view expands at) because it can cross diagonally from one side of the baffle to the other side of the window. Note in this figure that the heights of the vanes are defined so that they do not interfere with the field angle.

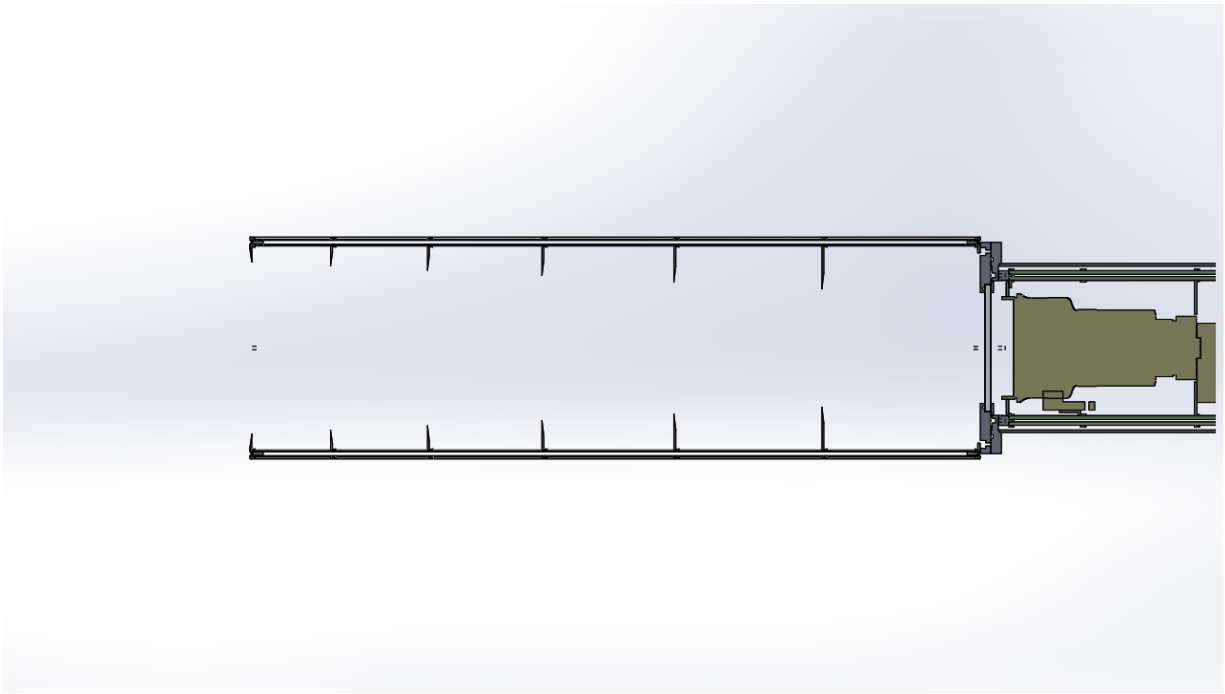
The longer the baffle, the smaller the rejection angle, so a longer baffle is preferred<sup>7</sup>. In the EBEX case the length of the baffle is limited to 34.5 in by its mount location on the gondola. Given a specified length, the baffle must be wide enough to at least accommodate the field angle, though wider is better. As we will explain in this section, a wider baffle allows for deeper wells between vanes, and therefore fewer vanes, and fewer vanes are preferred. In the EBEX case the inner radius of the baffle is limited to 4.755 in by the distance between the star cameras and the gondola.

Figure 4.11 shows the logic behind the placements of the vanes. The vanes are placed so

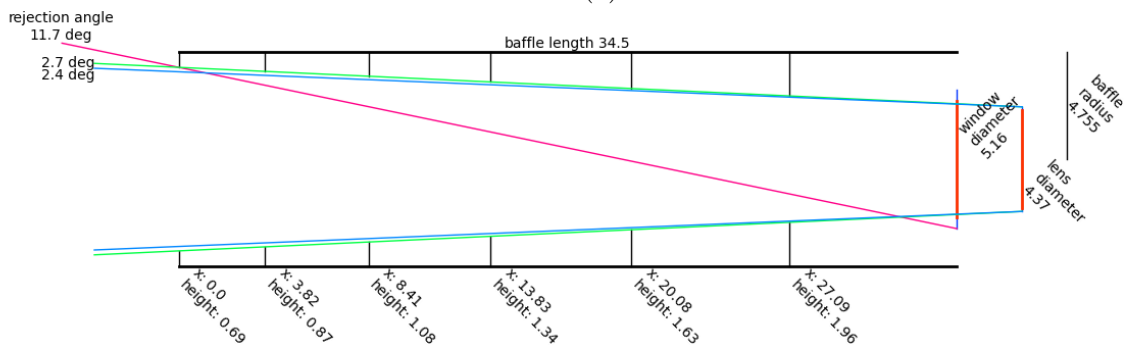
---

<sup>6</sup>For more information about baffle design, see [48].

<sup>7</sup>Note that in the limiting case of an infinitely long baffle, the rejection angle approaches the field angle.



(a)



(b)

Figure 4.10: (a) Cross section of a 3-D model of the star camera baffle and (b) optical drawing with specifications. In the drawing the black lines represent the baffle itself, which is mounted to the front of the star camera vessel, flush with the vessel window. The star camera lens sits a couple inches behind the window. The position of the lens differs in the two subfigures because subfigure (a) shows xsc1, while the baffle is designed for the more restrictive lens position which is the lens position in xsc0. Linear distances are in inches. In the 3-D model the vanes have finite thickness and are beveled at the inner edge.

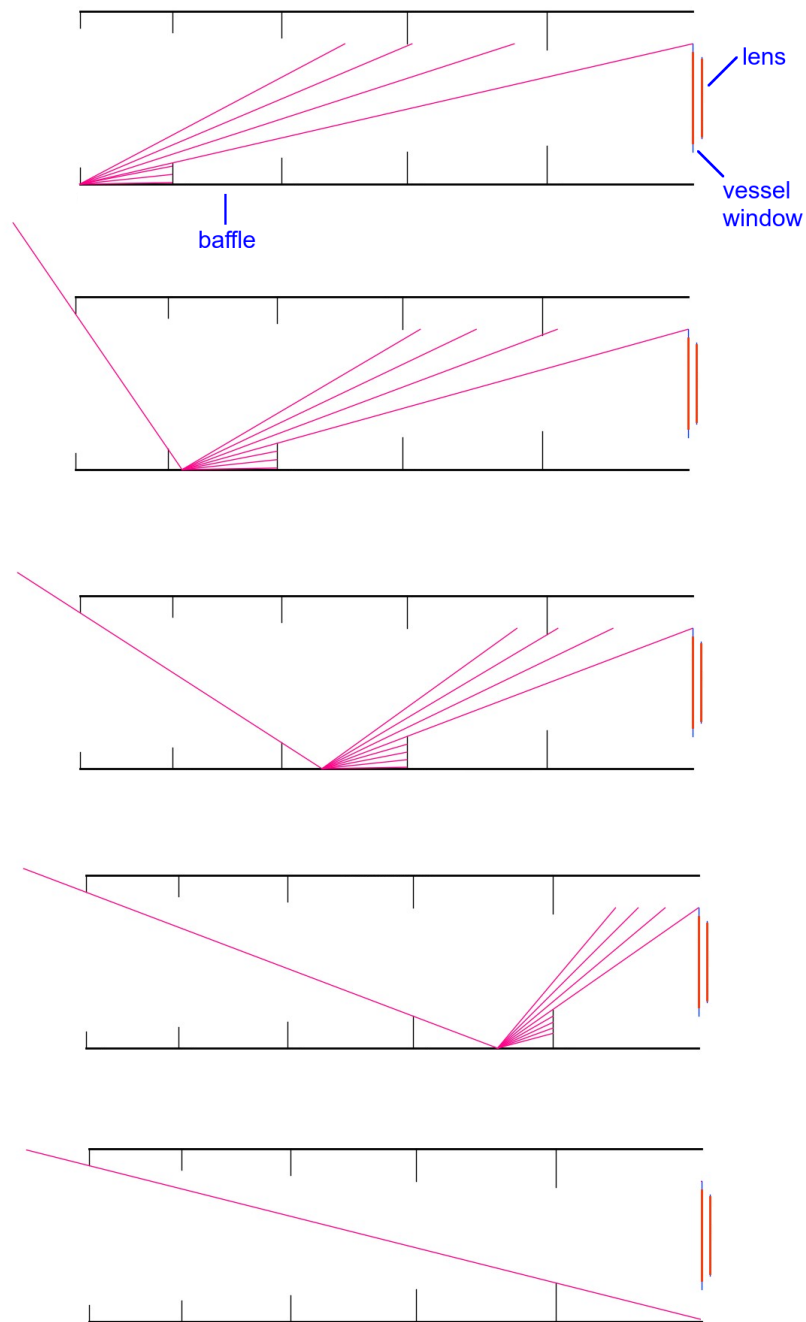


Figure 4.11: Diagram showing how the positions and heights of the vanes are defined to prevent light from reflecting off the tube directly into the vessel window. The procedure is described in detail in Section 4.3.

that no light can reflect off the inside of the tube directly into the vessel window. As the vanes have a finite thickness, however, light can reflect off them directly into the window. We therefore bevel the inner edges of the vanes to limit their surface area, and we minimize the number of vanes by placing them as far apart as possible. In Figure 4.11, the optical drawing is shown where light enters the baffle from the left and the vessel is located on the right. The first vane is placed at the left edge of the baffle. The second vane is placed in such a way that it blocks light that reflects from just inside the baffle behind the first vane, as shown in the top figure. The third vane blocks light that enters in such a way that it barely misses the second vane, as shown in the second figure from the top. This procedure is repeated until the next vane would be placed beyond the baffle. Note that since the position of a vane depends on its height, and the height depends on its position, the position and height must be solved for simultaneously.

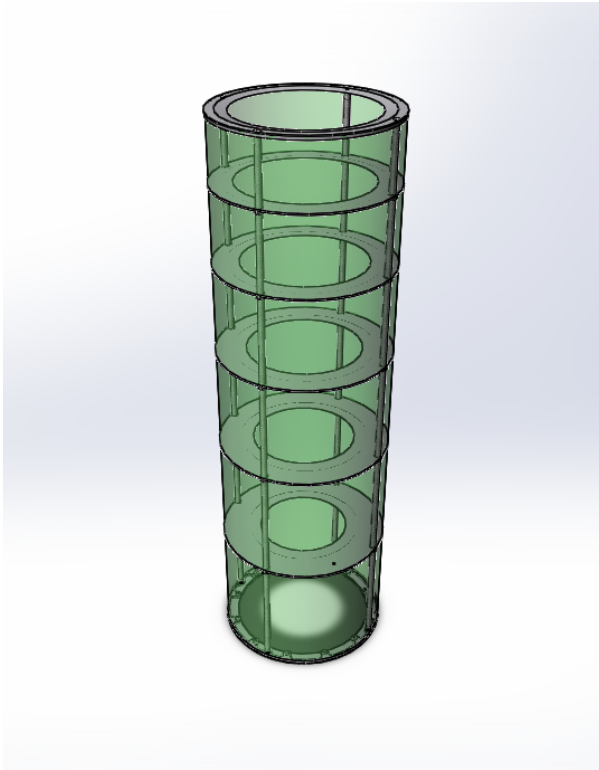
The 3-D model of the optical baffle is shown in Figures 4.10a and 4.7 and 4.12a. The EBEX baffles are made of carbon fiber rods, thin sheets of G-10 fiberglass, and thin black anodized aluminum vanes. The vanes are held together by the carbon fiber rods to form an internal structure. The internal structure is wrapped in a G-10 fiberglass sheet to form the tube. The inside of the sheet is painted with Krylon Ultra-Flat Black spray paint. The outside of the sheet is painted white for thermal reasons. Layers of paint were repeatedly added until no sunlight would be visible to the human eye when placing the sheet between the viewer and the Sun. Figure 4.12b shows one of the baffles after it is assembled but before the G-10 sheet is painted.

The constructed baffle weighs 3.7 lbs, and is estimated to deflect by  $\sim 1$  mils under its own weight and buckle under  $> 400$  lbs of force (or  $> 100$  g)<sup>8</sup>. The deflection specification ensures that the baffle will not obstruct the star camera fields of view, and the buckling specification ensures that the baffle will not buckle on launch or when the parachute opens upon descent, or during average landing conditions. In the EBEX 2012 Antarctic flight, both

---

<sup>8</sup>Calculations are courtesy of Asad Aboobaker.





(a)



(b)

Figure 4.12: A 3-D model (a) of the star camera baffle compared with an actual constructed baffle before painting (b).

baffles survived the accelerations of launch, parachute shock, and landing.

## 4.4 Pre-Flight Tests and Predictions

A number of tests were performed before flight in order to predict how well the star cameras would perform. In some cases we used these predictions to assess whether the star cameras were equipped with the hardware and software features necessary for successful in-flight performance. In other cases the tests simply provided us with ranges of values required for in-flight commandable parameters. Here we will discuss the notable pre-flight tests and predictions.

### 4.4.1 Sensitivity

At LDB float altitudes between roughly 32 km and 40 km the star cameras are still exposed to significant sky brightness that varies with altitude and viewing angle relative to the Sun. The optical baffles and red filters help limit this noise contribution. However, even with these features, it was important to predict the set of LDB conditions under which the cameras would be capable of solving.

We therefore set out to estimate the number of stars we would expect to find in images captured inside the EBEX science patch. In order to achieve this goal there were several steps. We first predicted the level of noise expected in an image of a given EBEX field based on predictions of sky brightness. Then, given this noise level, we determined the probability of identifying a given star based on its catalog magnitude. We then used these probabilities to predict how many stars would be identified in a random field of view in the EBEX science patch.

## Predicting the Noise

Given the levels of sky brightness the star cameras detect at float, photon noise is the dominant expected source<sup>9</sup>, so we model the level of sky brightness and use Poisson statistics to compute the level of noise. If the star camera detects a sky brightness signal level that corresponds to  $N$  photoelectrons per pixel, then the variation between pixels will be of magnitude  $\sqrt{N}$ .

In order to model the sky intensity we used *MODerate resolution atmospheric TRANsmission* (MODTRAN), a computer program that models atmospheric radiative transfer. We used this program with the caveat that we were not confident in the absolute value of the predictions because the optical transmission fraction along our star cameras' full optical paths is insufficiently well constrained. We did, however, trust the relative intensities between two different configurations for the same camera. A given configuration is a set of parameters that define the camera altitude, camera azimuth, camera elevation, Sun azimuth, Sun elevation, and Earth's surface albedo.

In order to obtain an absolute calibration of sky intensity, we turned to two recent flights: the EBEX 2009 test flight out of New Mexico, and the 2010 Antarctic flight of another experiment called BLASTPol [49]. In the EBEX 2009 flight we had flight data for XSC0 (XSC1 had not yet existed at that time). In the BLASTPol 2010 flight, we had access to data from one of their two star cameras: the "Other Star Camera" (OSC). In both cases we had access not only to multiple data points of measured sky intensity, but also the configurations associated with each measurement. For each of these data points we were able to use MODTRAN to calibrate the sky brightness to a fiducial EBEX 2012 Antarctic flight configuration using the procedure described below.

To define this fiducial Antarctic configuration, we conservatively used the value for each parameter that results in the largest sky brightness while still being plausible (within the

---

<sup>9</sup>We were not aware before flight that polar mesospheric clouds existed and would pollute some of the images.

99th percentile of possible outcomes). These parameters are:

- Altitude of 32.3 km. This prediction came from the flight profile of a previous experiment (CREST) flown on a similar balloon, lowered by 1000 ft to account for an expected higher EBEX weight.
- Star camera elevation of  $35^\circ$ . This is the lowest elevation scanned in a science or calibrator patch.
- Sun elevation of  $5^\circ$ , and a difference of  $180^\circ$  between the star camera azimuth and the Sun azimuth. The gondola was designed to never scan within  $90^\circ$  of the Sun. The brightest point within  $90^\circ$  of anti-Sun is anti-Sun itself<sup>10</sup>. We would always be pointing above the horizon, and the lowest elevation the Sun might reach during our flight window is  $5^\circ$ . For azimuth, we planned to scan nearly anti-Sun.
- Earth's albedo of 72%. This is the 99th percentile hottest case calculated from satellite observations by CSBF<sup>11</sup> (S. Cannon, personal communication, January 23, 2012).

To determine an absolute calibration using our EBEX 2009 data, we used MODTRAN to simulate the expected sky brightness for the EBEX camera using both the EBEX 2009 and the EBEX 2012 configurations. The ratio of those two intensities is used as the calibration scale factor.

In order to calibrate using BLASTPol 2010 data, we need an additional scale factor to account for the difference in cameras. We used two methods to determine this scale factor. The first method was to calculate the ratio from camera specifications. The second method was to determine the ratio empirically by putting the cameras side by side in a dark room and pointing them at the same diffuse light source. We list both of these methods in the table that summarizes the results. For the side by side method, there was a complication in that we found two different results depending on a parameter for which the value was not

---

<sup>10</sup>A consequence of Rayleigh scattering, which is the dominant source of sky brightness, is that the anti-Sun direction is a local maximum of sky brightness.

<sup>11</sup>CSBF stands for the Columbia Scientific Balloon Facility. It is the NASA facility responsible for flying the payload.

Table 4.1: Three methods of predicting the sky brightness for the EBEX Antarctic flight. The second method predicts two numbers as a consequence of not knowing the value of a parameter that could have been in two states during the BLASTPol 2010 flight.

Absolute Sky Brightness Reference	Scaling Method for Different Flight Profile	Scaling Method for Different Cameras	Result (kepsa)
BLAST 2010 OSC	modtran	camera specifications	681
BLAST 2010 OSC	modtran	measured	595 or 666
EBEX 2009 XSC0	modtran	N/A	673

known and could have been in two states during the BLASTPol 2010 flight. We therefore include two separate predictions for the side by side test in the summary table.

When computing a predicted sky brightness it is useful to define a unit corresponding to the quantity of *kilo electrons per second per fully open aperture*, or “kepsa” for short. This unit accounts for all the fixed specifications of the EBEX star cameras, but does not account for the three commandable parameters: the gain<sup>12</sup>, exposure time, and aperture size. Multiplying by the gain, the exposure time, and the fraction of the aperture in use converts a “kepsa” quantity into digital counts (or ADU, analog-to-digital units) in a pixel.

The resulting sky brightness estimates are listed in Table 4.1. The predictions are consistent to within 15%, and are generally between 600 and 700 kepsa. We conclude that to be safe, the star cameras must be capable of operating in 700 kepsa of sky brightness.

### Individual Star Identification Probabilities

We then determined the probability that a star would be identified in an environment with 700 kepsa of sky brightness, as a function of the star’s apparent magnitude. We choose nominal camera settings for flight:

- The minimum gain available, which is 0.040 95 ADU/e-.
- The aperture fully open, which has a diameter of 11.1 cm.

<sup>12</sup>The gain is the conversion factor in the CCD readout from photoelectrons to digital counts.

- A 240 ms exposure time. Note: a 240 ms exposure with 700 kepsa of sky brightness would saturate the CCD. Instead we capture two 120 ms exposures back to back, using the camera’s capability to store four buffers at a time, and a corresponding software feature called “multiple exposures” to co-add them, which will be discussed in Chapter 5.

We captured 12 images on a night sky from the ground with these parameters and added 700 kepsa of simulated sky brightness. The results are shown in Figure 4.13, in which we plot the fraction of stars successfully identified as a function of their apparent magnitude. There is a steep drop-off in identifiable stars between apparent magnitudes 6 and 8. These fractions are used in the next section.

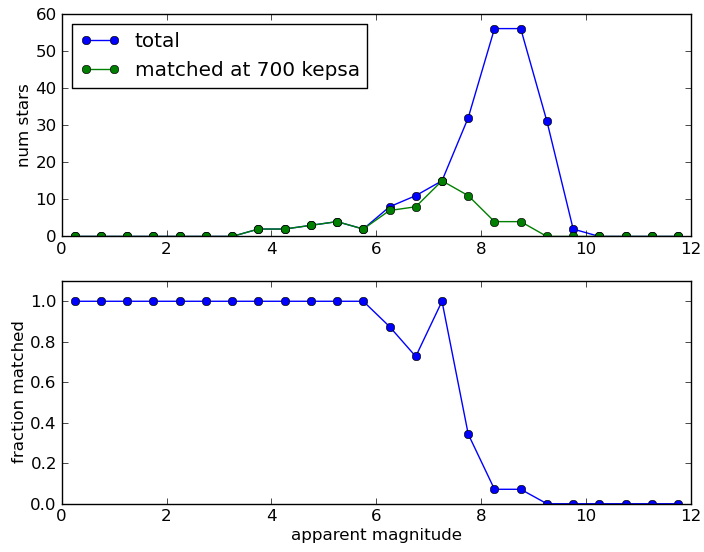
### **EBEX Science Patch Probabilities**

We use the identification probabilities from the previous section to predict how many identifiable stars would be in the images captured while the gondola is pointing at the EBEX Antarctic science patch. To do this we simulated 10,000 measurements. For each measurement we select a random star camera field of view from inside the EBEX science patch, and then calculate the expected number of identifiable stars based on the probabilities measured in the previous section. A cumulative histogram of these results are shown in Figure 4.13. For these parameters at 700 kepsa, the probability of identifying at least 4 stars is 93.1%, while the probability of finding at least 3 stars is 97.8%.

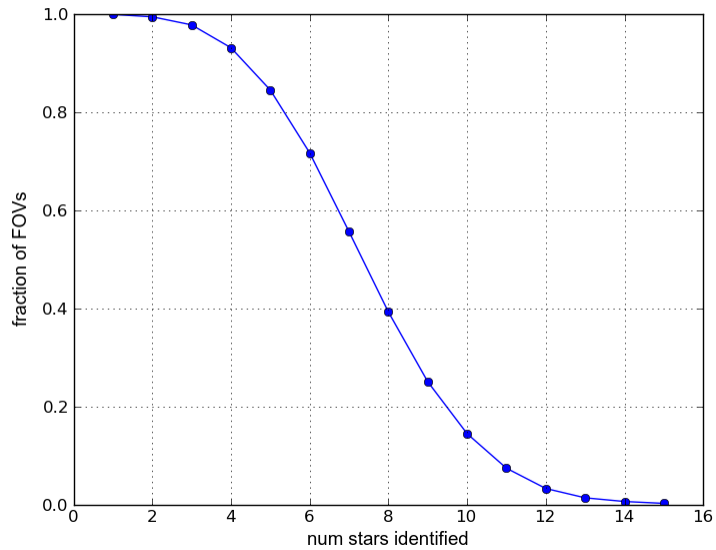
### **Actions Required**

We concluded that we would be well positioned to use the star cameras at float altitudes, but that if the brightness was as high as expected in the fiducial case, the double exposure feature would be critical.

An additional complication of increasing the exposure time, especially via multiple expo-



(a)



(b)

Figure 4.13: Subfigure (a) shows how many stars were identified in 240 ms exposures in an environment with 700 kepsa of sky brightness, as a function of apparent magnitude. It shows the total number of stars identified along with the total number of stars in the catalog (top), and then shows the fraction of catalog stars identified (bottom). Subfigure (b) uses the results from (a) to estimate how many stars will be identified in an arbitrarily selected star camera field from the Antarctic science patch. It shows a cumulative histogram showing the fraction of simulated fields of view (FOVs) that had at least N identified stars, as a function of N.

tures, is that the images may become susceptible to motion blur given the angular velocity of the gondola throughout the duration of the exposure. In order to accommodate potential motion blur, we also implemented a feature in STARS called “motion PSF”, where PSF refers to the point spread function. When enabled, STARS will collect gyroscope data from the flight computer that describes the rotation of the gondola for the duration of the exposure, and construct a corresponding kernel to imitate the streaked sources. This kernel is then convolved with the images to help pick out the otherwise low signal-to-noise sources. This feature is discussed in more detail in Chapter 5.

#### 4.4.2 Solution Uncertainty

When a star camera processes an image and finds a pointing solution, it also returns associated errors. We performed a test to determine whether the errors reported by the star camera software were correct, assuming no optical distortion.

To perform this test we simulated 1000 fields of view from the EBEX science patch. For each field of view we found its 5 brightest catalog stars, and determined their pixel locations within the field of view. From those five pixel locations, we used the star camera software to determine the attitude and associated errors. For each of the three attitude coordinates we could compare the recovered angle to the true angle to obtain the true error. We could then compare this true error to the error reported by the filter. With many iterations, we should find that the reported errors are consistently equal to the standard deviation of the true errors. Figure 4.14 shows this visually. For each attitude coordinate, a histogram of all the true errors are plotted in blue, and a histogram of the reported errors are plotting in green. The histogram of reported errors are centered at the standard deviations of the true errors to within roughly 4%.



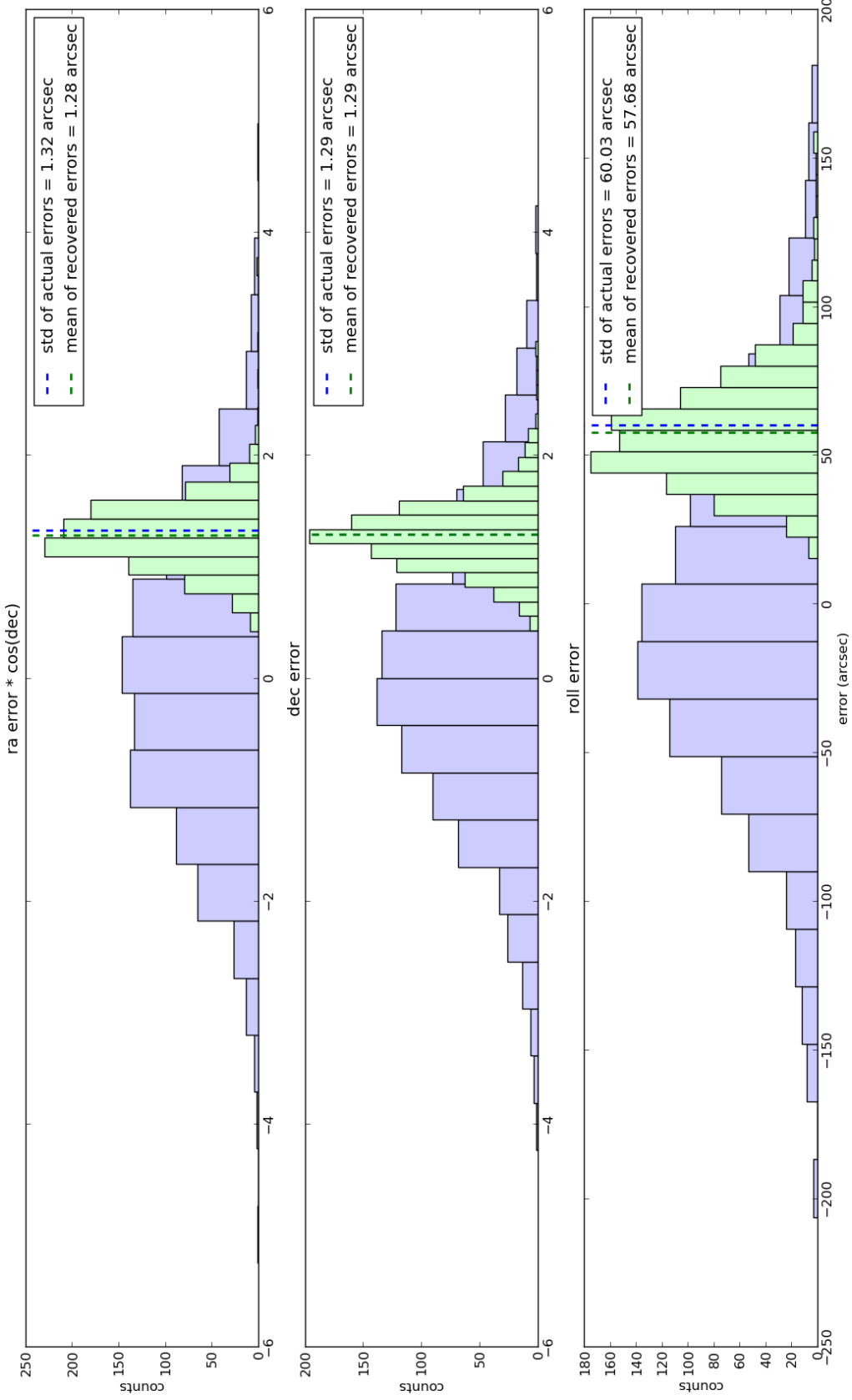
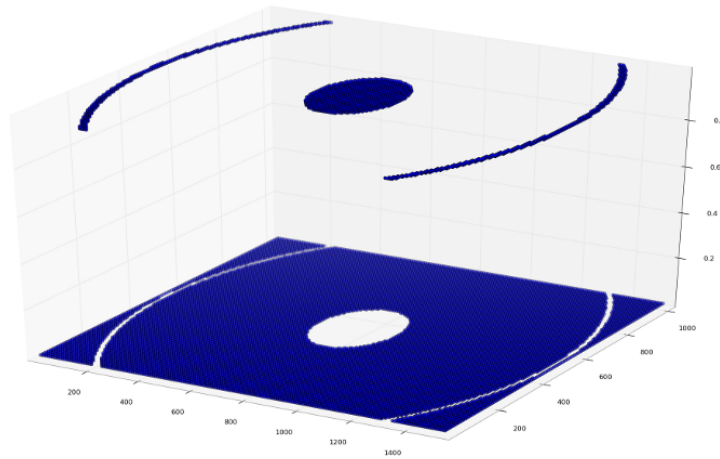
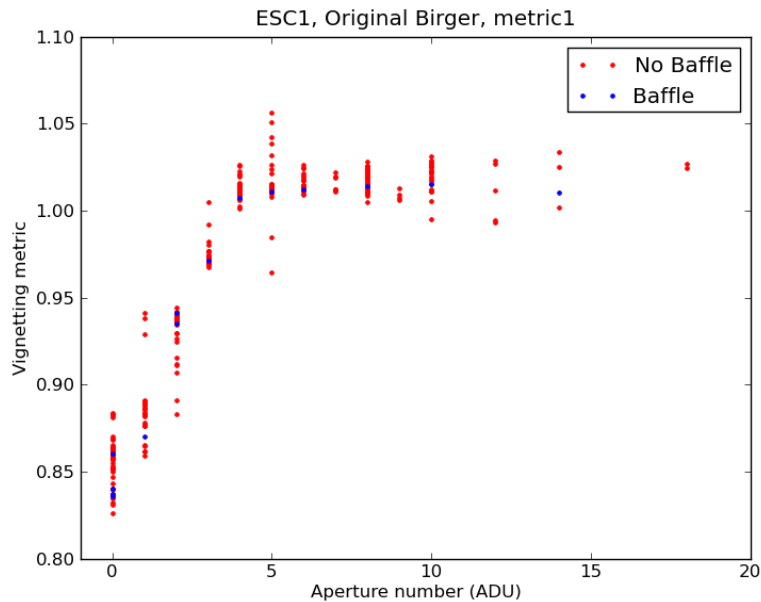


Figure 4.14: Results of a simulation testing the error reported by the star camera software. Each plot shows a comparison between the true errors (blue histogram) and the reported errors from the software (green histogram). The green histogram should be centered on the standard deviation of the blue histogram. There are three plots, one for each attitude coordinate.



(a)



(b)

Figure 4.15: Subfigure (a) shows a 3d plot representing all the pixels in an image and identifies those that are used to calculate the vignetting metric. The vignetting metric is the mean value of some representative edge pixels divided by the mean value of some representative center pixels. The raised pixels that in the subfigure that are far from the center are the representative edge pixels, and the raised pixels at the center of the image are the representative center pixels. Subfigure (b) shows the result of this metric for multiple images plotted as a function of aperture setting. Images taken with aperture settings 0 through 4 show vignetting. The test was done with and without the optical baffle installed, which has no noticeable effect.

### 4.4.3 Vignetting

We noticed that many of the star camera images contained vignetting, which is a reduction in brightness near the edges of an image. We determined that the vignetting only occurs when the aperture is fully open or close to fully open. Figure 4.15b shows the results of a test in which we demonstrate that large apertures on XSC1 result in images where the edges of images are reduced in brightness by up to 15%.

To perform this test we captured images at various aperture settings. Aperture setting 0 corresponds to fully open, while higher numbered aperture settings correspond to smaller diameter apertures. Aperture setting 6, for example, only has 60% the diameter of aperture setting 0. For each image we calculated a metric that represents the level of vignetting by dividing the mean level of pixels near the edge of the image by the mean level of pixels in the center of the image. The pixels chosen to represent the edge and represent the center are shown in Figure 4.15a. Figure 4.15b shows this metric as a function of aperture position for multiple images, some taken with and some taken without the optical baffle. Any effects from the optical baffle were found to be insignificant. The plot shows that at aperture setting 0 the edges only capture 85% as much light as the center does. It also show that by aperture setting 5 or 6 the vignetting is eliminated. Due to the results of this test we used aperture setting 6 during the EBEX 2012 Antarctic flight.

### 4.4.4 Disk Space

The star cameras are designed to record all the images captured for post-flight processing. This is because each image might not be solved during flight, and if it were solved during flight there might be refinements worth making post-flight that are not necessary in real-time attitude determination. As an aside, the star camera images captured during the 2012 Antarctic flight turned out to be of scientific interest outside of the field of cosmology (see Chapter 11).

Table 4.2: Comparison of lossless compression methods considered for the star camera images. The compression algorithms were tested on noisy images from the 2009 test flight. The compression ratio is the uncompressed size divided by the compressed size. Also listed is whether the compressed file is directly compatible with the *GNU Image Manipulation Program* (GIMP) or the *cfitsio* C library, which is also used by the *pyfits* library for python. Compatibility with each of these is highly convenient and increases productivity during star camera development and testing. Not shown is the “ezip” compression (“EBEX zip” - developed specifically for EBEX), which was comparable in compression ratio, but unfortunately was 2.5 to 7.7 times slower than the standard options.

Method	Compression Ratio	Compatibility	
		GIMP	cfitsio/pyfits
zip -1	1.54	no	no
zip -9	1.59	no	no
bzip2 -9	2.08	yes	no
cfitsio gz	1.59	yes	yes

The first step in predicting the in-flight disk space usage is to measure the size of the individual images, which depends on what kind of compression is used:

- **Lossy compression algorithms** were considered but rejected, due to the fact it is difficult to assess before flight whether a particular algorithm will negatively influence the images. As an example, an algorithm that discards high frequency noise (which is the least compressible part of an image) but maintains the large-scale structures would end up discarding the stars themselves. A notable lossy compression algorithm, which was used in another experiment, involved discarding parts of the images that did not contain stars, as determined by the real-time software [50]. This algorithm obviously results in a very high compression ratio, but unfortunately it relies on the real-time software application to correctly identify the stars, which is one of the primary failure modes we are trying to protect against by storing the images to disk.
- **Lossless compression algorithms** were also considered, some of which could reduce the disk space usage by a factor of  $\sim 2$ , as shown in Table 4.2. The embedded gz compression in the *cfitsio* library, listed as “cfitsio gz” in the table, was implemented as

Table 4.3: Disk usage calculations.

Exposure Mode	Num Images Per Day				Log File Space	Disk Space Required
	cmb	calibration	confusion	total		
single	2880	655	720	4255	0.324 GB	228 GB
double	5760	1309	720	7789	0.324 GB	418 GB

optional in STARS given its compatibility with the GNU Image Manipulation Library (GIMP) and pyfits. Further testing, however, showed that even the fastest compression algorithm (zip -1) cost too much CPU time to be used with double exposures during a calibrator scan. As a result, no compression algorithms were used during flight.

Each uncompressed image occupies  $\sim 3.15$  MB of disk space. Images are captured on each scan turnaround throughout flight. The star cameras can operate in single or double exposure modes, capturing either one or two images on each turnaround. A number of conservative assumptions are made in order to determine the amount of disk space required:

- 20 hours of cmb scans per day, with 50 s period throws
- 3 hours of calibrator scans per day, with 33 s period throws
- 1 hour of focusing / autofocusing / confusion per day, with an image every 5 s
- 17 day flight
- A log file of 20 MB per day

The resulting disk space calculations are shown in Table 4.3 and show that the drives must store at least 500 GB if double exposures are used. In order to use hard drives larger than 320 GB with the star camera computers, the maximum size available with IDE connections, we purchased and installed SATA to 44-pin IDE Converters from Addonics Technologies.

We purchased four disks that held at least 500 GB (two as spares), and consciously chose to fly a different brand in each star camera in order to guard against failure of one of them due to cosmic ray hits. Both disks operated successfully in during flight, though the caution

was well justified: star camera disk failures cut a flight of a different experiment short in the same season.

#### 4.4.5 Pointing Offsets

In this section we discuss the pre-flight procedure to estimate the pointing offsets between the star cameras and the microwave telescope boresight, which is necessary because the star cameras and the telescope do not point in the exact same direction. In the *Flight Control Program* (FCP), the pointing offsets for a given star camera consist of a cross-elevation and an elevation angle that are added to the star camera pointing stream to obtain the boresight pointing stream. In post-flight analysis, the pointing offsets consist of three rotations (delta-x, delta-y, and delta-roll) that are applied (in that order) to the star camera pointing stream. The FCP pointing offsets are flat sky approximations of delta-x and delta-y, and by not applying a delta-roll FCP assumes that the focal plane has zero roll.

The star cameras are mounted to aluminum blocks that are mounted to the gondola through slotted holes, allowing for a variable elevation angle. In addition, there are multiple structures located between the internal lenses of the cryostat and the internal lenses of the star cameras, all of which were constructed to some finite machine tolerances. As a result, upon assembling the gondola in Antarctica, we do not know the relative pointing between the star camera boresights and the microwave telescope boresight to better than a couple degrees.

As discussed in Section 3.3, the real-time pointing requirement is roughly  $0.5^\circ$  in cross-elevation and in elevation. This includes systematic errors, such as a pointing misalignment between the star cameras and the telescope. We therefore attempted to measure the angular offsets, in cross-elevation and elevation, between each star camera and the telescope boresight to an accuracy of  $\leq 0.5^\circ$ . We performed this test three weeks before flight, which was after the telescope was assembled and the star cameras were mounted in their final locations.

The challenge inherent in performing this measurement is that there are no astronomical sources that the EBEX telescope can map from the ground in Antarctica in the austral summer. Instead we must use our own millimeter-wave source, which is challenging because the telescope cannot point below  $15^\circ$  in elevation. Consequently, finding a location suitably far away to minimize near-field effects requires a high mounting platform. Given the height of the structures and vehicles available to us, this limits the distance that we can place the source to within the near-field region of the pointing system. The structure we used in Antarctica was an aerial work platform (also known as a “man lift”), placed only a couple dozen meters away.

If the telescope and star cameras were observing the same source, and if this source was infinitely far away, then the calibration procedure would be quite simple. We would point the telescope at the source, then point the star camera at the source, and record the amount that we rotated the gondola in between. When the source is in the pointing system’s near field, however, there is an additional complication. As the gondola rotates about its center of mass, the star cameras translate on their own lever arms, and the star cameras find themselves pointing directly at the source either sooner or later than they would have had the source been infinitely far away. To overcome this challenge we measured the distances from the center of rotation of the gondola to the microwave telescope’s primary mirror, the star camera’s lens, and the source. We then calculated and corrected for this near-field effect.

We performed this measurement to obtain the pointing offsets for each star camera. The error in the offsets measured before flight can be seen in Figure 3.2, which shows the difference between each sensor’s pointing solution and the reconstructed pointing solution from post-flight analysis. The errors reveal themselves as the overall displacement of each distribution from zero.

In post-flight analysis we determine the star camera offsets by comparing the location of the calibrator<sup>13</sup> in EBEX sky maps to its known location. The offsets obtained during

---

<sup>13</sup>The astronomical calibrator used by EBEX is an embedded star cluster named RCW 38

Table 4.4: The angular offsets between each star camera and the EBEX telescope boresight, as determined before flight and after flight. The offsets are added to the star camera pointing to obtain the microwave telescope pointing.

	pre-flight measurement (degrees)	post-flight measurement (degrees)
XSC0 cross-elevation	0.83	0.37
XSC0 elevation	2.51	2.71
XSC1 cross-elevation	0.72	0.34
XSC1 elevation	-1.49	-1.24

pre-flight testing and the pointing offsets obtained during post-flight analysis are shown in Table 4.4. The pre-flight measurements are shown to be accurate to within roughly  $0.5^\circ$ .

## 4.5 EBEX 2012 Performance

The star cameras performed successfully in the EBEX 2012 Antarctic flight despite a number of challenges that presented themselves during the flight. The star cameras captured viable images that were solved after the flight to a precision of  $1.5''$  in cross-declination,  $1.5''$  in declination, and  $48''$  in roll as shown in Figure 4.16. Roughly 40000 images were captured throughout the flight at scan end-points, which were generally 40 seconds apart, and we have valid pointing solutions at 92% of the end-points for which the telescope was pointed at least  $90^\circ$  away from the sun. The role of these images in the success of the post-flight pointing reconstruction is discussed in Section 10.2.3. During flight the star cameras solved with the same precision but less frequently, with solutions roughly every 40 to 100 seconds. These regular solutions were frequent enough to meet the real-time pointing requirement of  $0.5^\circ$  for the vast majority of flight, and to confirm that the star cameras were collecting viable images for post-flight analysis. The challenges that the star cameras faced in flight will be discussed in detail in Section 5.4, which describes the performance of the software package, STARS, in particular.



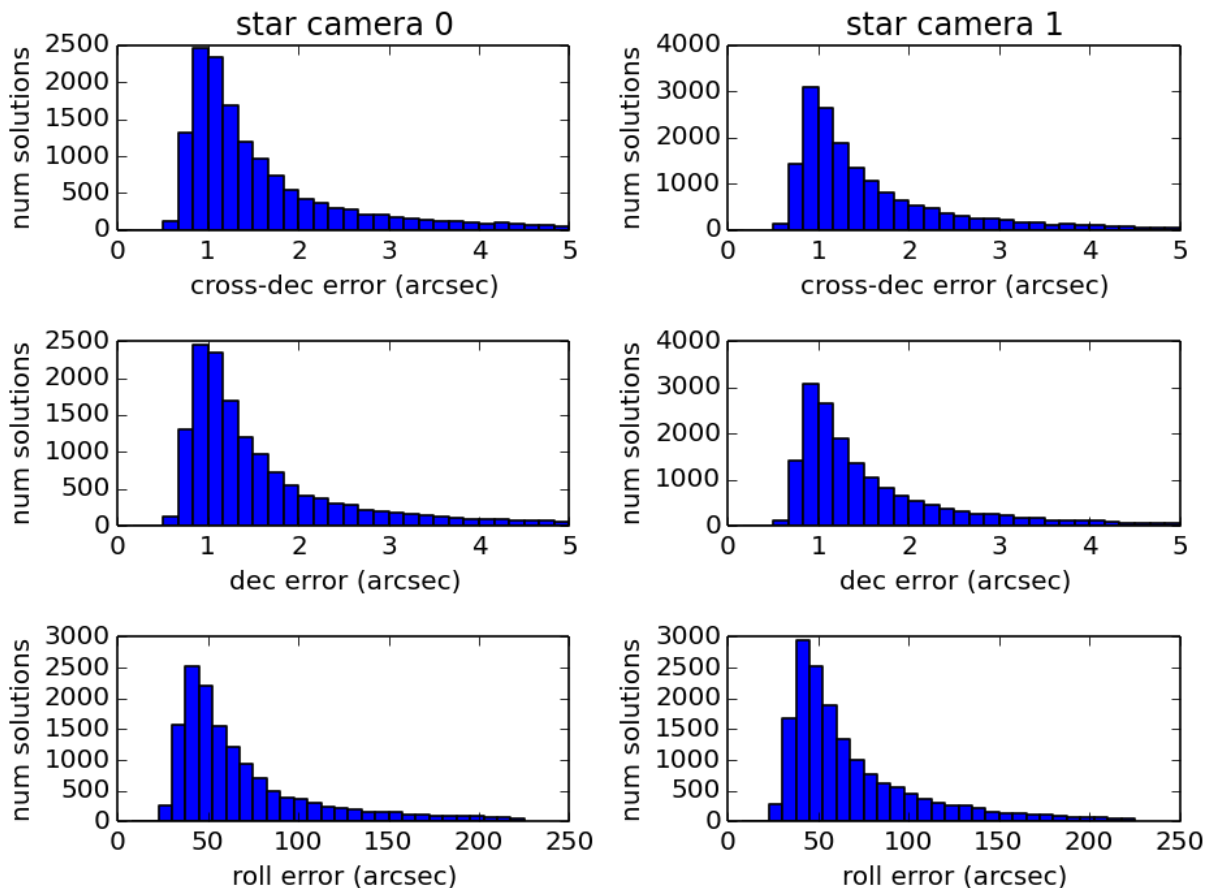


Figure 4.16: Histograms of the star camera solution uncertainties, as estimated by STARS during post-flight analysis of the images. The histograms contain 15050 Star Camera 0 solutions and 17175 Star Camera 1 solutions, which amount to  $>90\%$  of the solvable images from flight. The uncertainties are  $1.5''$  in cross-declination,  $1.5''$  in declination, and  $48''$  in roll. These uncertainties are smaller than the predicted uncertainties due to the fact that there are roughly 8 stars per image instead of 4. This was possible because the brightness at float was not as high as the conservative case for which the camera performance was modeled.

# Chapter 5

## Star Camera Software, “STARS”

### 5.1 STARS Design Requirements

When designing the *Star Tracking Attitude Reconstruction Software*, or STARS, we strove to satisfy a number of requirements as follows.

- **Frequent in-flight solutions.** The EBEX star cameras were designed to capture images roughly once every  $\sim 40$  s at turnarounds in the scan. Therefore, STARS must solve an appreciable fraction of these images in flight because the error on the star camera pointing stream increases with the amount of time that passes from the most recent solution, and because it is the only way that ground operators can have confidence that the star cameras are collecting viable data for post-flight analysis.
- **Robust solving operation in daylight conditions.** EBEX is designed to fly over Antarctica during the austral summer when the Sun is always above the horizon. Sky brightness, even at 120,000 ft, adds considerable noise to the images, making the signal-to-noise ratios of the sources a primary concern. Other unpredictable features may also pollute the images, including mesospheric clouds, satellites, cosmic ray hits, optical vignetting, and internal reflections. Therefore STARS must be able to process images with substantial noise and unpredictable features.

- **Autonomous operation.** STARS must be able to operate with limited manual intervention due to the low communication bandwidth between the ground station and the payload during most of the flight.
- **Minimal dependencies on other subsystems.** The star cameras depend on other components of the attitude control system for operation. They were designed to work in concert with the scan control algorithms in FCP, to use coarse attitude information from the other pointing sensors, to use commands from pre-defined observing schedules, and to use commands from ground operators using the telemetry system. However, issues can arise in flight with any number of subsystems<sup>1</sup>, so STARS must be robust to changes in other subsystems: it must continue to collect viable data and provide pointing solutions to FCP in instances where the inputs to the star camera systems are abnormal.
- **General robustness.** Any solving related functionality cannot be tested within four months of launch due to 24-hour daylight conditions in Antarctica, despite the fact that alterations are made to the flight computer program, with which STARS interfaces, and that the experiment is disassembled and reassembled for shipping. The software must therefore be robust enough to work with modifications in FCP and changes in the hardware without being re-tested before flight.

## 5.2 STARS Design Principles and Architecture

STARS is equipped with a number of features and functionalities that are designed to make it more robust (see Table 5.1). An increase in features and functionalities often leads to an increase in code complexity that can lead to a decrease in software reliability. For that reason, special attention was paid to good programming practices. We emphasized code readability, used logical abstractions and modularity, and enforced thread-safety. We also

---

<sup>1</sup>See Section 5.4 for examples from the EBEX Antarctic flight.

Table 5.1: List of key functionalities provided by STARS.

Notable In-Flight Feature	Discussed In
robust pattern matching	Section 5.3.4
robust source finding	Section 5.3.3
accurate and abundant video display	Section 5.3.6
accurate and abundant downlink	Section 5.3.8
selective masking	Section 5.3.3
multiple exposures	Section 5.3.7
motion PSF source finding	Section 5.3.3
non-stationary autofocus	Section 5.3.7

implemented extensive testing and performance verifications. As a result of these measures, throughout the 11 day flight of EBEX STARS reliably provided pointing solutions and never crashed.

### 5.2.1 Architecture

STARS is an object-oriented multi-threaded cross-platform application written in C++. During flight it runs on Windows due to lack of camera driver support in Linux. During development, testing, and post-flight analysis the software is primarily used in Linux.

STARS has multiple threads that can be grouped into four primary categories:

- Imaging, which handles lens control, camera control, and image capture either from the camera controller during flight or from a pre-captured image file during development, testing, and post-flight analysis.
- Solving, which handles the image processing and search algorithm for finding a pointing solution from an image.
- Networking, which receives commands from the flight computers and returns pointing solutions, debugging information, and optionally raw images for redundant storage.

- A main thread, which handles display, housekeeping, and shared memory.

## 5.2.2 Standard Operation

The top subfigure in Figure 5.1 shows the flow of data under normal operation. A camera object, running in its own thread, captures an image from the physical camera controller as soon as it becomes available, stores it to disk, and shares it with the solving object. The camera object also handles control of the camera gain, and lens objects control the focus and aperture of the lens. The solving object, also running in its own thread, will load the image when it is done working on the previous image (either because a solution is found or it timed out). Any information the solver extracts from the image, including statistics, the locations of the star sources, and a possible pointing solution, is shared with the main thread and passed on to the networking thread. The main thread displays this information for video downlink. The networking thread shares this information with the flight computer so that it can be used in the real-time pointing solution, stored to disk, and downlinked with the other EBEX numerical data streams.

## 5.2.3 Shared Memory

All data-sharing between threads is done using circular buffers in a thread-safe manner. Each circular buffer passes data in one direction between exactly two threads. Since different permutations of threads want to share different kinds of data, there are many possible data objects that can be created from a generalized circular buffer class for a given path<sup>2</sup>. This includes raw image objects, lens request objects, pointing solution objects, and many others.

Examples of shared data being passed between threads can be seen in Figure 5.1.

---

<sup>2</sup>Technically, these data objects are instantiations of a template circular buffer class, where the template argument is a class that contains the type of data to be passed. Multiple data-sharing paths might use the same class as their template argument.

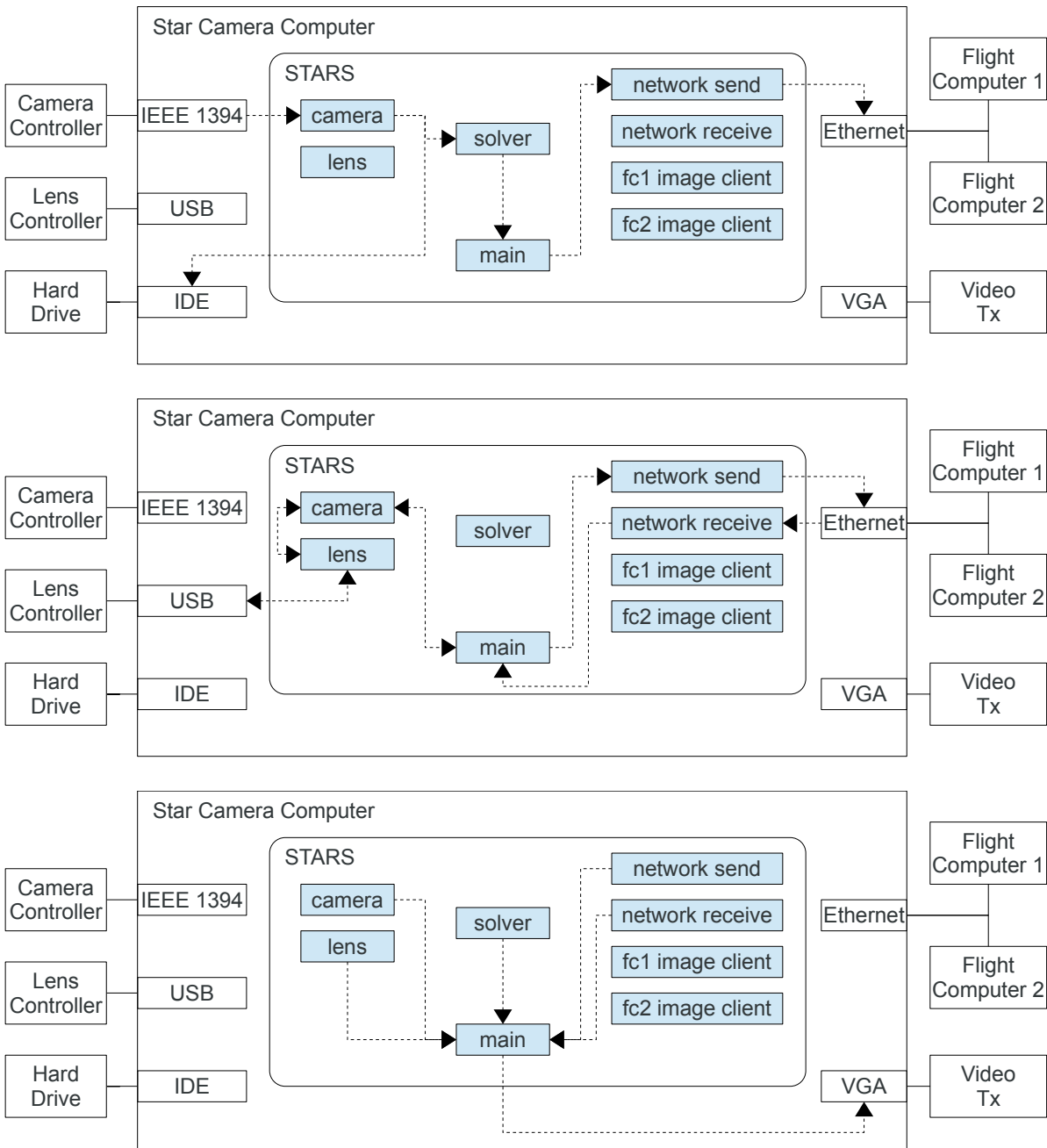


Figure 5.1: Block diagrams of STARS running on a star camera computer. The shaded boxes show the threads that belong to the STARS process. Arrows with dotted lines indicate the flow of data. Data sent between threads is done safely and efficiently with circular buffers. The top panel shows the standard operation: image data and/or pointing solutions make their way from the camera controller to the flight computers. The middle panel shows the path that lens requests and results follow from the flight computers to the lens controller and back. The bottom panel shows the paths of various objects for display purposes.

## 5.2.4 Settings Files

Settings files allow the user to adjust parameters without recompiling. Having many configurable settings is useful for development, testing, and post-flight image processing. However, too many configurable settings can be difficult to keep track of, especially when preparing for launch. To remedy this, STARS accepts two settings files, one called “flight.txt” and one called “custom.txt”.

The “flight.txt” settings file contains all 93 possible configurable parameters, with their default values for flight, and is committed to the repository. The “custom.txt” file is not committed to the repository, and any parameters in the custom.txt settings file override those in flight.txt. The custom file only contains 8 parameters that must be specified for flight because they are different between the two star cameras. These parameters are due to hardware differences, e.g. mounted roll, predicted focus position, and platescale. This helps curtail user error by reducing the number of parameters that must be reviewed during the pre-flight checklist from 93 down to 8. During testing and post-flight analysis, additional parameters are added to custom.txt for additional customization.

## 5.2.5 Testing

STARS was developed between 2010 to 2012, and all of its functionality was independently tested as features were implemented. However, the majority of its testing was done during integrated testing of other flight systems. Many telescope calibration tests, star camera hardware tests, and tests for other subsystems required fully functional star cameras. Some of these tests are discussed in other sections, including 4.4.1. During these tests aberrant software behavior on the part of the star cameras was not tolerated, in that any unexpected behavior was immediately fixed.

There are two notable features that were implemented to aid in the testing process: a sky brightness simulator and a log file parsing utility.

## Brightness Simulator

The sky brightness simulator is a feature in STARS that injects random Poisson noise into the images to simulate conditions at float. In the settings file one can specify the photon flux, the camera gain, and the exposure time to inject noise real-time into images during ground tests. This is particularly useful for making fully integrated ground tests as realistic as possible, and for performing the sensitivity test described in Section 4.4.1.

## Log Parsing Utility

The log parsing utility is a python module that, given a STARS log file path, returns to the user a list of python solution objects. This expedites development and testing, and is used in post-flight analysis of the images.

# 5.3 STARS Components

## 5.3.1 Solving

The solver operates in three parts. It first measures a number of image statistics: the mean, noise, gain, and number of pixels saturated. It then searches the image for sources, which are bright spots that could potentially be stars. To do that, the source finder convolves the image with potential point spread functions (PSFs) and selects the most significant sources by finding pixels that stand out above the average value of a set of neighboring pixels, defined in such a way that they are between 13 and 24 pixels distant from the potential source pixel. Once the sources in an image are located, they are passed to the pattern matcher. The pattern matcher builds triplet permutations of sources and compares them against pre-computed lists of triplets in the star catalog. When a catalog triplet potentially matches a source triplet, it then tries to match all the other sources in the image to catalog stars. If a list of conditions are met, then the set of source-to-catalog-star matches is accepted as the



solution. The star catalog is optimized for the EBEX star camera field of view and frequency response. The next three subsections discuss the three parts of the solver in detail (statistics, source finding, and pattern matching), and then the following subsection describes the star catalog in detail. The remaining subsections describe other STARS components.

### 5.3.2 Solving - Statistics

When the solver receives an image, it first calculates statistics on the image to be used in later steps of image processing, to be made available to the user, and to auto-level the image for displaying. Auto-leveling is necessary because the interesting features in the images only occupy a small fraction of the full image depth, due to the background sky brightness adding a constant level to the images. It is performed by building a histogram of the pixel values and defining the boundaries as the middle 98% of the data. As an optimization, only 1 in every 16 pixels are sampled - evenly spaced on a grid - when building the histogram. With this procedure the stars in the image have little influence on the auto-leveling, which is acceptable as the stars will show up as white circles.

The statistics measured from the image are the mean, noise, gain, and the number of pixels saturated. The image is broken up into 16x16 pixel cells, and these values are measured for each cell. The mean of the image is the mean of all the cell means. The noise of the image is the median of the cell noises; the median is used here to prevent structure within cells (e.g. stars and sharp gradients) or saturated cells from influencing this measurement. The gain of the image is the mean of the bottom one-sixth of the cell gains for all the “clean” cells - cells in which the mean is high enough to be considered significant and in which there are no saturated pixels. The gain of an individual cell is its variance divided by its mean.

### 5.3.3 Solving - Source Finding

The next step in solving an image is to find the bright spots that may be stars. Any object identified in an image that could potentially be a star is referred to simply as a “source”, because it may or may not correspond to an actual star, while objects loaded from the catalog of known stars are referred to as “stars”. This distinction is important for the next section in which we discuss the pattern matching algorithm, which attempts to match source objects to star objects.

The source finder primarily operates in two modes: normal and robust, though it also has a third mode called “motion PSF” that is also discussed here. Robust mode differs from normal mode in that it can also find sources that are quite out of focus ( $\sim 30$  px diameter) but at a significant computational cost (3.8 s on the EBEX star camera computers).

The source finder picks out sources that surpass some tunable threshold above the local level in that region of the image. To account for the fact that sources span multiple pixels, the source finder picks out sources from a filtered image (smoothed with a Gaussian kernel), and care is taken when calculating the local level in that region of the image. Thus a pixel is of interest when the following condition is met:

$$\text{smoothed pixel} > \text{leveled pixel} + \text{threshold} * \text{noise} \quad (5.1)$$

To calculate the local level for a particular pixel, the source finder takes the mean of a subset of pixels that are between 13 and 24 pixels away from the target pixel. This is akin to a low-pass filter, but it is performed in position space and excludes pixels that may be biased by the source. The specifics of this operation are shown in Figure 5.2, where each dark gray pixel in the leveled image takes the value of the mean of all the light gray pixels. This operation is performed coarsely on a  $4 \times 4$  downsampled copy of the image as an optimization: instead of making  $n \times 449$  pixel visits, where  $n$  is the number of pixels in the full-size image, it only makes  $n \times 3.8125$  pixel visits (ignoring boundary cases).

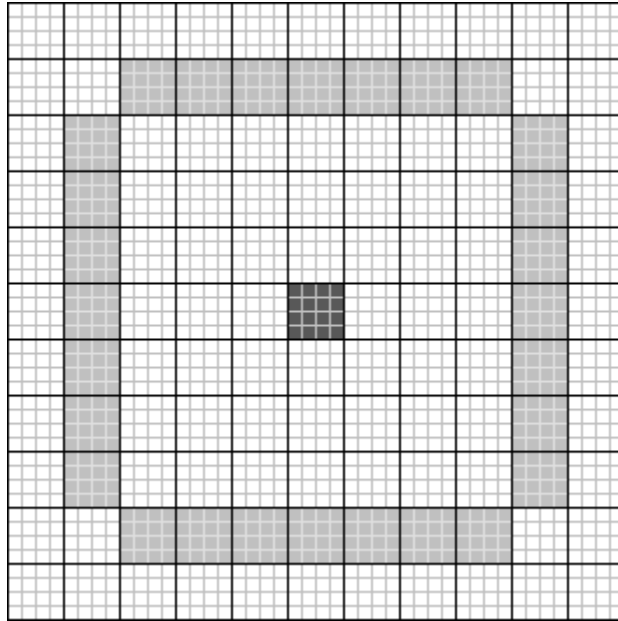


Figure 5.2: The algorithm for determining the regional level around a pixel. This image represents a  $44 \times 44$  px region of an image, where each pixel is shown as a small square with a gray outline. The STARS source finder compares a filtered version of the image to a leveled version of the image, where the leveled version is calculated by assigning the mean value of the 448 pixels shaded in light gray to the 16 pixels in the center. The light gray pixels are chosen because they are close enough to the dark gray pixels to be representative of the local level, but are far enough away that a potential source centered on a dark gray pixel itself does not bias the level. As an optimization, the operation is performed on a coarse version of the image. The coarse version of the image is a factor of  $4 \times 4$  smaller. In this example the coarse pixels are represented by black outlines.

The image is then smoothed with a  $3 \times 3$  px Gaussian kernel with  $\sigma = 1$  px and the source finder uses this smoothed copy to search for sources. The smoothing is a 2-D correlation with the Gaussian kernel over the image, and it serves to give weight to sources that extend beyond a single pixel. The image is broken up into  $128 \times 128$  px cells<sup>3</sup>, and up to two sources can be selected per cell according to equation 5.1.

If robust mode is enabled, this smoothing and searching procedure is repeated twice more, once with an  $11 \times 11$  px kernel with  $\sigma = 3.5$  px and once with a  $61 \times 61$  px kernel with  $\sigma = 15$  px. As an optimization, each correlation with an  $l \times l$  kernel is broken up into two operations: first a correlation with an  $l \times 1$  kernel, then a  $1 \times l$  kernel. This is mathematically identical if the kernel is Gaussian, and reduces the correlation from  $\Theta(nl^2)$  to  $\Theta(nl)$ .

Finally, the sources are sorted by their estimated significance (integrated flux / noise) and the vector of sources is cropped to  $10^3$ .

Note that we chose to develop robust algorithms for dealing with general artifacts rather than performing a pre-flight flat-fielding procedure. This is because the image structures seen in flight may not match those seen during controlled tests before flight, which was the case in the 2012 Antarctic flight (see Section 5.4).

### Selective Masking Utility

To be better prepared for image artifacts that could disrupt the source finder, STARS has the ability to disable  $128 \times 128$  px blocks of an image. This is demonstrated in Figure 5.3 and was used in flight (see Section 5.4).

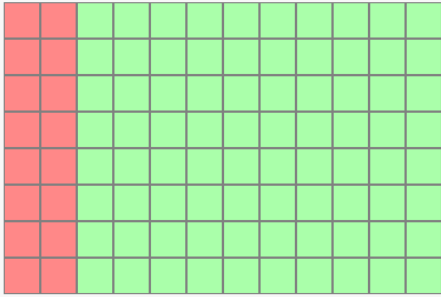
### Motion PSF

The motion PSF mode is an alternate source finding mode, and is meant to aid in finding sources when the exposure time is too long for the gondola to remain still to within a few pixels. This is especially common when using multiple exposures (see Section 5.3.7). In this

---

<sup>3</sup>Many of the numbers used in this section are the default flight values of tunable parameters.

### STARS Selective Masking Utility

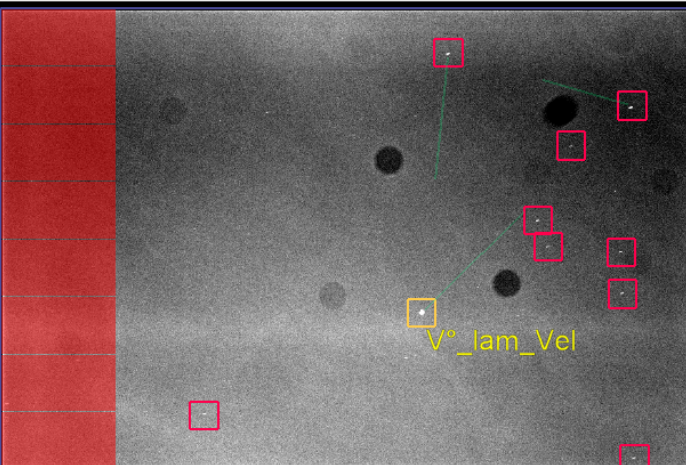


Result:

field0	field1	field2
50343939	805503024	3146496

```
./webexcmd xsc_selective_mask 2 1 50343939 805503024 3146496
imaging.selective_mask.enabled = true
imaging.selective_mask.field0 = 50343939
imaging.selective_mask.field1 = 805503024
imaging.selective_mask.field2 = 3146496
```

**STARS**



[name]

Housekeeping  
disk:

Lens  
Focus:  
Apert:

2012-12-30--08-27-56--625\_pX.fits : done, solved (1.5 s)

age: 5 s	mean: 1591.4	ra: 136.56°	match: 10 of 10
ctr_s: 1 [nc]	noise: 8.2	dec: -43.97°	pt error: 4.12"
ctr_f: 1600	num_sat: 0	az:	fit error: 0.97 px
lat:	gain: -1.1 dB	el:	plate: 9.509"
lst:		hroll:	mexp: 59 ms

Figure 5.3: Demonstration of the selective masking utility. This html/javascript utility allows users to toggle blocks on or off using the mouse and provides them with a command to send to STARS during flight. STARS then ignores the disabled blocks during source finding. In this example the utility is configured to block out the left  $\frac{2}{12}$  of the image, and the STARS screenshot shows that this area is shaded out of the image, indicating that the source finder will not search the left  $\frac{2}{12}$  of the image.

mode, FCP sends arrays of gyroscope readings to STARS for the duration of the exposures. STARS then reconstructs the motion blurred source shape for each exposure, and uses these shapes as smoothing kernels instead of Gaussian kernels. Note that individual exposures belonging to a set of multiple exposures must be smoothed separately, but they use kernels that originate from the same starting point so that the streaks for a given star from each exposure contribute to the same source. This is demonstrated in Figure 5.10.

### 5.3.4 Solving - Pattern Matching

The pattern matcher tries to match the sources in an image to corresponding catalog stars given a particular platescale and set of angles describing the orientation of the image frame using the following procedure:

1. It picks the brightest three sources in the image. These three sources form a triangle with known leg lengths (angular distances on the sky).
2. It selects all the triplets of catalog stars that have similar leg lengths. For each of these triplets:
  - (a) It performs a least-squares fit to find the pointing solution for the image that would align the selected sources with the selected stars.
  - (b) Using this pointing solution, it tries to match all the sources in the image to stars from the catalog. As an optimization it only checks stars in the region of the pointing solution.
  - (c) With all the possible source-star matches it fits a new pointing solution.
  - (d) If this tentative pointing solution satisfies all the requirements, it is accepted as the pointing solution.
3. If a pointing solution is not accepted, a new triplet or pair of stars is chosen. This is repeated until every unique triplet combination of the brightest 7 sources is exhausted, and then until every pair combination of two sources is exhausted.

User-specified parameters that may be tuned in order to define whether a solution is acceptable include:

- a limit on angular distance from a specified horizontal pointing location (azimuth and elevation)
- limits on horizontal roll
- limits on elevation range, which is convenient for placing constraints based on telescope hardware
- a limit on angular distance from a specified equatorial pointing location (right ascension and declination)
- limits on equatorial roll
- a limit on the pointing solution error
- a lower limit on the number of sources matched to stars

During flight the limits on pointing coordinates come from estimates of the pointing from the other absolute sensors, which are less accurate. Generally speaking they will provide an estimate to within 5 or 10 degrees, and in this case the code is set to require 4 or 5 source-to-star matches. However, STARS can operate in a lost-in-space mode in which it has no pointing estimate to limit the region of the sky that it searches. This “mode” is activated by disabling all the coordinate-based limits. Without these limits, we must require 7 or 8 source-to-star matches to be confident in a solution.

As will be discussed in the following section, the star catalog maintains a preprocessed list of star pairs and triplets in order to facilitate a fast search through the catalog. In this manner the STARS pattern matcher is highly influenced by the Pyramid algorithm described in [51].

The pattern matching step (2b), in which the solution from a triplet match is extended to try to match all the source in an image, allows the user to set a strict requirement on the number of sources matched. Allowing the requirement on the number of matches to be as

high as 20 allows the user to relax other constraints in the images, such as the platescale limits and the match tolerances that may become too strict under systematic optical distortions. The STARS pattern matcher is fast enough to run lost-in-space in flight (see Section 5.4).

### 5.3.5 Star Catalog

The EBEX star catalog is a list of stars, originally copied from an online database, that has been reduced to contain only as many stars as is useful for EBEX, and then compiled into different forms for the pattern matcher. We first discuss the process of reducing the size of the catalog for EBEX, and then the structure of the catalog itself.

#### Reducing the Number of Stars

Full sky catalogs often provide lists of all stars down to a given apparent magnitude (dimness). For the purposes of the EBEX star catalog, however, we prefer to have a roughly fixed number of stars per field of view. This distinction allows for an important optimization. We require at least 10 stars per field of view for even the most sparse regions of the sky. If we were to simply set a magnitude limit on our catalog, then we would have to go fairly deep (to high magnitudes, which are very dim stars) to support the sparse regions of sky, which would result in large numbers of stars in dense regions near the galactic plane.

The process of reducing the number of stars from a magnitude-limited catalog involves two steps. The first is to determine the magnitude of each star in the EBEX star camera frequency band. The second is to perform the reduction, picking only the brightest stars in a given field of view according to the EBEX band magnitude. Here are the details of the two steps:

1. Determining the flux or the magnitude of each star in the band specific to the EBEX star cameras is helpful because the next step performs comparisons between star brightnesses, so determining the EBEX band magnitude for each star allows the comparisons



to be more accurate. The available generic star catalogs tend to provide star magnitudes (and fluxes, by extension) in a number of common frequency bands. By fitting a blackbody function to the fluxes at each of the available frequency bands, we can construct a model of a star’s flux as a function of frequency. We then multiply this function by the transfer function of the EBEX star camera (which is the CCD responsivity multiplied by the transmission of the red filter) and integrate it to find the flux of each star in the EBEX star camera frequency band. The result of this process is shown in Figure 5.4. We perform this operation for every star in the original catalog.

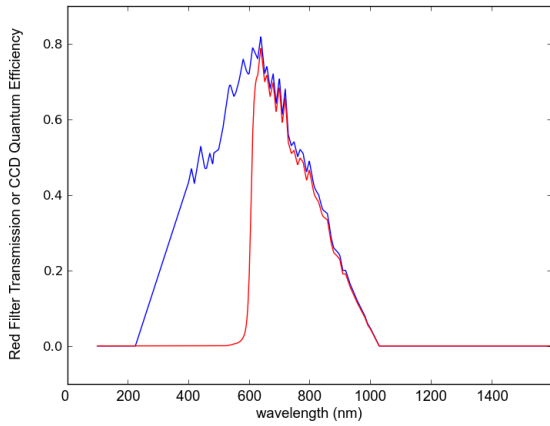
2. A simple and conservative approach is taken to reducing the number of stars down to a constant number per field of view. We developed a method that guarantees that every possible star camera field of view contains at least the N brightest stars (where N is tunable). We divide the sky into a many overlapping circular bins. We choose the bins to be small enough and overlapped enough so that any possible field of view contains at least one bin and every part of the sky is covered by at least one bin<sup>4</sup>. We then rank the stars in each bin so that the brightest star in a bin has a local ranking of 1. A star’s global rank, then, is the highest rank that it has in any bin. The catalog is then reduced to contain only stars down to a given global rank. This results in roughly homogeneous star density across the sky, while guaranteeing that any possible field of view has at least N stars in it. An example showing a resulting list of stars is shown in Figure 5.5.

## Catalog Structure

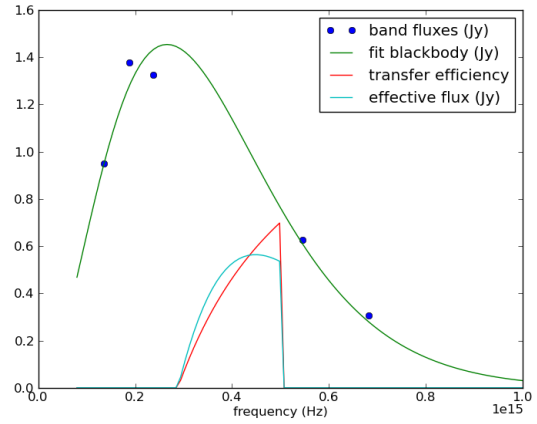
The list of stars is then organized into four different forms as an optimization for the pattern matching procedure: a flat list, pairs, triplets, and regions. We will now discuss these four forms.

---

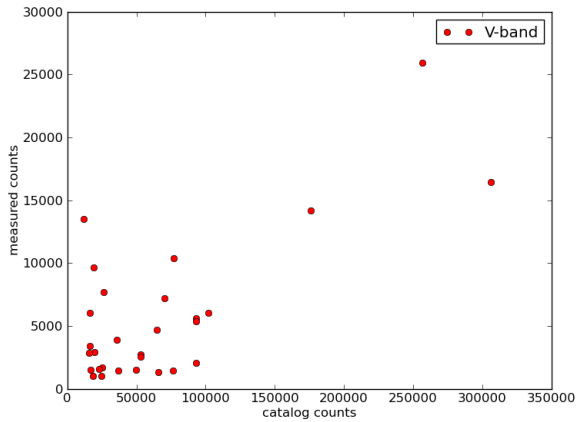
<sup>4</sup>For the EBEX star camera we use bins of radii  $1.003^\circ$  and spacing  $0.472^\circ$  for the star camera FOVs of size  $4.05^\circ \times 2.70^\circ$ .



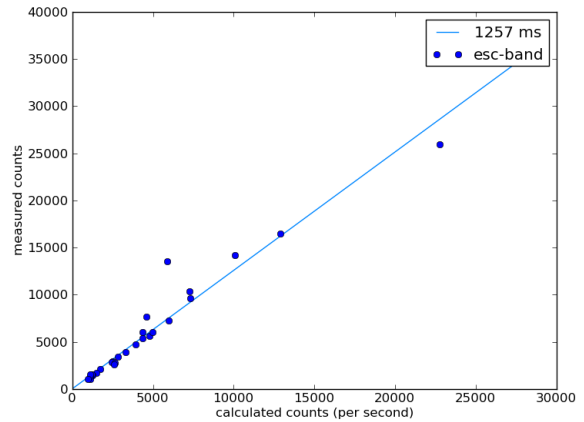
(a)



(b)



(c)



(d)

Figure 5.4: The process of determining the flux (and by extension magnitude) of a given star. Subfigure (a) shows the responsivity of the CCD (in blue) and the transfer function of the EBEX star camera (in red). The star camera’s transfer function is the CCD responsivity function multiplied by the red filter’s transfer function. Subfigure (b) shows the flux model of an example star in green, the EBEX transfer function in red, and the multiplication of the two in cyan. The integral of the cyan curve is the resulting flux of the star in the EBEX star camera band. The flux model comes from fitting a blackbody to the flux data points (shown as blue circles) at available band frequencies. Subfigure (c) shows the measured flux vs catalog flux for the stars in a single image before this procedure is applied (using the V-band), while subfigure (d) shows the same thing for the computed flux of the same stars in the EBEX band. The linear relationship in Subfigure (d) suggests an improvement for the stars in this image.

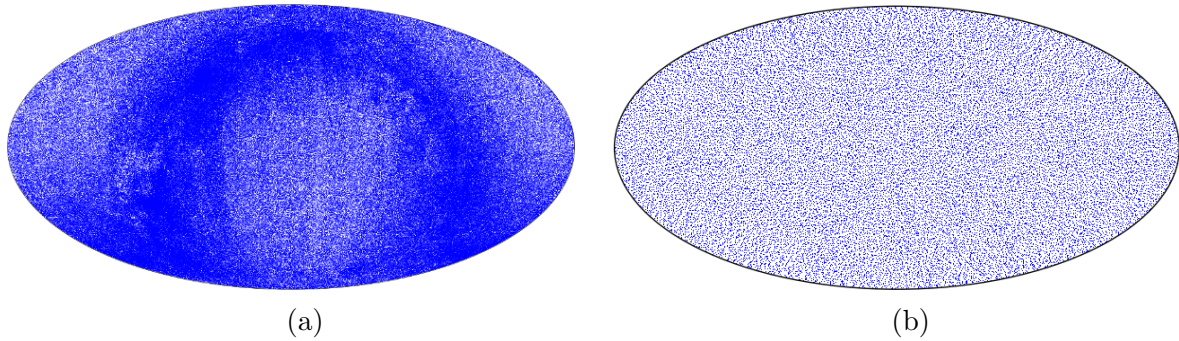


Figure 5.5: The catalog stars before (a) and after (b) reduction. The sky is shown as an equatorial Mollweide projection, and the blue dots are individual stars. It is apparent in Subfigure (a) that the Galaxy presents an inhomogeneity in the density of stars. Subfigure (b) plots an example list of stars that has been reduced down to only include stars with global rank 1. Note the homogeneity in this equal-area projection.

**Flat List** - All the stars in the catalog are stored in a simple text-based list. The list includes each star’s unique ID, right ascension, declination, flux, and name. This list is only actively used by STARS to display the name of identified stars, but it is also an intermediate step in producing a catalog, as it is the output of the reduction procedure described above.

**Triplets and Pairs** - The second step of the pattern matching algorithm, which involves loading triplet or pair permutations of stars from the catalog, would be slow if catalog permutations were generated dynamically. Instead the catalog pre-compiles every permutation that fits within a star camera field of view and stores it to disk.

Although there are more permutations of triplets than pairs, there are statistically fewer catalog triplets that can match a given source triplet than there are catalog pairs to match a given source pair. As a result, the pattern matcher can search for a given triplet faster than it can search for a pair, but the triplet section of the catalog takes significantly more disk space (and more time to compile during development) than the pair section.

This part of the catalog benefits the most from having a homogeneous star catalog since the number of triplets in a potential field of view is proportional to the cube of the number of stars in that field of view.

The pairs and triplets are binned into separate files for faster loading. Pairs are binned

by the distance in between the two stars, and the bins are stored as separate files, where the filename represents the bin's lower bound in arcminutes. Triplets are binned on two levels, first by the longest leg length and then by the second longest leg length of the triangle. The first level of binning forms a set of directories, where the directory names represent the bin lower bounds in arcminutes. Inside each directory the files represent the second level of binning, again with each filename representing the bin's lower bound in arcminutes. This binning allows the pattern matcher to quickly load all catalog permutations whose angular distances fall within specified ranges.

The triplets and pairs are stored sequentially in binary in each file. Each triplet permutation is stored as three angular distances and three parameters for each of the three stars. Likewise, each pair consists of a single angular distance and two stars.

**Regions** - Lastly, the catalog groups the individual stars into regions for step (2b) of the pattern matcher. This is an optimization so that in step (2b) the pattern matcher can quickly retrieve all the stars that may be nearby, without retrieving all the stars in the sky.

The regions are defined to be large enough so that for any FOV in the sky, the nearest region can be loaded and it will contain all the stars located in that FOV. For the EBEX field of view of  $4.05^\circ \times 2.70^\circ$ , we generate regions on a grid that is spaced by 10 degrees in declination, and spaced in right ascension by  $10^\circ / \cos(\text{dec})$ , and each region has a radius of  $10^\circ$ . The logic behind this spacing is shown visually in Figure 5.6, where any EBEX field of view is fully contained within the nearest circular region. The resulting pattern of regions is shown in Figure 5.7.

Each region is a file that begins with two binary float32 numbers: the right ascension and declination of the center of the region. It then contains a list of stars stored sequentially in binary, where each star consists of four parameters: an ID, ra, dec, and apparent magnitude.

STARS loads all the regions into memory at runtime for fast access.

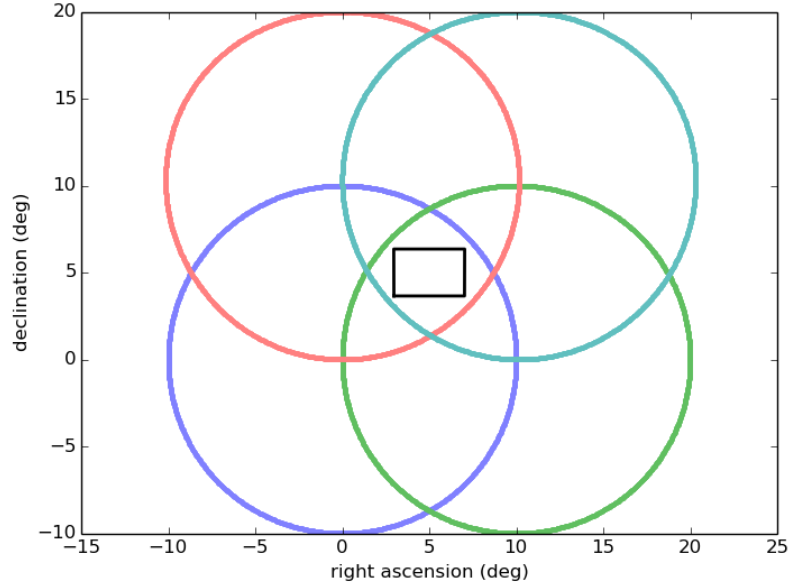


Figure 5.6: A zoom of part of the sky showing four adjacent circular catalog regions. Overlaid on the plot is a black rectangle representing an example EBEX star camera field of view. Given the size of the circular regions, and their spacing, any possible placement and rotation of the field of view will always be completely contained within the nearest region.

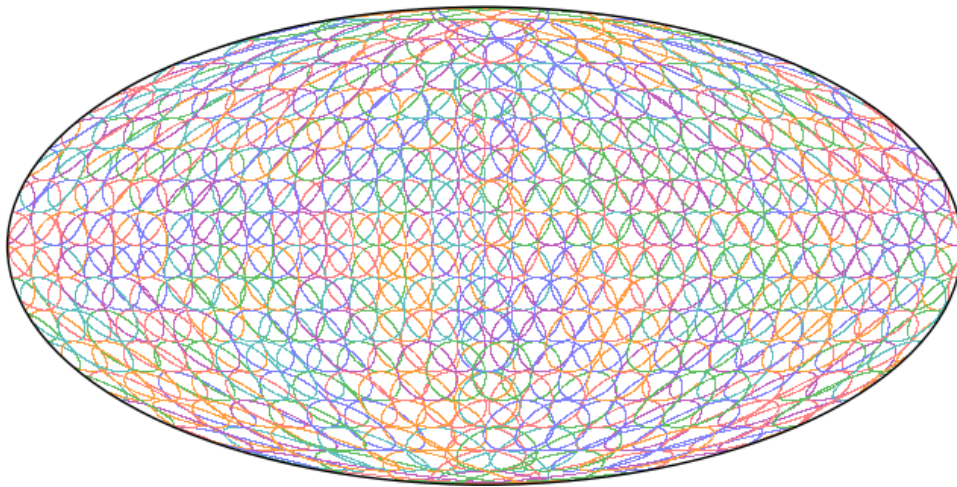


Figure 5.7: All the regions in the STARS catalog. Each region is a circle of radius  $10^\circ$ , and the regions are spaced on a grid separated by  $10^\circ$  in angular distance in both right ascension and declination. Given these parameters, the regions overlap. There is additional overlap at right ascension zero (an imaginary vertical center line in the plot) due to the fact that the distance around the sphere is not an integer multiple of the grid spacing at every declination.

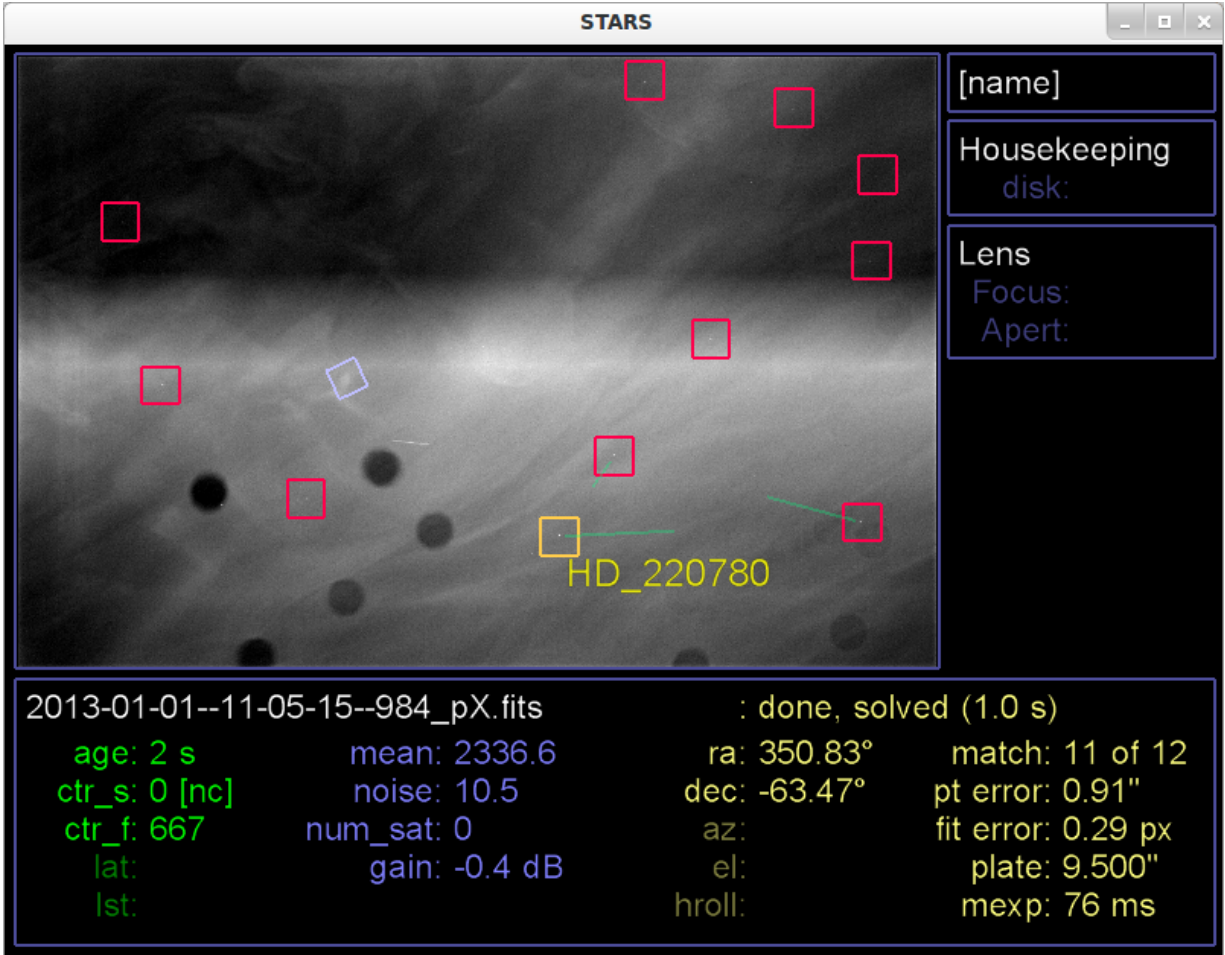


Figure 5.8: A screenshot of the STARS display, running post-flight on a ground computer on an image from the EBEX Antarctic flight.

### 5.3.6 Displaying

During the first day of flight when the gondola is in range for line-of-sight communication, and during pre-flight testing, the user can watch the display output of the star camera computer as captured from the VGA port. This video display can be crucial for debugging when the experiment first reaches float altitude. It is also useful during testing.

Figure 5.8 is an example screenshot taken during post-flight processing that shows much of the communication that the display provides. On the image itself, the sources are initially identified with rotating blue squares. When a solution is found, the squares around matched

sources change to fixed red squares, while the brightest of the matched sources is identified by name and a fixed yellow square. As the pattern matcher searches the catalog for matches from a set of source triplets, the outline of a green triangle is traced out as a progress bar connecting the three sources in the triplet. In this case, the solution was found just before making it half way through the list of potential catalog triplets for the three sources marked by the incomplete green triangle. If multiple source triplets are tested, old triangles fade away over time in order to not obstruct the image. To the right of the image various housekeeping, lens, and network connection information is displayed, though not in this post-flight run because it is not being executed on a star camera computer. Below is the status information for the image. Administrative information is displayed in green, statistics are displayed in blue, and information about the solution found is displayed in yellow.

The star camera image shown in Figure 5.8 contains 11 sources matched to catalog stars. The image also contains many of the undesirable features seen in flight: the dark circles primarily located in the bottom left of the image are dust spots in the optics; there is a large gradient due to internal reflections in the optics; there is a vertical slab of sharp reflections from the CCD itself on the left side; there is either a satellite streaking past the image or a cosmic ray CCD hit just above one of the dust spots; and lastly, this image contains mesospheric clouds. The blue square represents a source that was not matched; in this case it is a false positive picked out from a piece of cloud.

Not shown in Figure 5.8 is additional information that the display can also be commanded to show:

- A zoomed region of the image.
- The status of the autofocus routine, which includes a plot of the focus metrics<sup>5</sup> versus the focus position.
- The state of all the solution requirements.

---

<sup>5</sup>See Section 5.3.7

### 5.3.7 Imaging

#### Camera

STARS has a camera object that reads images from the camera controller over IEEE 1394, saves them to disk, and then shares them with the solver thread. The camera object runs this functionality in its own thread because downloading an image from the camera controller requires a blocking I/O call, and because the camera controller buffers should be flushed into STARS quickly, before the next set of triggers occurs.

STARS also has a camera object that loads images from disk to be used during testing and post-flight processing. Only one of the two camera objects is created when STARS runs, as determined by one of the parameters in the settings file.

#### Multiple Exposures

STARS has the ability to capture and process multiple exposures taken consecutively. The hardware in the EBEX star cameras allows for up to four exposures separated by gaps as short as 190 ms. Multiple exposures are employed when the user would like to capture more light for improved signal-to-noise, but is running up against the saturation limit of the CCD. STARS reads multiple exposures and effectively co-adds them for image processing. Figure 5.9 shows the results of a test in which multiple exposures were used to increase the statistical likelihood of solving an image.

When co-added multiple exposures are used on EBEX, they tend to be captured over a relatively large amount of time ( $\sim 800$  to  $1300$  ms) compared to a normal exposure ( $\sim 300$  ms), so the motion PSF feature discussed in Section 5.3.3 is more important in this situation. Figure 5.10 shows the pattern obtained when capturing multiple exposures while the gondola is rotating.



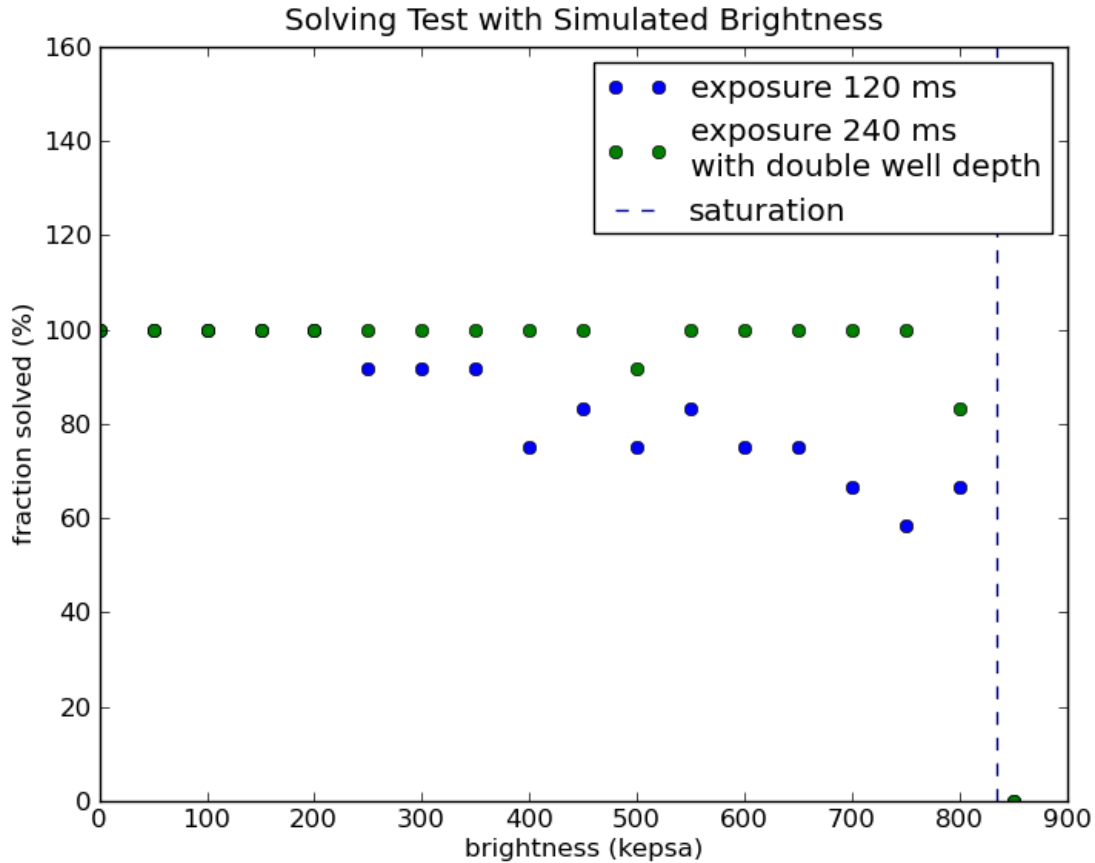


Figure 5.9: The probability that an image is solvable given the level of sky brightness. 12 images were captured on a dark night with 120 ms exposure times. Different levels of sky brightness were added to each image, using the brightness simulator discussed in Section 5.2.5, to test whether it was still solvable at that level. Shown in this plot is the fraction of the 12 images that were able to solve at that level of sky brightness (blue circles). The test was then repeated with effective 240 ms exposures, which were actually sets of two co-added 120 ms exposures (green circles). The longer effective exposure time produces better results because of increased signal-to-noise. A 240 ms exposure would normally saturate at 417 kepsa, but with two separate co-added exposures the saturation limit doubles to 833 kepsa, as indicated by the dotted blue vertical line.

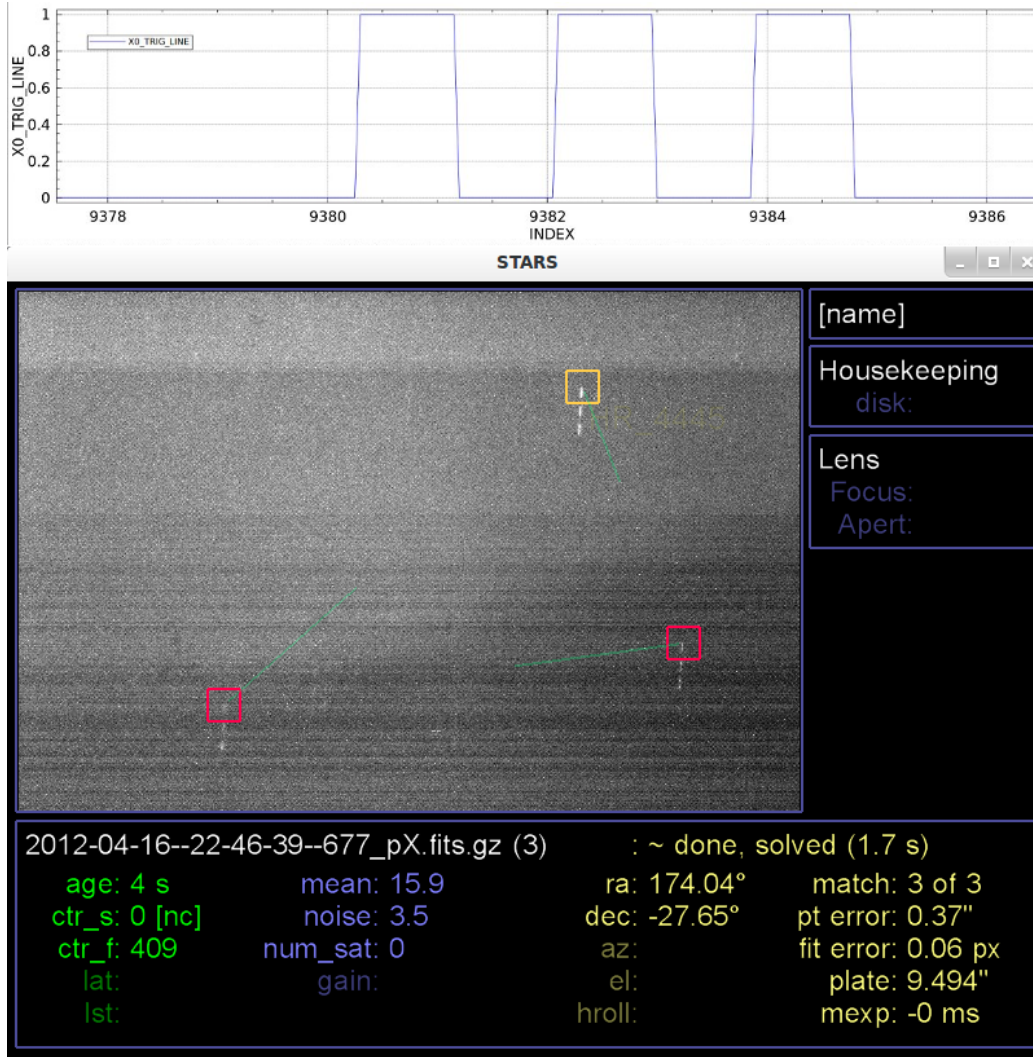


Figure 5.10: Capturing an image with multiple (triple) exposures while the gondola is in motion. The top plot shows a strip chart of the Star Camera 0 trigger line, which is the electronic line that drives the mechanical shutter in the camera. When the line is 1 the shutter is open, and when it is 0 the shutter is closed. Below the plot is the image captured. Each star shows the characteristic set of three streaks with gaps in between that appear for each star due to the shutter opening and closing three times. This may occur on smaller scales during the flight, which is why the Motion PSF source finding method is equipped to handle it. Note that the source finder was in motion PSF mode for this run, and as a result only the top end of the 3 streaks is identified for each source, even though the combined signal from all three streaks is used so that dimmer sources can be identified than with a single exposure.

## Lens Control

The lens thread controls the focus and aperture by communicating with a physical lens controller over a serial port. Lens requests typically originate in the networking thread - as ground commands are passed through the flight computers - and terminate in the lens thread. Once the lens thread has completed the request, the result is sent back to the networking thread to be sent back to the flight computers and ultimately downlinked.

However, the main thread should also know the state of the lens in order to display it to the user, and the camera thread should also know the state of the lens in order to store it in the saved image headers. The issue to be overcome is that these threads are asynchronous with the lens thread, and in fact the lens thread may not even know the state of the lens between when it sends a command over serial to the physical lens controller and when it receives a confirmation, which can be up to a couple seconds. The solution involves counters and passing lens requests/results sequentially.

When the flight computers make a lens request, they also assign it a count. This request (with a specific count) gets passed into the STARS networking thread, which passes it to the main thread, then the camera thread, and finally the lens thread which takes action. Once the action is complete, a result with a corresponding counter gets passed back to the flight computers via the same path. In this way, for example, if the main thread sees a focus request go to the lens, but has not yet seen the corresponding focus result come back from the lens (i.e. the counters differ), then it considers the focus value to be unknown, and displays it as such to the user. This does not allow every thread to immediately know the state of the lens, but it does allow every thread to know when action is being taken on the lens and therefore its state may be invalid, which is all that is necessary. This flow of requests and results can be seen in Figure 5.1.

This is an example of the way memory is shared between processes, and the emphasis placed on sharing accurate information with the user. The same method is used for the

camera gain, which is controlled by the camera thread, and the status of the autofocus routine, which both originates and terminates in the lens thread but takes a round-about path through the other threads for this reason.

## **Autofocus**

The autofocus works by stepping the lens through a range of focus positions and calculating metrics on an image taken at each step. STARS employs two separate metrics, described below, but will prefer the focus position found by the second metric if it is available.

The first metric is the peak value of the brightest source in the image, whether or not the solver finds a solution. The peak value of a source that represents an actual star is a good metric because, as the focus moves away from the optimal position, the peak value decreases with the square of the width of the source. If the gondola can point at a fixed patch of sky, such that over the course of an autofocus routine (minutes) the brightest star remains in the frame and another brighter star does not enter the frame, this is an effective metric.

The second metric relies on the solver finding a solution, and is the peak value of every source that corresponds to a matched star. This may appear to cause a circular dependency, in that the autofocus is run to obtain solvable images but solvable images are required for the autofocus to work. However it does not create a circular dependency because the STARS solver can find solutions when the focus is relatively far from the optimal position, and even when STARS is finding solutions we may still want to improve the focus (e.g. if we want more frequent solutions, or if in the future we will be pointing at a more challenging patch of sky that has more noise and artifacts). When a solution is found, the metric for each star identified is stored. Over the course of many focus steps, STARS actually builds multiple second metrics, one for each star, and it does so regardless of whether the stars are the brightest objects in the frame or whether the frame moves around. This metric has the advantage that it does not require a highly stationary gondola to determine the best focus, though it does require the gondola to remain in the same region, or revisit part of the same

region, as the focus steps past the optimal point.

### 5.3.8 Networking

STARS accepts network connections from both flight computers. Each flight computer attempts to establish a connection to each star camera every 9 seconds if it is not already connected. Once connected, FCP and STARS send updates to each other once every 0.5 s. When a gondola operator sends a command from the ground, FCP receives the command and passes it on to STARS over the network connection. STARS shares solution and other information with both flight computers, but only accepts commands from the flight computer that is in-charge, which is a designation assigned by a watchdog timer as the less-recently rebooted flight computer.

Counters are also used to protect STARS from spurious state changes in FCP. Sometimes FCP will change the value of a parameter even if the ground operators did not command such a change. These state changes have been observed many times, but would require significant changes in FCP to eliminate. To decrease the probability that STARS accepts a spurious command from FCP that did not originate from a ground command, it checks a timer to ensure that any particular state change from FCP coincides temporally with the corresponding ground command. With this safety mechanism, no spurious commands have been observed.

To help with potential in-flight debugging, STARS shares 44 variables with FCP to be downlinked to the ground operator. These variables include administrative counters, housekeeping measurements, the status of camera and lens parameters, and statistics and solving information about the current image.

## 5.4 STARS - Successful In-Flight Performance

Throughout the 11-day EBEX Antarctic flight STARS never crashed, and it reliably served pointing solutions to the flight computers with minimal intervention despite considerable challenges, which we will discuss in the following paragraphs. The fact that STARS required minimal intervention was especially important during sparse downlink times on the EBEX flight.

One of the challenges that STARS handled involved misinformation from other subsystems. A system clock failure on one of the flight computers and eventual failures or faulty readings from all three GPS systems caused FCP to share incorrect pointing information with STARS for various reasons and at various times throughout the flight. Nevertheless, STARS continued to provide pointing solutions due to the fast lost-in-space pattern matching capabilities discussed in Section 5.3.4.

Due to an issue with an azimuth motor controller the gondola was unable to remain stationary for more than one second at a time. A stationary gondola for at least three minutes is necessary to perform safe re-focusing procedures, in which we manually change the focus by small amounts to find the best focus position.<sup>6</sup> With no regular autofocus, due to thermal variations in the optics, the point spread function expanded and contracted from 2 px to 12 px in diameter over the course of hours and days. Nevertheless, STARS continued to solve on these out-of-focus sources due to the source finder's smoothing and leveling procedure discussed in Section 5.3.3, and because the resulting loss in signal-to-noise was recovered by the multiple exposures feature discussed in Section 5.3.7.

Another consequence of the azimuth motor controller issue was that the acceleration at scan end-points and the state variables in the scan control loop was different than in the original scan design. STARS continued to operate in these new conditions due to the architecture

---

<sup>6</sup>This is preferred over the autofocus routine which incurs more risk because it changes the mechanical focus by larger amounts. Considering this risk, and that the source finder continued to work on the out-of-focus sources, we opted not to test the non-stationary autofocus feature in flight.

described in Section 5.2 in which the camera and solver threads operate asynchronously on data whenever it becomes available, independent of the gondola's scan state.

Finally, STARS prevailed against various image artifacts: dust spots, mesospheric clouds, satellites and/or cosmic ray hits, optical vignetting, external reflections from the Sun, sharp internal reflections of the CCD, and broad gradients from internal reflections. The leveling and flux sorting procedures discussed in Section 5.3.3, along with the selective masking feature (Section 5.3.3), helped limit the number of false positives extracted by the source finder. The false positive sources that did get through did not prevent the pattern matcher from finding a solution due to the fast search algorithm, which allowed for many triplets to be tested in a matter of seconds, and due to the ability of the pattern matcher to identify all the sources in an image, rather than just those in the triplet. The multiple exposures feature enabled the source finder to find enough sources to meet this stringent matching criteria due to the increase in signal-to-noise.

# Chapter 6

## Real-Time Sky Maps with “QuickLook”

### 6.1 Description

*Quicklook* is a software package that processes a subset of the data that has been downlinked during flight into a map. The map serves two primary purposes. The first is to measure the sensitivity of the detectors to astronomical sources as a diagnostic tool. The second is to compare the location of the calibrator in the map to its true position in the sky in order to fine tune the pointing offsets between the pointing sensors and the microwave telescope boresight.

The quicklook software package consists of a front-end user interface and a back-end server. The user interface is a web page that shows a map created from the flight data and presents various options that specify what data to use in the map and how to use it. The map is displayed using the Google Maps library, which works by stitching together a number of “tiles”, or square images that constitute the map when placed side by side. The back-end server contains a map maker that is responsible for creating these tiles.

In this chapter we will discuss the back-end server, the user interface, and the testing and



performance of quicklook. In Chapter 10 we will discuss how the map maker was extracted from quicklook and used as the primary map maker for EBEX analysis purposes at least up until the time this document was written.

## 6.2 Back-End Server with Naive Map Maker

The back-end server is primarily responsible for delivering tiles, which are small square images, to the front-end interface, which assembles them into a cohesive view of the map. A Python program called “get\_tile.py” contains the core of the map making functionality. Based on the URL parameters passed by the user interface, this program loads detector timestreams and all relevant pointing timestreams from the selected section of flight. It then performs the necessary pointing conversions, first to the desired celestial coordinate system, and then through a Mollweide projection onto the requested tile. It then filters the detector timestream data to remove low frequency drifts (with a filter that we call a “destriper”) and high frequency noise (with a “smoother”). The destripping filter is a moving median removal filter that acts as a high pass, with a default window size of 2.10 s. The smoothing filter is a moving mean filter that acts as a low pass, with a default window size of 0.32 s. The default filter parameters were chosen before flight based on predictions of the scan strategy and detector noise frequency profiles. Finally, the tile program adds detector samples to tile pixels, normalizes each pixel by dividing by the number of detector samples added, and serves the tile back to the map in the user interface.

Depending on the user options, it can take many seconds for the map maker to generate a single tile out of hours of flight data. The longer it takes to generate tiles, the slower the user experience is, which can waste time during flight. We therefore implemented two important optimizations to speed up get\_tile.py. First, the program saves the tiles to disk when they are generated so that if the user happens to request the same tile again in the future, it can be loaded from disk instead of being dynamically generated (this is called memoiza-

tion). Second, various numerical functions are moved from Python to C++, including the timestream filters and the binning algorithm.

The server also has three other responsibilities outside of hosting `get_tile.py`:

- **Color Bar** - The map that is displayed is colored based on the intensity of the signal in each pixel. For the user to quantitatively know the level of signal in a pixel, they must also have access to a color bar. A Python program called “`get_color_bar.py`” creates the color bar as an image and serves it back to the user interface to be displayed alongside the map.
- **Logging** - The map making program `get_tile.py` writes log files to disk. When developing and testing the quicklook software, it is useful to be able to view these log files. A separate Python program called “`get_info.py`” loads these log files and serves them to the user interface, so that the user interface can display the logs. This speeds up development and testing of quicklook itself.
- **Focal Plane Visualization** - One of the options that the user can specify is a list of detectors that they would like to include in a map. In some cases the user chooses the detectors based on their location on the focal plane. For example, when scanning the calibrator, the user might want to view maps produced by a detector on the edge of the focal plane to ensure that it is scanning past the calibration source. When the user enters a list of detectors into the user interface, they can click on a link to view where the detectors are located on the focal plane. We therefore have a Python program called “`get_detector_visualization.py`” that produces a plot of the focal plane, highlights the specified detectors, and serves it back to the interface.

### 6.3 Quicklook User Interface with Google Maps

The user interface displays the map and allows the user to specify parameters that control how the map is built. It is written in HTML, JavaScript, and CSS.

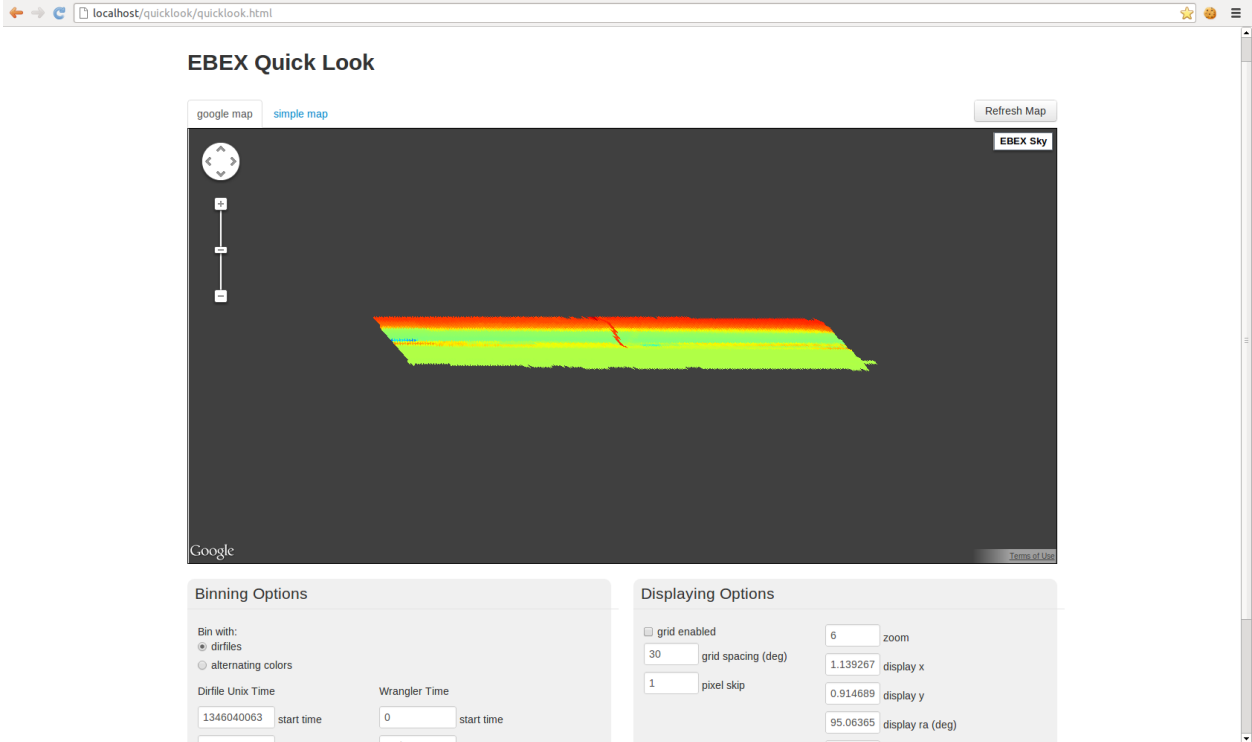


Figure 6.1: Example screenshot of the quicklook user interface. The map display is featured prominently in the screenshot. The screenshot was captured during ground tests, when the gondola was performing a science scan and the detector data was simulated to create a non-uniform map.

### 6.3.1 Map Display

The map display uses the Google Maps API (Application Program Interface) [52] to display the map data as a Google Maps object. Although most people are familiar with Google Maps as a way of displaying a map of the Earth in a Mercator projection, the library can be used to display any tile-based visualization, which in our case is a Mollweide projection of the sky. The Google Maps scrolling and zooming support is natural for navigating around large coverage areas and zooming in on small sections. When the map is initialized or the user interacts with the map via the standard methods (scrolling, swiping, zooming, pinching, etc), the map requests tiles from the back-end server to display.

In order to display an EBEX sky map, the Google Maps object is instantiated with a

custom `ebexMapType` object, whose URL callback defines a URL for map tiles that points to an address on a ground station that hosts the back-end server (as opposed to, for example, a URL that returns tiles of the Earth). The URL contains parameters that define the tile dimensions along with various user options on how to produce and display the map. These parameters come from the user options discussed in the next subsection, and are accessed by the URL callback with jQuery<sup>1</sup>. Figure 6.1 shows the map display.

### 6.3.2 User Options

The user options are divided into two categories: binning options that control how the map maker should bin detector samples on the map, and displaying options that control how the map is viewed.

The part of the interface that contains the binning options is shown in Figure 6.2. The “Coordinate System” radio buttons select which celestial coordinate system should be used. The “Solution Source” selects which source the pointing information should come from, whether it be the primary pointing stream or one of the individual star camera pointing streams. Selecting a star camera pointing stream here can simplify the process of calculating the pointing offsets between the star camera and the microwave telescope boresight. The “Dirfile Unix Time” and “Wrangler Time” boxes are two different methods of selecting which subset of data to include based on time in flight. The “Bin With” option provides “alternating colors” as an alternative to normal Dirfile map making for the purposes of testing the Google Maps API component, and the “healpix nside” parameter allows the user to specify the pixel sizes in the map via the standard HEALPix binning scheme<sup>2</sup>. The “Bolos” box allows the user to specify the detectors that they would like to include in the map, and the “where?” link allows the user to see a visual focal plane layout indicating where their selected detectors are located on the focal plane. The “Filtering” options are to specify

---

<sup>1</sup>jQuery is a javascript library that simplifies development [53].

<sup>2</sup>HEALPix is an algorithm to pixelize a sphere into equal area pixels, and stands for Hierarchical Equal Area isoLatitude Pixelization [54].

Binning Options
Apply
 Auto Apply

Note: may require lat and lst for conversions

<p><b>Coordinate System</b></p> <p><input type="radio"/> equatorial</p> <p><input checked="" type="radio"/> horizontal</p>	<p><b>Solution Source</b></p> <p><input checked="" type="radio"/> pointing solution</p> <p><input type="radio"/> x0</p> <p><input type="radio"/> x1</p>
<p><b>Dirfile Unix Time</b></p> <p><input type="text" value="end"/> start time</p> <p><input type="text" value="end"/> end time</p>	<p><b>Wrangler Time</b></p> <p><input type="text" value="0"/> start time</p> <p><input type="text" value="end"/> end time</p>
<p><b>Bin with</b></p> <p><input checked="" type="radio"/> dirfiles</p> <p><input type="radio"/> alternating colors</p> <p><input type="text" value="512"/> healpix inside</p>	<p><b>Bolos (comma separated)</b></p> <p><input type="text" value="64-2-11,54-1-4"/></p> <p><input type="checkbox"/> use offsets <a href="#" style="color: #0070C0;">where?</a></p>
<p><b>Filtering</b></p> <p><input type="text" value="2.10"/> destripping window size (s)</p> <p><input type="text" value="0.32"/> smoothing window size (s)</p>	

Figure 6.2: The binning options as displayed on the Quicklook user interface.

window sizes for high- and low-pass filters that are applied to the detector timestreams before binning.

The displaying options are shown in Figure 6.3. A redraw button allows the user to force the map to redraw, which may be useful if more data has become available in the time range specified. The grid related options control the overlay of grid lines on the map. The “pixel skip” option is an optimization that reduces the resolution in the maps to speed up tile delivery from the server. The “Positioning” options are for manually defining the center of the map and the zoom level. Specifying the positioning options is often less intuitive than

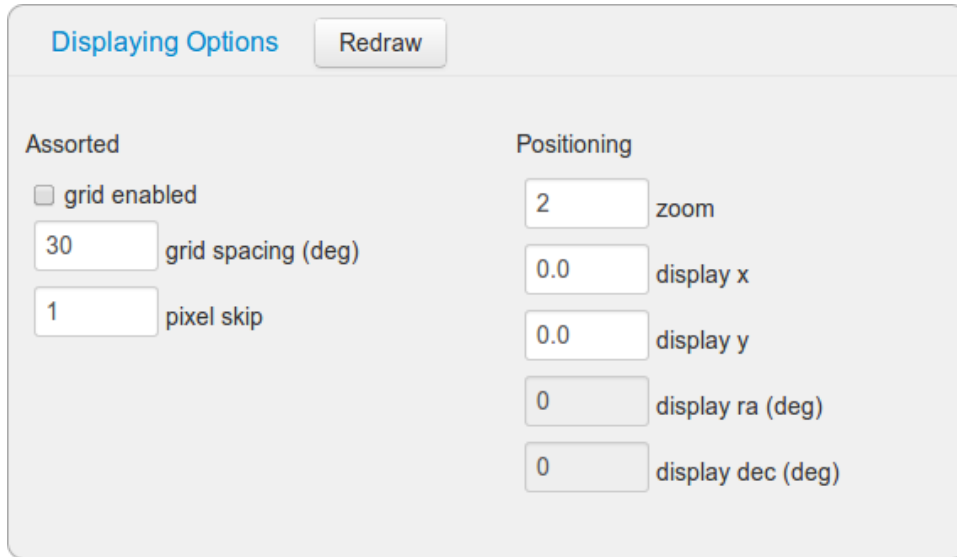


Figure 6.3: The displaying options as displayed on the Quicklook user interface.

scrolling and zooming on the map with the mouse, but it can be useful if the user needs to revisit the exact same view after navigating away.

## 6.4 Testing

The quicklook software package was tested before flight on both simulated data and real data. Though the real data did not contain interesting signals, we tested it anyway to ensure that the software would work with realistic scan parameters and detector timestreams. To simulate flight data, a simulation of the FCP scan function was written and then run to produce pointing timestreams from a science scan. We then used the pointing timestreams to create detector timestreams by sampling a Gaussian function at each timestep, using the pointing datapoints as parameters. We also simulated other functional forms, such as gradients, to see how the map maker would handle them.

## 6.5 In-Flight Performance

The quicklook software performed correctly during flight, and the gondola operators used it to check sky coverage. It correctly combined the detector data with the pointing data to make maps according to the user preferences. The ground operators used it to check sky coverage, in particular to see how many times the focal plane passed through the calibrator and to check on the science scan pattern. The ground operators also tried to use it to see signals from the calibration source (RCW 38) and the Galaxy, but unfortunately a bug in the FCP telemetry functions corrupted all the usable detector timestreams that were downlinked.

# Chapter 7

## 2012 Antarctic Science Flight

### 7.1 Flight Details

EBEX was launched on 2012-12-29 at 00:30 UTC from Williams Field, an air field near McMurdo Station in Antarctica. It rose to an altitude of 120 500 ft and remained between 110 500 ft and 120 500 ft (between 33.7 km and 36.7 km) for the next 11 days, during which time it collected scientific data. After 11 days the receiver exhausted its cryogenics and the scientific portion of the flight ended on schedule. For the next 14 days the telescope remained afloat with most systems powered down until it drifted over an acceptable landing site, at which point the flight was terminated. The altitude profile for the first 11 days of flight is shown in Figure 7.1, and the geographic profile for the entire flight is shown in Figure 7.2.

The telescope was terminated over a plateau a few hundred miles from McMurdo station. The hard drives were recovered in the following weeks, along with some of the more expensive and accessible components. The remainder of the telescope was recovered one year later.

For the most part the 2012 Antarctic flight was successful. Many aspects of the flight went as planned, though some did not, as is often the case in ballooning. There were several issues with various subsystems, some of which have been discussed in Chapter 5. Two subsystems had issues in flight that had notable effects on the data:



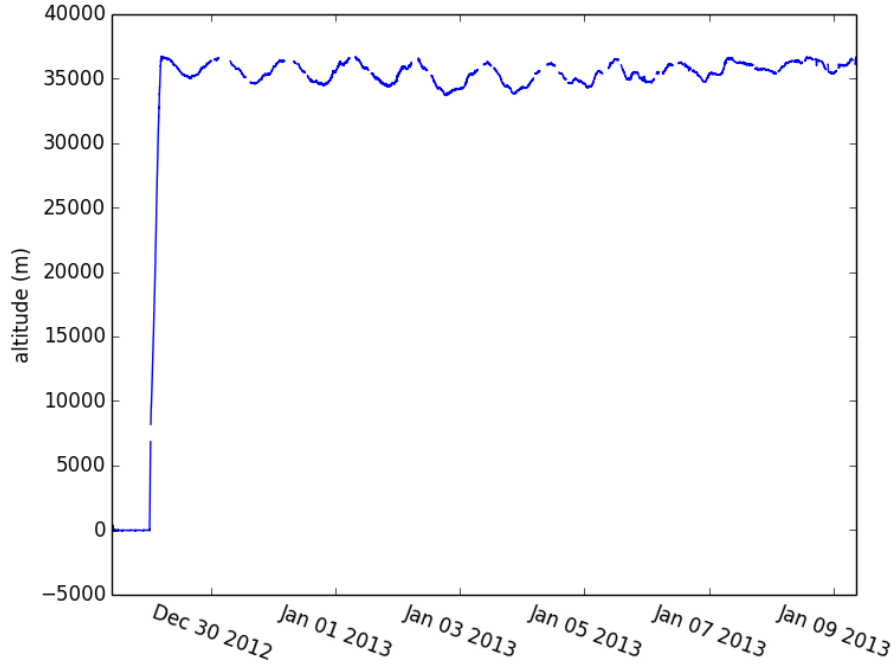


Figure 7.1: Altitude profile for the first 11 days of the EBEX 2012 Antarctic flight.

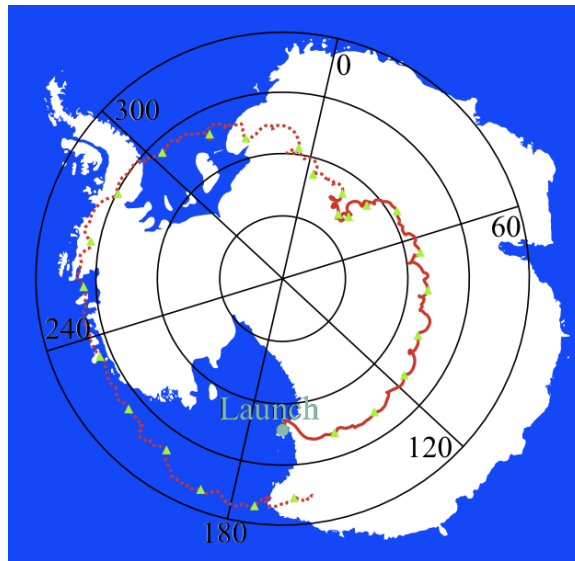


Figure 7.2: Geographic profile of EBEX in its 2012 Antarctic flight. The Antarctic continent is shown in white overlaid with a geographic grid with the lines of longitude labeled. The EBEX flight path for the first 11 days, during which time scientific data was collected, is shown in solid red. The flight path for the remaining 14 days is shown in dotted red.

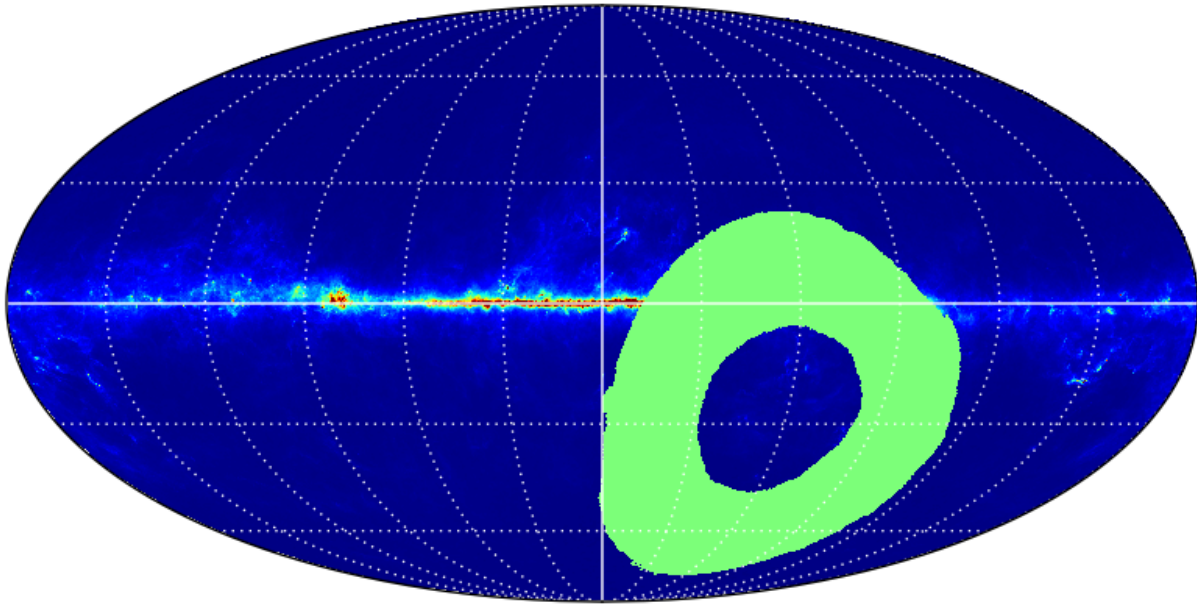


Figure 7.3: Sky coverage from the 2012 Antarctic flight, shown in galactic coordinates. The patch is circular, centered on the equatorial South pole, covers 5735 square degrees, and has a width of roughly  $27^\circ$  due to the telescope's latitude being  $10^\circ$  from the South pole and the focal plane being  $7^\circ$  wide. The calibrator is contained within this patch.

- Pivot Motor Controller** - The pivot motor controller overheated due to a flawed thermal design, which limited azimuth control. The telescope therefore rotated freely in azimuth for the majority of the flight, which resulted azimuth motion with two components. First, the telescope underwent full  $360^\circ$  rotations every 15 to 60 minutes. Second, oscillations of  $15^\circ$  to  $90^\circ$  with 80 s periods were superimposed on the rotations. To perform science scans, we drove the telescope to a fixed  $54^\circ$  elevation. To perform calibrator scans, we followed the calibrator's elevation when the telescope crossed the calibrator in azimuth, and targeted parts of the focal plane that had active detectors to make the most out of the limited number of crossings. The resulting sky coverage is shown in Figure 7.3. Another consequence of the  $360^\circ$  rotations is that the flight computers were occasionally exposed to direct sunlight, and had to be shutdown for one or two hours a day to cool.

- **FCP Downlink** - There were two issues with the downlink functionality in FCP (the *Flight Control Program*). First, an issue with data prioritization prevented continuous downlink of timestreams outside of line-of-sight communication (though they could be downlinked intermittently). The second issue affected the detector timestreams specifically. As discussed in Chapter 6, the *Quicklook* software was meant to be used in flight to confirm that the detectors were sensitive to astronomical sources and to refine the calibration offsets between the pointing sensors and the microwave boresight. To downlink enough detector and pointing data over the limited bandwidth to make maps during flight, the detector timestreams need to be downsampled. The FCP functions that downsampled these timestreams contained a bug that corrupted the data, making them unusable for map making. Both of these issues had an impact on the total observing time and resulted in limited coverage of the calibrator.

## 7.2 Data Extraction

All 18 hard drives (16 flight computer drives and 2 star camera drives) were recovered from the payload at its termination site. The data consists of numerical timestreams, flight computer log files, star camera images, and star camera log files. The timestreams contain pointing sensor data, detector data, half-wave plate data, and general gondola and receiver housekeeping data.

The timestreams and flight computer log files are stored on 16 disks that were accessed by the flight computers. These disks were divided into two separate pressure vessels, each with 8 disks and accessed by only one of the flight computers. The two vessels were intended to be redundant and store identical data, however due to a bug in FCP coupled with a system clock failure, 43% of the data in one of the vessels was overwritten during flight. One of the vessels contained 704 GB of data, while the other contained 402 GB of data that is mostly redundant with the first.

The star camera images and log files were recorded to the two star camera hard drives. The two star cameras are redundant in terms of function, however the two star cameras were pointed at different elevations by design and therefore collected non-identical data. Each star camera hard drive contained 62 GB of flight data.

In total the hard drives contained to 826 GB of unique (non-redundant) data. The 18 hard drives were imaged twice onto other hard drives, and these three identical sets of flight data were carried off-continent by three separate EBEX collaborators. The carriers also brought the hard drives from the ground station computers, which contain data that was downlinked during flight and pre-flight testing data. All of the recovered flight data made it safely back to collaborating institutions in North America and onto RAID arrays<sup>1</sup> for safekeeping.

---

<sup>1</sup>RAID stands for Redundant Array of Inexpensive Disks [55], and is a storage technology that is commonly used for the purposes of redundancy.

# Chapter 8

## Data Structures for Post-Flight Analysis

### 8.1 Introduction

During flight the *Flight Control Program* (FCP) receives timestreams of data from 31 different electronics components (described below) and writes the timestreams to disk in what we call “framefile streams”. A framefile stream is a set of files called “framefiles” that are sequential in time, with each framefile containing 30 minutes to an hour of data. A framefile is a file that contains a series of “data frames” that are sequential in time, each containing between 10 ms and 1 s of data. Each data frame contains samples from multiple timestreams, and may contain more than one sample of a given timestream. The number of samples of a timestream that a data frame contains is known as the SPF, or samples per frame. Not every timestream has the same SPF in a given data frame, but the timestreams inside a data frame are synchronous in that they each contain an integer multiple number of samples. We store many timestreams at a low SPF to limit the size of the data when a high sample rate is not necessary.

As an example consider two timestreams, “longitude” and “azimuth”, that belong to the

same framefile stream, which we will call “ACS” for now. These data frames are written to disk at a rate of 5.008 Hz, so we say that the ACS frame rate is 5.008 Hz. Each ACS data frame contains one sample of longitude data and 20 samples of azimuth data. Therefore the sample rate of the longitude timestream is 5.008 Hz and the sample rate of the azimuth timestream is 100.16 Hz.

The 31 electronic components that provide data to FCP are:

- **1 ACS bus** - A bus inside the Attitude Control System (ACS) crate contains data related to the attitude control system along with some housekeeping data from other subsystems.
- **2 HWP readout boards** - Two boards are responsible for reading half-wave plate (HWP) data.
- **28 bolometer readout boards** - Each of 28 bolometer readout boards sends two types of data to the flight computers: bolometer signal timestreams and bolometer settings timestreams. The settings timestreams are generated asynchronously from the bolometer signal timestreams.

FCP therefore stores 59 framefile streams to disk: 1 “ACS” framefile stream, 2 “HWP” streams, 28 “Bolo” streams that contain the bolometer signal timestreams, and 28 “Slow Streamer” (or “SS”) streams that contain the bolometer settings timestreams. These framefile streams are all generated asynchronously with respect to each other. The 59 framefile streams are listed in Table 8.1, along with each streams’s data frame rate and the SPF rates of the timestreams inside them.

The timestreams in the raw framefiles need to be processed for post-flight data anylisis for three reasons:

1. The framefile format described above is convenient for storing data, because for a given framefile stream FCP receives samples from multiple timestreams simultaneously and can write them all to a single file instead of having a separate file open for each

Table 8.1: List of framefile streams. The timestreams within a framefile stream are synchronous with each other, but each framefile stream is asynchronous with the framefile streams. There are 59 total, as listed here. Some contain timestreams stored at different samples per frame (SPF) rates, so the existing SPF rates are listed. The Bolo and HWP frame rates are more precisely defined as  $25.0 \times 10^6 \text{ Hz}/2^{18}$ .

Data Stream	Frame Rate	SPF rates
ACS	5.008 Hz	1, 20
Bolo Board 50	95.367 Hz	2
Bolo Board 51	95.367 Hz	2
...		
Bolo Board 77	95.367 Hz	2
HWP Board 78	95.367 Hz	1, 32
HWP Board 79	95.367 Hz	1, 32
SS Board 50	0.990 Hz	1
SS Board 51	0.990 Hz	1
...		
SS Board 77	0.990 Hz	1

timestream. For data analysis, however, the user generally wants to work with only a few timestreams at a time, without having to load all the timestreams in a framefile. Therefore for analysis it is more preferable to have each timestream be stored in its own file, as it is in the Dirfile structure defined in [56].

2. FCP stores each framefile stream across multiple sequential framefiles so that an individual framefile does not become too large. In many cases, two sequential framefiles are meant to be adjacent in time, and therefore it is preferable that the timestreams from the framefiles be merged together.
3. The data is stored on two separate sets of disks, one for each flight computer, and although much of the data is redundant between the two, neither set contains all the data because the computers may be shutdown or rebooted at different times during the flight. Therefore the two datasets must be merged into complete datasets.

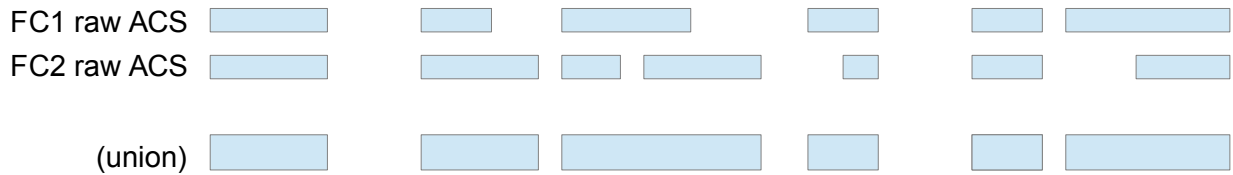


Figure 8.1: Diagram representing the union of ACS data from both flight computers. In this diagram time increases to the right, and each filled box represents a dirfile that contains multiple timestreams that are continuous for the length of the box. The top two lines of data are drawn to demonstrate a situation in which the flight computers stored some overlapping data and some unique data. The bottom line represents the desired result, which is the union of the top two lines.

The first step in converting this data into a convenient form for analysis is to parse the individual framefiles into Dirfiles, using a simple parsing program. The parsing program loads a framefile from disk, and then writes each timestream to a separate file in a new Dirfile. Then the Dirfiles from both flight computers are merged so that there is no redundant data, as we will discuss in the next section (Section 8.2), and the resulting merged data is written to disk in an organized structure which is described in Section 8.3. In the last section of this chapter, Section 8.4, we will describe how data products that are created during the analysis process fit into the organized data structure.

## 8.2 Merging Data from Redundant Flight Computers

### 8.2.1 Aligning Dirfiles

The two flight computers are named Flight Computer 1 (FC1) and Flight Computer 2 (FC2). The two flight computers are designed to obtain the exact same data and write all of it to disk, which would make the data completely redundant. However, this redundancy is designed into the system precisely because we anticipate the likelihood of issues in flight that prevent the flight computers from recording data the entire time. For example, if one flight computer overheats it may be shut down for a short period of time to cool off, or if there is a



memory leak the computer may need to be rebooted. As a consequence, each flight computer writes some unique data, so to re-create complete timestreams it is necessary to merge the timestreams from the two flight computers. Figure 8.1 shows a conceptual example of this, where the flight computers have written some overlapping ACS data and some unique ACS data for which the other flight computer was off, and the resulting product should be the union of the two.

Each flight computer begins writing framefiles to disk at different times, and assigns them timestamped filenames using its own system clock. In general this means that the parsed Dirfiles from the two flight computers will have filename timestamps that are consistent to within a few dozen seconds, given the magnitude of the system clock drifts, which is not sufficiently precise to know which frame in an FC2 Dirfile corresponds to a given frame in the corresponding FC1 Dirfile, a task that we refer to as “alignment”. Furthermore, in the 2012 Antarctic flight, a system clock failure led to most of FC2’s framefiles being given nearly the same filename timestamp, a timestamp that corresponds to the system’s factory default start time of January 1, 2002.

The solution to this problem is to use the timestreams themselves to determine how the Dirfiles from FC1 and FC2 should be aligned with each other. Every Dirfile has a timing channel, so we can look at the timing datapoints inside an FC2 Dirfile and try to find the same timing datapoints inside one of the FC1 Dirfiles. If we find an FC1 Dirfile that has the same timing datapoints, then we know that the two Dirfiles overlap as long as the timing timestreams never repeat the same value, and we know the offset from one Dirfile’s starting index to the other’s. If one of the Dirfiles is not completely contained within the other, then the union of the two Dirfiles results in a Dirfile that is longer than each of the individual Dirfiles. We can create a new Dirfile that is the union, and then continue the process of searching for overlapping data. By repeating this process until no new overlaps are found, we can chain together multiple Dirfiles into the largest continuous sections of data possible. We call these sections “subsegments”.

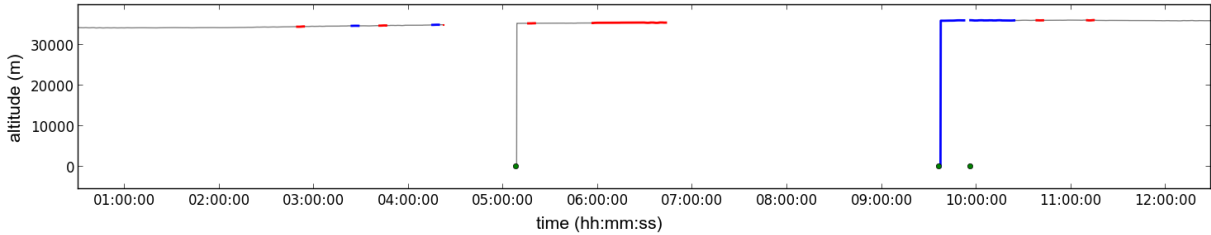


Figure 8.2: Example data demonstrating the merging of FC1 and FC2 data. The figure shows 12 hours of (ACS) altitude data. Gray lines represent data that is stored redundantly by both flight computers, blue lines represent data points that only existed on FC1, and red lines represent data that only existed on FC2. Green circles represent the start of continuous sections of data, and are at a y-value of 0 because the system has not yet initialized after powering up. The union of FC1 and FC2 data results in longer continuous timestreams than from a single flight computer.

Figure 8.2 shows an example of how this procedure results in long subsegments of continuous data. The figure shows 12 hours of altitude data from the ACS dataset. Gray lines represent data that is stored redundantly by both flight computers, blue lines represent data points that only existed on FC1, and red lines represent data that only existed on FC2. Green circles represent the start of continuous sections of data. In the figure, both FC1 and FC2 have data until 02:50, at which point the red section shows that only FC2 has recorded data. FC1 began recording data again 5 minutes later at 02:55. At 03:25 FC2 began missing data, but FC1 continued to record data as shown by the blue section. By alternating between FC1 and FC2, we can reconstruct continuous timestreams all the way up until the large gap at 04:20.

The large gaps in data beginning at 04:20 and 06:45 represent intentional system power downs that were done to allow the flight computers to cool. We refer to the collection of subsegments between intentional power cycling cool downs as “segments”. In total there are 35 segments in the EBEX 2012 Antarctic flight.

When the system is powered on at 09:35, only FC1 has recorded data for the first 50 minutes. Therefore, for the 4 minutes starting at 9:52 when FC1 rebooted, we have no FC2 data to fill in the gap. As this was not an intentional system power down, this gap delineates

subsegments instead of segments.

Note that the procedure for aligning Dirfiles from the two flight computers relies on the fact that the timing channels have non-repeating, or unique, data. If the timing channels only contain valid data, then this is the case because time increases monotonically. Unfortunately, the timing channels do not always contain valid data and are sometimes repetitive. An example of this data is shown in Figure 8.3, which shows the ACS timing channel, in Unix time (with 1970 epoch), for the entire flight. For most of the flight the time is valid, and therefore around 1.357e9, or 2013-01-01. However for periods near the second half of the flight the timing channel takes on the value of the FC2 system clock, and reverts to values around 1.0099e9, or 2002-01-01. In these cases the timing channel does not contain unique data, and cannot be used alone to align FC1 and FC2 data. Instead we require that both the timing channel and another channel have identical data. For the ACS dataset, for example, we used the altitude channel. By requiring that both channels have identical data points for a proposed alignment, we greatly decrease the probability of finding an incorrect alignment. All of the timing data in the different datasets suffer from non-uniqueness, and in each case a secondary channel is used to confirm alignment.

### 8.2.2 Resolving Conflicts

When FC1 and FC2 have overlapping data, the data points are for the most part identical because they are read from a digital bus. However there are some cases where the data points are not identical, either due to software bugs due to race conditions or hardware glitches due to cosmic rays. In these cases we have a conflict, and we attempt to choose from the flight computer that has the correct data. For a given conflict, to decide which flight computer has the valid data points and which has the invalid data points, we employ a two step resolution scheme:

1. **By Edge** - In many cases we find conflicts when one flight computer is beginning or

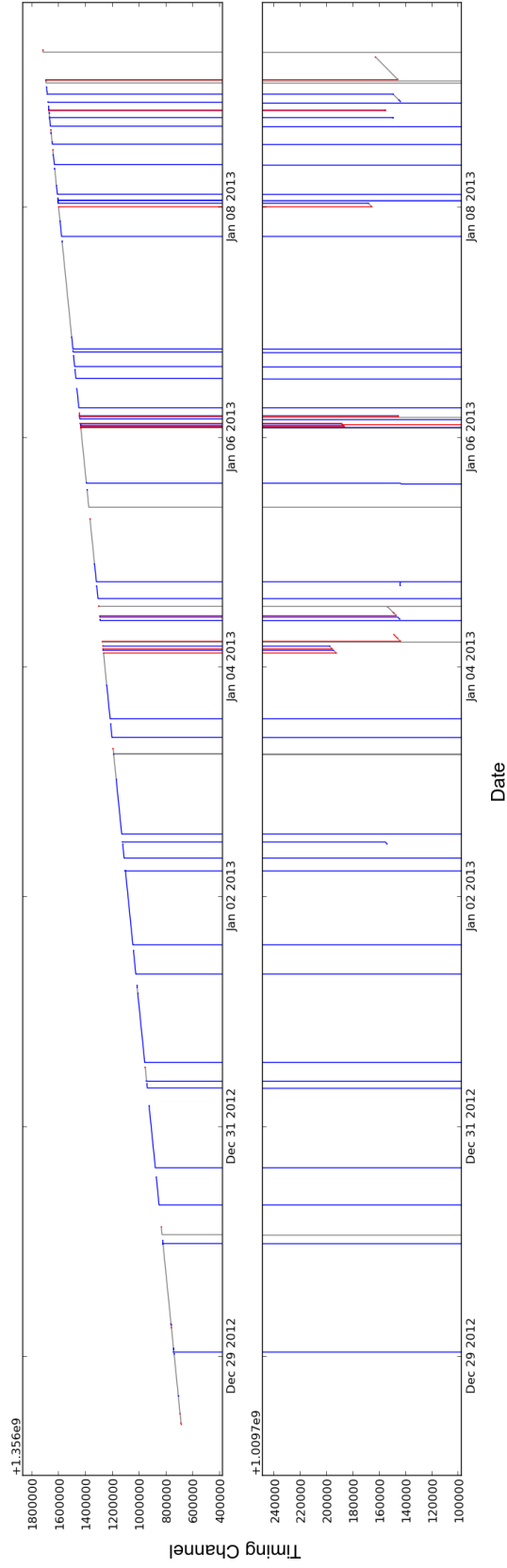


Figure 8.3: The ACS timing channel, in Unix time, for the entire flight. The two subplots show the same data at different y-ranges, calling out data that is valid (top) and data that is invalid (bottom). The colors follow the same scheme as in Figure 8.2.

ending a Dirfile, but the other flight computer is in the middle of a Dirfile. In these cases we assume that the flight computer that is just beginning or ending a Dirfile contains erroneous data, and prefer the data points from the flight computer that is in the middle of recording data. We assume this because a flight computer that is just beginning to write a Dirfile may still be in the process of initializing, and a flight computer that is just ending a Dirfile may be terminating FCP without correctly closing the framefile. An example of this is shown in Figure 8.4.

2. **By Expectation** - If both flight computers have data on both sides of the conflicting region, then it is not an edge case. For each section of conflicting data points, we build a temporary array of data points that represent our expectation of what values the data should take. These expectation data points are simply a linear interpolation between the data points on either side of the conflict. We then resolve the conflict by selecting the flight computer whose data points most closely match the expected data points. Specifically, we choose the flight computer for which the RMS between the actual data points and the expected data points is smallest. An example of this is shown in Figure 8.5. In the cases that we have inspected manually the rejected data points have clearly stood out as spikes given the variation in the data, indicating that this method chooses the correct data point rather than biasing the data by choosing between two reasonable data points.

No conflicts were found that could not be resolved by these two cases.

## 8.3 Base Data Structures

As discussed in the previous section, continuous sections of data are referred to as subsegments, and collections of subsegments between intentional power cycling cool downs are referred to as segments. Segments are named with the time of the start of the segment in UTC, in the format “YYYY-MM-DD--hh-mm-ss”. The date part of the filename follows the

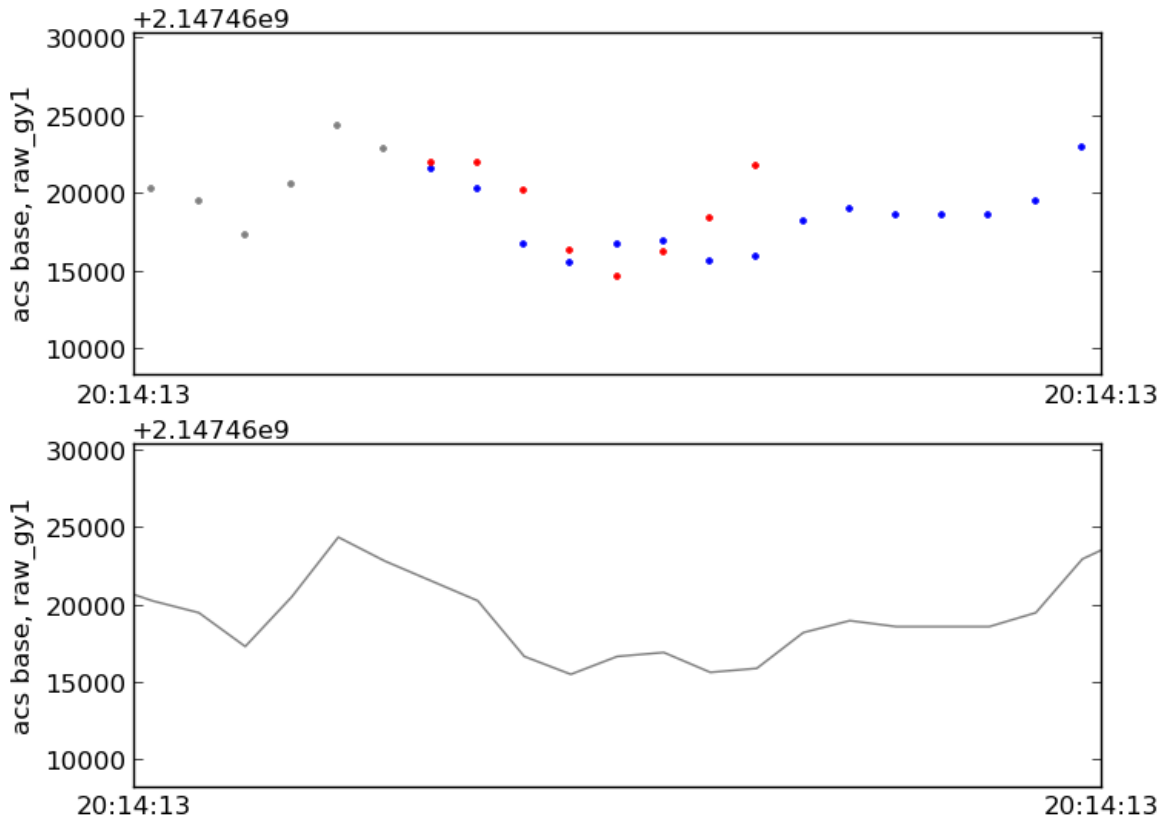


Figure 8.4: Example of conflict resolution in an ACS data stream called “raw\_gy1” using the “By Edge” method. In the top subplot gray data points are identical on both flight computers, blue data points are unique to FC1, and red data points are unique to FC2. Conflicts exist when the two flight computers have two different data points at the same index. The bottom subplot shows the resulting data stream after the conflict has been resolved in gray. The conflict occurs at the end of an FC2 dirfile, as evidenced by the transition in the top subplot from both red and blue data points to only blue data points. In this case FC1 is preferred, and the resulting data stream in the bottom subplot matches the blue data points.

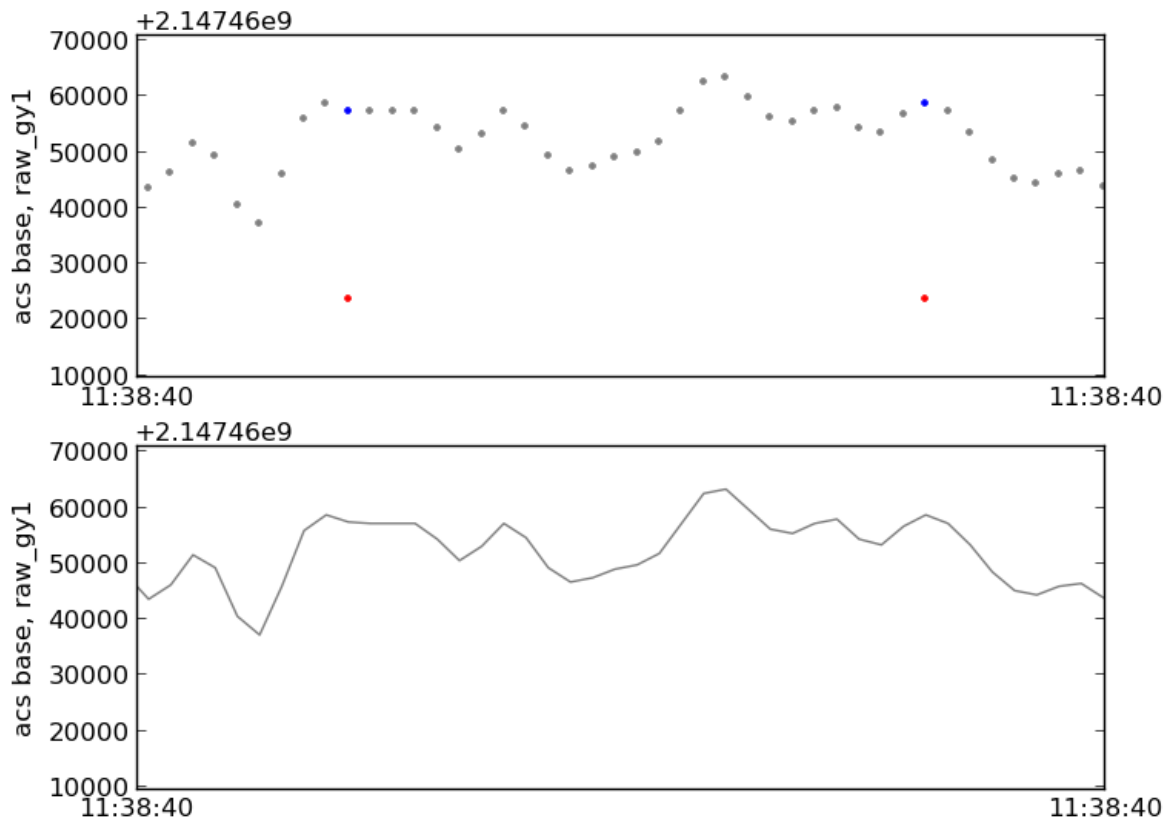


Figure 8.5: Example of conflict resolution in an ACS data stream called “raw\_gy1” using the “By Expectation” method. The plot layout is identical to that of Figure 8.4. In this case the conflicts are resolved by expectation, and it is evident that the resolved time stream in the bottom subplot does not contain spikes that would be due to the red outliers in the top subplot.

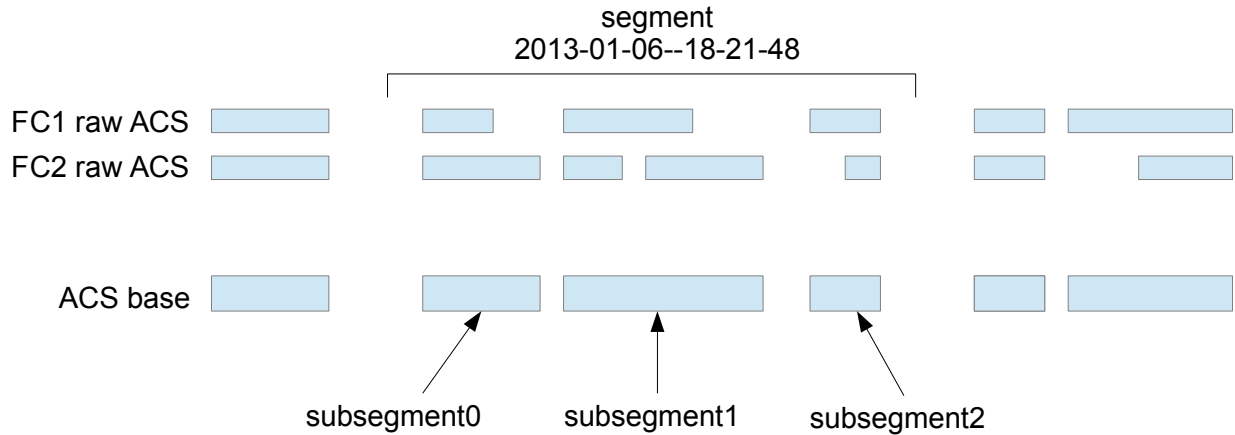


Figure 8.6: Diagram representing the union of ACS data from both flight computers, as in Figure 8.1, but with the one of the segments and its subsegments labeled.

ISO 8601 date standard, but the full filename does not follow the combined date and time standard because the standard contains the colon character which cannot be used in a file or directory name in Windows, and can cause problems in file or directory names in Unix-like operating systems (Linux and Mac OS). Subsegments are named “subsegmentN”, where N is an index that starts at 0 within each segment. Figure 8.6 revisits the diagram representing the union of FC1 and FC2 data, but labels it with example segment and subsegment names.

As described in Section 8.1, the 59 framefile streams can be categorized into four datasets (“ACS”, “Bolo”, “HWP”, and “SS”). We refer to these four datasets as flight “base” datasets. The word “base” draws a distinction between the datasets that contain unaltered flight data<sup>1</sup> and the “derived” datasets, which we have not yet discussed, that contain data that is derived from the flight data during analysis. The directory name of each dataset is the name of the category of data, along with a major revision number (X) and a minor revision number (Y), for example “acs\_v3-0”.

The ACS base dataset has the following directory structure:

---

<sup>1</sup>with the exception of merging the data from the two flight computers



```
acs_vX-Y/  
  2012-12-28--09-40-15/  
    subsegment0/  
      [Dirfile content]  
    subsegment1/  
      [Dirfile content]  
    subsegment2/  
      [Dirfile content]  
  2012-12-30--01-21-11/  
    subsegment0/  
      [Dirfile content]  
  ...
```

The Bolo base dataset has the following directory structure:

```
bolo_vX-Y/  
  2012-12-28--09-40-15/  
    board50/  
      subsegment0/  
        [Dirfile content]  
      subsegment1/  
        [Dirfile content]  
    board51/  
      subsegment0/  
        [Dirfile content]  
      subsegment1/  
        [Dirfile content]  
    ...  
  ...
```

The HWP base data set has the following directory structure:

```
hwp_vX-Y/  
  2012-12-28--09-40-15/  
    board78/  
      subsegment0/  
        [Dirfile content]  
    ...  
    board79/  
      subsegment0/  
        [Dirfile content]  
    ...  
  ...
```

The SS (slow streamer) base data set has the same structure as the Bolo base dataset.

Note that the any of the dirfile content in acs\_vX-Y will not be synchronous with any of the data in bolo\_vX-Y. In addition, the data in bolo\_vX-Y's board50 directory is not

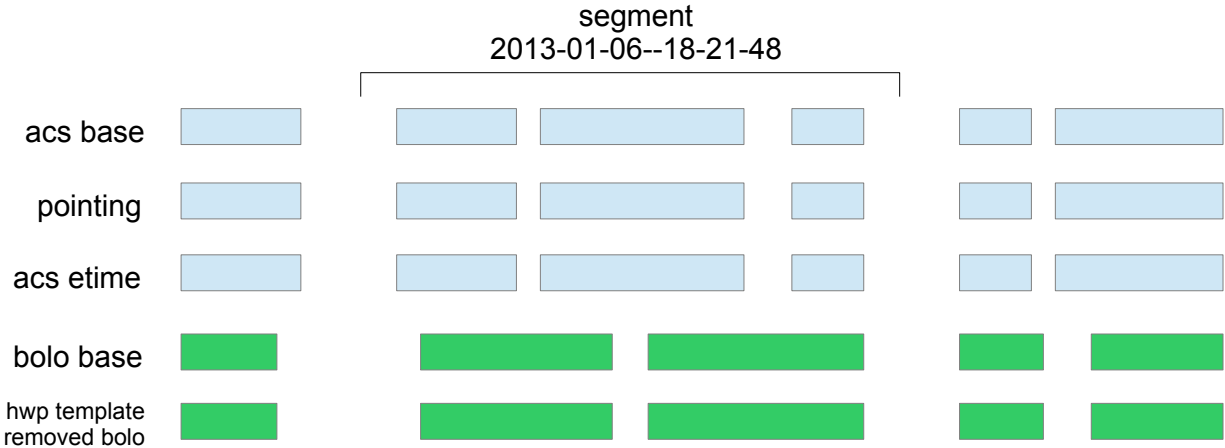


Figure 8.7: Examples of “derived” datasets (“pointing”, “acs etime”, and “hwp template removed bolo”) and the base dirfiles with which they are aligned (“acs base” and “bolo base”), in diagram form similar to that of Figure 8.6. Note that each derived dataset has subsegments that are aligned with, synchronous with, and of the same frame length as their respective base datasets.

synchronous with data in the board51 directory since the two boards are from different framefile streams (see Table 8.1).

## 8.4 Derived Data Structures

The four base datasets are not meant to be altered once they are created. During analysis users will need to write timestreams to disk after various pipeline stages. This is an optimization so that at a given pipeline stage a user can load input data directory from disk, instead of taking the time to generate it in memory by running all the previous pipeline stages up until that point. When writing timestreams to disk, the users will create new datasets called “derived” datasets. A derived dataset has an identical directory structure to one of the four base datasets, and its Dirfiles are the same length and aligned with those of the base dataset.

An example of a derived dataset is the pointing dataset, which is the result of the pointing reconstruction analysis. It contains pointing angles (e.g. right ascension, declination, etc),



Figure 8.8: Table of the four base datasets and some of their potential derived datasets.

and it is aligned with the ACS base dataset. Another example is that the timing channels in each of the four base datasets have various types of glitches that need to be fixed. Therefore, for each base dataset, a new dataset must be created that contains a clean timing channel, which we call “etime”. Another example is that new bolometer timestreams must be created, for which a half-wave plate signal must be removed. This derived dataset is called the “hwp template removed bolo” dataset, and is aligned with the Bolo base dataset. Another example is a derived dataset that exists only to hold flag fields describing the states of the timestreams in its base dataset. Some of these examples are shown in Figure 8.7.

The base datasets, and all of the derived datasets described in the paragraph above, are shown in table form in Figure 8.8. The four base sets are shown on the bottom, and their corresponding derived datasets (color coded appropriately) are shown to rest on top. As of the time of this document, all of the datasets described here, except for some of the “flags” datasets which were not necessary, have been created and are actively used for map making.

# Chapter 9

## The “LEAP” Software Framework

The post-flight analysis procedure is discussed in Chapter 10. In this chapter we describe a software framework called “LEAP”, which is used to support the software applications written for many of the analysis pipeline stages discussed in Chapter 10.

### 9.1 Terminology

We first review some programming terminology that is used in this chapter and the next:

- **object** - In object-oriented programming, an object is a data structure that contains both variables and functions.
- **classes and instantiation** - A class is the description of a type of object. When a new object is created, it contains the functions and variables described by the class. When an object is created from a class definition it is called instantiation. Multiple objects may be instantiated from a class.
- **method** - A function that belongs to an object is called a method.
- **module** - In Python, a file containing code is called a module.
- **dictionary** - In Python, a dictionary is a data structure that holds key and value pairs, where the value can be retrieved by specifying the key.

- **multiprocessing or multithreading pool** - A pool in this context is one paradigm for executing code in parallel. If there are many “jobs” to be performed (e.g. functions to be executed), the pool queues jobs and feeds them to the processors in the computer that can take on work (called “workers”). When a job finishes, the processor becomes underutilized and the pool passes it the next job.
- **repository** - A repository is a storage location for software. Sometimes when multiple programmers work on a single project they share code by storing it in a repository.
- **svn** - Subversion [57], or “svn”, is a program that is sometimes used to manage a repository and use it for version control.
- **framework** - A software framework is an environment that provides functionality to make it easier to write programs, such as a suite of libraries and support programs.

## 9.2 Overview

Each stage in the analysis pipeline can be executed through an individual program or a set of programs. Multiple collaborators from multiple institutions contribute their expertise in the form of one or more pipeline stages. Most pipeline stages, however, share a considerable amount of functionality, such as the capability to read and write the EBEX LDB (long duration balloon)-specific flight data structures discussed in Chapter 8, or to transform between celestial coordinate systems, or to deglitch timestreams. I therefore created a software framework in Python, along with another graduate student, Joy Didier, for collaborators to write programs in that share functionality<sup>1</sup>. We call it LEAP, or the *LDB EBEX Analysis Pipeline*. The framework primarily consists of an “apps” directory, which contains the programs that generally correspond to individual pipeline stages, and a libraries (“lib”) directory, which contains the functions that are shared between apps. LEAP is stored in a Subversion (svn) repository at Columbia. At the time of this document, LEAP has been

---

<sup>1</sup>Kevin MacDermid also contributed significantly to the early stages of LEAP.

used by over a dozen collaborating scientists from five different institutions.

The top level of LEAP contains these directories:

- **apps** - This directory contains the apps. The apps are discussed in more detail in Chapter 10, in which we discuss the post-flight analysis process. In that chapter we describe the first phase of the analysis pipeline, which involves processing raw flight data into temperature and polarization sky maps. Every pipeline stage in that description, which is shown graphically in Figure 10.1, is currently part of the LEAP framework as a series of apps.
- **lib** - This directory contains the libraries, which are Python modules that contain functions that are useful for multiple apps. The libraries are discussed in more detail Section 9.3.
- **resources** - This directory contains Python modules that contain data, rather than functions, that are useful for multiple apps. For example, it contains the hard-coded coordinates of some astronomical sources and a table of detector locations on the focal plane. The resources are discussed in Section 9.4
- **documentation** - This directory contains documentation files. It includes, for example, documentation on how to convert between different bolometer units, a block diagram of how all the EBEX detectors are wired, documentation on coordinate system conventions, and documentation on the orientation of sensors axes with respect to the gondola.
- **ldb\_data** - This directory contains links to all the flight data structures that will be loaded by the apps. Different users store the flight data structures in different locations on their computers, so the links must be created manually by each user, and nothing in this directory is committed to the repository.
- **output** - As will be discussed in Section 9.3.1, any time an app writes data to disk, that data is written somewhere inside this directory. Specifically, any app that writes data

to disk will first create a directory with a unique name inside the output directory, and write data there. The unique name will consist of the app name and the timestamp of the app's execution. Users generally run their apps thousands of times during development and testing, so they do not want to keep the output of every execution, only a select few for presentations or as final products. Therefore the output directory is not meant to store data long term, instead it is meant to be occasionally cleared out manually by the user. Nothing in this directory is committed to the repository.

- **long\_term\_output** - This directory is where users should manually copy the output directories that they would like to keep long term. The user is not meant to delete the contents of this directory. Nothing in this directory is committed to the repository.

## 9.3 Libraries

### 9.3.1 Parent App

LEAP apps are written as classes that inherit from a parent `leap_app` class. The parent class endows each LEAP app with a suite of functionality that is useful for development. The functionality that the parent class provides is:

#### Run Time

When a LEAP app is initialized it prints the name of the app to the screen and the start time of execution. When the app is finished it prints the run time of the app.

#### Settings

Every LEAP app is meant to have two settings files in its directory. A “`default_settings.py`” file contains the default settings. This shows the user what settings are meant to be available. It is also intended to be left in a clean state, so that a user with an untouched copy of the

app from the repository can run it and see the output one would expect from such an app. For example, the default settings for the map making app configure it to show a plot of the calibrator using a select few detectors. A “custom\_settings.py” file must also exist, and any settings in this file overwrite those in default\_settings.py. The custom\_settings.py file is not committed to the repository. The LEAP app automatically has access to the settings parameters through an object that is accessed as “self.settings”.

## Output

The parent class provides a method called create\_output(). When this method is called, a new directory is created inside leap/output/. The name of the directory is the name of the app (extracted from the name of the class and re-formatted) appended with a timestamp of the execution time.

This method also creates a directory, inside of this new app directory, named “admin”, and writes a number of files to it:

- It copies both settings files into the admin directory. It also writes a file that contains the contents of the settings object. These allow the user to easily recreate the settings files in case they need to re-run the app some time in the future and cannot remember the state of the settings.
- It also records the output of an “svn diff” command. This command outputs any local changes that have been made to any of the code with respect to the version on the repository. The output of “svn diff” is stored in a file whose filename contains the repository version number that it was compared against. This, combined with the settings file, gives the user the ability to recreate the state of the code. This can be useful to recreate a plot, recreate a data product, or investigating data that was created at any time in the past.
- It automatically records any log files to the admin directory. When the multiprocessing pool is used, a different log file is recorded for each pool worker.



After an app calls `create_output()`, it has access to a variable that it can access as “`self.out_path`”, which points to the newly created directory. Any data written to disk thereafter is meant to be written to the app’s directory using `out_path`.

## Logging

The parent app provides a logger that the LEAP app can access as “`self.logger`”. All content that is passed to the logger is written to file, and if the content is marked as having high enough priority it is also printed to the screen.

## Style Checking

The LEAP framework also encourages the use of the standard Python style guide, which is called “PEP8”. When any LEAP app runs, its code is automatically checked for PEP8 compliance, and if non-compliance is found a warning message is printed to the screen. This encourages users to adhere to a single style guide, which promotes consistency throughout LEAP and therefore increases productivity.

## Profiling

The parent app provides a method called “`profile`”, which runs the app wrapped in a profiler. The profiler is a program that measures how much time is spent inside the various methods of the LEAP app and prints the results to the screen. It is useful for finding the bottleneck in a program when it is time to optimize.

### 9.3.2 IO Management

The `io_management` library is tailored to the EBEX data structure. With this library the user can load any combination of ACS data, pointing data, bolometer (Bolo) data, half-wave plate (HWP) data, and slow streamer (SS) data, and it will be arranged in a meaningful

way and interpolated when necessary.

To load data the user instantiates the “Params” class, which contains a set of default parameters. The user then overwrites the parameters that they would like to customize. The most important parameters relate to the location of the data on disk, selecting sections of time that should be loaded, and selecting timestreams that should be loaded. The full list of parameters is shown in Appendix D.

As an example, consider loading the ACS altitude timestream, called “alt”, for the entire flight. The user will customize a Params object, and pass it to the dirfile loading function:

```
segments = dirfile_loading.load(params)
```

The load function returns a list of segment objects, that the user should call segments. Each segment object contains a number of dataset objects, which are named for the type of data they hold, such as “acs”, “pointing”, or “bolo”. These datasets contain the timestreams. For example:

- **segment.acs.times** is a timestream (an array) of timestamps that the dataset object obtained from the “etime” dirfiles.
- **segment.acs.channels** is a dictionary of channels, or timestreams, where the key is the name of the channel and the value is the timestream as an array.

Therefore if the user wants to plot altitude vs time for the entire flight, they simply write:

```
segments = dirfile_loading.load(params)
for segment in segments:
    pylab.plot(segment.times, segment.channels["alt"])
```

The io\_management library also contains the information required to automatically convert many of the channels in the ACS timestreams to standard units, e.g. radians, meters, and seconds. The library also converts the pointing timestreams from equatorial coordinates into galactic coordinates, if desired.

Similarly, the user can load a particular detector, or bolometer, channel. The bolometer timestream is accessed as “bolo.signals”. Note that the bolometer timestreams are not synchronous with the ACS or the pointing timestreams, so the following code to plot bolometer 0’s timestream against declination would likely produce an error:

```
segments = dirfile_loading.load(params)
for segment in segments:
    pylab.plot(segment.pointing.channels["dec"],
               segment.bolos[0].signals)
```

Instead, the library provides the user with a second copy of any ACS or pointing data that is interpolated onto the bolometer time base, so the user can write:

```
segments = dirfile_loading.load(params)
for segment in segments:
    pylab.plot(segment.bolos[0].pointing.channels["dec"],
               segment.bolos[0].signals)
```

noting that `segment.pointing` is different from `segment.bolo.pointing`. Every non-bolo dataset can be accessed either directly from the `segment` object, in which case its timestreams are on their native time base, or from the `bolo` object, in which case their timestreams are interpolated to the bolometer time base. With this in mind, the general structure of a `segment` object is:

```
* segment
  * name
  * acs
  * pointing
  * hwp
  * bolo[i]
    * name
    * times
    * signals
    * acs      <--- interpolated to bolo.times
    * pointing <--- interpolated to bolo.times
    * hwp      <--- interpolated to bolo.times
    * ss       <--- interpolated to bolo.times
```

An important optimization is that the list of segments returned by the `load` function is actually a special kind of list called a “generator”. Without explaining the distinction, this

is important because it prevents memory from being accumulated when multiple segments are loaded. Instead, only the segments that are actively being used are kept in memory. This is also true for the list of bolo objects that belong to the segment object.

### 9.3.3 Other Libraries

LEAP also contains other libraries that simplify the process of developing apps. All of the libraries are grouped into directories, which contain multiple files with various functions. Some of the directories are:

- **Mapping** - The functions that bin timestreams into sky maps are located in this directory. Different apps may want to create their own maps, such as the pointing calibration app or the signal calibration app.
- **Numerical** - This directory contains various mathematical functionalities, including gaussian fitting, fourier analysis, resampling, and wrapping.
- **Physics** - This directory contains functions to model blackbody spectra and scale galactic dust models.
- **Plotting** - This directory supplements the standard plotting libraries in python with functions that plot large arrays efficiently and format plots in ways that are common for working with EBEX data.
- **Time Domain Processing** - This directory contains various filters for modifying time domain data (as opposed to map domain data). This includes moving mean and median filters, butterworth low- and high- pass filters, and deglitching functions. These are commonly applied to detector timestreams or gyroscope timestreams.
- **Timing** - This library includes a progress indicator that helps the user estimate run-times, and a timer class.
- **Tools** - This directory contains assorted tools. Examples include:
  - A module for listing bolometers given a focal plane, wafer, or board

- A module with functions to help merge flag channels together (e.g. find overlapping flags or continuous unflagged sections)
  - A function to group objects together by a given attribute or unravel groups of objects into a list
  - A multiprocessing pool class that has more features than the standard multiprocessing pool class
  - A module that aids in loading tables from text files and arranging them into lists of objects
- **Units** - These modules contain functions that convert between different units of time, different units of angle, and different units of detector signal (e.g. between raw bolometer ADC units, noise equivalent power, noise equivalent temperature, kelvin CMB, etc).

## 9.4 Resources

The resources directory in LEAP contains any kind of information that is not part of the flight datasets or documentation. Some examples include:

- tables defining the frequency bands of the EBEX detectors
- tables listing when the detectors were active in flight
- coordinates of astronomical objects of interest
- sample rates of various EBEX subsystems
- tables defining angular offsets between the star cameras, the microwave telescope bore-sight, and individual detectors
- tables of star camera solutions

# Chapter 10

## Data Analysis

### 10.1 Overview

In this chapter we describe the post-flight analysis procedure. Broadly speaking, the analysis procedure consists of two phases: processing raw flight data into maps and then extracting scientific parameters from the maps. Here we describe the first phase of the pipeline and discuss in detail some of the pipeline stages with which I was heavily involved. We then show preliminary results and discuss the next steps that will lead to scientific results.

Figure 10.1 is a data flow diagram that gives an overview of the EBEX analysis pipeline from the raw data through to map making. Green ovals represent individual pipeline stages, which generally correspond to apps, and blue boxes represent the data that is input to or output from the app. The “flight base construction” stage converts the raw flight data into a useful format. The “star camera solving” and “pointing reconstruction” stages are used to produce pointing timestreams. The “etime reconstruction” stage cleans up the timestamps that allow us to align asynchronous timestreams to each other. The “hwp”-related stages and the “timestream cleaning” stages produce timestreams that are necessary for polarization analysis and prepare the detector timestreams for map making. The “map making” stage produces temperature and polarization maps from the timestreams. The

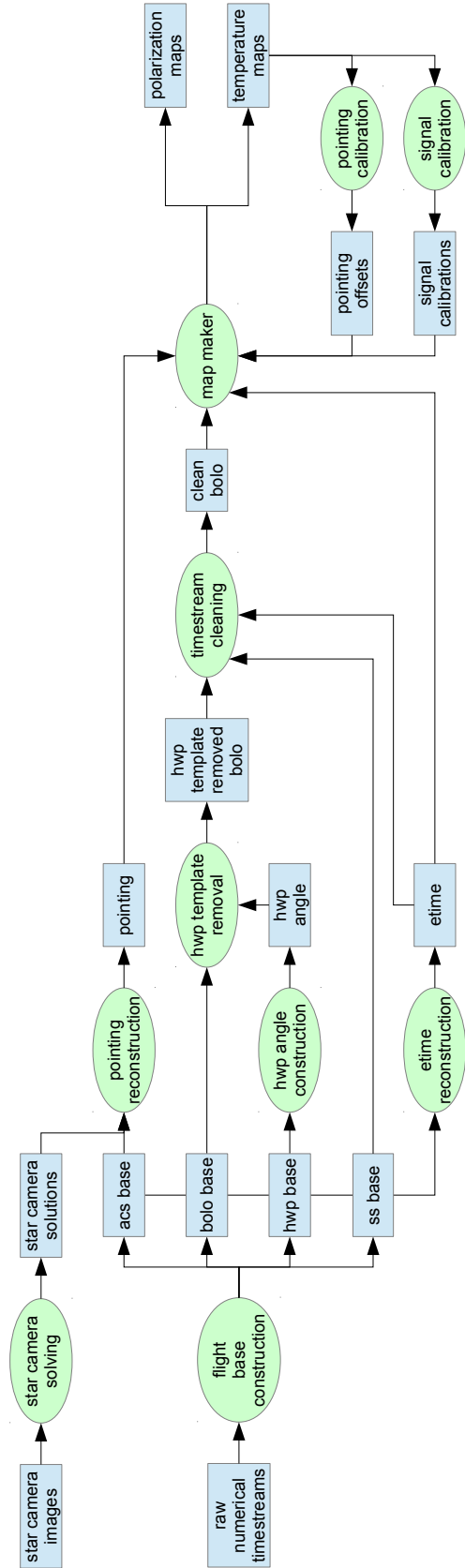


Figure 10.1: Data flow diagram of the analysis pipeline from raw flight data to maps. The rest of the pipeline (after maps) is not shown. Green ovals represent individual programs (apps), and blue boxes represent data (input and output for the apps). Every app shown here is part of the LEAP framework. The pointing and signal calibration results are iteratively improved using the temperature maps and therefore exist in a feedback loop with the map maker.

“pointing calibration” stage uses the maps to determine the angular offsets between the star cameras and the detectors, and the “signal calibration” stage uses the temperature maps to calibrate detector responsivities.

## 10.2 Selected Pipeline Stages

Every pipeline stage discussed in this chapter has been implemented, either by myself or another collaborator. In this section we review some of the pipeline stages with which I was heavily involved.

### 10.2.1 Flight Base Creation

The first step in the data analysis pipeline is to convert the raw numerical timestreams into the data structures described in Chapter 8 using the procedure discussed there.

### 10.2.2 Star Camera Solving

The star camera solving app converts star camera images into pointing solutions that can be aligned with the ACS dirfiles. The app therefore has two separate tasks. The first is to determine with what index in the ACS dirfiles each image aligns. The next step is to find a pointing solution for each image, and record this solution along with the timestream index.

As discussed in Chapter 4, the flight computers control the star camera trigger lines that open and close the camera shutters in flight. They record the status of the star camera shutters (open or closed) into numerical timestreams with the other ACS data. These timestreams are called trigger lines. When the value is 1 the shutter is open, and when it is 0 the shutter is closed. An event where the trigger line rises to 1 and falls back to 0 is called a “trigger”. When a trigger event occurs, the digital camera in the star camera automatically captures an image and stores it into a buffer, and shortly thereafter STARS finds that the



buffer has a new image, downloads it, and writes it to disk.

FCP keeps a counter called “counter\_fcp” that it increments after each trigger event (excluding multiple exposures). STARS keeps a counter called “counter\_stars” that it increments after each new image is found (also excluding multiple exposures). Both programs share their counters with each other. FCP stores both counters in numerical timestreams that are synchronous with the trigger\_line, and STARS stores both counters in the headers of the image files. After a trigger event roughly 0.5 s is required for both programs to increment their counters and share the new values with each other. If this happens, then when the following trigger event occurs, the counters written to the dirfile will correctly correspond to the counters written in the image headers. If there is a problem with the network connection then counter\_stars will be incorrect in the dirfiles and counter\_fcp will be incorrect in the image header, and we will know that the images require investigation. Figure 10.2 shows the FCP and STARS counters for both star cameras. Since the entire EBEX system was shut down twice a day during flight for thermal reasons, the FCP counters and the STARS counters reset to 0 multiple times throughout the flight.

The star camera solving directory inside the LEAP apps directory actually contains 4 separate apps:

1. **Header Listing** - This app loads every image header to extract the counters for each image, and stores them in a table.
2. **Trigger Listing** - This app loads the numerical timestreams and finds the trigger events by looking for rising and falling edges in trigger\_line. It records the start and end index of each trigger event to a table, along with the values of the counters in the timestreams at the time of the events.
3. **Trigger Alignment** - This app takes the two tables produced by the first two apps and matches the trigger events to the images. Since the counters resets to 0 multiple times throughout the flight, this app also relies upon the timestamp of each image to

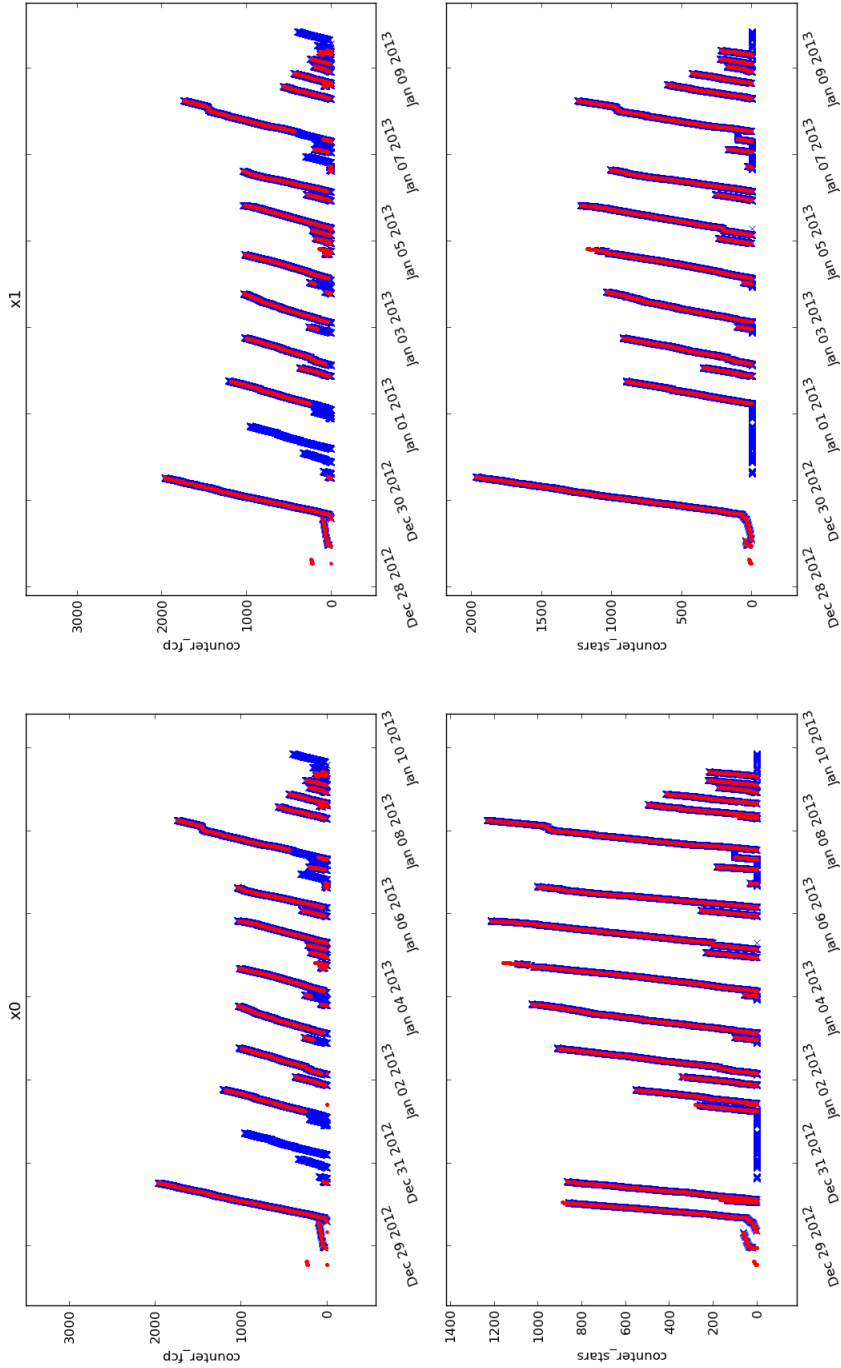


Figure 10.2: The fcp\_counters (top) and stars\_counters (bottom) for both star cameras (left and right) for the entire flight. For each star camera, both FCP and STARS keep a copy of both counters. The blue data points show FCP's copy of the counter from the numerical timestreams, and the red data points show STARS's copy of the counter from the image headers. When there are only blue points, the flight computers were running and sending triggers, but the star cameras were off. Most notably around December 31 the gondola operators were debugging the pivot motor controller issue instead of collecting scientific data. Each time the flight computers were powered on counter\_fcp reset to zero. Each time the star cameras were powered on counter\_stars reset to zero.

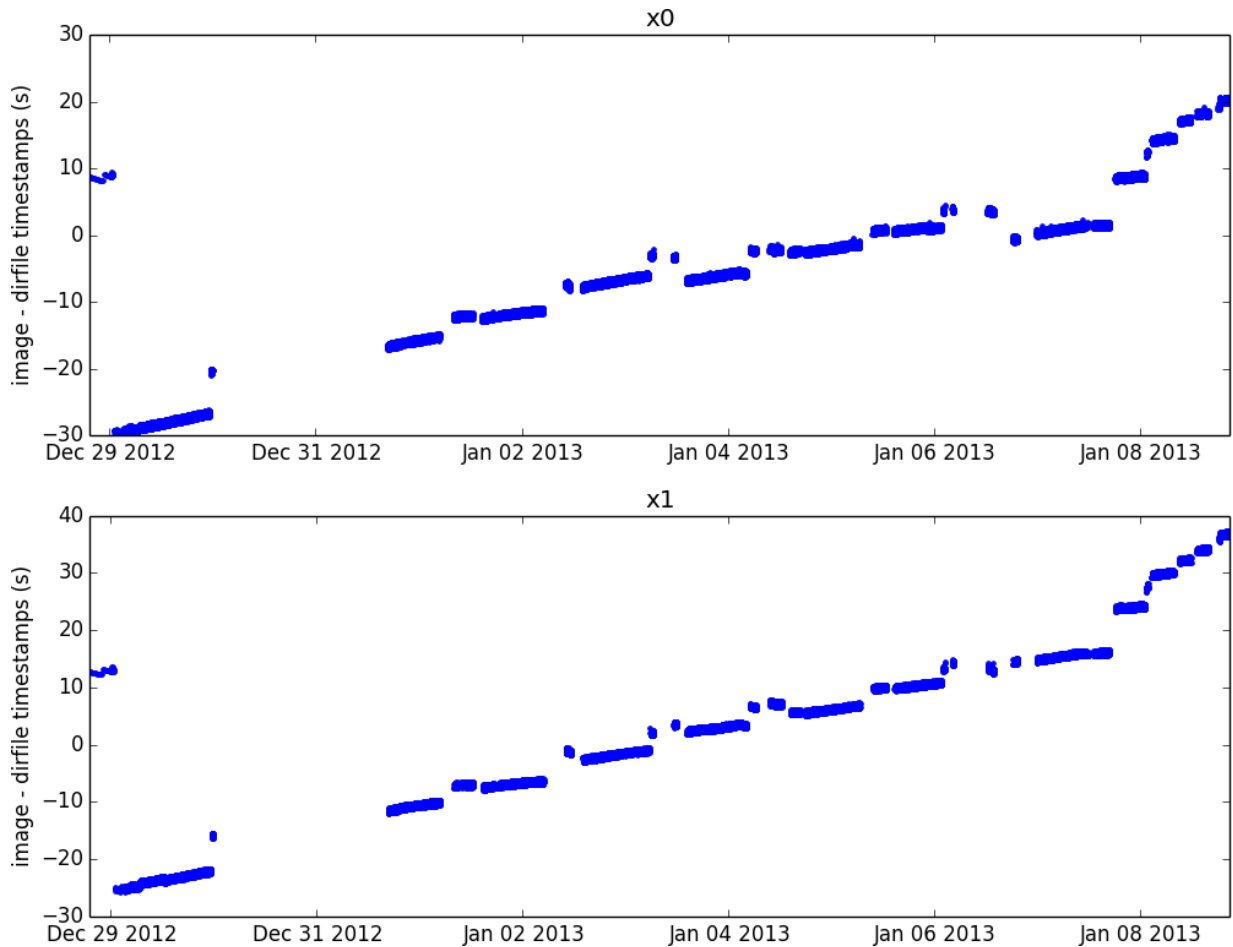


Figure 10.3: These plots show, for each star camera, a data point for each image that has been aligned to the dirfiles. The data points represent the difference between the timestamp associated with the dirfile at that index (which comes from the flight computer system clock) and the timestamp associated with the image (which comes from the star camera system clock), plotted against time. A non-zero slope represents a drift between the two machines' system clocks. A relative drift of this magnitude is not a problem for analysis because we primarily use the “tigger line” timestream and a series of counters to align the star camera images to the ACS timestreams. The fact that none of the differences exceed 40s indicates that the trigger alignment app did not make any alignments that mistakenly cross resets.

ensure that an alignment isn't made that crosses resets. The output of this app is a table that contains the filename and corresponding timestream index for each image. The app also outputs a metric, shown in Figure 10.3, to demonstrate that no false alignments were produced.

4. **Solving** - This app iterates through every image and passes it to STARS to be solved, along with the configuration parameters that STARS requires. The output of this app is a table that contains each image's filename, dirfile index, and pointing solution. The pointing solution consists of the right ascension, declination, and roll coordinates, along with an uncertainty for each coordinate.

The first time that the solving app runs, no information from the coarse sensors is used and each image must solve entirely lost-in-space. We choose to disregard the coarse sensor data because the geographic GPS data that is necessary to convert it into the equatorial frame is faulty, whereas the method described here is quite simple. The solving app therefore requires that STARS match 8 stars to find a solution. The output of the solving app is then fed into the pointing reconstruction app, which will be discussed in the next section. The pointing reconstruction app combines these star camera solutions with the gyro data to find a pointing solution at every index, not just when there is a star camera solution. The output of the pointing reconstruction app can be fed back into the solving app, and this time the solving app will use the the pointing solution stream as a guess for STARS. If the uncertainty on the pointing solution stream at the index of the image is less than  $0.5^\circ$ , then the solving app tells STARS to include a  $2^\circ$  search radius and match 5 stars. Otherwise, if the uncertainty is less than  $3^\circ$  then the solving app tells STARS to use a  $12^\circ$  search radius and match 6 stars. Otherwise, STARS runs lost-in-space. This iteration between the solving app and the pointing reconstruction app can be repeated until no new solutions are found.

## Preliminary Results

At this point we have only performed 3 iterations because we are satisfied with the number of solutions. After the first iteration, which was lost-in-space, we solved 92.6% of the images. This statistic excludes images that are saturated or are pointed within  $30^\circ$  of the Sun. After the next iteration we solved 92.8%, and after the third iteration 92.9%. If we were to tune the solving parameters more carefully, we would expect to be able to solve at least 99% of the images that are not saturated or close to the Sun based on visual inspections of the images. However, with 93% of the images solved the average error in the reconstructed pointing streams is sufficiently low for our purposes.

### 10.2.3 Pointing Reconstruction

#### Overview of the Filter

Most of the flight data consists of 40s azimuth throws, which are azimuth rotations at a fixed elevation. At the end of an azimuth throw the gondola velocity is zero and there is a star camera solution. The pointing reconstruction app estimates the pointing solution at every time step. The pointing solution that it estimates represents the pointing of one of the star cameras, as opposed to the microwave telescope boresight as one might expect. At the moment we use Star Camera 0, though in the future we may decide to incorporate solutions from both star cameras. To obtain the pointing solution for Star Camera 0 at every 100.16 Hz time step we use the Star Camera 0 solutions whenever they are available, and the gyroscopes (gyros), which are available at every time step. We combine the star camera solutions, which we call observations, with the gyro measurements in an unscented Kalman filter (UKF) [58].

A Kalman filter iterates through every time step and holds an estimate of the state in a vector  $\hat{\mathbf{x}}$  and an estimate of the errors in a covariance matrix  $\mathbf{P}$ . In our case the state consists of the three equatorial angles - right ascension ( $\phi$ ), declination ( $\theta$ ), and roll ( $\psi$ ) - along with

three gyro bias values  $b_1$ ,  $b_2$ , and  $b_3$ . The bias values are angular velocities that are added to the gyroscope measurements to compensate for a systematic error in the gyroscopes, which we discuss below. By including them as part of the state in the Kalman filter we allow the filter to estimate their values along with the pointing angles. The estimate of the state  $\hat{\mathbf{x}}$ , at time step  $k$ , when it includes all the observations up to and including time step  $k$ , is written:

$$\hat{\mathbf{x}}_{k|k} = \begin{pmatrix} \theta \\ \psi \\ \phi \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} \quad (10.1)$$

At each time step the Kalman filter estimates  $\hat{\mathbf{x}}$  and  $\mathbf{P}$  with two phases, a prediction phase and an update phase. The prediction phase uses the state from the previous step  $k - 1$  to predict the state at the current step  $k$ . In our case, this means integrating the gyro measurements to evolve the pointing angles. As we do this at points increasingly far from star camera measurements, the error on the pointing stream increases. In the update phase, the state is corrected with a new observation, decreasing the error on the pointing solution. In our case, the new observation is a star camera solution that is not available at every timestep, so the update phase is not executed at every time step. This is similar to FCP's real-time attitude determination filter, described in Chapter 3.5.1, which itself is a modified Kalman filter.

We predict the state at time step  $k$  using the previous state  $\hat{\mathbf{x}}_{k-1|k-1}$  and the equatorial rates  $\dot{\phi}$ ,  $\dot{\theta}$ , and  $\dot{\psi}$ :

$$\hat{\mathbf{x}}_{k|k-1} = \hat{\mathbf{x}}_{k-1|k-1} + \begin{pmatrix} \dot{\theta}\Delta t \\ \dot{\psi}\Delta t \\ \dot{\phi}\Delta t \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad (10.2)$$

where  $\Delta t$  is the time between time steps, or  $1/(100.16 \text{ Hz})$ . Note that since this does not include the update step, we have not yet incorporated a possible observation from time step

$k$ , so this estimates the state  $\hat{\mathbf{x}}_{k|k-1}$  instead of  $\hat{\mathbf{x}}_{k|k}$ . We calculate the equatorial rates from the measured gyroscope rates  $\omega_1$ ,  $\omega_2$ , and  $\omega_3$  using this equation:

$$\begin{pmatrix} \dot{\theta} \\ \dot{\psi} \\ \dot{\phi} \end{pmatrix} = \begin{pmatrix} \cos \psi & 0 & \sin \psi \\ \sin \psi \tan \theta & 1 & -\cos \psi \tan \theta \\ -\frac{\sin \psi}{\cos \theta} & 0 & \frac{\cos \psi}{\cos \theta} \end{pmatrix} \mathbf{RO} \begin{pmatrix} \omega_1 + b_1 \\ \omega_2 + b_2 \\ \omega_3 + b_3 \end{pmatrix} \quad (10.3)$$

where  $\phi$ ,  $\theta$ , and  $\psi$  are the current state estimates of the pointing solution,  $\mathbf{O}$  is a matrix that orthogonalizes the gyroscopes into an orthogonal frame, and  $\mathbf{R}$  is a matrix that rotates the orthogonal gyro frame into the star camera frame. The matrices  $\mathbf{O}$  and  $\mathbf{R}$  we discuss below. The left-most matrix rotates the star camera frame into the equatorial frame.

We use an unscented Kalman filter (UKF) instead of a standard Kalman filter because this system of equations is non-linear. The unscented Kalman filter picks a set of sampling points around the current attitude state to propagate through the non-linear equations. The sampling points are chosen with a technique called the unscented transform [59]. The state at the next time step is the mean of the propagated sampling points.

We adapted a generalized unscented Kalman filter from Python to C++. Writing the filter in C++ sped it up by an order of magnitude. Reconstructing the pointing for the entire 11 day flight, which contains almost 100 million time steps, takes 80 minutes on a single 2.1 GHz processor. The filter runs each segment of flight separately, so we multiprocessed the app to run multiple segments in parallel as an additional optimization.

Kalman filters only use the data leading up to time step  $k$  to estimate the state at  $k$ . Therefore if we run the filter forwards, at step  $k$  we have only used the information in the range  $[0, k]$ , and if we run the filter backwards then at step  $k$  we have only used the information in the range  $[k, N]$  where  $N$  is the total number of time steps. Thus if we want to use all information available at every time step, we can run the filter in both directions and take a weighted average - a procedure known as Kalman smoothing.

Figure 10.4 shows the results of this process for about 200 seconds of simulated data. Simulated pointing streams were generated that mimic the data we have from flight. Star

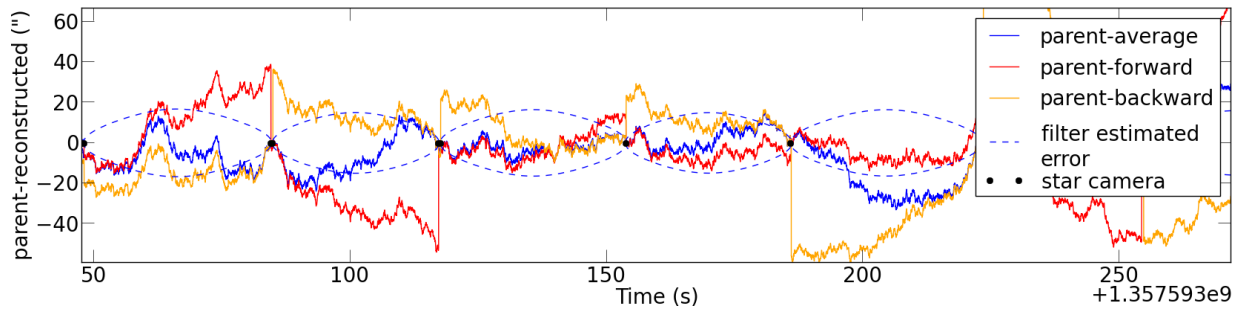


Figure 10.4: The declination results from running the unscented Kalman filter forwards and backwards on simulated data, as explained in detail in Section 10.2.3 and in [60].

camera and gyro measurements were created from the simulated pointing streams, and run through the UKF. The black dots represent star camera solutions, which occur every 40 s. The red curve shows the error in the declination coordinate estimated by the forward run of the UKF. The error is the true (“parent”) declination minus the output of the filter. Note that the red curve increases in magnitude as time progresses and then snaps back to near-zero at each star camera reading. The orange curve shows the same thing but for the backwards run. The blue curve shows the error for the average of both runs. This error is smaller than for the forward or backward runs individually. The figure also shows the error as estimated by the filter itself, namely the declination-declination term in the covariance matrix.

After the filter is run in either direction (or in both directions) on real data, we can calculate a metric for how well the filter performed. At the end of an azimuth throw we calculate the difference between filter’s estimated pointing solution and the new star camera solution before it is incorporated into the filter. This provides an estimate of the filter’s error after integrating the gyros for the length of time of the throw. If we choose, we can combine the metric from all the throws into a single metric. We discuss the use of this metric below.



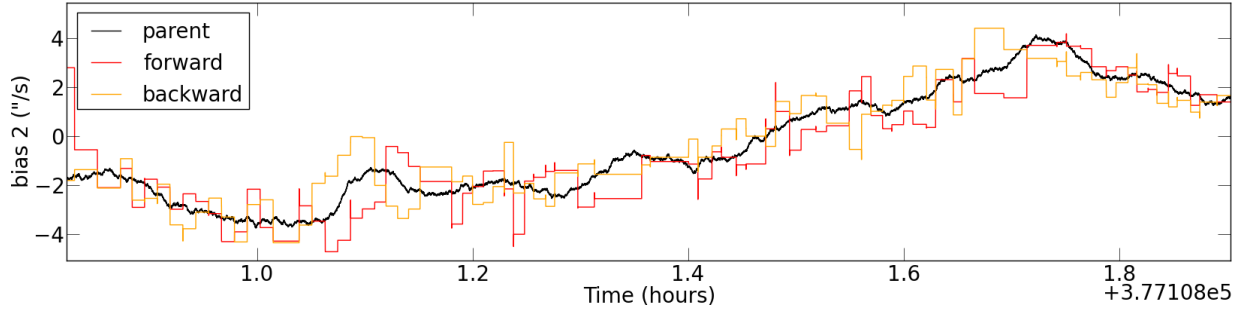


Figure 10.5: The result from estimating the Gyro 2 bias in a simulation [60]. The black curve shows the true simulated bias of Gyro 2. The red and orange curves show the estimated bias from the forward and backward runs of the unscented Kalman filter.

## Systematic Errors

There are two systematic errors that we will detail in this section. The first is the gyro bias, and the second is the gyro alignment angles.

**Gyro Bias** - For our application it is useful to consider the frequency profile of the gyro noise as two separate components: a white noise component and a  $1/f$  component. The  $1/f$  component can be thought of as a slow-drifting bias, which is constant for timescales shorter than 200 seconds. Considering these two noise sources, a gyro measurement consists of three components:

$$\text{gyro measurement} = \text{true angular velocity} + \text{white noise} + \text{bias}$$

The white noise is accounted for by the Kalman filter. The bias, however, is not. Since we know from measurements that it is a good approximation to take the bias as a constant value for the timescales of 40 s throws [60], we model the bias as part of the filter state as shown in Equation 11.5. Since we do not alter the bias in the prediction phase, the filter will estimate the bias as a constant value for each 40 s throw.

Figure 10.5 shows the bias estimates for about an hour of the simulation described previously for Figure 10.4. The black curve shows the true, or “parent”, bias of Gyroscope 2.

The red and orange curves show the estimated bias from the forward and backward runs of the filter. Simulations show that the filter estimates follow the true bias sufficiently well to meet the pointing requirement.

**Gyro Alignment Angles** - As shown in Equation 10.3, we must rotate the three gyro measurements into the star camera reference frame. The equation writes the orthogonalization and rotation matrix separately: the orthogonalization matrix adjusts the gyros so that they are in an orthogonal frame, while the rotation matrix rotates this frame to align it with the star camera frame. These two matrices can also be multiplied together into a single matrix. In either case, there are only 6 degrees of freedom. As separate matrices, there are only 3 angles required to create an orthogonal gyro frame<sup>1</sup>, and 3 angles required to rotate that frame into the star camera frame (as for any frame rotation). As a single matrix, it can simply be thought of as 2 angles for each of the 3 gyros, where the 2 angles rotate a gyro axis to align it with one of the axes of the star camera frame.

In the pointing reconstruction app, we run the unscented Kalman filter through a least-squares optimizer to find the 6 angles. Each time the Kalman filter is run, it automatically calculates the metric described above, and returns this metric to the optimizer. The optimizer runs a standard Levenberg-Marquardt algorithm to minimize the metric by altering the 6 angles. On our data the optimizer generally runs the full Kalman filter 90 times before converging on the 6 parameters. Simulations show that the optimizer finds the 6 angles to within  $\sim 0.001$  rad [60].

## Performance

The pointing reconstruction app has been run on the data from the EBEX 2012 Antarctic flight to produce timestreams that are currently being used for map making. To evaluate the output we wish to report an estimate of the RMS pointing error for a standard 40 s azimuth

---

<sup>1</sup>Only 3 angles are required to orthogonalize a gyro basis because one axis of the frame can be chosen so that it is already aligned with one of the gyros, and another axis can be chosen to be in the same plane as another gyro.

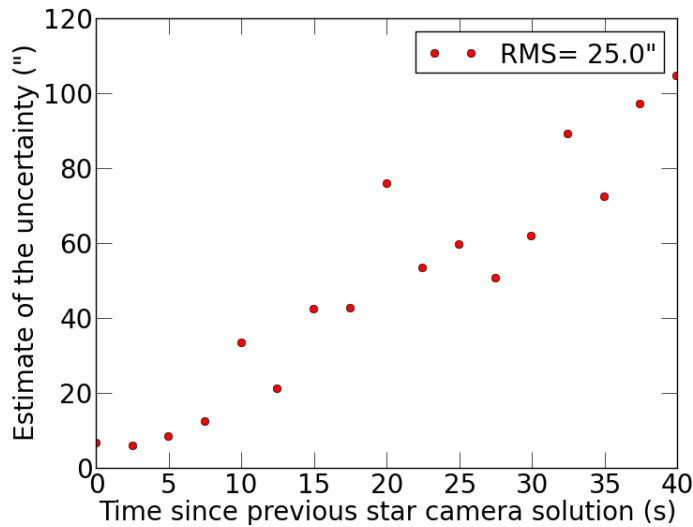


Figure 10.6: Binning the average filter error as a function of time since the last star camera solution.

throw, and to compare it to the RMS requirement of  $54''$ .

In order to estimate the error at each point in the throw given that we have a direct estimate only at endpoints, we make use of the fact that we have variable-length throws. We therefore calculate the metric described above for each throw, and then bin the metrics by time length of throw. This is shown in Figure 10.6, which plots the average error in a bin as a function of throw length. This yields an estimate of the filter error as a function of the time into a throw, when integrating the filter in a single direction. Given this information we can calculate what the RMS error would be on a 40s throw that averages forward and backward runs together. The result is  $25''$ , considerably less than the requirement of  $54''$ . In Section 3.3 we listed the pointing requirement differently as  $10''$  in the map domain. If we convert the RMS error on a throw into the map domain using the sky coverage from flight, we find a map-domain error of  $4.6''$ . See [60] for more information about the pointing reconstruction app.

## 10.2.4 Map Making

Here we discuss the details of the map maker. The map maker is extracted and expanded upon from the Quicklook software package described in Chapter 6. It is also integrated into LEAP. We first describe the procedure for generating temperature maps, then we discuss how it is implemented in LEAP, and finally we discuss how the map maker is altered to also produce polarization maps.

### Procedure

There are several steps involved in producing maps from timestreams:

1. The map maker loads template removed, deglitched detector timestreams, pointing timestreams, and half-wave plate timestreams if polarization maps are being generated. The pointing and half-wave plate timestreams are interpolated to the time base of the detector timestreams.
2. For each detector, the pointing timestreams are rotated so that they represent the pointing of the detector rather than the star camera. This is done in two steps: the first step rotates the pointing solution from the star camera frame into the microwave boresight frame, while the second step rotates the boresight frame into the frame of the specific detector.
3. The timestreams are then reduced to only include valid data by merging a set of flags that indicate when samples should be excluded from map making. The flags are generated from various methods such as:
  - Samples are flagged when the detector noise is found to be above a certain threshold. The noise level is measured by analyzing detector timestream power spectra.
  - Samples are flagged when the detectors are found to be non-responsive, as determined by analysis in the time domain.

- Samples are flagged when the pointing timestreams are invalid and, if producing polarization maps, when the half-wave plate angle timestreams are invalid.
  - Samples are flagged when the timestreams contain glitches, determined by time domain analysis.
  - Samples are flagged when the pointing timestreams indicate that the gondola velocity was too high.
  - Samples are flagged when the pointing timestreams indicate that the pointing uncertainty is above a certain threshold.
4. The timestreams are then high-passed to remove low frequency drifts. It should be noted here that the DC offset and other low frequency components of a detector timestream are not scientifically meaningful, and must be removed from the data. Indeed, the map generated by EBEX is not intended to measure the DC level of the CMB or signals at low angular scales. If the data were not high-passed, then the low frequency drift in the detector timestreams would cause a striping pattern on the map. The high-pass is performed by removing the result of a moving median filter from the data. The default window size for this filter is 15 s, though this is a preliminary value that may change as we refine the analysis procedure.
  5. The detector timestreams are then low-passed. This eliminates residual half-wave plate template at high frequencies, and for temperature maps eliminates sky synchronous polarization signal. The low-pass filter is a butterworth filter with a default (preliminary) cutoff frequency of 1 s.
  6. If producing a polarization map, the timestreams are then modulated according to the orientation of the selected polarization on the sky. This is explained below.
  7. The timestreams for each detector are then binned into a map according to the pointing associated with each detector. The final map is normalized by the total number of hits.
  8. A noise covariance matrix is estimated for the map. This is discussed in Section 10.3.

In the next section we explain the implementation of these steps in software.

## Implementation in LEAP

In the settings file of the map making app the user specifies which detectors and which sections of time they would like to incorporate into the map. They also specify a number of parameters associated with filtering the detector timestreams (e.g. filter cutoff frequencies), and map parameters (e.g. pixel size and which regions of the map to plot).

The map making app uses a multiprocessing pool as an optimization<sup>2</sup>. The map making app loops through every detector in every section of flight specified by the settings file, and adds each section of each detector to the pool queue as a job.

When a job runs, it uses the LEAP `io_management` library to load the detector data, the pointing data, and any flags that need to be used. These timestreams are loaded as part of a single object that we call a “dataset”. As discussed in Section 9.3.2, the `io_management` library takes care of a number of tasks that are useful for map making. It interpolates the pointing timestreams to the time base of the detector timestreams. It also rotates the pointing angles so that they represent the pointing of the detector rather than the star camera. It also converts the pointing from equatorial coordinates into galactic coordinates if desired.

The job then combines all the relevant flags (described above) to identify continuous sections of valid data that should be included in the map. Some of the flags are loaded from disk, and some are generated dynamically based on the data. For example, a timestream representing the angular velocity of the telescope is loaded, and a flag is generated to cut data that was taken when the gondola was moving too fast. The dataset is then divided into multiple smaller datasets, called “chunks”, that are usually between 1 minute long and 30 minutes long and only contain timestreams of continuous valid data. The job then iterates

---

<sup>2</sup>We use a multiprocessing pool instead of a multithreading pool because Python threads cannot take advantage of parallel computing.

through each of the chunks.

For each chunk, the job then filters the detector timestream. It first high-passes, then low-passes, then, if building a polarization map, modulates the detector timestreams. Once the detector timestream for a chunk is filtered and modulated if necessary, it is added to a map. Each job creates two maps: a hit map and an unnormalized signal map. The hit map records the number of times that samples are added to each pixel. The signal map holds the sum of the detector samples that belong to each pixel (based on the pointing). Ultimately we want the signal map to contain the average of all the detector samples that belong to each pixel, but at this stage we do not yet divide by the number of samples - we simply record them in the hit map. Once the job has added the detector timestreams from all the chunks, it returns the hit map and the unnormalized signal map.

The map making app creates a “total” hit map and “total” unnormalized signal map. Any time a job returns, the app simply co-adds the returned hit map to the total hit map, and co-adds the returned unnormalized signal map to the total unnormalized signal map. Once all the jobs are finished, the app can co-divide the total unnormalized signal map by the total hit map to obtain a normalized signal map containing the data from all the detectors. After that the map can be plotted or written to disk. In its current state the map maker does not estimate a noise covariance matrix of the map, though we will discuss this point in Section 10.3. It also does not weight the detector samples by their noise, it only rejects sections of timestreams for which detector noise is high.

The map making app employs a number of optimization strategies to help it run fast. The first optimization has already been described: individual sections of individual detectors are processed in parallel using a multiprocessing pool. With a 24 core machine that is being used at Columbia this can result in a  $24\times$  speedup. Another optimization is that the loop that bins detector samples into the signal map is performed in C++ instead of Python. This results in a  $45\times$  speedup due to the fact that the numerical computations compile to native machine code when using C++. The final optimization is that some of the processing done

to the pointing timestreams is preprocessed and included directly in the pointing dataset. Specifically, the pointing reconstruction app was modified to store the pointing timestreams in an additional mathematical form, called “quaternions”, rather than just in Euler angles. The `io_management` library used by the map maker now loads the quaternion values directly instead of calculating them, bypassing a time consuming step.

## Polarization

In order to study the polarization of the CMB, we produce two polarization maps in addition to a temperature map. The two polarization maps are called “ $Q$ ” and “ $U$ ”, where  $Q$  and  $U$  are Stokes parameters that describe polarization states of the light entering the telescope. Here we review the  $Q$  and  $U$  Stokes parameters and discuss the polarization capabilities of the map maker.

In the general case an electromagnetic wave has elliptical polarization. It can be defined as having an intensity  $E_0$ , an ellipticity angle  $\beta$ , and a semi-major axis that is oriented at an angle  $\chi$  from the x-axis. The Stokes parameters for a general electromagnetic wave are then defined as:

$$\begin{aligned}
 I &= E_0^2 \\
 Q &= E_0^2 \cos(2\beta) \cos(2\chi) \\
 U &= E_0^2 \cos(2\beta) \sin(2\chi) \\
 V &= E_0^2 \sin(2\beta)
 \end{aligned}
 \tag{10.4}$$

The  $I$  parameter corresponds to the total intensity. The  $I$  map is often referred to as the T map where T is the temperature of the CMB. The  $Q$  and  $U$  parameters describe the linear polarization intensity. The  $Q$  parameter represents linear polarizations aligned with the axes of the map, while the  $U$  parameter represents linear polarizations offset by  $45^\circ$  from the axes. The  $V$  parameter describes circular polarization, which we do not include in this analysis as it does not correspond to any of the physical mechanisms described in Chapter 1.



We explain above how the core of the map maker adds samples from detector timestreams into an  $I$  map. The core of the map maker has been modified to produce three separate maps -  $I$ ,  $Q$ , and  $U$  - and return all three to the main loop of the map making app. The main loop then co-adds and normalizes each of the three maps independently.

To produce the two polarization maps, the detector timestreams are modulated before being added to the map. Before being modulated, the signal in a detector  $s(t)$  depends on the total intensity ( $I_{\text{sky}}$ ) and the  $Q$  and  $U$  components on the sky ( $Q_{\text{sky}}$  and  $U_{\text{sky}}$ ):

$$s(t) = \frac{1}{2} [I_{\text{sky}} + Q_{\text{sky}} \cos(4\omega + 2\psi) + U_{\text{sky}} \sin(4\omega + 2\psi)] \quad (10.5)$$

where  $\omega$  is the half-wave plate angle with respect to the telescope frame, and  $\psi$  is the roll angle of the telescope with respect to the galactic frame. To produce a  $Q$  map we modulate the detector timestream with  $\cos(4\omega + 2\psi)$ , which produces:

$$\begin{aligned} s(t) \cos(4\omega + 2\psi) &= \frac{1}{2} [I_{\text{sky}} \cos(4\omega + 2\psi) + Q_{\text{sky}} \cos^2(4\omega + 2\psi) \\ &\quad + U_{\text{sky}} \sin(4\omega + 2\psi) \cos(4\omega + 2\psi)] \\ &= \frac{1}{2} I_{\text{sky}} \cos(4\omega + 2\psi) + \frac{1}{4} Q_{\text{sky}} + \frac{1}{4} Q_{\text{sky}} \cos(8\omega + 4\psi) \\ &\quad + \frac{1}{4} U_{\text{sky}} \sin(8\omega + 4\psi) \end{aligned}$$

in which  $\frac{1}{4}Q_{\text{sky}}$  is the only term without a sin or cos function, meaning that after averaging many samples with random half-wave plate angles, the pixel will only contain the  $Q_{\text{sky}}$  term. Similarly, we modulate the timestream with  $\sin(4\omega + 2\psi)$  to produce a  $U$  map.

### 10.3 Preliminary Results and Next Steps

Figure 10.7 shows a preliminary temperature map of the EBEX calibration source, RCW 38. Using this map we have refined the star camera pointing offsets to an accuracy of 5'. The

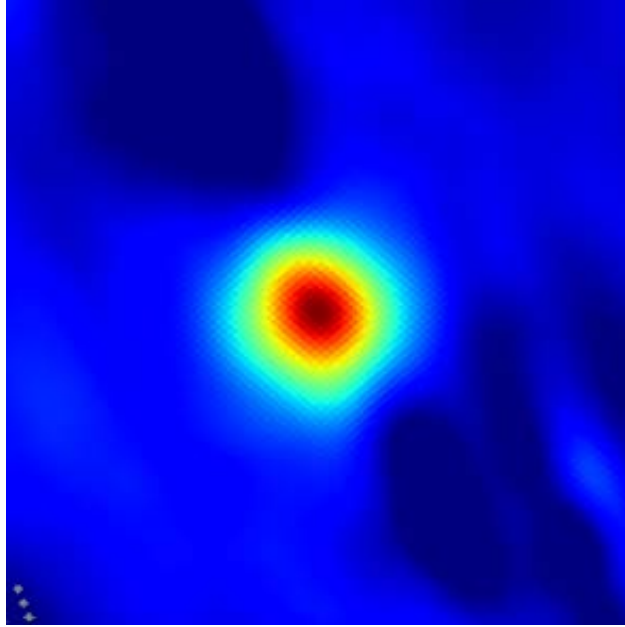


Figure 10.7: A preliminary temperature map of the EBEX calibration source, RCW 38, created from EBEX 2012 Antarctic flight data using the LEAP map maker. The map includes 2 hours of data from 91 150 GHz detectors.

current limitation on the accuracy of the pointing offsets is due to inaccuracies in the focal plane model. The consequence of this limitation is that calibrator maps made by different detectors show the calibrator source in different locations on the sky. We are currently integrating a more sophisticated model of the focal plane into the map maker, which will allow us to further refine the pointing offsets.

Figure 10.8 shows a preliminary map of part of the galactic plane in  $I$ ,  $Q$ , and  $U$ . The majority of the polarized signal from the galactic plane is expected to be found in  $Q$  because the sources of the polarized signal, dust grains, are aligned with the galactic magnetic field. We are in the process of calibrating a polarization angle offset from the telescope axis for each detector using pre-flight ground tests. In the meantime, however, to demonstrate our polarization sensitivity we produced these maps by calibrating the polarization offset angle for each detector using WMAP data. When the polarization angle offsets from pre-flight measurements become available we will integrate them into the map maker. In addition, the

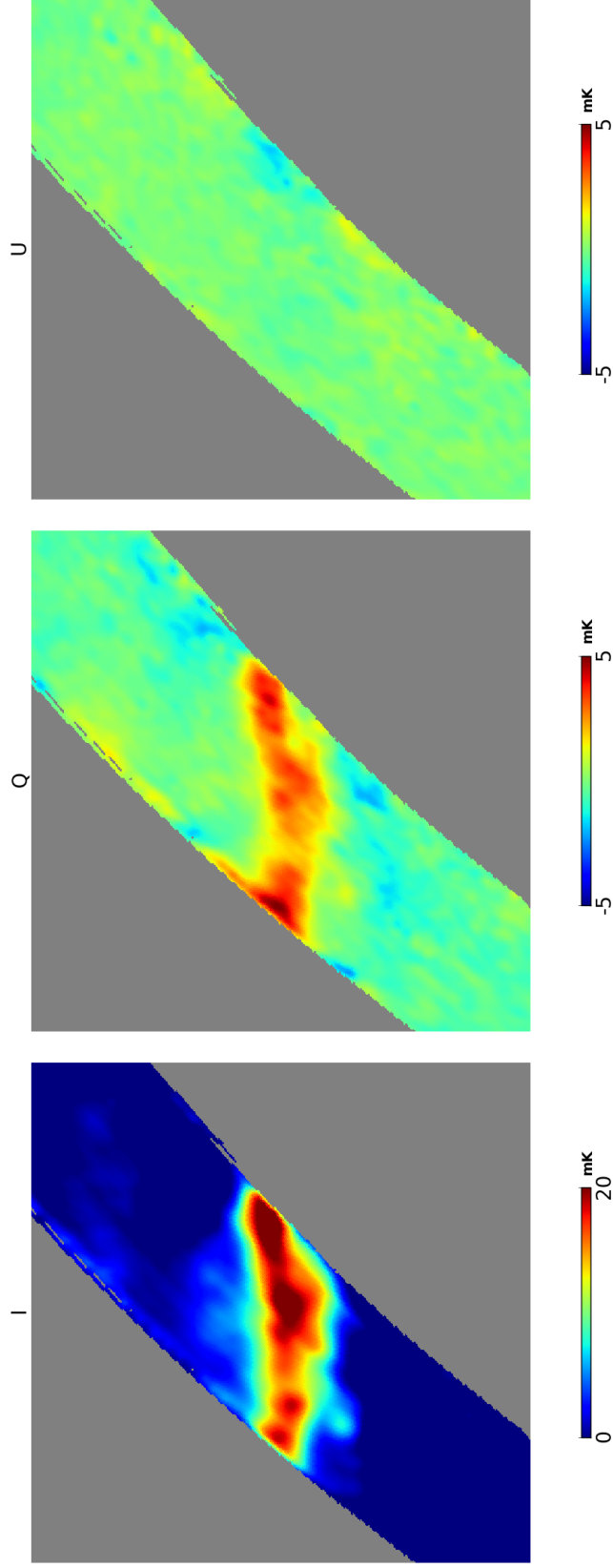


Figure 10.8: Preliminary maps in  $I$ ,  $Q$ , and  $U$  of part of the galactic plane, created from EBEX 2012 Antarctic flight data using the LEAP map maker. The map includes data from 91 250 GHz detectors from a 17 minute section of flight. The  $I$  map is a map of total intensity. The  $Q$  and  $U$  maps represent different alignments of polarization on the sky.

intensity map is currently being used for calibrating the absolute signal of the detectors by determining the conversion factor between raw detector counts and intensity using the Planck temperature data.

With improved calibrations of the pointing offsets, signal conversion factors, and polarization angles, we will proceed with generating polarization maps and extracting scientific parameters. We might create maps using a more sophisticated map maker that better accounts for the correlated detector noise (i.e. the unknown slow-drifting detector bias). We will estimate the noise covariance matrix either with monte carlo simulations or, depending on the map maker, simultaneously while calculating the map. The noise covariance matrix is necessary for evaluating the error on the extracted parameters.

We will combine the  $Q$  and  $U$  maps as  $Q + iU$  and  $Q - iU$ , and decompose the results into spherical harmonics using ladder operators [61]:

$$Q \pm iU = \sum_{l,m} a_{lm}^{\pm 2} [Y_{lm}(\theta, \phi)]$$

similar to what we did for the temperature spectrum in Equation 1.3.1. We can then quantify the E-mode and B-mode patterns that correspond to the physical phenomena described in Chapter 1 with:

$$E \equiv \sum_{l,m} \left( -\frac{1}{2} (a_{lm}^{(2)} + a_{lm}^{(-2)}) \right) Y_{lm}(\theta, \phi)$$

$$B \equiv \sum_{l,m} \left( -\frac{1}{2i} (a_{lm}^{(2)} - a_{lm}^{(-2)}) \right) Y_{lm}(\theta, \phi)$$

We will produce separate polarization maps at the three frequencies that EBEX is sensitive to, in regions of the sky selected for their limited foreground contribution, and produce E-mode and B-mode power spectra. We will use the power spectra to characterize the frequency dependence of the remaining foreground contaminants and remove them. Finally,

we will use the two-point correlation functions (Equation 1.3.1) of the E and B coefficients to measure or place an upper limit on  $r$ , the tensor-to-scalar ratio.

# Chapter 11

## Polar Mesospheric Clouds

In this chapter we describe the preliminary analysis of a non-cosmological data set that was obtained serendipitously in the EBEX 2012 Antarctic flight. As discussed in Chapters 4 and 5, the star cameras were designed to save every image captured and to continue finding solutions when faced with certain types of unanticipated challenges. In the 2012 Antarctic flight, polar mesospheric clouds (PMCs) emerged in roughly half of the star camera images. From the perspective of the EBEX pointing system, the clouds are a source of noise that must be and has been overcome. From the perspective of atmospheric science, however, the clouds in the images may ultimately provide an unprecedented look at small scale dynamics in the mesopause region of the atmosphere. Given that we stored every image to disk and were successful at identifying stars through the cloud layers, we now have roughly 20000 images of polar mesospheric clouds, from a vantage point not yet exploited by atmospheric scientists<sup>1</sup>, for which we have attitude solutions and geographic coordinates. Example images containing PMCs are shown in Figure 11.1.

Polar mesospheric clouds are clouds that form at an altitude of  $\sim 82$  km near the Earth's poles in the summer time. Given the brightness of the lower atmosphere, it is difficult to observe these clouds from the ground unless they are viewed at a shallow angle, resulting in

---

<sup>1</sup>to our knowledge, after research and expert consultation

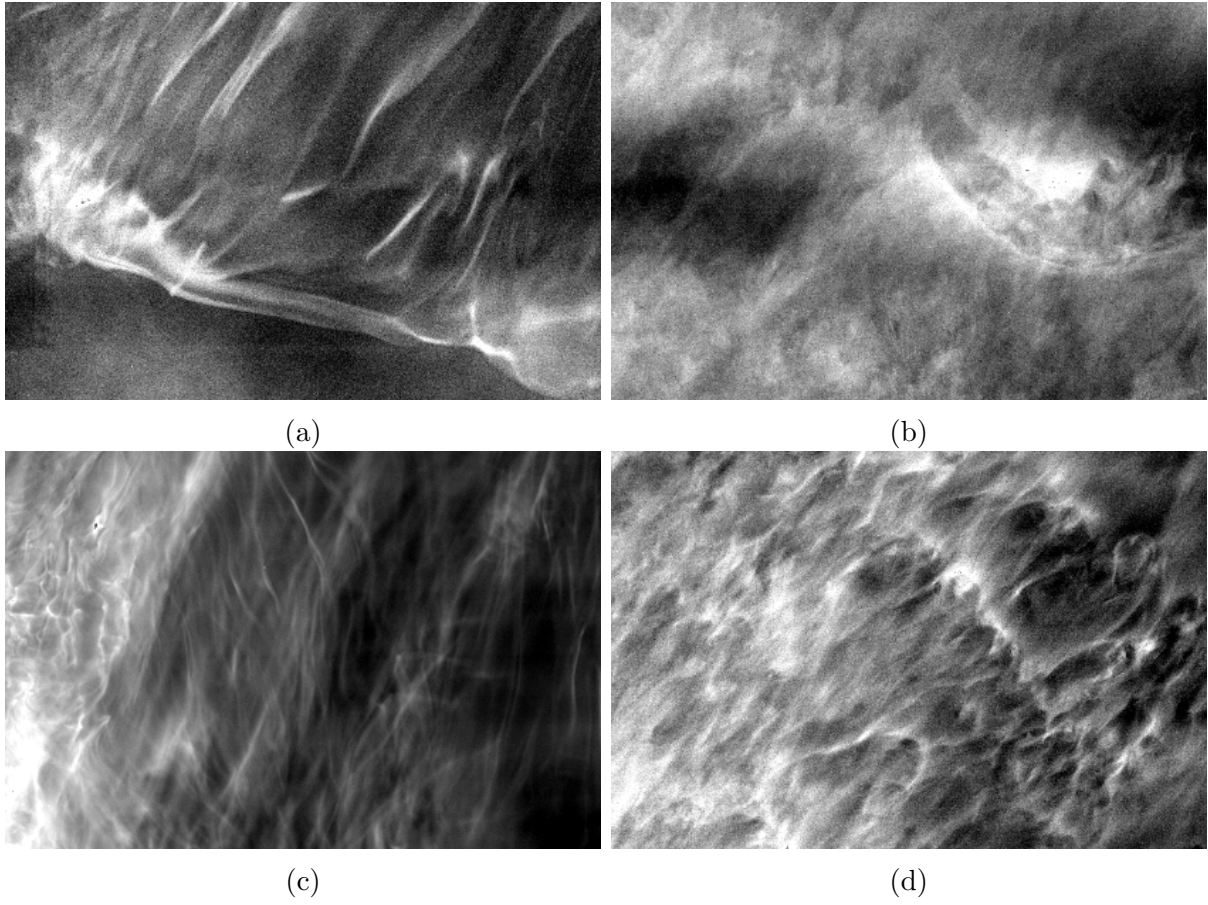


Figure 11.1: Example images containing polar mesospheric clouds.

distortion [62][63]<sup>2</sup>. The vantage point of an LDB (long duration balloon) payload provided the high resolution EBEX star cameras with a non-shallow view of the clouds from close proximity that was above the bright lower atmosphere.

Polar mesospheric clouds are of scientific interest because their observable features trace atmospheric gravity wave dynamics in the mesosphere and lower thermosphere [64][62][63]. Atmospheric gravity waves transfer momentum between the layers of the atmosphere, and characterizing the dynamics of these waves on the small scales at which we have observed them may help complete models of large-scale dynamics used by meteorologists and climatologists. The analysis of the PMC images therefore involves identifying morphological features in the clouds, measuring their characteristics, and tracking their evolution. Note that the gravity waves discussed in this chapter are different from the gravitational waves discussed in the EBEX science chapter. “Gravity waves” are waves at the interface of two media in which gravity provides the restoring force, whereas “gravitational waves” are waves in spacetime.

In this chapter we discuss the first two steps taken in the PMC analysis process: characterizing the existing data set and processing the images to enable feature tracking and characterization across multiple images.

## 11.1 Data Set Characterization

To characterize the  $\sim 40,000$  star camera images from flight based on their PMC content, we first developed an automated metric to find the sections of flight that had significant cloud activity, then reviewed the images in those sections manually.

To find sections of flight that have significant cloud activity we calculated a metric for each star camera image. The metric employed is the median absolute deviation about the median (MAD) of all the pixels in each image, after the image has been flat-fielded to eliminate dust spots and internal camera reflections. After reviewing the MAD of roughly 100 randomly

---

<sup>2</sup>PMCs are sometimes referred to as noctilucent clouds, given that they can be seen when the Sun is below the horizon but the clouds are high enough to still be illuminated.



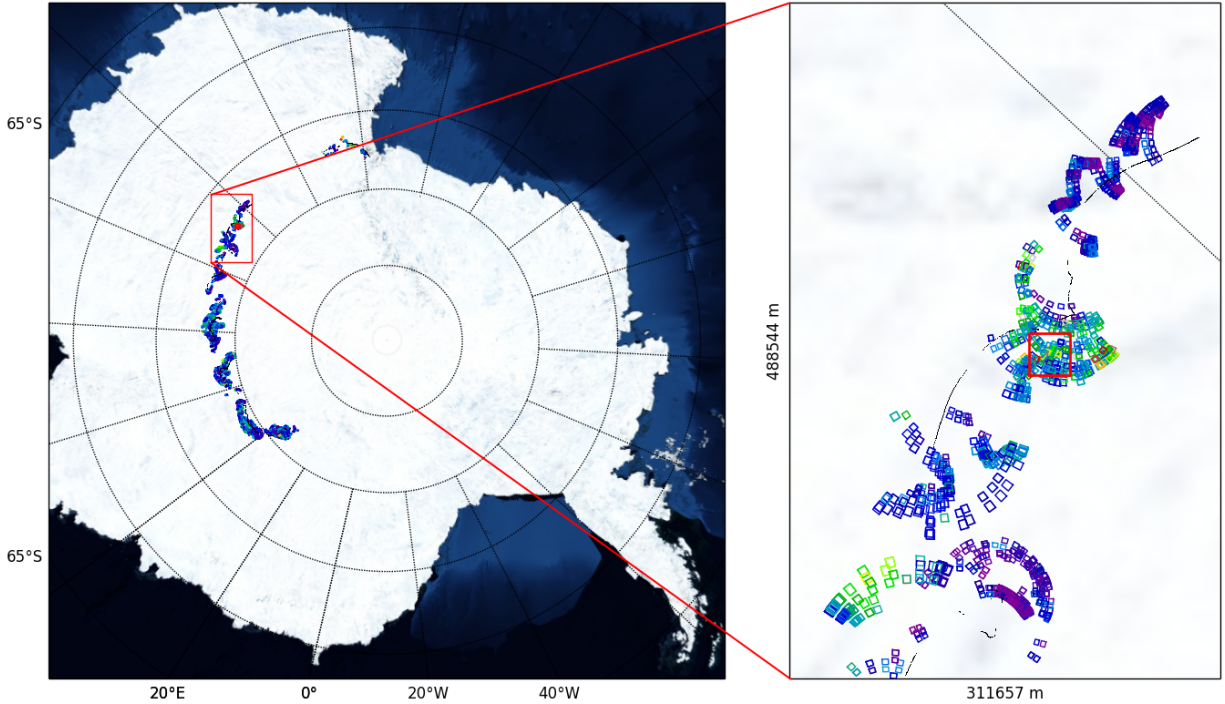


Figure 11.2: Coverage plot of the star camera images over Antarctica. Each rectangle represents the location of an image, and the color of the rectangle represents the MAD value. Purple and dark blue represent little to no cloud activity, light blue represents moderate cloud activity, and green, yellow, and red represent significant cloud activity. The left plot shows all the images, and the right plot shows a zoomed region that has significant cloud activity.

selected images, we determined that images whose MAD exceeded 3 or 4 counts had at least some cloud activity, and images whose MAD exceeded roughly 7 counts had high cloud activity. We find that roughly half of the images have at least some cloud activity.

We found the regions of flight that have significant cloud activity by plotting the geographic locations of the images along with their MAD values. Figure 11.2 shows the resulting coverage, where each rectangle represents the location of an image over Antarctica and the color of a rectangle represents the image's MAD value. It is evident that there are a several regions that contain at least some cloud activity, but a select few that contain high cloud activity.

We then manually reviewed the images from the sections of flight that contain high cloud activity, and identified features in the clouds that correspond to physical phenomena

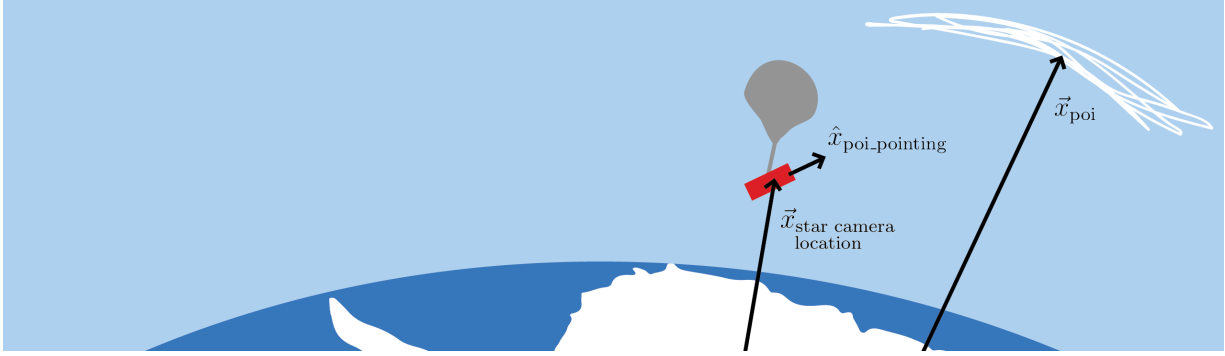


Figure 11.3: Visual explanation of the vectors involved in projecting the point of interest (POI) onto the cloud layer (shown in white). The EBEX star camera, shown in red, is suspended  $\sim 35$  km above the Earth. It observes polar mesospheric clouds that are at an altitude of  $\sim 82$  km. Figure courtesy of Michael D’Anvers.

associated with gravity waves. We find that, of the images with high cloud activity, roughly 5-10% show wave-like structures while the remaining images show turbulence. Figure 11.1 shows four example images. The top left image likely represents a gravity wave breaking. The top right image likely represents a vortex ring, which is a dynamic effect that accompanies gravity wave breaking. The bottom left image shows instabilities in the gravity waves, and the bottom right image shows the transition between wave instabilities and turbulent flow.

## 11.2 Feature Tracking and Characterization

In order to identify features that span multiple images, and to track cloud evolution over time, it is necessary to display the images in such a way that the cloud features appear in their geographic locations and orientations. We therefore produce plots in which we project selected star camera images onto the 82 km cloud layer. In this section we explain the projection and then discuss the preliminary results.

### 11.2.1 Projection Onto the Cloud Layer

For each pixel in an image, or “point of interest” (POI), we want to determine the geographic coordinates (longitude and latitude) at which the line of sight of the pixel intercepts the cloud layer at 82 km. To find where the POI intercepts the cloud layer it is useful to define a reference frame and three vectors. The reference frame is the equatorial celestial reference frame, whose origin is at the center of the Earth. The three vectors, shown visually in Figure 11.3, are:

- $\vec{x}_{\text{star camera location}}$  - This vector defines the position of the star camera with respect to the origin of the reference frame. This vector depends on the payload’s longitude, latitude, and altitude, as well as the time (due to the rotation of the Earth).
- $\hat{x}_{\text{poi\_pointing}}$  - This unit vector defines the pointing direction of the point of interest (POI) in an image. It depends on the star camera pointing solution (ra, dec, and roll), the pixel coordinates of the POI within the image (x and y), and the platescale of the image (p).
- $\vec{x}_{\text{poi}}$  - This vector defines the position of the POI at which it intersects the cloud layer, with respect to the origin of the reference frame.

The relationship between these vectors is:

$$\vec{x}_{\text{poi}} = \vec{x}_{\text{star camera location}} + \alpha \hat{x}_{\text{poi\_pointing}} \quad (11.1)$$

where  $\alpha$  can be solved given the following constraint:

$$|\vec{x}_{\text{poi}}| = \text{radius of Earth} + 82 \text{ km} \quad (11.2)$$

To use Equation 11.1 we must transform the information we have about each image from its natural form into cartesian equatorial coordinates. The information we have about each image is:

- the geographic coordinates (latitude, longitude, and altitude) of the payload
- the time at which the image was taken
- the pointing solution of the center of the image, which consists of three euler angles in the equatorial celestial frame (right ascension, declination, and roll)
- the platescale of the star cameras

The transformations are described here:

### Geographic Coordinates $\longleftrightarrow$ Equatorial Cartesian Coordinates

To transform  $\vec{x}_{\text{star camera location}}$  and  $\vec{x}_{\text{poi}}$  between geographic and equatorial coordinates we use the following relationships between spherical parameters:

$$\begin{aligned}
 \rho &= \text{radius of Earth} + \text{altitude} \\
 \phi &= \text{longitude} - \text{Greenwich sidereal time} \\
 \theta &= \frac{\pi}{2} - \text{latitude}
 \end{aligned}
 \tag{11.3}$$

We can then convert between spherical quantities and cartesian quantities using the standard formulalae.

### Image Coordinates $\longleftrightarrow$ Equatorial Cartesian Coordinates

Here we define the relationship between three quantities:

- $(x, y)$  - The pixel coordinates of a POI in an image
- $(\phi_1, \theta_1)$  - The equatorial pointing coordinates of the POI
- $(\phi_0, \theta_0, \psi_0)$  - The equatorial star camera solution that represents the attitude of the center of the image

We first recognize that the star camera optics project an image of the sky onto the flat surface of the CCD through a gnomonic projection, which allows us to define the CCD position  $(u, v)$  of the POI as:

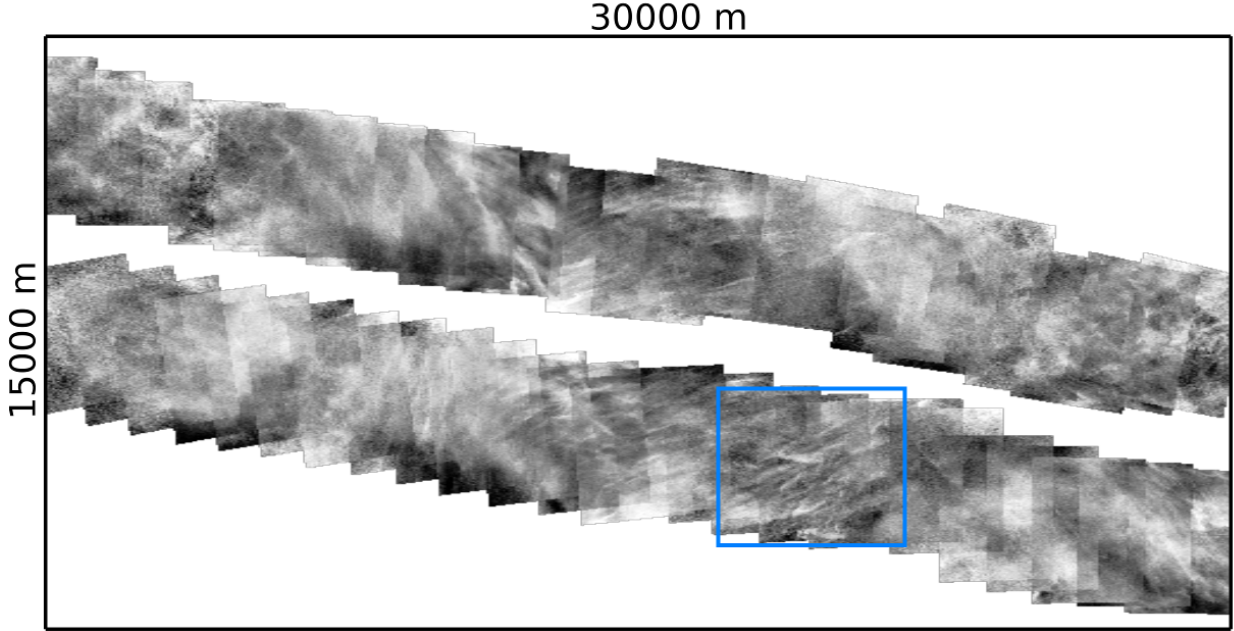


Figure 11.4: Example plot showing the projection of dozens of star camera images onto the cloud layer. Cloud features that span multiple images can be seen.

$$\begin{aligned}
 u &= \frac{\cos \theta_1 \sin (\phi_1 - \phi_0)}{\sin \theta_0 \sin \theta_1 + \cos \theta_0 \cos \theta_1 \cos (\phi_1 - \phi_0)} \\
 v &= \frac{\cos \theta_0 \sin \theta_1 + \sin \theta_0 \cos \theta_1 \cos (\phi_1 - \phi_0)}{\sin \theta_0 \sin \theta_1 + \cos \theta_0 \cos \theta_1 \cos (\phi_1 - \phi_0)}
 \end{aligned}
 \tag{11.4}$$

We then multiply by the camera's platescale,  $p$ , to convert from radians to pixels, and we account for the roll of the image by rotating the POI about the center of the image:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos \psi_0 & -\sin \psi_0 \\ \sin \psi_0 & \cos \psi_0 \end{pmatrix} \begin{pmatrix} -up \\ vp \end{pmatrix}
 \tag{11.5}$$

The minus sign in the term  $-up$  accounts for a parity inconsistency between the equatorial reference frame and the CCD frame.

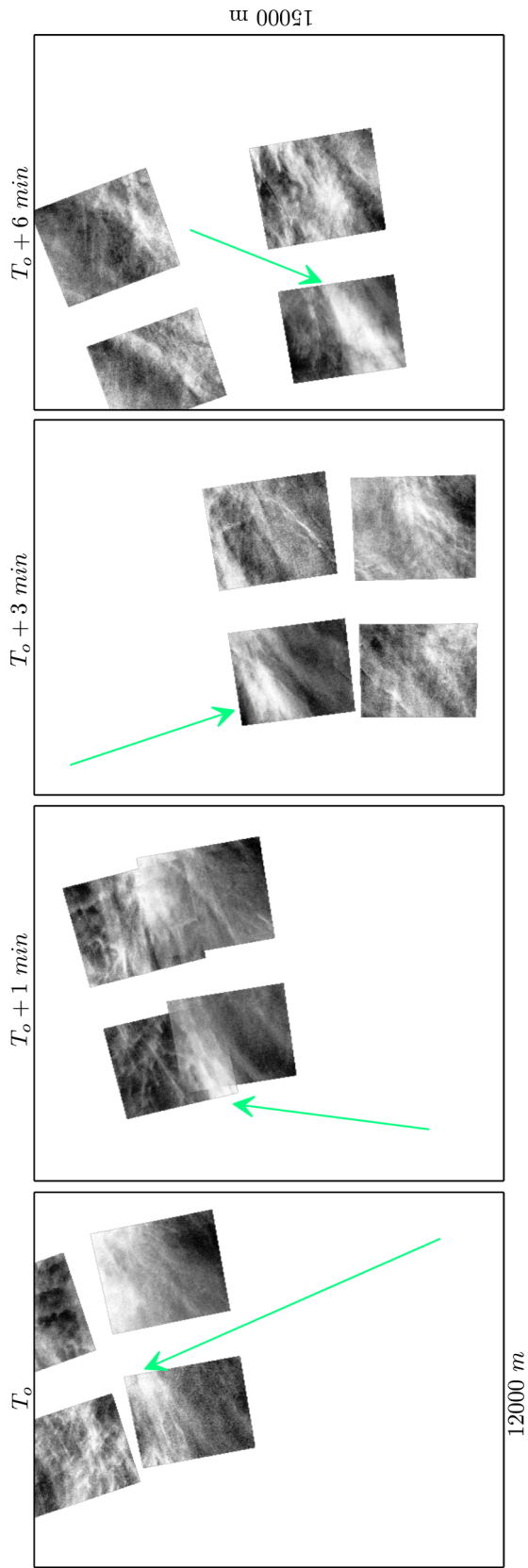


Figure 11.5: Example of feature tracking and evolution observation. Four plots are shown, each plot containing four images taken in the same one minute window. The four plots span 6 minutes, and a unique feature, identified by the green arrow, can be seen to move downward over time.

## 11.2.2 Preliminary Results

We have written a program to implement these transformations in order to project images onto the cloud layer so that the morphological features in the clouds are placed in their correct geographic location and orientation. Figure 11.4 shows an example plot in which dozens of images are projected onto the cloud layer. In this plot, features that span multiple images can be identified. Figure 11.5 shows an example in which 16 images are captured over the course of 6 minutes, and a unique feature within the images can be identified, tracked, and observed for morphological evolution.

# Bibliography

- [1] Planck Collaboration, P. A. R. Ade, N. Aghanim, C. Armitage-Caplan, M. Arnaud, M. Ashdown, F. Atrio-Barandela, J. Aumont, C. Baccigalupi, A. J. Banday, and et al. Planck 2013 results. XVI. Cosmological parameters. *ArXiv e-prints*, March 2013.
- [2] D. Baumann, M. G. Jackson, P. Adshead, A. Amblard, A. Ashoorioon, N. Bartolo, R. Bean, M. Beltrán, F. de Bernardis, S. Bird, X. Chen, D. J. H. Chung, L. Colombo, A. Cooray, P. Creminelli, S. Dodelson, J. Dunkley, C. Dvorkin, R. Easther, F. Finelli, R. Flauger, M. P. Hertzberg, K. Jones-Smith, S. Kachru, K. Kadota, J. Khoury, W. H. Kinney, E. Komatsu, L. M. Krauss, J. Lesgourgues, A. Liddle, M. Liguori, E. Lim, A. Linde, S. Matarrese, H. Mathur, L. McAllister, A. Melchiorri, A. Nicolis, L. Pagano, H. V. Peiris, M. Peloso, L. Pogosian, E. Pierpaoli, A. Riotto, U. Seljak, L. Senatore, S. Shandera, E. Silverstein, T. Smith, P. Vaudrevange, L. Verde, B. Wandelt, D. Wands, S. Watson, M. Wyman, A. Yadav, W. Valkenburg, and M. Zaldarriaga. Probing Inflation with CMB Polarization. In S. Dodelson, D. Baumann, A. Cooray, J. Dunkley, A. Fraisse, M. G. Jackson, A. Kogut, L. Krauss, M. Zaldarriaga, and K. Smith, editors, *American Institute of Physics Conference Series*, volume 1141 of *American Institute of Physics Conference Series*, pages 10–120, June 2009.
- [3] S. Dodelson. *Modern cosmology*. 2003.
- [4] A. R. Liddle and D. H. Lyth. *Cosmological Inflation and Large-Scale Structure*. June 2000.



- [5] S. Seager, D. D. Sasselov, and D. Scott. How Exactly Did the Universe Become Neutral? *Ap. J. Suppl.*, 128:407–430, June 2000.
- [6] A. A. Penzias and R. W. Wilson. A Measurement of Excess Antenna Temperature at 4080 Mc/s. *Ap. J.*, 142:419–421, July 1965.
- [7] G. F. Smoot, C. L. Bennet, A. Kogut, E. L. Wright, J. Aymon, N. W. Boggess, E. S. Cheng, G. De Amici, S. Gulkis, M. G. Hauser, G. Hinshaw, P. D. Jackson, M. Janssen, E. Kaita, T. Kelsall, P. Keegstra, C. Lineweaver, K. Lowenstein, P. Lubin, J. Mather, S. S. Meyer, S. H. Moseley, T. Murdock, L. Rokke, R. F. Silverberg, L. Tenorio, R. Weiss, and D. T. Wilkinson. Structure in the COBE Differential Microwave Radiometer First-Year Maps. *Ap. J.*, 396:L1–L5, 1992.
- [8] S. Hillbrand. *The E and B EXperiment: A balloon-borne cosmic microwave background anisotropy probe*. PhD thesis, Columbia University, 2014.
- [9] Planck Collaboration, P. A. R. Ade, N. Aghanim, C. Armitage-Caplan, M. Arnaud, M. Ashdown, F. Atrio-Barandela, J. Aumont, C. Baccigalupi, A. J. Banday, and et al. Planck 2013 results. XV. CMB power spectra and likelihood. *Astron. Astrophys.*, 571:A15, November 2014.
- [10] QUIET Collaboration, D. Araujo, C. Bischoff, A. Brizius, I. Buder, Y. Chinone, K. Cleary, R. N. Dumoulin, A. Kusaka, R. Monsalve, S. K. Næss, L. B. Newburgh, R. Reeves, I. K. Wehus, J. T. L. Zwart, L. Bronfman, R. Bustos, S. E. Church, C. Dickinson, H. K. Eriksen, T. Gaier, J. O. Gundersen, M. Hasegawa, M. Hazumi, K. M. Huffmanberger, K. Ishidoshiro, M. E. Jones, P. Kangaslahti, D. J. Kapner, D. Kubik, C. R. Lawrence, M. Limon, J. J. McMahon, A. D. Miller, M. Nagai, H. Nguyen, G. Nixon, T. J. Pearson, L. Piccirillo, S. J. E. Radford, A. C. S. Readhead, J. L. Richards, D. Samtleben, M. Seiffert, M. C. Shepherd, K. M. Smith, S. T. Staggs, O. Tajima, K. L. Thompson, K. Vanderlinde, and R. Williamson. Second Season QUIET Observations: Measure-

- ments of the Cosmic Microwave Background Polarization Power Spectrum at 95 GHz. *Ap. J.*, 760:145, December 2012.
- [11] D. Hanson, S. Hoover, A. Crites, P. A. R. Ade, K. A. Aird, J. E. Austermann, J. A. Beall, A. N. Bender, B. A. Benson, L. E. Bleem, J. J. Bock, J. E. Carlstrom, C. L. Chang, H. C. Chiang, H.-M. Cho, A. Conley, T. M. Crawford, T. de Haan, M. A. Dobbs, W. Everett, J. Gallicchio, J. Gao, E. M. George, N. W. Halverson, N. Harrington, J. W. Henning, G. C. Hilton, G. P. Holder, W. L. Holzapfel, J. D. Hrubes, N. Huang, J. Hubmayr, K. D. Irwin, R. Keisler, L. Knox, A. T. Lee, E. Leitch, D. Li, C. Liang, D. Luong-Van, G. Marsden, J. J. McMahon, J. Mehl, S. S. Meyer, L. Mocanu, T. E. Montroy, T. Natoli, J. P. Nibarger, V. Novosad, S. Padin, C. Pryke, C. L. Reichardt, J. E. Ruhl, B. R. Saliwanchik, J. T. Sayre, K. K. Schaffer, B. Schulz, G. Smecher, A. A. Stark, K. T. Story, C. Tucker, K. Vanderlinde, J. D. Vieira, M. P. Viero, G. Wang, V. Yefremenko, O. Zahn, and M. Zemcov. Detection of B-Mode Polarization in the Cosmic Microwave Background with Data from the South Pole Telescope. *Physical Review Letters*, 111(14):141301, October 2013.
- [12] R. K. Sachs and A. M. Wolfe. Perturbations of a Cosmological Model and Angular Variations of the Microwave Background. *Ap. J.*, 147:73, January 1967.
- [13] D. Larson, J. Dunkley, G. Hinshaw, E. Komatsu, M. R. Nolta, C. L. Bennett, B. Gold, M. Halpern, R. S. Hill, N. Jarosik, A. Kogut, M. Limon, S. S. Meyer, N. Odegard, L. Page, K. M. Smith, D. N. Spergel, G. S. Tucker, J. L. Weiland, E. Wollack, and E. L. Wright. Seven-year Wilkinson Microwave Anisotropy Probe (WMAP) Observations: Power Spectra and WMAP-derived Parameters. *Ap. J. Suppl.*, 192:16–+, February 2011.
- [14] R. B. Friedman, P. Ade, J. Bock, M. Bowden, M. L. Brown, G. Cahill, P. G. Castro, S. Church, T. Culverhouse, K. Ganga, W. K. Gear, S. Gupta, J. Hinderks, J. Ko-

vac, A. E. Lange, E. Leitch, S. J. Melhuish, Y. Memari, J. A. Murphy, A. Orlando, C. O’Sullivan, L. Piccirillo, C. Pryke, N. Rajguru, B. Rusholme, R. Schwarz, A. N. Taylor, K. L. Thompson, A. H. Turner, E. Y. S. Wu, M. Zemcov, and QUa D Collaboration. Small Angular Scale Measurements of the Cosmic Microwave Background Temperature Power Spectrum From QUaD. *Ap. J. Lett.*, 700:L187–L191, August 2009.

- [15] M. Lueker, C. L. Reichardt, K. K. Schaffer, O. Zahn, P. A. R. Ade, K. A. Aird, B. A. Benson, L. E. Bleem, J. E. Carlstrom, C. L. Chang, H.-M. Cho, T. M. Crawford, A. T. Crites, T. de Haan, M. A. Dobbs, E. M. George, N. R. Hall, N. W. Halverson, G. P. Holder, W. L. Holzapfel, J. D. Hrubes, M. Joy, R. Keisler, L. Knox, A. T. Lee, E. M. Leitch, J. J. McMahon, J. Mehl, S. S. Meyer, J. J. Mohr, T. E. Montroy, S. Padin, T. Plagge, C. Pryke, J. E. Ruhl, L. Shaw, E. Shirokoff, H. G. Spieler, B. Stalder, Z. Staniszewski, A. A. Stark, K. Vanderlinde, J. D. Vieira, and R. Williamson. Measurements of Secondary Cosmic Microwave Background Anisotropies with the South Pole Telescope. *Ap. J.*, 719:1045–1066, August 2010.
- [16] J. W. Fowler, V. Acquaviva, P. A. R. Ade, P. Aguirre, M. Amiri, J. W. Appel, L. F. Barrientos, E. S. Battistelli, J. R. Bond, B. Brown, B. Burger, J. Chervenak, S. Das, M. J. Devlin, S. R. Dicker, W. B. Doriese, J. Dunkley, R. Dünner, T. Essinger-Hileman, R. P. Fisher, A. Hajian, M. Halpern, M. Hasselfield, C. Hernández-Monteagudo, G. C. Hilton, M. Hilton, A. D. Hincks, R. Hlozek, K. M. Huffenberger, D. H. Hughes, J. P. Hughes, L. Infante, K. D. Irwin, R. Jimenez, J. B. Juin, M. Kaul, J. Klein, A. Kosowsky, J. M. Lau, M. Limon, Y.-T. Lin, R. H. Lupton, T. A. Marriage, D. Marsden, K. Martocci, P. Mausekopf, F. Menanteau, K. Moodley, H. Moseley, C. B. Netterfield, M. D. Niemack, M. R. Nolta, L. A. Page, L. Parker, B. Partridge, H. Quintana, B. Reid, N. Sehgal, J. Sievers, D. N. Spergel, S. T. Staggs, D. S. Swetz, E. R. Switzer, R. Thornton, H. Trac, C. Tucker, L. Verde, R. Warne, G. Wilson, E. Wollack, and Y. Zhao. The Ata-

- cama Cosmology Telescope: A Measurement of the 600  $<\ell < 8000$  Cosmic Microwave Background Power Spectrum at 148 GHz. *Ap. J.*, 722:1148–1161, October 2010.
- [17] W. Hu and M. White. A CMB polarization primer. *New Astronomy*, 2:323–344, 1997. astro-ph/9706147.
- [18] W. Hu. CMB temperature and polarization anisotropy fundamentals. *Annals of Physics*, 303:203–225, January 2003.
- [19] S. M. Carroll. *Spacetime and geometry. An introduction to general relativity*. 2004.
- [20] B. Gold, N. Odegard, J. L. Weiland, R. S. Hill, A. Kogut, C. L. Bennett, G. Hinshaw, X. Chen, J. Dunkley, M. Halpern, N. Jarosik, E. Komatsu, D. Larson, M. Limon, S. S. Meyer, M. R. Nolta, L. Page, K. M. Smith, D. N. Spergel, G. S. Tucker, E. Wollack, and E. L. Wright. Seven-year Wilkinson Microwave Anisotropy Probe (WMAP) Observations: Galactic Foreground Emission. *Ap. J. Suppl.*, 192:15–+, February 2011.
- [21] J. M. Kovac, E. M. Leitch, C. Pryke, J. E. Carlstrom, N. W. Halverson, and W. L. Holzapfel. Detection of polarization in the cosmic microwave background using DASI. *Nature*, 420:772, December 2002. astro-ph/0209478.
- [22] H. C. Chiang, P. A. R. Ade, D. Barkats, J. O. Battle, E. M. Bierman, J. J. Bock, C. D. Dowell, L. Duband, E. F. Hivon, W. L. Holzapfel, V. V. Hristov, W. C. Jones, B. G. Keating, J. M. Kovac, C. L. Kuo, A. E. Lange, E. M. Leitch, P. V. Mason, T. Matsumura, H. T. Nguyen, N. Ponthieu, C. Pryke, S. Richter, G. Rocha, C. Sheehy, Y. D. Takahashi, J. E. Tolan, and K. W. Yoon. Measurement of Cosmic Microwave Background Polarization Power Spectra from Two Years of BICEP Data. *Ap. J.*, 711:1123–1140, March 2010.
- [23] M. L. Brown, P. Ade, J. Bock, M. Bowden, G. Cahill, P. G. Castro, S. Church, T. Culverhouse, R. B. Friedman, K. Ganga, W. K. Gear, S. Gupta, J. Hinderks, J. Kovac, A. E.

- Lange, E. Leitch, S. J. Melhuish, Y. Memari, J. A. Murphy, A. Orlando, C. O’Sullivan, L. Piccirillo, C. Pryke, N. Rajguru, B. Rusholme, R. Schwarz, A. N. Taylor, K. L. Thompson, A. H. Turner, E. Y. S. Wu, M. Zemcov, and The QUa D collaboration. Improved Measurements of the Temperature and Polarization of the Cosmic Microwave Background from QUaD. *Ap. J.*, 705:978–999, November 2009.
- [24] C. Bischoff, L. Hyatt, J. J. McMahon, G. W. Nixon, D. Samtleben, K. M. Smith, K. Vanderlinde, D. Barkats, P. Farese, T. Gaier, J. O. Gundersen, M. M. Hedman, S. T. Staggs, B. Winstein, and CAPMAP Collaboration. New Measurements of Fine-Scale CMB Polarization Power Spectra from CAPMAP at Both 40 and 90 GHz. *Ap. J.*, 684:771–789, September 2008.
- [25] J. L. Sievers, C. Achermann, J. R. Bond, L. Bronfman, R. Bustos, C. R. Contaldi, C. Dickinson, P. G. Ferreira, M. E. Jones, A. M. Lewis, B. S. Mason, J. May, S. T. Myers, N. Oyarce, S. Padin, T. J. Pearson, M. Pospieszalski, A. C. S. Readhead, R. Reeves, A. C. Taylor, and S. Torres. Implications of the Cosmic Background Imager Polarization Data. *Ap. J.*, 660:976–987, May 2007.
- [26] T. E. Montroy, P. A. R. Ade, J. J. Bock, J. R. Bond, J. Borrill, A. Boscaleri, P. Cabella, C. R. Contaldi, B. P. Crill, P. de Bernardis, G. De Gasperis, A. de Oliveira-Costa, G. De Troia, G. di Stefano, E. Hivon, A. H. Jaffe, T. S. Kisner, W. C. Jones, A. E. Lange, S. Masi, P. D. Mauskopf, C. J. MacTavish, A. Melchiorri, P. Natoli, C. B. Netterfield, E. Pascale, F. Piacentini, D. Pogosyan, G. Polenta, S. Prunet, S. Ricciardi, G. Romeo, J. E. Ruhl, P. Santini, M. Tegmark, M. Veneziani, and N. Vittorio. A Measurement of the CMB  $\langle EE \rangle$  Spectrum from the 2003 Flight of BOOMERANG. *Ap. J.*, 647:813–822, August 2006.
- [27] BICEP2 Collaboration, P. A. R. Ade, R. W. Aikin, D. Barkats, S. J. Benton, C. A. Bischoff, J. J. Bock, J. A. Brevik, I. Buder, E. Bullock, C. D. Dowell, L. Duband, J. P.

- Filippini, S. Fliescher, S. R. Golwala, M. Halpern, M. Hasselfield, S. R. Hildebrandt, G. C. Hilton, V. V. Hristov, K. D. Irwin, K. S. Karkare, J. P. Kaufman, B. G. Keating, S. A. Kernasovskiy, J. M. Kovac, C. L. Kuo, E. M. Leitch, M. Lueker, P. Mason, C. B. Netterfield, H. T. Nguyen, R. O’Brient, R. W. Ogburn, IV, A. Orlando, C. Pryke, C. D. Reintsema, S. Richter, R. Schwarz, C. D. Sheehy, Z. K. Staniszewski, R. V. Sudiwala, G. P. Teply, J. E. Tolan, A. D. Turner, A. G. Vieregg, C. L. Wong, and K. W. Yoon. BICEP2 I: Detection Of B-mode Polarization at Degree Angular Scales. *ArXiv e-prints*, March 2014.
- [28] Planck Collaboration, R. Adam, P. A. R. Ade, N. Aghanim, M. Arnaud, J. Aumont, C. Baccigalupi, A. J. Banday, R. B. Barreiro, J. G. Bartlett, and et al. Planck intermediate results. XXX. The angular power spectrum of polarized dust emission at intermediate and high Galactic latitudes. *ArXiv e-prints*, September 2014.
- [29] E. Komatsu, K. M. Smith, J. Dunkley, C. L. Bennett, B. Gold, G. Hinshaw, N. Jarosik, D. Larson, M. R. Nolta, L. Page, D. N. Spergel, M. Halpern, R. S. Hill, A. Kogut, M. Limon, S. S. Meyer, N. Odegard, G. S. Tucker, J. L. Weiland, E. Wollack, and E. L. Wright. Seven-year Wilkinson Microwave Anisotropy Probe (WMAP) Observations: Cosmological Interpretation. *Ap. J. Suppl.*, 192:18–+, February 2011.
- [30] Britt Reichborn-Kjennerud et al. EBEX: A balloon-borne CMB polarization experiment. *Millimeter, Submillimeter, and Far-Infrared Detectors and Instrumentation for Astronomy V*, 7741, 2010.
- [31] B. Reichborn-Kjennerud. *Building and Flying the E and B Experiment to Measure the Polarization of the Cosmic Microwave Background*. PhD thesis, Columbia University, 2010.

- [32] S. Hanany and D. P. Marrone. Comparison of Designs of Off-Axis Gregorian Telescopes for Millimeter-Wave Large Focal-Plane Arrays. *Applied Optics*, 41:4666–4670, August 2002.
- [33] C. E. Tucker and P. A. R. Ade. Thermal filtering for large aperture cryogenic detector arrays. *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, 6275, July 2006.
- [34] K. Zilic. *Calibration and Design of the E and B EXperiment (EBEX) Cryogenic Receiver*. PhD thesis, University of Minnesota, 2014.
- [35] I. Sagiv et al. The ebex cryostat and supporting electronics.
- [36] S. Hanany, H. Hubmayr, B. R. Johnson, T. Matsumura, P. Oxley, and Thibodeau M. Millimeter-wave achromatic half-wave plate. *Applied Optics*, 44:4666–4670, August 2005.
- [37] J. Klein et al. A cryogenic half-wave plate polarimeter using a superconducting magnetic bearing. *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, 8150, September 2011.
- [38] F. Aubin, A. M. Aboobaker, P. Ade, C. Baccigalupi, C. Bao, J. Borrill, C. Cantalupo, D. Chapman, J. Didier, M. Dobbs, W. Grainger, S. Hanany, J. Hubmayr, P. Hyland, S. Hillbrand, A. Jaffe, B. Johnson, T. Jones, T. Kisner, J. Klein, A. Korotkov, S. Leach, A. Lee, M. Limon, K. MacDermid, T. Matsumura, X. Meng, A. Miller, M. Milligan, D. Polsgrove, N. Ponthieu, K. Raach, B. Reichborn-Kjennerud, I. Sagiv, G. Smecher, H. Tran, G. S. Tucker, Y. Vinokurov, A. Yadav, M. Zaldarriaga, and K. Zilic. First implementation of TES bolometer arrays with SQUID-based multiplexed readout on a balloon-borne platform. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 7741 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, July 2010.

- [39] Kevin D. MacDermid et al. The performance of the bolometer array and readout system during the recent flight of the e and b experiment (EBEX). *Millimeter, Submillimeter, and Far-Infrared Detectors and Instrumentation for Astronomy VII*, 9153, 2014.
- [40] M. Milligan. *The E and B EXperiment: Implementation and Analysis of the 2009 Engineering Flight*. PhD thesis, University of Minnesota, 2011.
- [41] N. N. Gandilo et al. Attitude determination for balloon-borne experiments. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 9145 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, July 2014.
- [42] James R. Wertz. *Spacecraft Attitude Determination and Control*. D. Reidel, 1980.
- [43] J. Didier et al. A High-resolution Pointing System for Fast Scanning Platforms: the EBEX Example. *Manuscript submitted for publication.*, 2015.
- [44] W. Hu, M. M. Hedman, and M. Zaldarriaga. Benchmark parameters for CMB polarization experiments. *Phys. Rev. D.*, 67:043004–+, February 2003. astro-ph/0210096.
- [45] rudolph emil kalman. a new approach to linear filtering and prediction problems. *transactions of the asme—journal of basic engineering*, 82(series d):35–45, 1960.
- [46] C. D. Alexander, W. R. Swift, K. Ghosh, and B. D. Ramsey. Design of a day/night star camera system. In R. E. Fischer and W. J. Smith, editors, *Current Developments in Optical Design and Optical Engineering VIII*, volume 3779 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pages 47–54, October 1999.
- [47] Yury Vinokurov. *EBEX, a balloon-borne telescope for observing the polarization of the cosmic microwave background*. PhD thesis, Brown University, 2010.



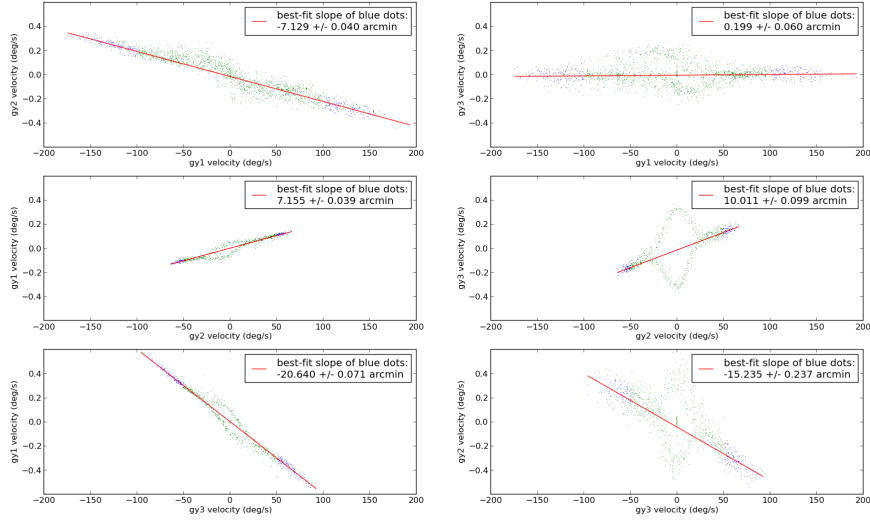
- [48] Y. S. Lee, Y. H. Kim, Y. Yi, and J. Kim. A Baffle Design for an Airglow Photometer onboard the Korea Sounding Rocket-III. *Journal of Korean Astronomical Society*, 33:165–172, December 2000.
- [49] B. J. Dober et al. The next-generation BLASTPol experiment. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 9153 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, July 2014.
- [50] m. rex. *the balloon-borne large aperture submillimeter telescope (blast)*. PhD thesis, university of pennsylvania, 2007.
- [51] Daniele Mortari, Malak A. Samaan, Christian Bruccoleri, and John L. Junkins. The pyramid star identification technique. *Navigation*, 51:171–184, Fall 2004.
- [52] Google Maps API. <https://developers.google.com/maps>.
- [53] jQuery Team. jQuery. <http://jquery.com/>.
- [54] K. M. Górski, E. Hivon, A. J. Banday, B. D. Wandelt, F. K. Hansen, M. Reinecke, and M. Bartelmann. HEALPix: A Framework for High-Resolution Discretization and Fast Analysis of Data Distributed on the Sphere. *Ap. J.*, 622:759–771, April 2005.
- [55] David A. Patterson, Garth A. Gibson, and Randy H. Katz. A case for redundant arrays of inexpensive disks (raid). Technical Report UCB/CSD-87-391, EECS Department, University of California, Berkeley, Dec 1987.
- [56] Donald Victor Wiebe. Dirfile Standards. <http://getdata.sourceforge.net>.
- [57] Apache. Subversion. <https://subversion.apache.org/>.
- [58] Eric A. Wan and Rudolph van der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of Symposium 2000 on Adaptive Systems for Signal Process-*

- ing, Communication and Control (AS-SPCC)*, Lake Louise, Alberta, Canada, October 2000. IEEE.
- [59] Simon J. Julier, Jeffrey, and K. Uhlmann. Unscented filtering and nonlinear estimation. In *Proceedings of the IEEE*, pages 401–422, 2004.
- [60] J. Didier et al. A High-resolution Pointing System for Fast Scanning Platforms: the EBEX Example. *Manuscript submitted for publication*.
- [61] M. Kamionkowski, A. Kosowsky, and A. Stebbins. Statistics of Cosmic Microwave Background Polarization. *Phys. Rev. D.*, 55:7368–7388, June 1997.
- [62] E. J. Jensen and G. E. Thomas. Numerical simulations of the effects of gravity waves on noctilucent clouds. *Journal of Geophysical Research*, 99:3421–3430, 1994.
- [63] G. Baumgarten and D. C. Fritts. Quantifying Kelvin-Helmholtz instability dynamics observed in noctilucent clouds: 1. Methods and observations. *Journal of Geophysical Research (Atmospheres)*, 119:9324–9337, August 2014.
- [64] D. C. Fritts, J. R. Isler, G. E. Thomas, and Ø. Andreassen. Wave breaking signatures in noctilucent clouds. *Geophysics Research Letters*, 20:2039–2042, 1993.

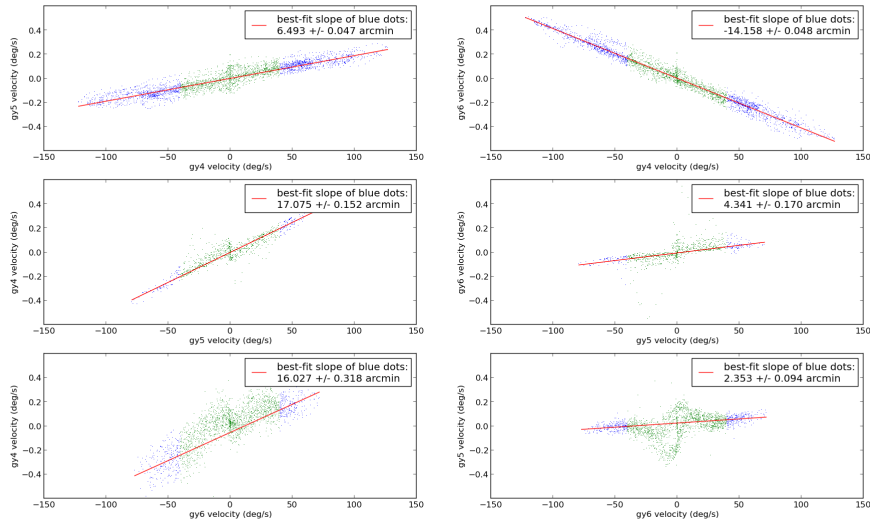
# Appendix A

## Gyro Orthogonalization Results

In August 2012, before shipping the telescope to Antarctica, we performed the gyro orthogonalization procedure discussed in Section 3.4.1 on both gyro boxes. Gyro box A contains gyros labeled “1”, “2”, and “3”. Gyro box B contains gyros labeled “4”, “5”, and “6”. As discussed in Section 3.4.1, the intermediate results of the orthogonalization procedure are the slopes of the best fit lines shown in Figure A.1, which are used as approximations for the elements in the inverse orthogonalization matrix. The inverse orthogonalization matrix is then inverted to give the final matrix.



(a)



(b)

Figure A.1: The results of the gyro orthogonalization procedure discussed in Section 3.4.1 for gyro boxes A and B, shown respectively in Subfigures (a) and (b). The slopes of the best fit lines provide the six off-diagonal elements of the inverse orthogonalization matrix. In the plots, the center bulges are a result of accidental rotations when changing directions. We therefore grouped the data into center points (green) and edge points (blue), and only used the edge points to find the slope.

# Appendix B

## Star Camera Assembly Procedure

**Star Camera 0** disassembly procedure (reverse for assembly):

1. Lay camera on its side
2. Unscrew back flange screws (5/16" allen/hex)
3. Slide out back structure (back half)
4. Disconnect cables connecting front and back halves
  - (a) Lens AD590 (BOB slot 5 of 6; 1 being power slot)
  - (b) Flange AD590 (BOB slot 6 of 6)
  - (c) Birger power
  - (d) 4-wire white connector
  - (e) Birger USB
  - (f) TDP from camera head
5. Unscrew front flange screws (special thin 9/16" drive socket)
6. Slide out front structure

**Star Camera 1** disassembly procedure (reverse for assembly):

1. Stand camera on its head

2. Unscrew back-flange screws (3/16" allen / hex)
3. Stand camera on its bottom
4. Unscrew front-flange screws (3/16" allen / hex)
5. Remove front flange
6. Unscrew front-weld-flange screws (9/64" allen / hex)
7. Disconnect front flange heater and AD590
8. Put front flange back on if desired
9. Slide vessel off
10. Remove o-ring from back-weld-flange

# Appendix C

## Star Camera 1 Electrical Documentation

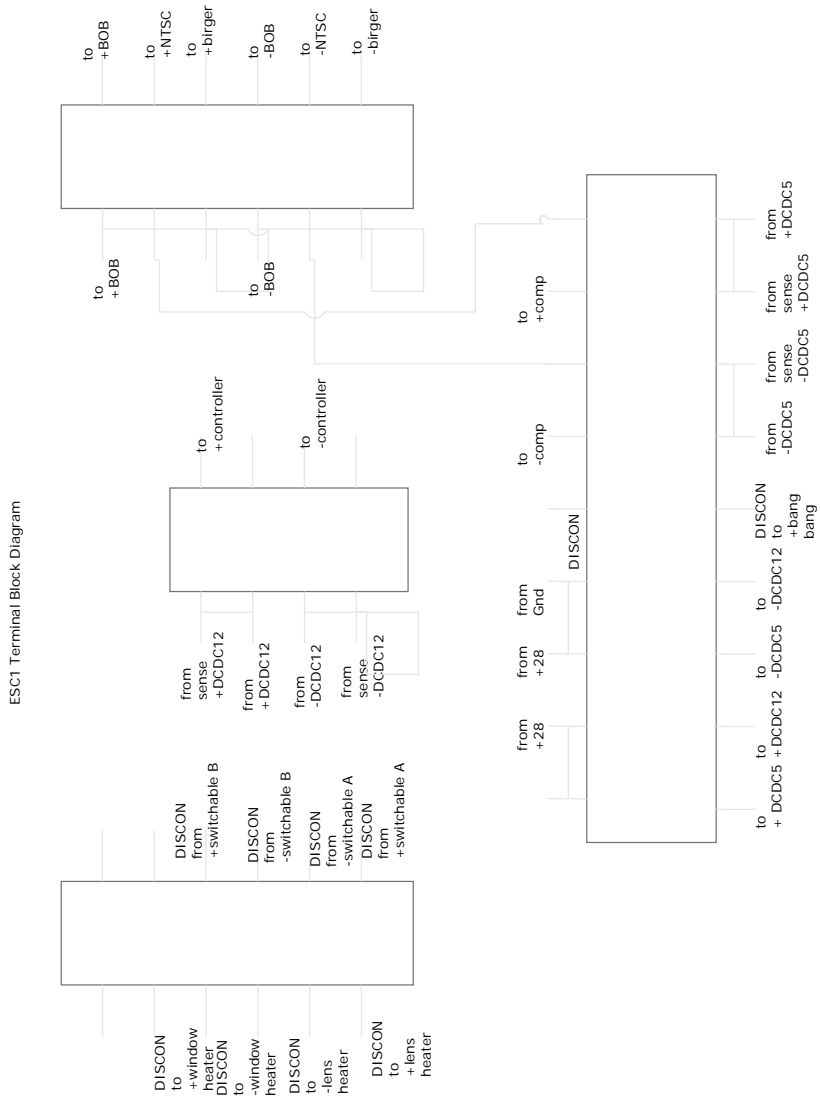


Figure C.1: Diagram documenting Star Camera 1 terminal blocks wiring. The top left terminal block labeling is deprecated, while the terminal block labeling shown in the “Heater Wiring Diagram”, also in the appendix, is up to date.



Sheet1

	PIN (PT) on SC end	WIRE	FUNCTION	DEVICE	PIN (JT) on ACS end
MS3114H-14-12P	A	red	+28 V DC	heater A	A
	B	black	ret	heater A	B
	C	red	+28 V DC	heater B	D
	D	black	ret	heater B	E
	E	white	trigger+	trigger	H
	F	gray	trigger-	trigger	F
	G	blue	hsync	vga	J
	H	purple	gnd-hsync	vga	K
	J	red (thick)	+28 V DC	power A	G
	K	black (thick)	ret	power A	C
	L	red (thick)	+28 V DC	power B	L
	M	black (thick)	ret	power B	M
	MS 3114H-14-19P	A	yellow	tx+	ethernet
B		orange	tx-	ethernet	
C		blue	rx+	ethernet	
D		purple	rx-	ethernet	
E		red	vcc	usb	
F		gray	data-	usb	
G		white	data+	usb	
H		black	gnd	usb	
J		blue	?	keyboard	
K		green	?	keyboard	
L		yellow	?	keyboard	
M		purple	?	keyboard	
N		brown	red	vga	
P		gray	green	vga	
R	white	blue	vga		
S	orange	gnd-vsync	vga		
T	yellow	vsync	vga		
U	blue	empty	empty		
V	green	empty	empty		
TNC	pin		video+	NTSC	
	shield		video-	NTSC	

Figure C.2: XSC1 connector pinout.



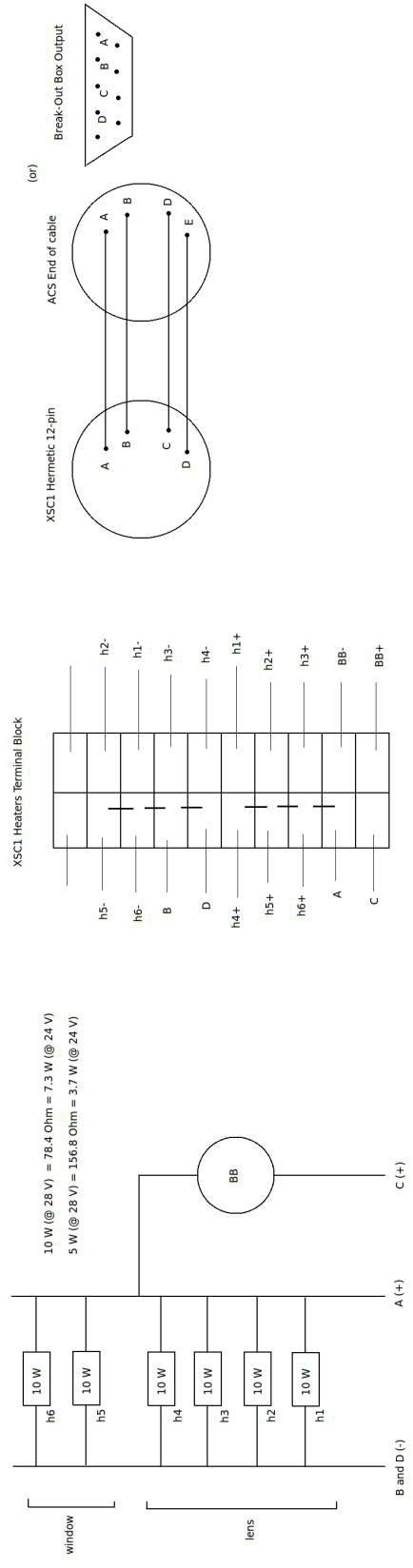


Figure C.4: XSC1 Heater Wiring Diagram. (Left) Block diagram of the components in the heating system. There are six 10 W heaters: two on the vessel window, and four on the lens. They are activated by either software control (positive line A) or bang-bang control (positive line C), implemented in parallel so as to act as a logical OR. (Center) Wiring configuration on the heater terminal block. (Right) Pinout of the two power lines in the 12-pin cable.

# Appendix D

## Leap IO Management Loading

### Parameters

The LEAP I/O (input/output) management library discussed in Section 9.3.2 provides a Params class that can be instantiated and customized for specifying which data to load and how to load it. The full list of customizable parameters is listed and explained here (taken from a comment string in the Params module):

```
MANDATORY VARIABLES: path to data
@ data_path (str): /path/to/data. This should either lead to
  * a leap_aligned directory containing all the segments or
  * a seth_aligned directory containing all the segments or
  * a flight_base directory with the following structure:
  acs/
    acs [link to acs vector base]
    pointing [link to latest pointing release]
    timing [link to timing release for acs/pointing]
  bolo/
    bolo [link to bolo vector base]
    timing [link to timing release for bolo]
  hwp/
    hwp [link to hwp vector base]
    timing [link to timing release for hwp]
  ss/
    ss [link to ss vector base]
    timing [link to timing release for ss]
```

#### TIME SELECTION: Method 1:

- @ segment\_list (list): list of segments you want to load. If None, all segments are loaded
- @ time\_range (list): beginning and end time (etime) for loading the data. this range will apply to all datasets (acs, hwp, bolo, ss)
- @ frame\_range[dataset\_name] (list): [start, end]: frames to load for a particular dataset. this parameter cannot be specified if time\_range is specified also. dataset\_name can ONLY be in ["acs", "bolo", "hwp", "ss"]

Note: you can specify time\_range OR frame\_range, not both

#### TIME SELECTION: Method 2: specify one of the following:

- @ segment\_and\_time\_list (dict): dictionary with desired segments as keys, and etime list for each segment.  
Ex: segment\_and\_time\_list =  
{"2012-12-31--13-17-57":[[etime1, etime2],[etime3, etime4]],  
"2013-01-06--18-21-48":[[etime5, etim6]]}
- @ segment\_and\_frame\_list (dict): dictionary with desired segments as keys, and frame\_range dictionary for each segment. Ex:  
segment\_and\_frame\_list =  
{"2012-12-31--13-17-57": {"acs": [[0, 100], [200, 400]],  
"bolo":[[0, 20]]}}  
segment\_and\_frame\_list[segment] is a dictionary with possible dataset\_name that can ONLY be in ["acs", "bolo", "hwp", "ss"]

#### CHANNEL SELECTION

- @ channels[dataset\_name] (list): list of all the channel names you want to load for a particular dataset. this includes usual channels, time channels and flags.
- @ bolo\_names (list): list of the bolos you want to load. Ex: ["64-1-3"]
- @ bolo\_load\_signals (bool): when bolo\_datset.load() is called

#### LESS COMMON VARIABLES

- @ progress\_indicator\_enabled (bool): do you want to see a progress indicator as you are iterating through segment?
- @ progress\_indicator\_allow\_printing (bool): progress\_indicator output includes line feeds to allow for printing in between status lines
- @ interpolate\_acs (bool): when loading bolo\_dataset, do you want to load the acs channels, interpolate to the bolo rate, as bolo\_dataset.acs? True by default.
- @ interpolate\_hwp (bool): same as for acs. False by default.

@ interpolate\_ss (bool): same as for acs

#### RARE VARIABLES

@ print\_frames\_from\_time\_range (bool): print the frames loaded for a given  
time range

@ xsc\_solutions\_path (str): /path/to/xsc/solution/txtfile

@ bolo\_delay (float): time in seconds the bolometer signal should be  
delayed compared to acs/pointing data

@ rollover (bool): do you want the bolo signal to be unwrapped

@ galactic\_xsc\_pointing (bool): generate galactic lon/lat for pointing  
dataset from pointing.channels ra/dec

@ print\_warnings (bool)