2009

Computational Techniques for Reducing Spectra of the Giant Planets in Our Solar System

Holly L. Grimes
Portland State University

COMPUTATIONAL TECHNIQUES FOR REDUCING SPECTRA OF THE GIANT PLANETS IN OUR SOLAR SYSTEM

by

HOLLY L. GRIMES

A thesis submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE
in
COMPUTER SCIENCE

Portland State University
2009

Dedication

To Mom, Dad, and Sarah.

Acknowledgements

I would like to thank my advisor, Dr. Bryant York, for his guidance during my research and during the writing of this thesis. His many suggestions and questions improved the clarity and completeness of this paper. I would also like to thank my thesis reader, Dr. David Maier, for his thorough commentary on later drafts of this thesis.

I want to thank Dr. Glenn Orton, my mentor at the Jet Propulsion Laboratory, and Dr. Leigh Fletcher, Dr. Orton's post-doctoral assistant, for their guidance and advice during my research. I also thank the Jet Propulsion Laboratory for providing computing resources used in this research and for funding this work through their Research Apprenticeship Program.

Table of Contents

List of Tables

List of Figures

Glossary

absolute calibration: the process of converting the intensity axis of a spectrum from units of photon count to units of radiance, such as W/cm^2/sr/µm.

aperture width: the number of rows of pixels that are averaged together to produce an extracted spectrum.

calibration images: spectral images that are used only for the calibration of scientific images.

data reduction: the calibration of raw data in preparation for analysis.

DetermineDispersion1: the dispersion axis determination algorithm discussed in the COMICS Data Reduction Manual.

DetermineDispersion2: the first version of the newly developed dispersion axis determination algorithm.

DetermineDispersion3: the second version of the newly developed dispersion axis determination algorithm.

DetermineSpatial: the spatial axis determination algorithm.

dispersion axis: the x-axis of a spectral image.

extraction: the process of obtaining a one or more spectra from a spectral image.

flat spectral image: a calibration image that captures the non-uniformities in the spectrometer detector's response to light.

InImage: the input image for an algorithm.

orthogonalization: the process of transforming a spectral image so that the image's dispersion and spatial axes are perpendicular, the dispersion axis is horizontal, and the spatial axis is vertical.

photon count: the average number of photons that have hit the pixels in a column of the detector array. This term may also be used to denote the number of photons that have hit a single pixel. The intended meaning will be clear from the context.

radiance: a measure of the intensity of the radiation emitted from an object. Radiance may be expressed in units such as W/cm^2/sr/µm.

read-out pattern noise: the noise generated by detectors when they read data.

resolution: a measure of the degree of separation of wavelengths along the x-axis of a spectrum.

scientific images: spectral images of objects that are of interest to astronomers. For the purposes of this research, all scientific images are the spectral images of planets and stars.

sky emission: the infrared (thermal) radiation emitted by Earth's atmosphere.

spatial axis: the y-axis of a spectral image.

spectral image: a two-dimensional image obtained from a spectrometer that reports the intensity of the electromagnetic radiation emitted from an object (such as a star or a planet) as a function of wavelength and of position along the spectrometer slit. The x-axis of such an image is the dispersion axis, which keeps track of the wavelength of the radiation; the y-axis is the spatial axis, which keeps track of the position along the slit.

spectral line: a line within a spectral image that reports the amount of radiation emitted at a single wavelength as a function of position along the spectrometer slit.

spectrometer: an instrument that separates light according to wavelength, much as a prism does.

spectrum: a one-dimensional plot of the intensity of the electromagnetic radiation emitted from an object (such as a star or a planet) as a function of wavelength.

spectrum calibration: the process of converting the axes of a spectrum to units that are meaningful to scientists. This process involves two types of calibration: wavelength calibration and absolute calibration.

theta: the angle between the spatial and dispersion axes of a spectral image.

Transformation1: The initial version of the image transformation algorithm that is used during orthogonalization.

Transformation2: The first refinement of the image transformation algorithm.

Transformation3: The second refinement of the image transformation algorithm.

wavelength calibration: the process of converting the wavelength axis of a spectrum from units of pixel number to units of wavelength, such as micrometers (µm).

Chapter 1 : Introduction

1.1 Problem Statement

The dynamic atmospheres of Jupiter, Saturn, Uranus, and Neptune provide a rich source of meteorological phenomena for scientists to study. To investigate these planets, scientists obtain spectral images of these bodies using various instruments including the Cooled Mid-Infrared Camera and Spectrometer (COMICS) at the Subaru Telescope Facility at Mauna Kea, Hawaii. These spectral images are two-dimensional arrays of double precision floating point values that have been read from a detector array. Such images must be reduced before the information they contain can be analyzed. The reduction process for spectral images from COMICS involves several steps:

1. *Sky subtraction:* the background radiation from Earth's atmosphere must be subtracted from the spectral images.

2. *Read-out pattern noise reduction:* the noise related to reading data from detectors must be subtracted from the spectral images.

3. *Division by the flat:* the spectral images must be corrected for non-uniformities in the detector array response.

4. *Orthogonalization:* the spectral images must be transformed so that the images' axes are perpendicular.

5. *Extraction:* individual spectra must be extracted from the spectral images. These spectra are plots of pixel intensity as a function of position along the x-axis of a spectral image.

6. *Calibration:* the x- and y- axes of the extracted spectra must be converted to units that are meaningful for scientific analysis.

In earlier work, the author developed software tools to support the first three steps in this reduction process. This thesis presents algorithms for performing the next two reduction steps, namely orthogonalization and extraction. More specifically, this thesis addresses the following research question: What are proper methods of orthogonalizing spectral images in preparation for extraction?

1.2 Motivation and Background

The research presented in this thesis is a continuation of work begun by the author as a student intern at the Jet Propulsion Laboratory, California Institute of Technology (JPL/CIT) in the summer of 2008. During this internship, the author worked under the direction of Dr. Glenn Orton, an astronomer who studies the atmospheres of Jupiter, Saturn, Uranus, and Neptune. Dr. Orton and his colleagues in the Earth and Planetary Atmospheres group observe these planets at telescope facilities around the world, collecting data in the form of infrared images and spectra. The images are obtained using cameras and provide data similar to that contained in a photograph, while the spectra are obtained using spectrometers and provide data as described in Section 1.3 of this thesis. These data must be reduced before they can be analyzed. The Earth and Planetary Atmospheres group has been collecting imaging data for decades now, so they have a vast software base to aid in the reduction of such data. However, this group has only recently begun collecting spectroscopic data. Traditional reduction techniques for images cannot, in general, be applied to spectra, so new software needed to be developed to reduce the new type of data. During the summer of 2008, the author produced software to facilitate the reduction of spectra from the Cooled Mid-Infrared Camera and Spectrometer (COMICS) at the Subaru Telescope; the reduction process used for this instrument along with the format of the data after reduction are described in Section 1.3. Partial support for the reduction of COMICS spectra was implemented over the summer, and this thesis is a continuation of that work.

Once spectroscopic data have been reduced, the data can be analyzed and published in peer-reviewed journals. These published data provide scientists with information about the atmospheric composition of the giant planets in our solar system; this composition information in turn provides insight into the structure and dynamics of these planetary atmospheres [15] [16][18][19]. In addition to advancing scientists' understanding of these important bodies in our solar system, studying the outer planets also provides a knowledge base that scientists can use as they seek to understand the composition and dynamics of the atmospheres of giant planets in other planetary systems [16]. In providing an easy method for Dr. Orton and his group to use to reduce their data, the algorithms presented here will help these scientists as they seek to increase humanity's knowledge about our universe. It may also be possible to use these algorithms to process data from other instruments (in addition to COMICS);

these instruments may collect data for use in scientific inquiry or in military applications.

This data reduction problem was chosen because, even after the author's summer internship, suitable algorithms had not been developed for performing all of the reduction steps required for COMICS spectra. Questions regarding what computational techniques should be used to complete these reduction steps provided an interesting research topic to address in a thesis. This research was conducted in a computer science department because the work described here did not require the use of scientific principles to develop a new process for reducing data from the COMICS spectrometer. Instead, this research involved looking at a preexisting data reduction process for COMICS spectra (see Section 1.3) and developing new algorithms, where necessary, to support this reduction process. The novel algorithms presented here apply standard image processing techniques to the problem of reducing COMICS spectra. Such development of new algorithms falls under the realm of computer science.

1.3 Terminology

To perform their research, Dr. Orton and his colleagues observe Jupiter, Saturn, Uranus, and Neptune using various ground-based telescopes including the Subaru Telescope, NASA's Infrared Telescope Facility (IRTF), the Gemini North and South Telescopes, and the ESO Very Large Telescope (VLT). These telescopes are all equipped with spectrometers that are sensitive to infrared (thermal) radiation. A spectrometer is an instrument that separates light according to wavelength, much as a prism does [15]. The schematic of a spectrometer is shown in Figure 1-1. This spectrometer works as follows: the telescope is pointed at an object of interest, such as a planet or a star. Light from the object enters the telescope and is directed through a slit and into the spectrometer. Once inside the spectrometer, the light is focused onto a diffraction grating that separates the light into its component wavelengths. The ability of this grating to separate light of different wavelengths is indicated by its resolution; the higher the resolution, the greater the distance that separates the different wavelengths after the light leaves the grating. From the grating, the light is directed onto a detector that records the data. This detector may be thought of as a two-dimensional array of pixels where each pixel collects a portion of the light that hits the detector.

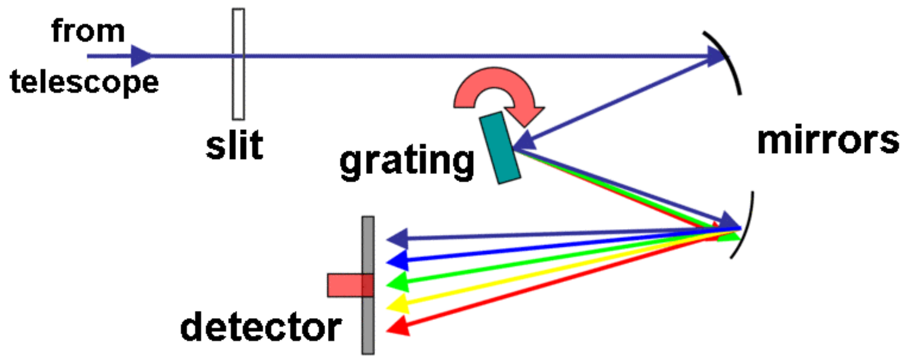


Figure 1-1. The schematic of a spectrometer. Each of the colored arrows in the schematic represents a different wavelength of light. This figure is a modified version of the image http://en.wikipedia.org/wiki/Image:Spectrometer_schematic.gif made by Kkmurray.

The data obtained from a telescope's spectrometer are reported in the form of a spectral image such as the one shown on the right half of Figure 1-2. A spectral image is a two-dimensional image that reports the intensity of the electromagnetic radiation emitted from an object as a function of wavelength and of position along the spectrometer slit. The spectral image's x-axis is the dispersion axis, which keeps track of the wavelength of the light that was captured by the detector; the y-axis of this image is the spatial axis, which keeps track of the position along the spectrometer's slit. This spectral image contains multiple vertical columns of bright pixels, and each of these columns is called a spectral line. Each spectral line reports the intensity of the radiation emitted at a particular wavelength from each position along the spectrometer slit.

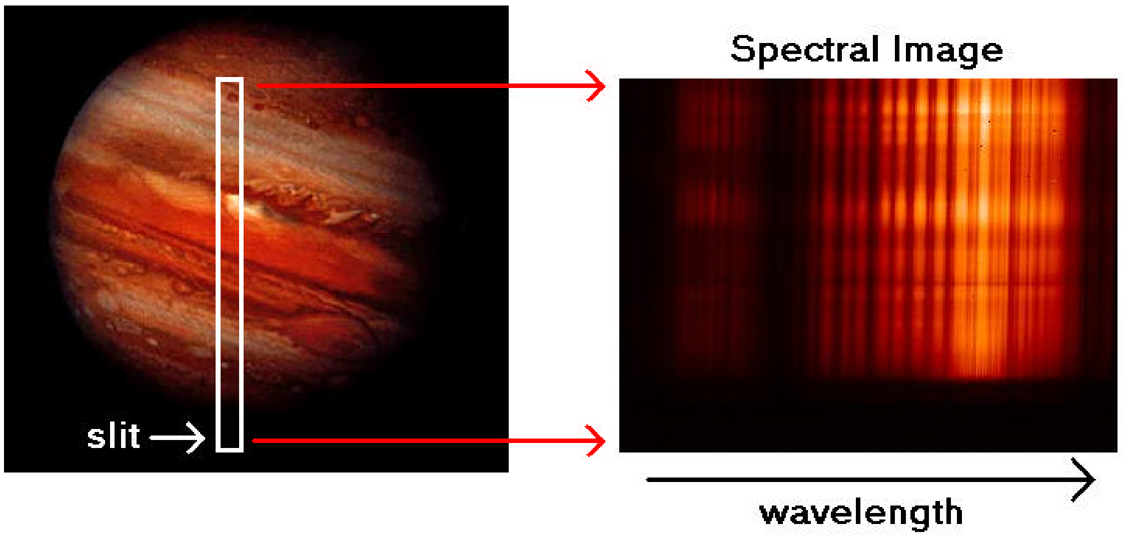


Figure 1-2. (left) The position of the spectrometer slit on an object of interest (Jupiter), and (right) the spectral image obtained at that slit position. The shades of red in this spectral image indicate the intensity of each pixel, not the wavelength of the light associated with those pixels.

These spectral images do not present the data in a form that is easy for scientists to analyze. Therefore, the data must be reduced, or calibrated, to prepare them for analysis. The data reduction process for spectral images involves several major steps, which are outlined below for infrared data [10][12][13]. During this reduction procedure, two types of spectral images are used: we will refer to them as scientific images and calibration images. Scientific images are the spectral images of objects that are of interest to astronomers; for the purposes of this research, the scientific images are the spectral images of planets and stars. Calibration images are spectral images that are used only for the calibration of scientific images.

1. *Sky subtraction.*

   Earth's atmosphere emits a significant amount of infrared radiation; the intensity of this sky emission is high enough to overwhelm the signal resulting from the infrared radiation from the outer planets. A calibration image of this background radiation must be subtracted from the scientific images.

2. *Read-out pattern noise reduction.*

   The electrical circuits present in spectrometer detectors generate a certain amount of noise when the data are read from the detector and stored to disk. This noise pattern must be subtracted from the scientific images.

3. *Division by the flat.*

   The detectors used with telescope spectrometers may be viewed as two-dimensional arrays of pixels. Each pixel in such a detector array responds differently to the light incident upon it. To correct for these non-uniformities in the detector array response, each scientific image must be divided by a flat calibration image. Here, the term flat refers to the fact that the flat image captures the detector's response to a uniform light source.

4. *Spectral image orthogonalization.*

   As shown in Figure 1-3, the spatial and dispersion axes of a scientific image may not be perpendicular to one another; this axis skew is caused by the characteristics of the COMICS spectrometer. Orthogonalization is the process of transforming a spectral image so that these axes are perpendicular. The size of the spectral image is the same both before and after orthogonalization.

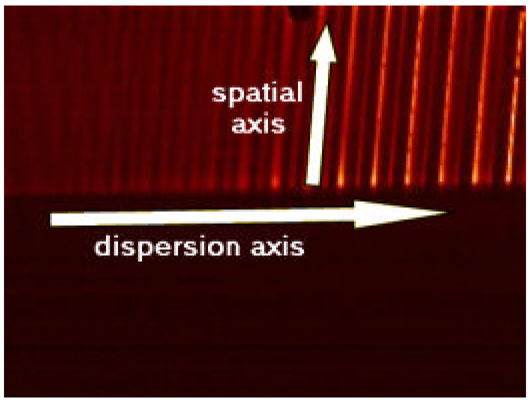


Figure 1-3. A spectral image for which the spatial axis is not perpendicular to the dispersion axis.

5. *Spectrum extraction.*

   To reformat the data contained in a science image so that scientists may analyze it, one or more spectra must be extracted from the spectral image. A spectrum is a plot of the intensity of the electromagnetic radiation emitted from an object as a function of wavelength. The intensity axis is often expressed in units of photon counts, where the photon count is the average number of photons that have hit the pixels in one column of the orthogonalized image. Instead of being specified in units of wavelength, the wavelength axis of a newly extracted spectrum is often expressed in terms of pixel number, where the pixel number is the position along the dispersion axis of the column of pixels whose photon count is being reported.

6. *Spectrum calibration.*

   The intensity and wavelength axes of a newly extracted spectrum are not expressed in terms of useful units. Spectrum calibration is the process of converting the axes to units that are more meaningful for scientists; this process requires two types of calibration: wavelength calibration and absolute calibration. Wavelength calibration is the process of converting the wavelength axis from units of pixel number to units of wavelength, such a micrometers (µm). Absolute calibration is the process of converting the intensity axis from units of photon count to units of radiance, such as W/cm^2/sr/µm; radiance is a measure of the intensity of the radiation emitted from an object.

Figure 1-4 provides an illustration of these six reduction steps [10].

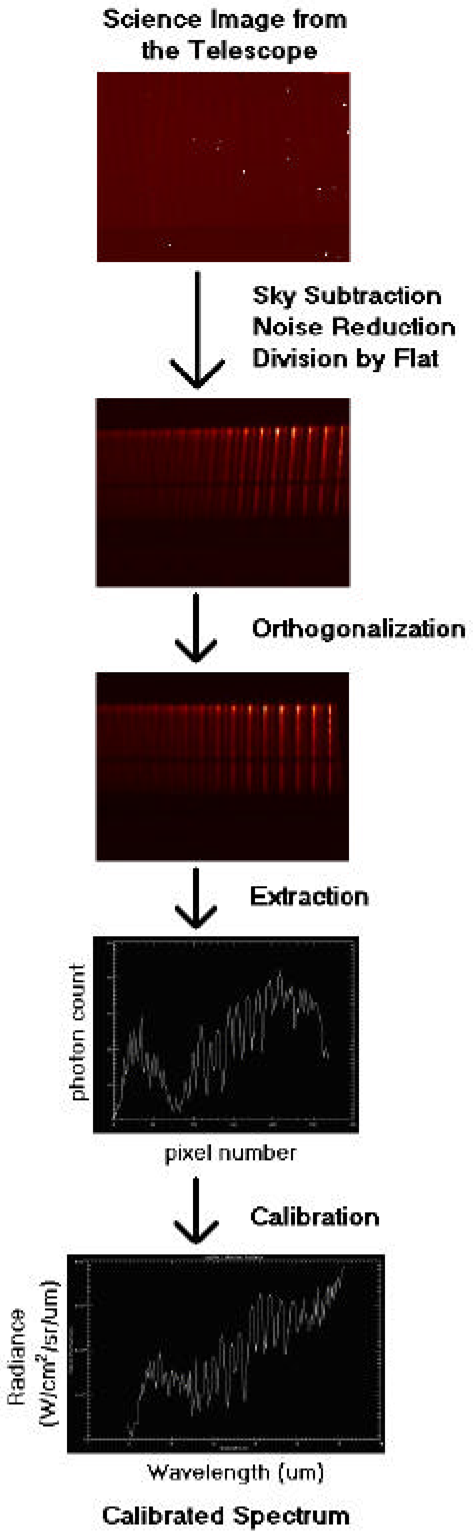


Figure 1-4. An illustration of the process of data reduction for spectral images.

1.4 Telescopes and their Properties

The Earth and Planetary Atmospheres Group at JPL/CIT routinely use the infrared spectrometers at several ground-based telescopes for their observations. The 8.2-m Subaru telescope is located on Mauna Kea in Hawaii and is operated by the National Astronomical Observatory of Japan (NAOJ) [15]. Subaru is equipped with the Cooled Mid-Infrared Camera and Spectrometer (COMICS), an instrument that is sensitive to infrared radiation in the wavelength range from 7.8 to 24.5 µm. The COMICS spectrometer is equipped with low (250), medium (2,500), and high (10,000) resolution gratings that cover the range from 7.8 to 13.3 µm. COMICS is also outfitted with five spectroscopy detectors; only one of these detectors is used for low-resolution observations, while all five detectors are used during medium- and high- resolution observations[10]. The 3-m Infrared Telescope Facility is also located on Mauna Kea and is run by NASA and the University of Hawaii; the IRTF is equipped with an infrared spectrograph called SpeX (a spectrograph is similar to a spectrometer). The 8.1-m Gemini North and South telescopes are a pair of identical telescopes, one in the northern hemisphere and the other in the southern hemisphere, that are operated by the International Gemini Consortium. Gemini North is located on Mauna Kea and is equipped with the Michelle infrared spectrometer. Gemini South is located at Cerro Pachon in Chile and houses the T-Recs infrared spectrometer. The Very Large Telescope contains four 8.2-m telescopes that can operate individually or as a single unit. VLT is operated in La Paranal, Chile by the European Southern Observatory (ESO) and houses the VISIR camera and spectrometer [15].

Chapter 2 : Methodology and Related Work

2.1 Approach

The approach presented in this thesis for spectral image orthogonalization is based on the work of Dr. Glenn Orton and on the work of the COMICS team at the Subaru Telescope Facility. Before describing this approach, it will be useful to describe the data sets for which the algorithms presented in this thesis were developed (Section 2.1.1). Then we will discuss the general approaches used for developing (Section 2.1.2) and testing (Section 2.1.3) the algorithms presented in this thesis.

2.1.1 Data Sets

For the purposes of this research, a data set is a collection of spectral images that were obtained using the COMICS spectrometer during one night of making observations; Dr. Orton's data sets typically contain 25 to 100 images. These spectral images are encoded using the Flexible Image Transport System (FITS) format, a standard file format used by astronomers. The FITS files used in this research contain two components: a 320×240 array of double-precision floating-point numbers, and an array of strings. The floating-point array represents a spectral image, while the string array is the file header. This FITS header contains a description of the data stored in the file along with information about the conditions under which the data were collected. Such FITS files are typically g-zipped with the compressed file sizes ranging from 500 to 600 kB. More information about the FITS format is available online at http://fits.gsfc.nasa.gov.

2.1.2 Approach for Algorithms

To orthogonalize the images in a data set, the spectral images of stars are used to determine the spatial axis orientations associated with the images in the data set. Then the spectral image of a planet is used to determine the dispersion axis orientation associated with the data. Once the spatial and dispersion axes have been determined, the spectral images in the data set are transformed so that the axes are perpendicular, with the dispersion axis being horizontal and the spatial axis being vertical. Spectra can then be extracted from the orthogonalized spectral images. All of these algorithms operate on individual spectral images.

2.1.3 Approach for Testing

The algorithms developed to perform orthogonalization have been tested for both correctness and efficiency; the tests used here are based on techniques used by Dr. Orton and by the COMICS team. Correctness has been evaluated using data provided by Dr. Orton. Spectral images have been orthogonalized using the algorithms presented in this thesis. Some of these algorithms were developed by the author, and others were developed by the COMICS team. The spatial and dispersion axes have been re-determined for the orthogonalized images using the axis determination algorithms described in this thesis. The angle between these determined axes was also calculated for the orthogonalized images. If the orthogonalized images' axes are perpendicular, with the spatial axes being vertical and the dispersion axes being horizontal, then the output of the orthogonalization algorithm is correct. A second measure of correctness could be to check that, for any given spectral image, the sum of all the pixels' intensities is the same before and after orthogonalization. This correctness criterion cannot be used here because the spectral images are the same size before and after orthogonalization: as a result of this size restriction, some pixels from the original image are mapped outside the bounds of the transformed image by the orthogonalization transformation, resulting in "lost pixels" and a sum of pixel values that is smaller after the transformation than before the transformation. The efficiency of this algorithm has been measured in terms of running time: the implementation of the orthogonalization algorithm must be able to process a single spectral image in less than 5 minutes. The performance of the orthogonalization algorithm based on these correctness and efficiency criteria are reported in Chapter 3 of this thesis.

2.2 Distinctive Aspects of this Approach

The orthogonalization algorithm is divided into three stages: spatial axis determination, dispersion axis determination, and image transformation. The spatial axis determination and image transformation algorithms developed by the COMICS team were implemented for this thesis. However, a new algorithm for dispersion axis determination was developed because the algorithm presented by the COMICS team for this purpose was strongly connected to wavelength calibration. Dr. Orton needed an algorithm that did not involve such a connection because he wanted the software for each reduction step to be designed as a separate module. Modularity will simplify the task of generalizing these algorithms so that they can be used to reduce data from other instruments, a task that will be undertaken in future work. The new dispersion axis determination algorithm uses general image processing techniques to separate the axis determination procedure from wavelength calibration.

2.3 Orthogonalization Algorithm

After a spectral image has undergone the reduction steps of sky subtraction, read-out pattern noise reduction, and division by the flat, that image is ready to be orthogonalized. Orthogonalization is the process of transforming a spectral image so that the image's dispersion and spatial axes are perpendicular to one another, with the dispersion axis being horizontal and the spatial axis being vertical. The COMICS Data Reduction Manual [10] states that the orthogonalization process may be implemented in three steps:

1. Determine the orientation of the spatial axis for a high S/N spectral image of a standard star. The orientation of this axis, y, changes as a function of x (see Figure 2-1a); this variation in orientation can be captured by a quadratic function of the form $y = b_0 x^2 + b_1 x + b_2$. This step should be done once for each set of low-resolution spectral images and five times for each set of medium-resolution spectral images; these numbers of repetitions are used because low-resolution spectra are obtained using one of the five spectroscopy detectors available on COMICS while the medium-resolution spectra are obtained using all five spectroscopy detectors[10]. The algorithm used to perform this step may involve user interaction.

2. Determine the orientation of the dispersion axis for a high S/N spectral image of a planet. The orientation of this axis, x, changes as a function of y (see Figure 2-1b); this variation in orientation can be captured by a linear function of the form $x = a_0 y + a_1$. This dispersion axis is described by a linear function instead of a quadratic function because the orientation of this axis can be inferred from the orientation of the spectral lines in an image, and these spectral lines are described by linear equations. This step normally should be performed once for each set of low-resolution spectral images and once for each set of medium-resolution images because the dispersion axis orientation is usually the same for images of the same resolution in the same data set (see the results presented in Section 3.3.1.2). The algorithm used to perform this step may involve user interaction.
3. Using information about the axes determined in steps 1 and 2, transform the images so that the spatial and dispersion axes are perpendicular. This step should be performed once for each image in a data set, and the algorithm used to perform this step should not require user interaction.

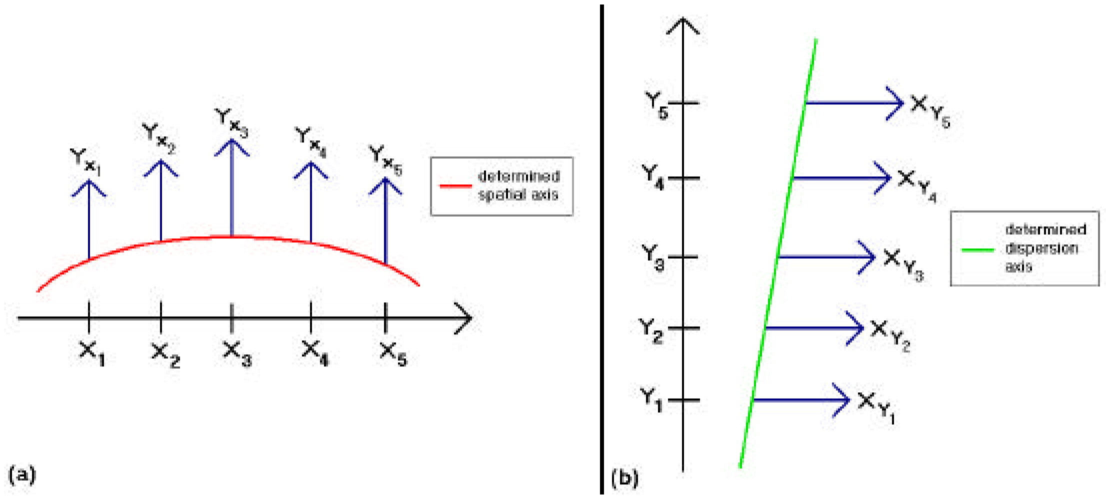


Figure 2-1. A conceptual illustration of (a) determining the spatial axis and (b) determining the dispersion axis. The red curve gives the location of the "base" of the spatial (y) axis as a function of x; the blue arrows labeled Yx_1 through Yx_5 show the position of the spatial axis for the x-values x_1 through x_5, respectively. Likewise, the green line give the location of the "base" of the dispersion (x) axis as a function of y; the blue arrows labeled XY_1 through XY_5 show the position of the dispersion axis for the y-values Y_1 through Y_5, respectively.

Section 2.3.1 describes the algorithm for spatial axis determination, Section 2.3.2 describes three algorithms for dispersion axis determination, and Section 2.3.3 describes three algorithms for transforming an image so that its axes are perpendicular. In these sections, the input image will be referred to as InImage.

2.3.1 Spatial Axis Determination

This algorithm, which will be referred to as DetermineSpatial in the remainder of this thesis, requires the following input parameters:

1. **InImage:** The spectral image of a standard star represented as an 320×240 array of floating point values. The dimensions of this array will be labeled x and y.

2. **x_{min}:** The minimum x-value for which we will perform calculations; the COMICS manual suggests using a value of 30. The rationale for choosing this value for

x_{min} is not explained in the manual, but the value was probably chosen to exclude pixels near the edge of the image from the calculation: due to the characteristics of the spectroscopy detector, the pixels near the edge are less accurate than the pixels in the middle of the image.[10]

3. **x_{max}:** The maximum x-value for which we will perform calculations. The COMICS manual suggests using a value of 290; the reason for choosing this value for x_{max} is the same as that used to choose a value for x_{min}.

To determine the spatial axis of InImage, first, automatically determine the y-values (y_{min} and y_{max}) between which the star spectrum appears in this image (see Figure 2-2). These boundary values are obtained using a 2-standard-deviation threshold above the mean pixel value; the user is allowed to adjust these values if they are not reasonable.

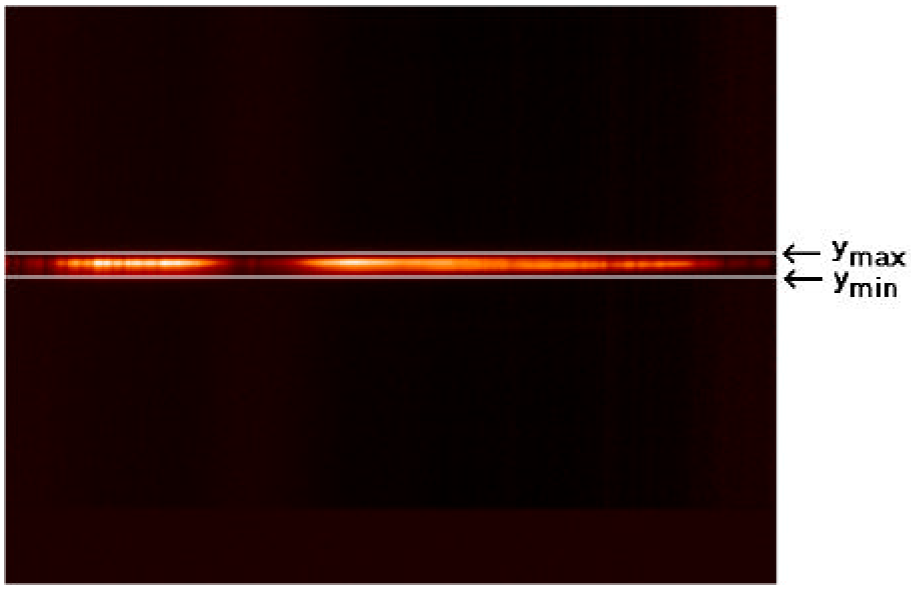


Figure 2-2. The two-sigma threshold estimates for the minimum (y_{min}) and maximum (y_{max}) y-values between which the star spectrum appears.

Next, locate the peak position (in terms of intensity) of the star spectrum for each x-value between x_{min} and x_{max}. To determine the coordinates of these peak positions this algorithm employs a two-step method that is used by the q-series task *q_startrace*; q-series is free software available on the Internet at http://canadia.ir.isas.ac.jp/comics/open/rbin/rbin.html.

<u>Step 1</u>: For each value of y between y_{min} and y_{max}, compute the arithmetic mean of the x-values in the row corresponding that value of y. Fit these average values with a Gaussian of the form

$$A \cdot \exp\left(\frac{-z^2}{2}\right) + D \cdot x + E, \qquad (2.1)$$

where $z = \dfrac{x - B}{C}$.

To perform this Gaussian fit, the algorithm implemented by the Interactive Data Language's *gaussfit* function is used; *gaussfit* uses gradient expansion to calculate a non-linear least squares fit. The initial guesses for the fit parameters are as follows, where *m* is the vector containing the average for each y-value and *n* is the size of vector *m*:

$$A = m\left[\frac{n}{2}\right] - \frac{m[n-1] + m[0]}{2.0}$$

$$B = \frac{n}{2.0}$$

$$C = 1.5 \qquad (2.2)$$

$$D = m[0]$$

$$E = \frac{m[n-1] - m[0]}{n}$$

Step 2: For each value of x between x_{min} and x_{max}, take the y-values corresponding to that value of x and fit those y-values with a Gaussian of the same form as was used in Step 1. Use the fit parameters associated with the final Gaussian from Step 1 as initial guesses for the fit parameters associated with the Gaussians here. The x- and y-coordinates of the Gaussian centers for each x-value are recorded for later use. Figure 2-3 shows a star spectrum with the calculated Gaussian centers displayed over the top.

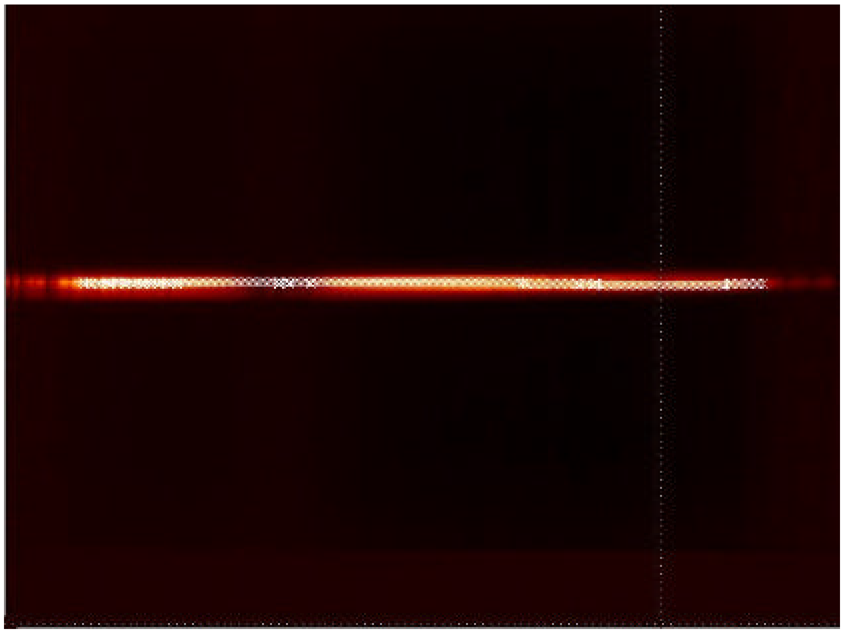


Figure 2-3. A star spectrum with the calculated Gaussian centers plotted over the top.

These peak positions are fit with both a quadratic equation of the form $y = b_0 x^2 + b_1 x + b_2$ and a linear equation of the form $y = b_1 x + b_2$. Both fits are displayed as shown in Figure 2-4 with a plot of each fit superimposed upon the star image. The user is asked to identify which fit, if any, best describes the axis; here, the user is an astronomer or a student worker who is using an implementation of this algorithm to reduce spectra. The parameters of the fit chosen by the user are the parameters that describe the spatial axis of InImage.

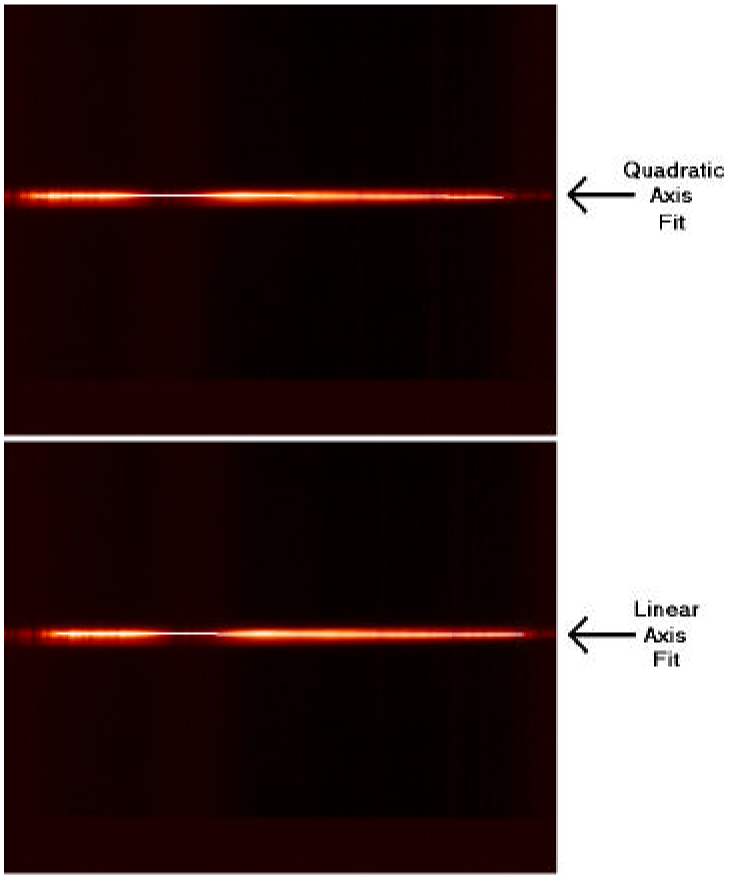


Figure 2-4. Quadratic and linear spatial axis fits for the spectral image of a standard star.

2.3.2 Dispersion Axis Determination

The COMICS Data Reduction Manual [10] proposes an algorithm for determining the dispersion axis, an algorithm that is partially implemented by the q-series task *q_sky_nlow*. This algorithm (which will be referred to as DetermineDispersion1) has an undesirable property, so a new algorithm (DetermineDispersion2) was developed to perform this determination. The newly developed algorithm was later refined after testing revealed problems; this refined algorithm will be called DetermineDispersion3. The COMICS Data Reduction Manual algorithm and both versions of the newly developed algorithm are described below. All three of these algorithms require one input parameter: InImage, the spectral image of a planet represented as a two-dimensional array of floating point values; the dimensions of this array are labeled x and y.

2.3.2.1 The COMICS Data Reduction Manual Algorithm (DetermineDispersion1)

The following algorithm will determine the dispersion axis of InImage:

For each value of y in the set {30, 40, 50, … , 220}, perform the steps described below. This set was chosen because the values are evenly spaced across the image and can be used to obtain a good approximation of the orientation of the dispersion axis.

1. Identify the row in InImage that is associated with the current value of y.
2. Initialize an array **x1** to hold the x-coordinates of all the pixels in the row and the array **y1** to hold the value associated with each pixel in this row.
3. Calculate the wavelength calibration parameters for this row using the procedure outlined below. The wavelength calibration fit equation is

$$\lambda(x) = Ax + B \qquad (2.3)$$

   where λ is the wavelength in microns of the pixel with x-coordinate x and where A and B are the desired calibration parameters.

   a. Locate the "peaks" in the array **y1**, where a peak is located at position i in the array if **y1**[i] is greater than both **y1**[i-1] and **y1**[i+1].
   b. Create an array $\mathbf{y_{data}}$ that contains approximate Gaussian peaks at each position i where a peak was found in the **y1** array. An approximate

Gaussian peak is created at position i in $\mathbf{y_{data}}$ by adding 1 to $\mathbf{y_{data}}$[i] and adding 0.382546 to $\mathbf{y_{data}}$[i-1] and to $\mathbf{y_{data}}$[i+1].

c. Initialize an array $\mathbf{x_{sky}}$ to hold the wavelengths (in microns) of the Earth's atmospheric emission lines.

d. Test a range of values for the calibration parameters from equation (2.3) as follows:

   For A = 0.01965; A = 0.020145; A = A + 0.000245

   For B = 7.45; B = 7.849; B = B + 0.199

   i. Substitute the current values for A and B into equation (2.3) to calculate the x-coordinates associated with each wavelength found in the array $\mathbf{x_{sky}}$.

   ii. Generate an array $\mathbf{y_{sky}}$ that contains Gaussian peaks at the x-positions calculated in Step 1. A Gaussian peak is produced at position p in the $\mathbf{y_{sky}}$ array by finding the equation of the Gaussian with a mean of p and a standard deviation of 0.721347; this value for the standard deviation captures the effect that the spectrometer's slit has on the width of the lines in Earth's atmospheric emission spectrum. The function values for this Gaussian are calculated for positions p-1, p, p+1, and p+2; these function values are added to the corresponding elements in the $\mathbf{y_{sky}}$ array.

   iii. Calculate the correlation coefficient for the arrays $\mathbf{y_{data}}$ and $\mathbf{y_{sky}}$ (see Figure 2-5).

e. The values of A and B that give the highest correlation coefficient are the wavelength calibration parameters that are reported for the current row in InImage.

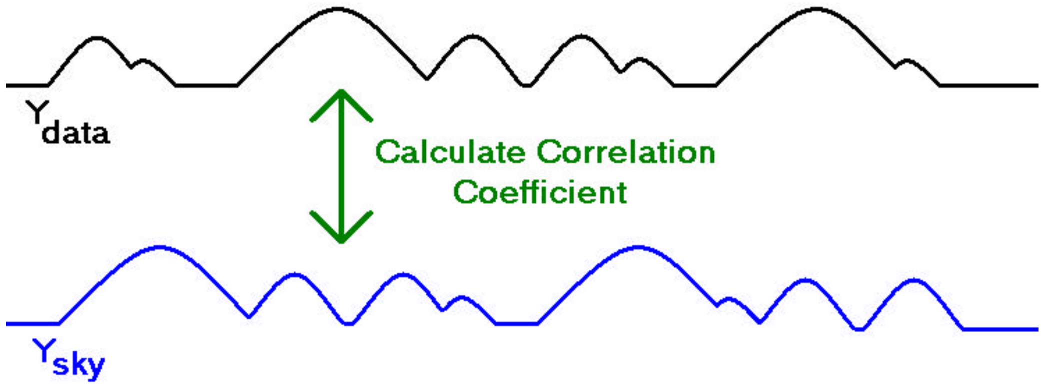


Figure 2-5. Gaussian peaks from the $\mathbf{y_{data}}$ and $\mathbf{y_{sky}}$ arrays that are compared using a correlation coefficient. The $\mathbf{y_{data}}$ array contains peaks from one row in the inputted image while the $\mathbf{y_{sky}}$ array contains peaks corresponding to Earth's atmospheric emission lines.

Once the wavelength calibration parameters have been calculated for the 20 values of y from 30 to 220, the y-dependence of these parameters must be determined. For this purpose, we fit the parameters using the equations $A = a_0 y + a_1$ and $B = a_2 y + a_3$ using least-squares linear regression. The resulting fit coefficients a_0, a_1, a_2, and a_3 are the parameters that describe the dispersion axis. These fit parameters are different from the dispersion axis fit parameters discussed at the beginning of Section 2.3 because the dependence of λ on x and y cannot, in general, be captured using only two parameters. However, DetermineDispersion2 and DetermineDispersion3 can use only two parameters because these algorithms are independent of wavelength calibration.

2.3.2.2 The Newly Developed Algorithm: Initial Version (DetermineDispersion2)

The algorithm from the COMICS Data Reduction Manual has one significant problem: The dispersion axis determination procedure is strongly tied to wavelength calibration. This property of the algorithm is undesirable because wavelength calibration is generally an independent step in the spectral reduction process and therefore should not be tied to any other reduction steps. For this reason, the following new algorithm was developed for dispersion axis determination. This algorithm assumes that all of the spectral lines in InImage have the same slope; a test that was performed to determine the validity of this assumption is described in Chapter 3.

First, remove noise from InImage using a median filter. Convolve the resulting filtered image with the Sobel operator shown below, producing a first-derivative image; in this first-derivative image, the vertical edges are enhanced.

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Convolve the resulting first-derivative image with the above Sobel operator to produce a second-derivative image. In this second-derivative image, the edges are the boundaries between regions with positive and negative pixel values. Set all of the positive values in the second-derivative image to 1 (white) and all of the negative values to 0 (black). In the resulting image (see Figure 2-6), the boundaries between black and white regions correspond to the vertical edges in the original spectral image.

This procedure for producing a black and white second-derivative image is based on ideas presented by Gonzalez and Woods [11].

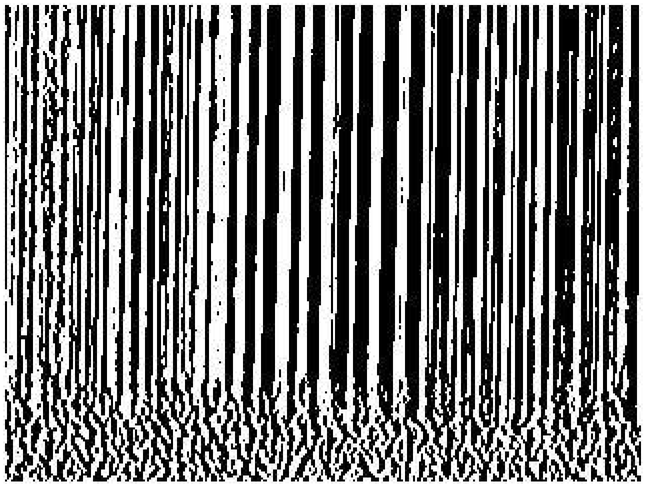


Figure 2-6. A second-derivative image in which edges are indicated by the boundaries between black and white regions. This figure is the second derivative of the spectral image on the right side of Figure 1-2.

Next, ask the user (an astronomer or a student worker) to identify a rectangular region in this black and white image that contains a well-defined edge; here, a rectangle contains a well-defined edge if, for each row of pixels contained in the rectangle, the rectangle contains a boundary between black and white in that row. Figure 2-7 shows an example of a rectangle that contains a well-defined edge. This rectangular region must be at least 20 pixels high.

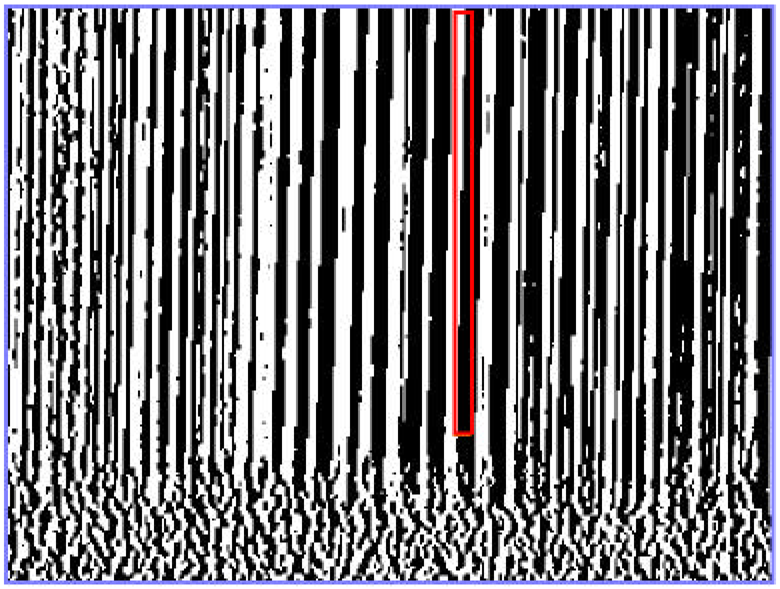


Figure 2-7. A second-derivative image with a well-defined edge enclosed in the red rectangle.

After the user has selected a rectangular region, locate 20 equally-spaced points along the edge contained within the region; if the region contains more than one edge, the leftmost such edge is used. If 20 edge points could not be found, then the selected rectangle did not contain a well-defined edge, and the user is asked to select a different region. Once 20 points along an edge have been located, fit those points with a linear equation of the form $x = a_0 y + a_1$. The parameters of this linear equation are the parameters that describe the dispersion axis of the inputted image. Figure 2-8 shows the dispersion axis fit plotted over the spectral image of a planet.

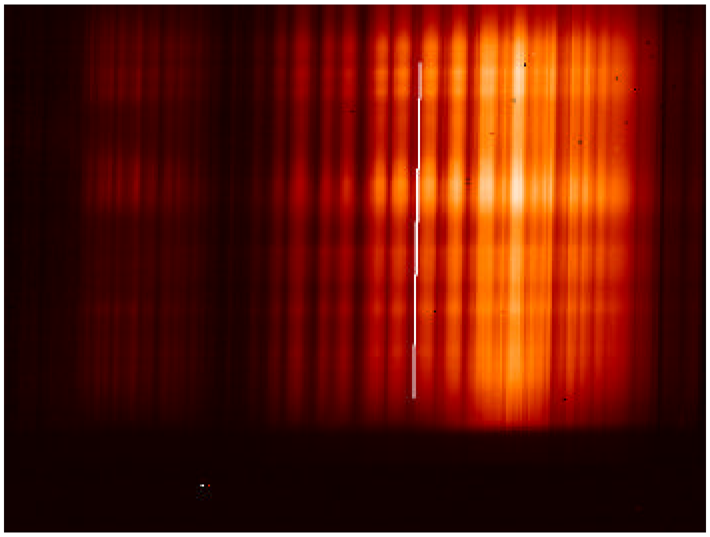


Figure 2-8. The dispersion axis fit for the spectral image of a planet.

2.3.2.3 The Newly Developed Algorithm: Refined Version (DetermineDispersion3)

The dispersion axis determination algorithm just described suffers from two major difficulties: (1) the algorithm requires the user to specify a region of interest in InImage, and (2) the algorithm assumes that all of the spectral lines in InImage have the same slope when, in actuality, the slopes of the spectral lines can vary across the image. As part of the contribution of this thesis, DetermineDispersion2 has been refined to deal with these difficulties. To deal with difficulty (1), a procedure was developed to automatically locate well-defined spectral lines in InImage; however, the user is still given the option of specifying a region of interest manually in case the automatic procedure produces unreasonable results. To deal with difficulty (2), multiple spectral lines are identified in InImage; the slopes of these lines are averaged to produce an equation for the dispersion axis that is characteristic of the image as a whole. This improved algorithm is described below. However, since we are still using a single number to describe the slope of the dispersion axis, this algorithm assumes that all of the spectral lines in InImage have comparable slopes—that is, that the orthogonalization transformation will not be sensitive to the differences in spectral line slope that occur within the image.

The Algorithm

First, produce a second-derivative image as was done in the initial version of this algorithm. Set all of the positive values in the second derivative image to 1 (white) and all of the negative values to 0 (black). In the resulting binary image, the boundaries between black regions and white regions correspond to the spectral lines in the original image (see Figure 2-6). Isolate the edges in this black and white image as follows to produce a binary edge image:

- Create a new array of integers that is the same size as the black and white image; this new array, called edge_image, will represent the edge image.
- For each pair of consecutive columns I and I+1 in the black and white image, perform the following steps:
    - For each pixel in column I, do the following:
        - Let P be the position of the current pixel in column I.
        - If the current pixel has the same value as the pixel at position P in column I+1, assign the value 0 to the pixel at position P in column I of the edge_image array. Otherwise, assign the value of 1 to the pixel at position P in column I of edge_image.

In the resulting edge image, the edges are represented as white lines that are one pixel thick (see Figure 2-9).

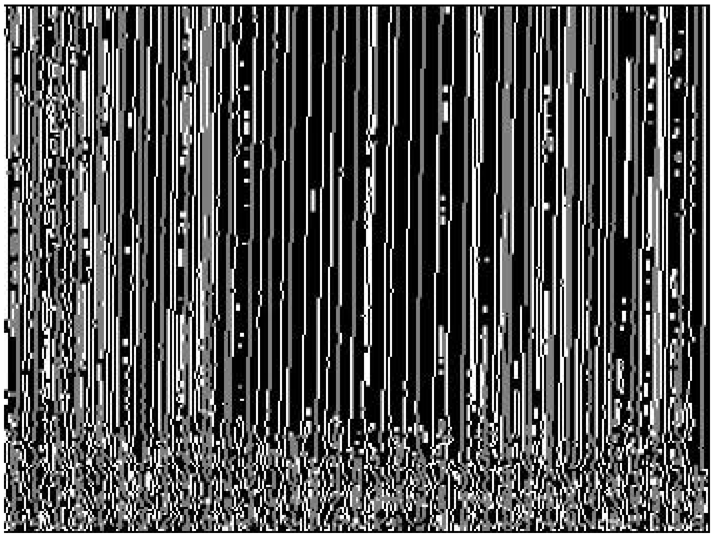


Figure 2-9. A binary edge image where each white line represents an edge from the second derivative image.

Next, identify well-defined edges in this image; here, an edge is well defined if it does not contain any discontinuities or branching. The user (an astronomer or student worker) is asked whether this identification process should occur manually or automatically. If the user chooses manual mode, the user must identify a rectangular region in the second derivative image that contains one or more well-defined boundaries between black and white. A boundary between black and white is well-defined if the black region associated with the boundary is not disjoint – in other words, if the black region (1) does not contain any white pixels, (2) does not contain any "broken cycles" of black pixels, and (3) is at least one pixel wide in every row contained within the rectangular region. For example, in Figure 2-10, the boundary enclosed in the dark blue rectangle is not well-defined because its black region contains white pixels, the boundary enclosed in the green rectangle is not well-defined because its black region contains a broken cycle of black pixels, and the boundary enclosed in the pink rectangle is not well-defined because its black region is zero pixels wide in many of the rows contained within the rectangle.

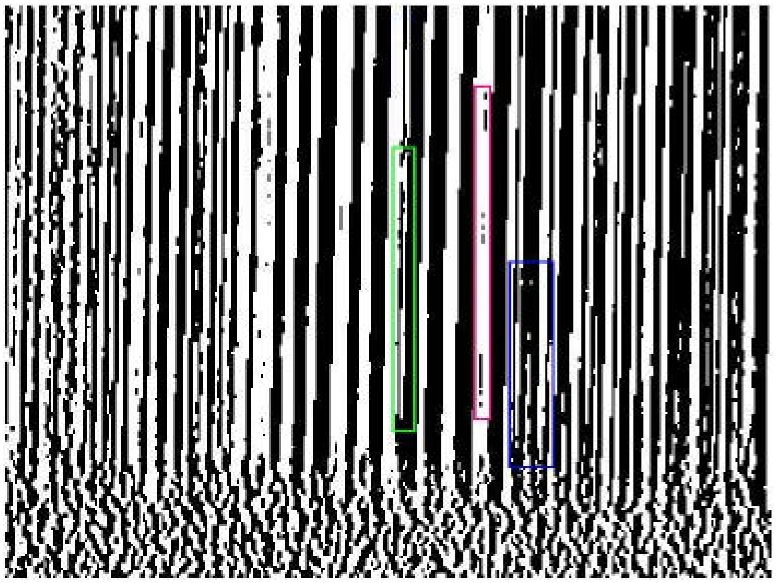


Figure 2-10. A second derivative image containing three ill-defined boundaries between black and white. The boundary enclosed in the dark blue rectangle is ill-defined because its black region contains white pixels, the boundary enclosed in the green rectangle is ill-defined because its black region contains a "broken cycle" of black pixels, and the boundary enclosed in the pink rectangle is ill-defined because its black region is zero pixels wide in many of the rows contained within the rectangle.

Once the user has identified a region of interest in the second derivative image, the corresponding region in the binary edge image is isolated; the well-defined edges contained in this image region are the well-defined edges that will be used in the remainder of this algorithm (see Figure 2-11).

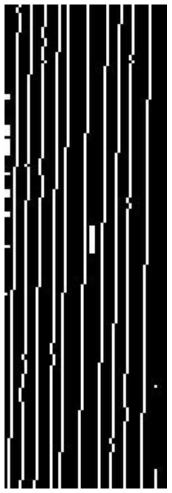


Figure 2-11. A region of interest containing several well-defined edges that were identified by the user in manual mode.

If the user chooses automatic mode, well-defined edges in the edge image are identified as follows. The procedure described here is based on ideas presented by Fisher, Perkins, Walker, and Wolfart[9]. First, the Hough transform of the edge image is calculated; for the Hough transform, only theta values from -6.3° to -0.6° were considered and the step size separating consecutive theta values was given by the following formula:

$$\text{step\_size} = \frac{\pi}{\text{ceiling}\left(\pi\sqrt{x^2 + y^2}\right)}$$

where $x = \dfrac{xsize - 1}{2.0}$, $y = \dfrac{ysize - 1}{2.0}$, xsize is the width of the edge image, and ysize is the height of the edge image. The resulting transformed image is then histogram-equalized, converted to a binary image using a threshold of 0.9995 times the image's maximum pixel value for medium-resolution spectra or 0.995 times the image's maximum pixel value for low-resolution spectra, and then is thinned using the following hit and miss structural elements along with their 90 degree rotations:

Hit structural elements:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}$$

Miss structural elements:

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

Lastly, the Hough backprojection is calculated for the resulting thinned binary image. This backprojection is converted to a binary image using a threshold of zero; the resulting image contains the well-defined edges that will be used in the remainder of this algorithm (see Figure 2-12). For the interested reader, descriptions of the Hough transform, histogram equalization, and thinning are given in the appendicies.

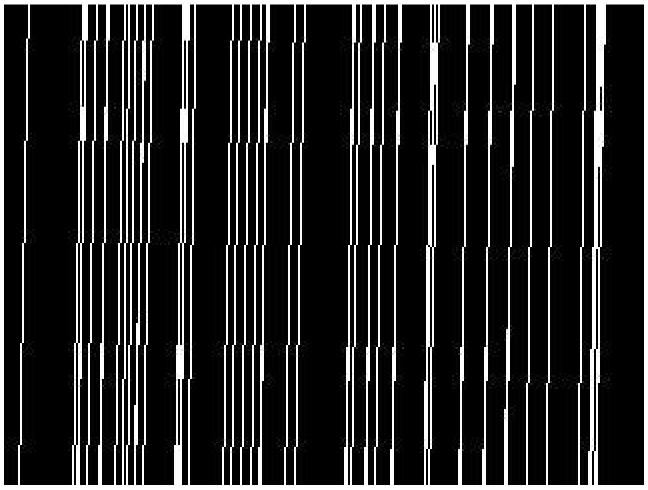


Figure 2-12. An image containing the well-defined edges that were identified by the algorithm in automatic mode. In this figure, each set of connected white pixels forms a single edge.

Once well-defined edges have been identified, DetermineDispersion3 identifies which pixels are associated with each edge. To simplify this identification process, the following assumptions are made:

1. All edges are vertical, or nearly vertical.
2. None of the edges intersect.
3. For each pixel that is part of an edge, all of the pixel's neighbors are part of the same edge or are background pixels.
4. If an edge is more than one pixel thick at any row within the image, only the leftmost pixel in that row needs to be accounted for.

The first and second assumptions are reasonable because each edge represents a spectral line: all spectral lines are nearly vertical, and two distinct spectral lines do not intersect. The third assumption indicates that all 1-pixels that are adjacent to one another are part of the same edge; this assumption is valid because, if two adjacent pixels were part of two different edges, then those edges would be close enough that it would be difficult to distinguish between the two. The fourth assumption is reasonable because the orientation of an edge can be described using a least squares linear fit of the coordinates of the leftmost pixels associated with that edge, and information about the orientation of each edge is the only information necessary for determining the orientation of the dispersion axis.

To identify which pixels are associated with each edge, a vertical sweep line that is one pixel thick moves from the left side of the image to the right side; each 1-pixel encountered by the sweep line is given a negative number that indicates which line the pixel is associated with. If there is more than one 1-pixel on the sweep line at any given time, these 1-pixels are considered in order from bottom to top. The value assigned to each 1-pixel depends on the values of that pixel's neighbors, so the following notation will be used to refer to the neighbor pixels' values. In the diagram below, Current-Pixel is the 1-pixel whose value is currently being determined.

Left-Above		
Left	**Current-Pixel**	
Left-Below	Below	

If Current-Pixel is in the leftmost column of the edge image, then Left-Above, Left, and Left-Below are all assigned values of zero; if Current-Pixel is in the last row of the edge image, then Left-Below and Below are both assigned values of zero. Five different cases, considered in the order given, must be handled according to the values assigned to a 1-pixel's neighbors:

Case 1: Left-Above, Left, and Left-Below are all 0.

If Below is zero, assign to Current-Pixel a negative value that has not yet been given to any other pixels. Otherwise, assign to Current-Pixel the same value as Below.

Case 2: Left is non-zero and negative

If Below is zero, assign to Current-Pixel the same value as Left. Otherwise, assign to Current-Pixel the same value as below and assign Left, along with any other pixels with the same value as Left, to the value of Below. For example, if Left = -4 and Below = -3, assign to Current-Pixel the value -3. Then locate all pixels having the value -4 (including Left) and assign to those pixels the value -3.

Case 3: Left-Above is zero and Left-Below is negative (or vice-versa)

If Below is zero, assign to Current-Pixel the same value as Left-Below (or Left-Above, in the alternative case). Otherwise, assign to Current-Pixel the same value as Below and assign Left-Below (or Left-Above), along with any other pixels with the same value as Left-Below (or Left-Above), to the value of Below. For example, if Left-Above = 0, Left-Below = -5, and Below = -3, assign to Current-Pixel the value -3. Then locate all pixels having the value -5 (including Left-Below) and assign to those pixels the value -3.

Case 4: Left-Above is equal to Left-Below

If Below is zero, give Current-Pixel the same value as Left-Above. Otherwise, assign to Current-Pixel the same value as Below and assign Left-Above, along with any other pixels with the same value as Left-Above, to the value of Below.

Case 5: Left-Above is not equal to Left-Below

Let ‘higher’ be the pixel of maximum value between Left-Above and Left-Below, and let ‘lower’ be the pixel of minimum value. Assign to Current-Pixel the same value as ‘higher’ and assign ‘lower’, along with any other pixels with the same value as ‘lower’, to the value of ‘higher’.

After each 1-pixel in the edge image has been assigned a negative number, the algorithm thins the edges so that each edge is one pixel thick. This thinning is done by considering each row in the image; in each row, the first pixel having a particular value is retained while the remaining pixels having that value are set to zero. Figure 2-13 shows a version of the image from Figure 2-12 where each edge is assigned to different negative number; the different negative numbers are represented by the different colored lines in the figure.

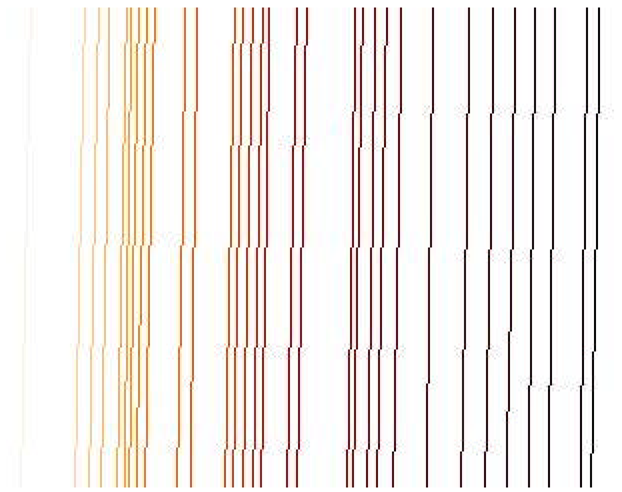


Figure 2-13. An image containing the well-defined edges from Figure 2-12 where each of the edges has been assigned a different color that corresponds to a different negative number.

Once each edge has been associated with a different number, a vertical scan line is used to extract and store the coordinates of the pixels associated with each edge. Edges that do not extend for the entire height of the image or that contain discontinuities are ignored. Each extracted line is then fit with a linear equation of the form $x = a_0 y + a_1$. Lines for which the value of a_0 is more than 2 standard deviations away from the mean value for a_0 (over all the lines) are discarded. For the remaining lines, the mean of the a_0 values and the median of the a_1 values are calculated and reported as the fit parameters of the spectral axis. This completes the description of the newly developed DetermineDispersion3 algorithm.

2.3.3 Image Transformation

As with the dispersion axis determination algorithm, an initial algorithm was developed to perform the image transformation and then was refined twice after testing revealed problems. The initial algorithm will be referred to as Transformation1, while the two refined versions of the algorithm will be called Transformation2 and Transformation3. These three algorithms are described below.

2.3.3.1 Initial Algorithm (Transformation1)

This algorithm uses the following input parameters:

1. **InImage:** the image to be transformed.
2. **a_0:** the coefficient to the linear term in the equation $x = a_0 y + a_1$ that describes the dispersion axis.

3. **b$_0$:** the coefficient of the quadratic term in the equation $y = b_0 x^2 + b_1 x + b_2$ that describes the spatial axis.
4. **b$_1$:** the coefficient of the linear term in the equation describing the spatial axis.

First select a set of tie points that will be used to perform the transformation; a tie point is a point for which we know its x- and y- coordinates in InImage and for which we can calculate the x- and y- coordinates where the point will be in the transformed image. As was done in the q-series task *q_transtable2*, all of the pixels in InImage are used as tie points. The coordinates where the tie points will be in the transformed image are calculated from the coordinates of the tie points in the original image using the following equations; this computation is done using floating-point arithmetic.

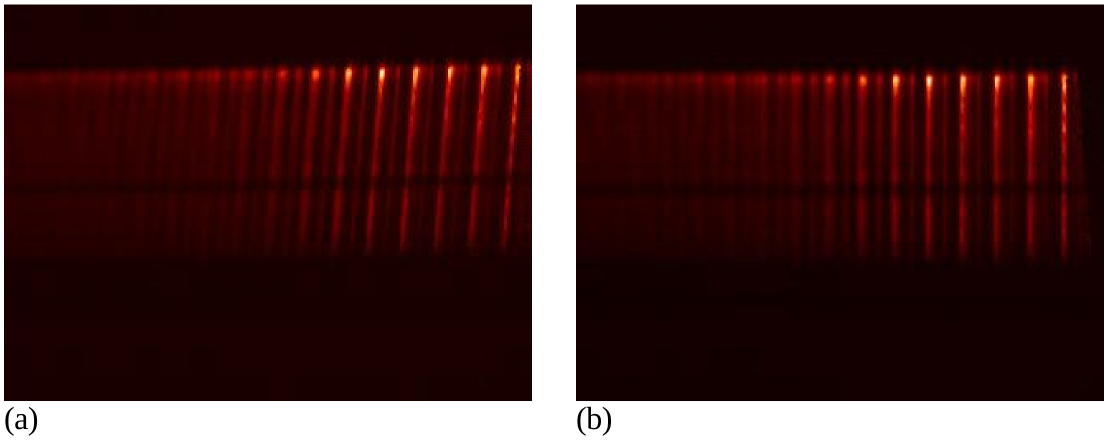
$$\begin{aligned} x_{\text{transformed}} &= x_{\text{original}} - a_0 y_{\text{original}} \\ y_{\text{transformed}} &= y_{\text{original}} - \left(b_0 x_{\text{original}}^2 + b_1 x_{\text{original}}\right) \end{aligned} \qquad (2.4)$$

The obtained coordinates of the tie points in the original image and in the transformed image are inputted into the Interactive Data Language (IDL) built-in function *warp_tri*, which performs the transformation and returns the resulting orthogonalized image. The *warp_tri* function is so named because triangulation is one of the steps in the procedure used by this function to warp, or transform, images. According to the IDL online help manual, *warp_tri* works as follows:[14]

> First, the *warp_tri* function triangulates the irregular grid defined by the coordinates of the tie points in the transformed image. Then, the function calculates the coordinates of the points in the original image that are associated with each pixel in the transformed image. These calculated coordinates might

have non-integer values, while all of the pixels in the original image have integer-valued coordinates. Therefore, the original image is linearly interpolated to compute the values that should be given to each of the points identified in the original image. The values given to these points are also assigned to the corresponding pixels in the transformed image, thereby producing the final image that is outputted by *warp_tri*.

Figure 2-14 shows a spectral image as it appears before and after orthogonalization.



(a) (b)

Figure 2-14. A spectral image of Saturn both before orthogonalization (a) and after orthogonalization using Transformation1 (b).

2.3.3.2 Refined Algorithm 1 (Transformation2)

Through testing, it has been shown that Transformation1 does not properly handle the case where the spatial axis is described by a quadratic equation (that is, where b_0 is nonzero); the right side of Figure 2-15 shows a standard star spectrum that was transformed using a quadratic equation for the spatial axis. This result prompted a closer examination of the transformation procedures described in the COMICS Data Reduction Manual[10] and it was found that the IDL procedure *warp_tri* does not perform the same transformation as is implemented in the software used by the COMICS team; it was also noticed that the programs used by the COMICS team conserve InImage's total flux (that is, the total brightness of the image[21]) before and after the transformation, while the initial version of the algorithm presented here does not conserve the total image flux. The improved image transformation algorithm is described below.

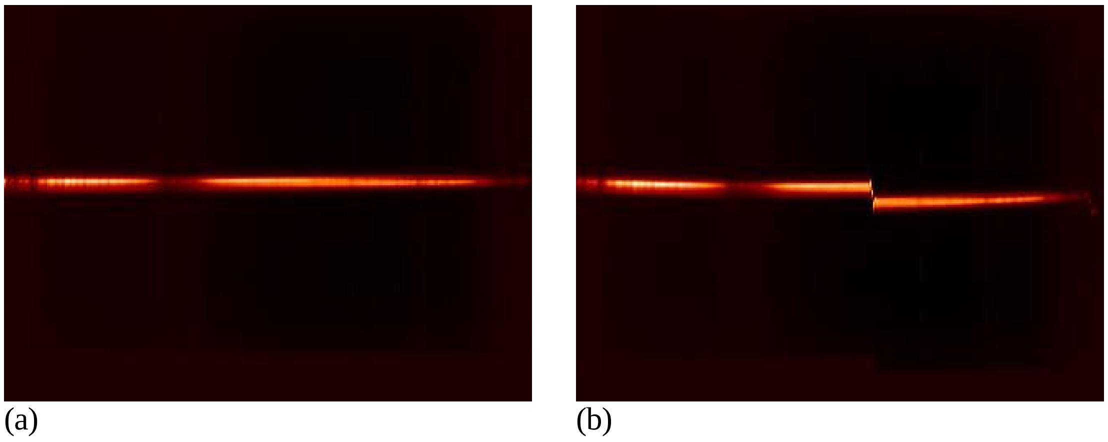


Figure 2-15. The spectral image of a standard star both before (a) and after (b) being orthogonalized using a quadratic fit for the spatial axis with Transformation1.

The Algorithm

Transformation2 calculates the tie points in the original and transformed images with the same procedure as was used in the initial algorithm. However, instead of using the *warp_tri* function to perform the transformation, this algorithm uses the *polywarp* and *poly_2d* IDL built-in functions. The *polywarp* procedure uses least squares estimation to calculate polynomial transformations that map the tie point coordinates from the coordinate system of the transformed image to the coordinate system of the original image. This coordinate system transformation is described by the following equations:

$$x_{\text{original}} = \sum_{I=0}^{2}\sum_{J=0}^{2} Kx[I,J]\, x_{\text{transformed}}^{J}\, y_{\text{transformed}}^{I}$$
$$y_{\text{original}} = \sum_{I=0}^{2}\sum_{J=0}^{2} Ky[I,J]\, x_{\text{transformed}}^{J}\, y_{\text{transformed}}^{I} \qquad (2.5)$$

where *Kx* and *Ky* are 2-dimensional arrays of coefficients. Using these polynomial transformation functions, the *poly_2d* function performs the transformation and produces the orthogonalized image; poly_2d uses bilinear interpolation (see Appendix D) to produce the output image and uses the output value 0.0 for pixels whose (x_{original}, y_{original}) coordinates refer to a point outside of the bounds of InImage.

Once the transformation is complete, the orthogonalized image is multiplied by the Jacobian of the coordinate transformation to ensure that the total image flux is conserved. The Jacobian is calculated using the following equations:

$$\text{Jacobian} = \frac{\partial(x_{\text{original}}, y_{\text{original}})}{\partial(x_{\text{transformed}}, y_{\text{transformed}})} = \frac{\partial x_{\text{original}}}{\partial x_{\text{transformed}}}\frac{\partial y_{\text{original}}}{\partial y_{\text{transformed}}} - \frac{\partial x_{\text{original}}}{\partial y_{\text{transformed}}}\frac{\partial y_{\text{original}}}{\partial x_{\text{transformed}}} \qquad (2.6)$$

where,

$$\frac{\partial x_{\text{original}}}{\partial x_{\text{transformed}}} = Kx[0,1] + Kx[1,1]\, y_{\text{transformed}} + 2Kx[0,2]\, x_{\text{transformed}} + 2Kx[1,2]\, x_{\text{transformed}} y_{\text{transformed}} + 2Kx[2,2]\, x_{\text{transformed}} y_{\text{transformed}}^{2} + Kx[2,1]\, y_{\text{transformed}}^{2}$$

$$\frac{\partial y_{\text{original}}}{\partial y_{\text{transformed}}} = Ky[1,0] + Ky[1,1]\, x_{\text{transformed}} + Ky[1,2]\, x^2_{\text{transformed}} + 2Ky[2,2]\, x^2_{\text{transformed}} y_{\text{transformed}} + 2Ky[2,0]\, y_{\text{transformed}} + 2Ky[2,1]\, x_{\text{transformed}} y_{\text{transformed}}$$

$$\frac{\partial x_{\text{original}}}{\partial y_{\text{transformed}}} = Kx[1,0] + Kx[1,1]\, x_{\text{transformed}} + Kx[1,2]\, x^2_{\text{transformed}} + 2Kx[2,2]\, x^2_{\text{transformed}} y_{\text{transformed}} + 2Kx[2,0]\, y_{\text{transformed}} + 2Kx[2,1]\, x_{\text{transformed}} y_{\text{transformed}}$$

$$\frac{\partial y_{\text{original}}}{\partial x_{\text{transformed}}} = Ky[0,1] + Ky[1,1]\, y_{\text{transformed}} + 2Ky[0,2]\, x_{\text{transformed}} + 2Ky[1,2]\, x_{\text{transformed}} y_{\text{transformed}} + 2Ky[2,2]\, x_{\text{transformed}} y^2_{\text{transformed}} + Ky[2,1]\, y^2_{\text{transformed}}$$

Figure 2-16 shows the image from the left side Figure 2-15 after it has been orthogonalized using this refined algorithm.

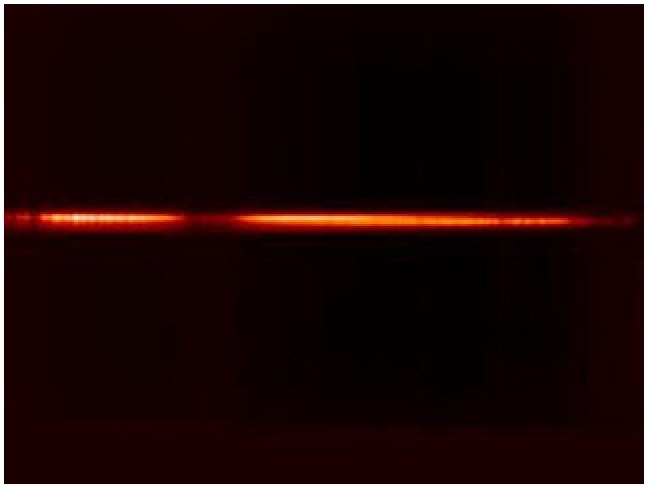


Figure 2-16. A spectral image of the standard star from Figure 14 after being orthogonalized using a quadratic fit for the spatial axis with the refined transformation algorithm.

2.3.3.3 Refined Algorithm 2 (Transformation3)

When Transformation2 was tested, an additional problem was revealed: The transformation equation for the x-axis described in the COMICS Data Reduction Manual evenly distributes the pixels in the spectral image along the x-axis, but the transformation equation used here for the x-axis does not perform a similar function. This difference between the algorithm described here and the algorithm used in the COMICS Data Reduction Manual may lead to additional pixels being mapped outside of the image by the refined algorithm. Tests performed on low-resolution spectra show that 304 pixels are mapped outside of the image by the transformation used by the COMICS team, while 646 pixels are mapped outside of the image by the transformation used here. To fix this problem, the equation used to calculate the x-coordinates of the tie points in the transformed image was changed to the following:

$$x_{\text{transformed}} = x_{\text{original}} - a_0 y_{\text{original}} + \frac{a_0 y_{\text{size}}}{2} \qquad (2.7)$$

where y_{size} is the number of rows of pixels in InImage. With this change, the number of pixels mapped outside of the image by this transformation was reduced to 389, a value which is comparable to the number of pixels lost with the transformation used by the COMICS team.

2.4 Extraction Algorithm

Extraction is the process of obtaining one or more spectra from a spectral image; if multiple spectra are generated, each spectrum is extracted from a different spatial position in the spectral image. This algorithm is based on the spectrum extraction procedure described in the COMICS Data Reduction Manual[10]. Given InImage, the spectral image from which spectra will be extracted, the algorithm operates as follows.

First, the user identifies a rectangular region that contains the spectrum to be extracted from InImage. This rectangular region will be referred to as the *spectrum region*; an example of such a region is delimited by the white rectangle in Figure 2-17. The positions of the boundaries of this rectangular region are initialized using a two-standard-deviation threshold above the mean pixel value, and the user is allowed to adjust these boundaries as necessary. The user is also asked to identify two more rectangular regions—a sky emission region that appears above the spectrum region and a sky emission region that appears below the spectrum region; examples of such sky regions are shown in Figure 2-17. If there is no sky region above (or below) the spectrum region, the user may indicate this condition by specifying a line instead of a rectangular region above (or below) the spectrum region as was done for the sky region above the spectrum in Figure 2-17. The left and right boundaries of these sky regions must be the same as the left and right boundaries of the spectrum region; the upper and lower boundaries of the sky regions may be varied, and these boundaries are initialized to appear at a fixed distance above or below the spectrum region. These sky regions are not required to abut the image borders.

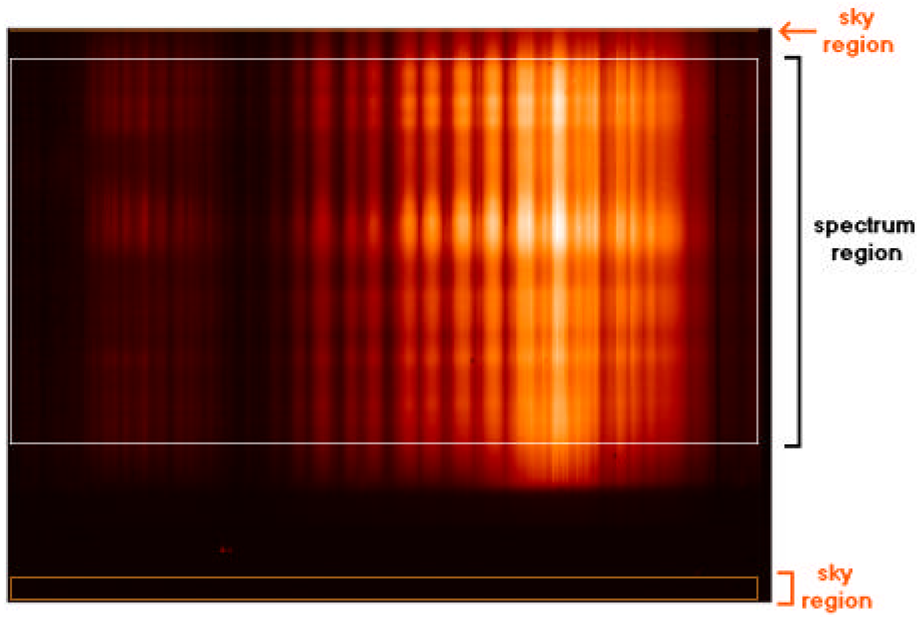


Figure 2-17. A spectral image in which a spectrum region, containing the spectrum that will be extracted from the image, has been identified by the white rectangle. Sky emission regions have also been identified above and below the spectrum region. The sky region above the spectrum region is specified by a line instead of a rectangle, indicating that there is no sky emission present above the spectrum.

Once these sky and spectrum regions have been identified, the user is asked to choose the aperture width to be used for extraction; the possible widths (in pixels) are 1, 3, 5, 7, 9, and the total number of rows in the spectrum region. Spectra are then extracted from the image using the selected aperture width. The following procedure is used to perform the extraction process:

Let r_{min} be the bottommost row of pixels in the spectrum region and let r_{max} be the topmost row. Further, let w be the aperture width selected by the user. For each row r of pixels beginning at row $\mathrm{r}_{\min} + \frac{\mathrm{w}-1}{2}$ and ending at row $\mathrm{r}_{\max} - \frac{\mathrm{w}-1}{2}$, calculate the pixel-wise average of rows $r - \frac{w-1}{2}$ through $r + \frac{w-1}{2}$. This process is illustrated in Figure 2-18 for an aperture width of 3 with a spectrum region that contains 5 rows.

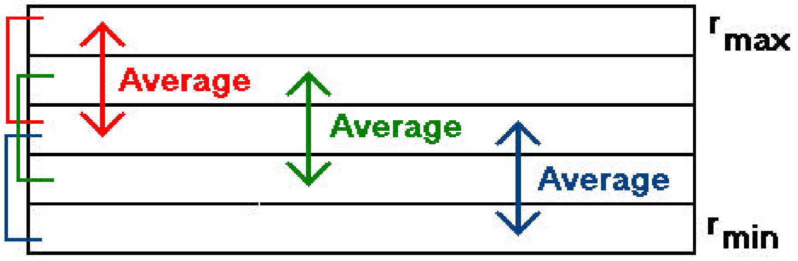


Figure 2-18. An illustration of the spectrum extraction process for a spectrum region that contains 5 rows of pixels (from r_{min} to r_{max}) and for an extraction aperture width of 3.

The averaging process generates $(r_{max} - r_{min} + 1) - w + 1$ extracted spectra where each spectrum is the average of *w* consecutive rows from the spectrum region.

Next, the sky regions that were identified above and below the spectrum region are isolated from InImage and then merged into a single rectangle of pixels; any sky regions that were identified using a line instead of a rectangle are not incorporated into this rectangle of sky pixels, so if both sky regions were identified using lines, the rectangle of sky pixels is left undefined. If the rectangle of sky pixels is defined, the standard deviation is calculated for each column of pixels in the rectangle. This standard deviation calculation gives a noise level estimate for each point in the extracted spectra. However, if the rectangle of sky pixels is not defined, this noise estimate is not calculated.

Once this process is complete, each extracted spectrum is packed in a two-dimensional array of double-precision floating-point numbers. This array has three rows, and the contents of each row are as follows:

row 1: The x-coordinates associated with each column in the spectrum.

row 2: The spectrum itself.

row 3: The noise level estimate for each column in the spectrum. If a noise level estimate was not calculated, this row contains all zeros.

The resulting spectra are then saved in separate files. Figure 2-19 shows an extracted spectrum.

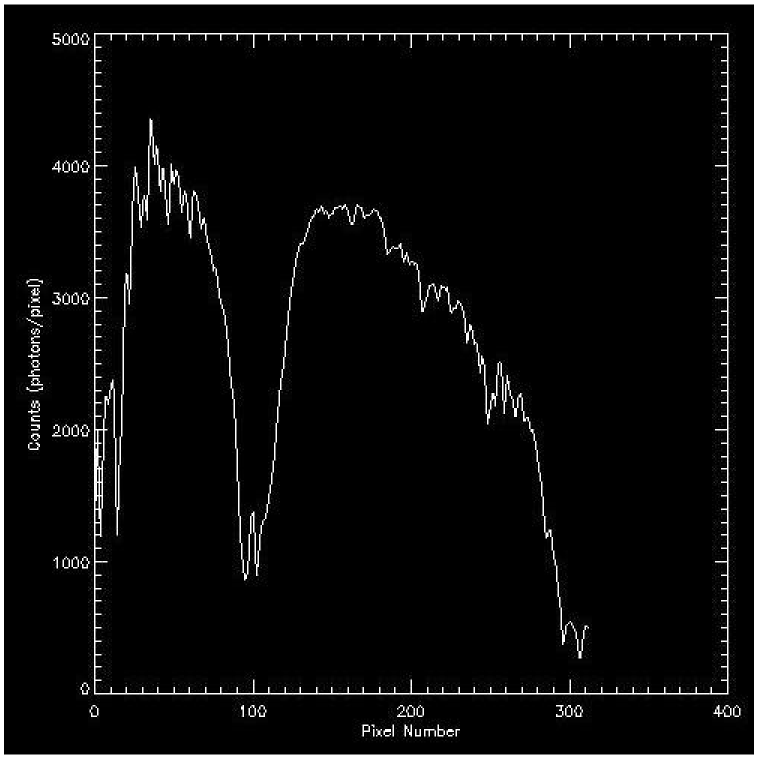


Figure 2-19. An extracted spectrum of a star.

2.5 Future Approach for Spectrum Calibration

Algorithms for spectrum calibration have not been developed for this thesis. However, the COMICS team and Dr. Glenn Orton have general techniques that may be used in the future to calibrate spectra. The calibration process requires two types of calibration: wavelength calibration, and absolute calibration (see Section 1.3). To perform wavelength calibration, Dr. Orton's technique involves comparing the spectrum of a star with Earth's atmospheric absorption spectrum. Some of Earth's absorption lines can be seen in the stellar spectrum, allowing us to determine the pixel-to-wavelength correspondence for the stellar spectrum. This pixel-to-wavelength correspondence allows us to convert the wavelength axes of all images in a single data set from units of pixel number to units of wavelength, such as μm.

To perform absolute calibration, Dr. Orton and the COMICS team use the spectrum of a star to correct for the effects of atmospheric absorption on the spectrum of a planet. Let Observed(Planet) be the spectrum of a planet X from one of our data sets Y, let Standard(Planet) be the corrected spectrum of planet X, let Observed(Star) be the stellar spectrum of a star Z from data set Y, and let Standard(Star) be the atmospheric-absorption-corrected spectrum of star Z. Here, Standard(Planet) is the end result of absolute calibration, and Standard(Star) is a standard spectrum that has been calculated by Cohen et al.[3] Standard(Planet) may be calculated using the following formula:

$$\text{Standard}(\text{Planet}) = \frac{\text{Observed}(\text{Planet})}{\text{Observed}(\text{Star})}\text{Standard}(\text{Star})$$

This equation may be used to perform absolute calibration on any planetary spectrum; after this calibration step is complete, the intensity axis of the calibrated spectrum has been converted from units of photon count to units of radiance, such as W/cm^2/sr/µm. Dr. Orton does not perform absolute calibration on stellar spectra because he is only interested in studying planets.

2.6 Literature Review

This thesis presents an algorithm for orthogonalizing spectral images obtained using COMICS; the orthogonalization process involves transforming images so that their spectral axes are vertical and their dispersion axes are horizontal. The need to transform images also appears in the reduction processes required for other instruments and in the preprocessing procedures used in stereovision applications. Based on how they differ from the orthogonalization algorithm presented here, the transformation algorithms used for data reduction can be divided into three categories: algorithms that fix the orientation of one axis during the transformation (Section 2.6.1), algorithms that use different interpolation techniques during the transformation (Section 2.6.2), and algorithms that use similar polynomial transformation techniques to transform different types of images (Section 2.6.3). Section 2.6.4 discusses the differences between the transformations used in stereovision applications and the orthogonalization transformation presented in this thesis.

2.6.1 Rectification With One Fixed Axis

Cushing et al.[4] present algorithms to reduce spectral images from the SpeX spectrograph at IRTF, including an algorithm to rectify spectral images that are curved. This rectification algorithm differs from the orthogonalization algorithm presented here because Cushing et al.'s algorithm assumes that the spatial axes of all spectral images are vertical, so the algorithm only needs to straighten the dispersion axis. The orthogonalization algorithm presented here, on the other hand, must straighten both the spatial and dispersion axes.

2.6.2 Interpolation Techniques

Barrett et al.[1] and Dressel et al.[5] present a new algorithm to rectify spectral images obtained from the Space Telescope Imaging Spectrograph; this rectification algorithm is used to prepare spectral images for extraction. The new algorithm uses wavelet interpolation to produce a final image that is more accurate than could be produced using bilinear interpolation: Bilinear interpolation assumes that the light collected by a detector pixel is concentrated at the center of that pixel. This assumption gives accurate results when the aperture width that will be used for extraction is large. In reality, the light collected by a detector pixel is distributed over the entire pixel area; wavelet interpolation takes this fact into account to produce a more accurate rectified image in the case where the extraction aperture width is small. The COMICS team uses bilinear interpolation while orthogonalizing spectral images from their instrument[10], so the transformation algorithm discussed in this thesis uses bilinear interpolation instead of wavelet interpolation.

2.6.3 Polynomial Transformations

Wang[21] presents algorithms to reduce camera images from WIRCAM on the Canada-France-Hawaii Telescope and from MOIRCS on the Subaru Telescope; among the algorithms discussed is a procedure to correct distorted images using a polynomial transformation. A polynomial transformation is also used in the orthogonalization algorithm presented here, but Wang's algorithm is specialized for camera images while the algorithm introduced in this thesis is specialized for spectral images.

2.6.4 Rectification of Stereo Image Pairs

Image transformations are used to rectify stereo image pairs in stereovision applications[2][17][20]. This rectification process aligns a pair of images so that the epipolar lines in both images are horizontal; if images A and B form a stereo image pair, then the epipolar lines indicate which pixels in A and B correspond with one another. Thus the image transformations in stereovision are concerned with aligning image pairs along one dimension (that is, along the dimension denoted by the direction of the epipolar lines) while the image transformations used in this thesis are concerned with aligning individual images along two dimensions (that is, the dimensions denoted by the spatial and dispersion axes).

Chapter 3 : Implementation and Experimental Results

3.1 Implementation

The algorithms described in this thesis for orthogonalizing spectral images and for extracting one-dimensional spectra from two-dimensional spectral images have been implemented using the Interactive Data Language (IDL). IDL is an array-based language with built-in image processing functions and GUI development tools that facilitate the development of user-friendly image processing applications. More information about IDL is available Online at http://www.ittvis.com/ProductServices/IDL.aspx.

3.2 Experimental Design

The correctness of DetermineSpatial (DS), DetermineDispersion2 (DD2), and DetermineDispersion3 (DD3) cannot be evaluated until after images have been orthogonalized using the parameters determined for the axes. However, in developing DD2 and DD3, assumptions were made about the variations in the slopes of the spectral lines within a single spectral image. Experiments designed to test these assumptions are described in Section 3.2.1. Section 3.2.2 describes tests used to evaluate the correctness of the image transformation algorithm. Section 3.2.3 describes a sensitivity analysis that was performed for the transformation algorithm, and Section 3.2.4 describes efficiency tests that were performed for the orthogonalization algorithm.

3.2.1 Testing Assumptions About Spectral Line Slope

DD2 assumes that all of the spectral lines in the inputted spectral image have the same slope. This assumption was tested as follows: Three or four well-defined spectral lines were manually identified in each of seven low-resolution and five medium-resolution spectral images; DD2 was used to calculate the slope of each selected spectral line. If all of the spectral lines identified in the low-resolution spectral images have the same slope, then the assumption made by DD2 is valid for low-resolution images. Likewise, if the spectral lines identified in the medium-resolution spectral images all have the same slope, then the assumption is valid for medium-resolution images. DD3 assumes that any variations in the slopes of the spectral lines within a single image are small enough that the orthogonalization transformation will not be sensitive to the difference. To test this assumption, a region containing at least fifteen well-defined spectral lines was manually identified in each of seven low-resolution and five medium-resolution spectral images. DD3 was used to calculate the slope of each spectral line in the selected regions; the implementation of the algorithm was modified for this test so that the slope of each identified line in the selected region would be reported to the user. The variation in spectral line slope for both the low- and medium- resolution images was recorded and a sensitivity analysis was performed to determine the transformation's sensitivity to changes in spectral line slope (the procedure used to perform the sensitivity analysis is described in Section 3.2.3). If the variation in spectral line slope for the low-resolution images is within the insensitive range of the transformation, then the assumption made by DD3 is valid for low-resolution images. Likewise, if the spectral line slope variation seen in the medium-resolution images is within the insensitive range of the transformation, then the assumption made by DD3 is valid for medium-resolution images. The results for these tests on DD2 and DD3 are reported in Section 3.3.1.

3.2.2 Correctness Tests for the Image Transformation Algorithm

To test the correctness of the image transformation algorithm, both low- and medium- resolution spectra were orthogonalized using the transformation algorithm; for all images, linear equations were used to describe the spatial and dispersion axes both before and after the transformation. The angle between the axes of each image was calculated both before and after the transformation according to the equation

$$\theta = \cos^{-1}\left(\frac{-b_1 - a_0}{\sqrt{(1+a_0^2)(1+b_1^2)}}\right) \qquad (3.1)$$

where θ is the angle between the axes, a_0 is the coefficient of the linear term in the equation $x = a_0 y + a_1$ that describes the dispersion axis, and b_1 is the coefficient to the linear term in the linear equation $y = b_1 x + b_2$ that describes the spatial axis. The values of the coefficients a_0 and b_1 before the transformation were also compared with the coefficients' values after the transformation. If θ is between 89.73° and 90.27°, and if $|a_0|$ and $|b_1|$ are reduced by 1-2 orders of magnitude, then the output of the transformation algorithm is reasonable. The results of these angle calculations and parameter comparisons are presented in Section 3.3.2 for Transformation3; for these tests, DD3 was used for axis determination both before and after the transformation.

3.2.3 Sensitivity Analysis

The sensitivity of the transformation to changes in the parameters a_0, b_0, and b_1 was measured using a pair of medium resolution spectral images: The image of a standard star and the image of a planet. To determine the transformation's sensitivity to changes in a_0, the following procedure was used: first the values of a_0, b_0, and b_1 were initialized to the correct values calculated for the pair of images using DD3 and DS. Keeping the values of b_0 and b_1 constant, the value of a_0 was decreased until it was visually obvious that the pair of images, after being transformed, were not properly orthogonalized. Then values of a_0 were selected from the range extending from the correct value of a_0 to a value of a_0 for which the transformed image was clearly not orthogonal; for each selected value of a_0, the angle between the dispersion and spatial axes was calculated. To determine the dispersion axes of the images in preparation for these angle calculations, DD3 was used in manual mode. After returning a_0 to its correct value, the same process was repeated, only this time a_0 was increased instead of being decreased. A similar procedure was used to determine the sensitivity of the transformation to changes in b_0 and in b_1. The results of this sensitivity analysis are given in Section 3.3.3 for Transformation3 and are used to define ranges of values that a_0, b_0, and b_1 can take on without adversely affecting the quality of the transformation. These results are also used to evaluate the assumption made by DD3 as described in Section 3.2.1.

3.2.4 Efficiency Tests

Efficiency tests were performed for the implementations of three algorithms: DS, DD3, and Transformation3. These tests were performed using IDL's code profiling procedure, *profiler*. For each algorithm, the running time of the implementation was averaged over five runs using five different input images; these running times do not include the time spent waiting for user I/O. The execution times for DS, DD3, and Transformation3 were added together to calculate the amount of time required to process a single spectral image. If this running time is less than 5 minutes, and if it is possible that the processing time will be less than 5 minutes after user I/O time is added, then the IDL implementation of the orthogonalization algorithm meets the efficiency requirements described in Section 2.1.3. The results of these tests are presented in Section 3.3.4.

3.3 Results

The following sections report the results of experiments described in Section 3.2.

3.3.1 Dispersion Axis Determination Tests

3.3.1.1 Results for DetermineDispersion2 (DD2)

Figure 3-1 and Figure 3-2 report the results of tests designed to evaluate the validity of the assumption made by DD2 – that all of the spectral lines in the inputted spectral image have the same slope.

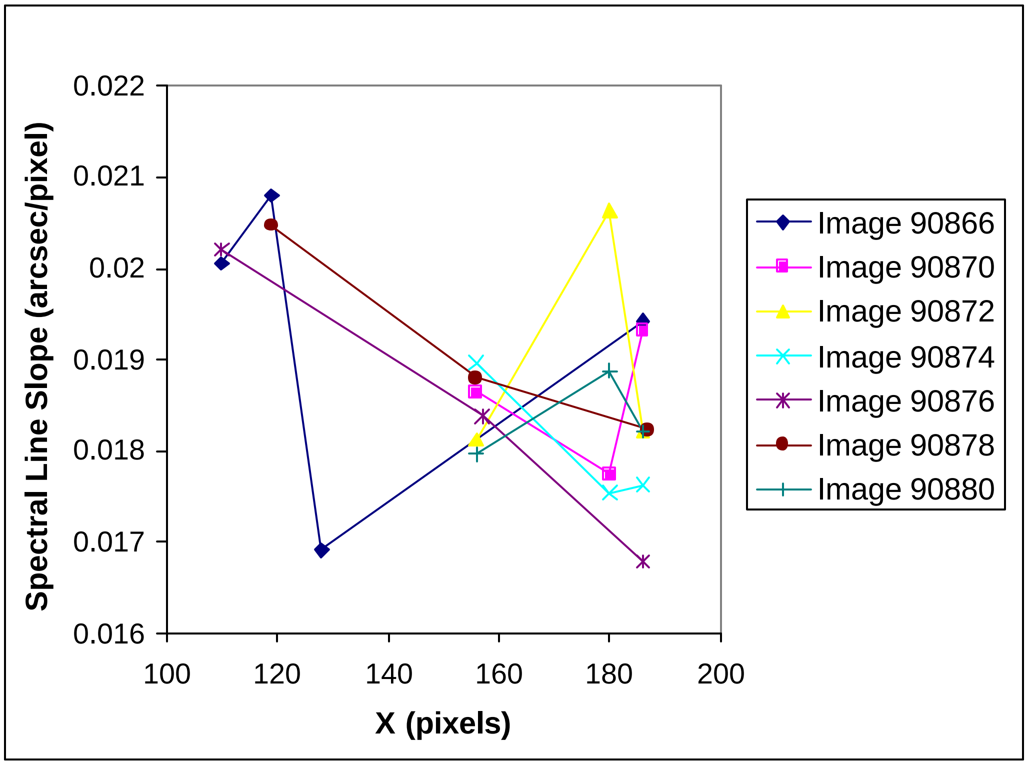


Figure 3-1. The relationship between spectral line slope and position along the image's x-axis as calculated for low-resolution spectra by DD2. The range of the x-axis indicates the region in the images that contain easily identified spectral lines.

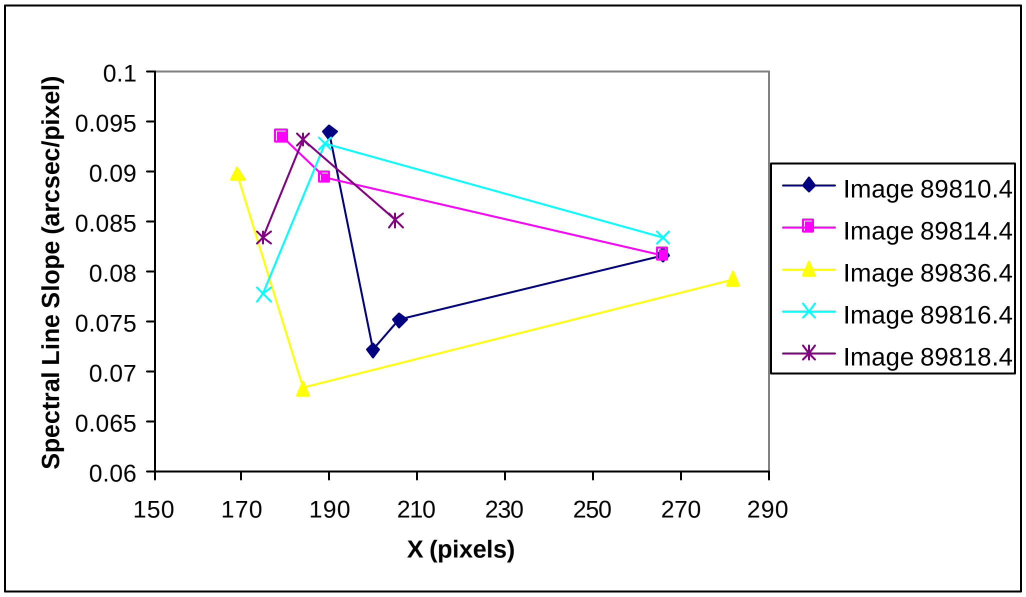


Figure 3-2. The relationship between spectral line slope and position along the image's x-axis as calculated for medium-resolution spectra by DD2. The range of the x-axis indicates the region in the images that contain easily identified spectral lines.

The slopes plotted in Figure 3-1 vary from 0.0167 to 0.0208, a range of 0.0041, while the slopes plotted in Figure 3-2 vary from 0.0684 to 0.0940, a range of 0.0256. This variation in spectral line slope may be real or it may be due to noise. If the variation is real, then the assumption made by DD2 is incorrect. However, if the variation is simply due to noise, then DD2's use of a single spectral line to determine the orientation of the dispersion axis may result in inaccurate algorithm output: if the user selects a spectral line whose slope is at one of the extreme ends of the slope range for the image, the slope of the determined dispersion axis will not be characteristic of the image as a whole. This test, therefore, indicates that there is a flaw in the design of DD2.

3.3.1.2 Results for DetermineDispersion3(DD3)

Figure 3-3 and Figure 3-4 present the results of tests designed to evaluate the validity of the assumption made by DD3, namely that any variations in the slopes of the spectral lines within a single spectral image are small enough that the orthogonalization transformation will not be sensitive to the difference.

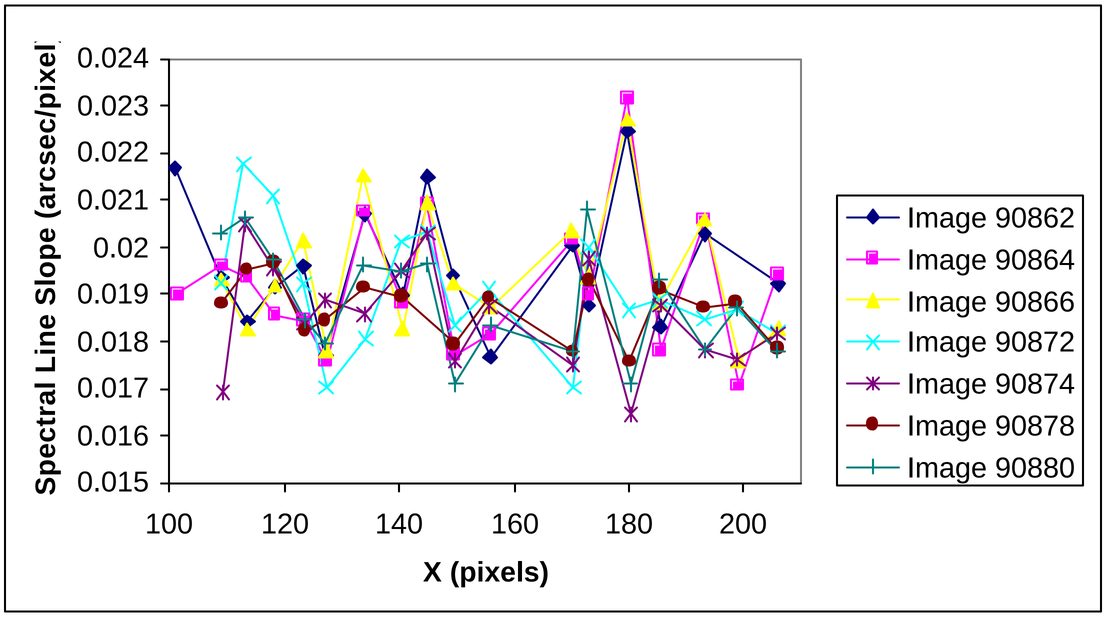


Figure 3-3. The relationship between spectral line slope and position along the image's x-axis as calculated for low-resolution spectra by DD3. The range of the x-axis indicates the region in the images that contain easily identified spectral lines.

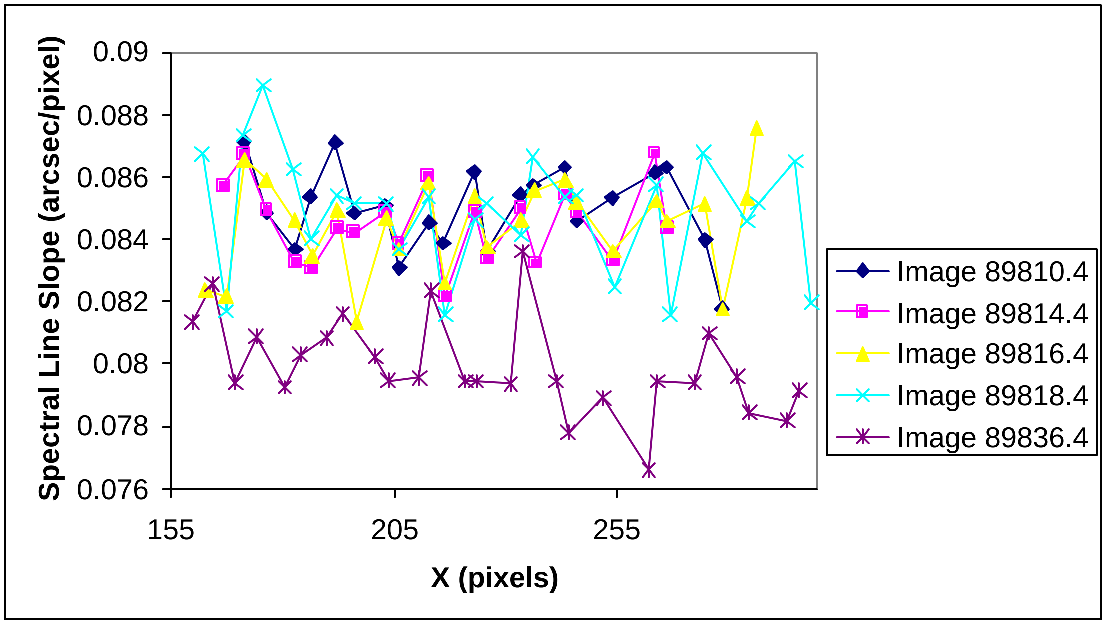


Figure 3-4. The relationship between spectral line slope and position along the image's x-axis as calculated for medium resolution spectra by DD3. The range of the x-axis indicates the region in the images that contain easily identified spectral lines.

The spectral line slopes shown in Figure 3-3 vary from 0.0165 to 0.0232, a range of 0.0067. Here, the range is calculated over multiple images because the dispersion axis determined using one spectral image should be applicable to other images as well; therefore, the assumption made by DD3 must be tested across multiple images. These results show that if the orthogonalization transformation is sensitive to variations of less than 0.0067 in spectral line slope, the transformation algorithm is sensitive to the slope variations in low-resolution spectra and the assumption made by DD3 is invalid for low-resolution spectra. The spectral line slopes shown in Figure 3-4 appear in two separate "bands": the lower band consists of the slope values obtained from Image 89836.4 and the upper band consists of the slope values obtained from the remaining images. The slope values in the lower band range from 0.0766 to 0.0836 while the values in the upper band range from 0.0814 to 0.0890, yielding a range of 0.007 for the lower band and a range of 0.0076 for the upper band. Thus, if the orthogonalization transformation is sensitive to variations of less than 0.0076 in spectral line slope, the transformation algorithm is sensitive to the slope variations in medium-resolution spectra, leaving the assumption made by DD3 invalid for medium-resolution spectra. The sensitivity analysis results needed to determine the validity of this assumption is presented in section 3.3.3.

3.3.2: Orthogonalization Transformation Tests

Table 3-1. Angle calculation results for the transformation test on Transformation3. For each star and planet spectral-image pair, Original θ is the angle between the spatial and dispersion axis before transformation, Transformed θ is the angle between the axes after transformation, $\Delta\theta$ the difference between Transformed θ and Original θ, and Error is the difference between Transformed θ and 90°.

Star Image Number	Planet Image Number	Original θ (degrees)	Transformed θ (degrees)	$\Delta\theta$ (degrees)	Error (degrees)
90886	90880	90.89	90.04	-0.85	0.04
89848.1	89836	92.46	90.06	-2.40	0.06
89848.2	89836	93.21	90.06	-3.15	0.06
89848.3	89836	94.43	90.08	-4.35	0.08
89848.4	89836.4	95.60	90.11	-5.49	0.11
89848.5	89836	96.46	90.13	-6.33	0.13

Table 3-2. Axis parameter values before and after transformation for Transformation3. For each star and planet spectral-image pair, values of a_0 and b_1 both before and after the transformation are reported. The value of b_0 was held constant at zero for these tests.

Star Image Number	Planet Image Number	Original a_0	Transformed a_0	Original b_1	Transformed b_1
90886	90880	1.8×10^{-2}	6.2×10^{-4}	-2.7×10^{-3}	8.7×10^{-5}
89848.1	89836	7.8×10^{-2}	1.4×10^{-3}	-3.5×10^{-2}	-3.4×10^{-4}
89848.2	89836	7.8×10^{-2}	1.4×10^{-3}	-2.2×10^{-2}	-3.8×10^{-4}
89848.3	89836	7.8×10^{-2}	1.4×10^{-3}	-1.1×10^{-3}	3.3×10^{-5}
89848.4	89836.4	7.8×10^{-2}	1.4×10^{-3}	1.9×10^{-2}	5.4×10^{-4}
89848.5	89836	7.8×10^{-2}	1.4×10^{-3}	3.4×10^{-2}	7.7×10^{-4}

Table 3-1 shows that for all of the image pairs listed in the table, the angle between the spatial and dispersion axes after orthogonalization is less than 0.14° away from 90°. The results in Table 3-2 indicate that the transformation brings the axis parameters a_0 and b_1 one to two orders of magnitude closer to zero than they were before the transformation; if the orthogonalization procedure worked perfectly, a_0 and b_1 would both be zero after the transformation. This low error level coupled with the corresponding decreases in the magnitudes of a_0 and b_1 suggest that the Transformation3 works reasonably well in the case where both the spatial and dispersion axes are described by a linear equation. No results are presented for the case where the spatial axis is described by a quadratic equation because none of the spectral images available for use in testing during this research were well described by a quadratic spatial axis.

3.3.3: Sensitivity Analysis

The following graphs show the results of a sensitivity analysis on each of the parameters a_0, b_0, and b_1. In these figures and in the discussion below, theta is the angle between the spectral and dispersion axes of a spectral image.

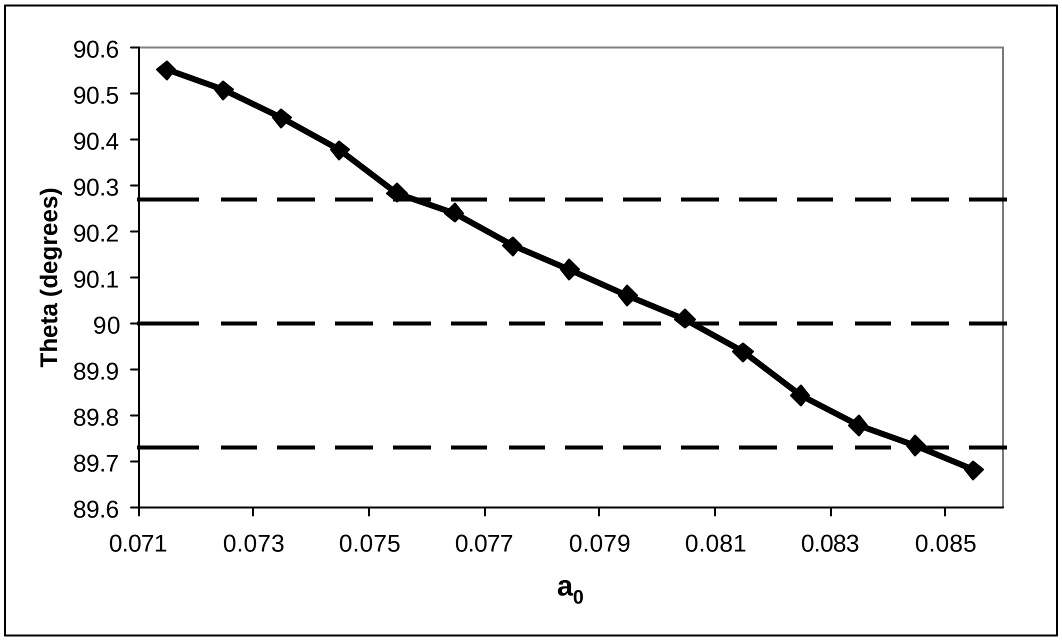


Figure 3-5. The variation of theta as a function of the transformation parameter a_0, where theta is the angle between the spatial and dispersion axes of the spectral images used in the sensitivity analysis. The dashed lines indicate, from top to bottom, angles of 90.27°, 90.0°, and 89.73°.

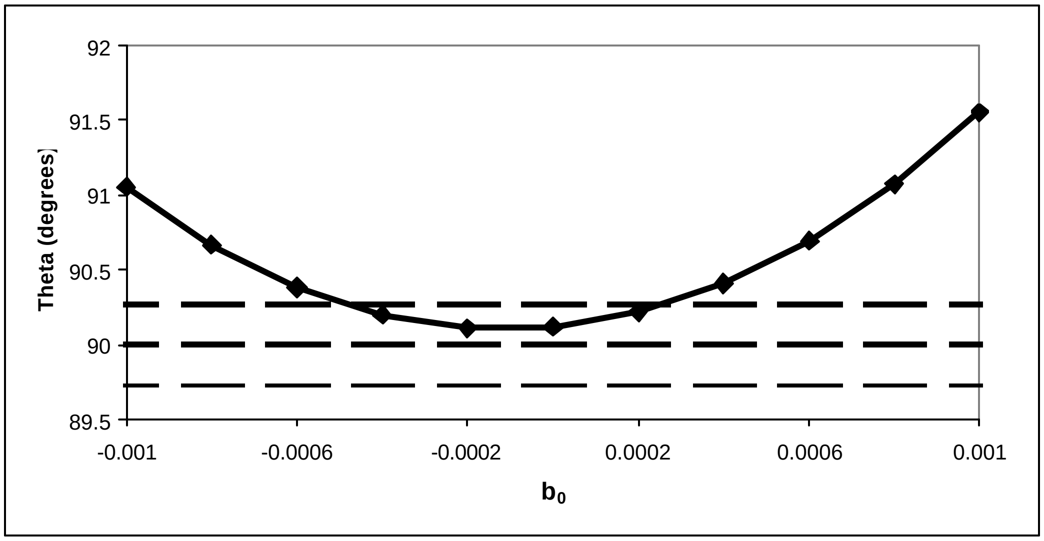


Figure 3-6. The variation of theta as a function of the transformation parameter b_0, where theta is the angle between the spatial and dispersion axes of the spectral images used in the sensitivity analysis. The dashed lines indicate, from top to bottom, angles of 90.27°, 90.0°, and 89.73°.

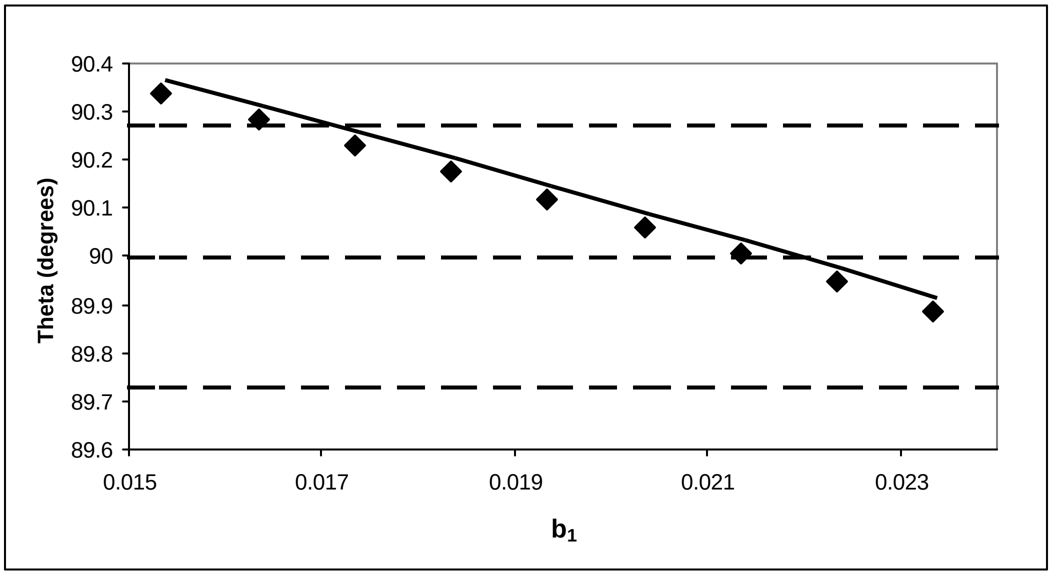


Figure 3-7. The variation of theta as a function of the transformation parameter b_1, where theta is the angle between the spatial and dispersion axes of the spectral images used in the sensitivity analysis. The dashed lines indicate, from top to bottom, angles of 90.27°, 90.0°, and 89.73°.

The goal of this sensitivity analysis was to compute the range of values that a_0, b_0, and b_1 can take on without unacceptably reducing the quality of the transformation.

To carry out this calculation, a range of theta-values was selected such that a spectral image may be considered satisfactorily orthogonalized if its theta value appears in that range. The following reasoning was used to select this range:

> Table 3-1 in Section 3.3.2 indicates that 90.89° is the smallest angle theta for a spectral image that has not been orthogonalized; thus, for the data examined here, 0.89° is the smallest deviation from 90° that we can expect before orthogonalization. If the image transformation can decrease this deviation to 30% of its original value (that is, to 0.27°), then the departure from perpendicularity will probably not be visually detectable and the image will be satisfactorily orthogonalized. This maximum deviation of 0.27° means that theta must be between 90.27° and 89.73° for the transformation to be correct.

For this range of theta-values, the data in Figure 3-5, Figure 3-6, and Figure 3-7 indicate that acceptable orthogonalization occurs for a_0 ranging from 0.076 to 0.084, for b_0 ranging from –0.0004 to 0.0002, and for b_1 ranging from 0.017 to 0.023. Thus a_0 may vary with a range of 0.008 without reducing the transformation quality to an undesirable extent; when viewed in conjunction with the results from Section 3.3.1, these sensitivity analysis results indicate that the assumption made by DD3 is valid. In addition, the sensitivity analysis results show that b_0 may vary with a range of 0.0006 and b_1 may vary with a range of 0.006 without unacceptably reducing the transformation quality.

3.3.4: Efficiency Tests

Table 3-3. Running times for DetermineSpatial (DS), DetermineDispersion3 when run in automatic mode (DD3A), DetermineDispersion3 when run in manual mode (DD3M), and Transformation3 (T3). For each algorithm, the computation time (CPU time), the disk I/O time, and the sum of the computation and disk I/O times are reported. Each running time is an average over five runs.

Algorithm	Computation Time (seconds)	Disk I/O Time (seconds)	Computation + Disk I/O Time (seconds)
DS	3.20	0.35	3.55
DD3A	6.54	0.35	6.89
DD3M	2.51	0.35	2.86
T3	13.37	0.35	13.72

The data reported in Table 3-3 give the computation and disk I/O times for DetermineSpatial, DetermineDispersion3, and Transformation3. These results show that 23.11 seconds of computation time are required to determine an image's spatial and dispersion axes and then transform the image. After adding the disk I/O time to this total, the time required to process a single spectral image increases to 24.16 seconds. This running time is clearly less than five minutes; if the time spent waiting for user I/O were added to this total, the running time could potentially be under 5 minutes, depending on the amount of time that the user spends entering input. Thus, the implementations of these algorithms have met the efficiency requirements laid out in Chapter 2.

Chapter 4 : Conclusions and Future Work

4.1 Conclusions

The purpose of this thesis was to develop algorithms that properly orthogonalize spectral images from the COMICS spectrometer. To be correctly orthogonalized, the spatial and dispersion axes of a spectral image must be perpendicular to one another, with the spatial axis being vertical and the dispersion axis being horizontal. The results presented in Section 3.3.2 demonstrate that the orthogonalization algorithms described in this thesis produce correct output relative to these criteria. In addition to the correctness requirements, the IDL implementations of these algorithms must meet efficiency requirements so that the software may be used to reduce large quantities of data quickly. These requirements state that it must be possible to determine the spatial and dispersion axes of a single image and then transform the image so that the axes are perpendicular in less than 5 minutes. The results presented in Section 3.3.4 indicate that the implementation used for this thesis meets these efficiency requirements. Thus, we have presented algorithms that may be used by astronomers to correctly and efficiently orthogonalize COMICS spectral images as part of the data reduction process.

In addition to the correctness and efficiency requirements just discussed, the dispersion axis determination algorithm used during orthogonalization should be independent of wavelength calibration. This thesis has introduced a new algorithm (DetermineDispersion3) that has this independence property. Independence allows the orthogonalization and wavelength calibration algorithms to be implemented as separate modules in software, making the software easier to extend and maintain. Since the dispersion axis determination algorithm presented in the COMICS manual (DetermineDispersion1) does not have the independence property, this thesis has introduced a new technique for separating orthogonalization from wavelength calibration.

4.2 Future Work

There is still more work to be done on the research problem discussed in this thesis. First, techniques must be developed for calibrating spectra after they have been extracted from spectral images. Then, the data reduction algorithms that have been developed for COMICS must be generalized so that those algorithms can be used to reduce data from other instruments including the Michelle spectrometer on the Gemini North telescope, the T-Recs spectrometer on the Gemini South telescope, and the VISIR spectrometer on the Very Large Telescope. The procedures that should be used to reduce data from these instruments differ according to the hardware characteristics of each instrument. Generalizing the COMICS-specific algorithms so that they apply to these varied instruments involves modifying the algorithms so that the methods they use support the other instruments' reduction procedures. In addition, new algorithms will need to be developed to support reduction steps that are required for these instruments but are not required for COMICS.

References

[1] P. Barrett and L. L. Dressel, “Spectral Extraction of Extended Sources Using Wavelet Interpolation,” in *The 2005 HST Calibration Workshop*, pp. 260-266.

[2] Z. Chen, C. Wu, and H. T. Tsui, “A New Image Rectification Algorithm,” *Pattern Recognition Letters*, vol. 24, pp. 251-260, 2003.

[3] M. Cohen et al., “Spectral Irradiance Calibration in the Infrared. X. A Self-Consistent Radiometric All-Sky Network of Absolutely Calibrated Stellar Spectra,” *The Astronomical Journal*, vol. 117, pp. 1864-1889, Apr. 1999.

[4] M. C. Cushing, W. D. Vacca, and J.T. Rayner, “Spextool: A Spectral Extraction Package for SpeX, a 0.8-5.5 Micron Cross-Dispersed Spectrograph,” *Publications of the Astronomical Society of the Pacific*, vol. 116, pp. 362-376, Apr. 2004.

[5] L. Dressel, P. Barrett, P. Goudfrooij, and P. Hodge, “Improving the Rectification of Spectral Images,” in *The 2005 HST Calibration Workshop*, pp. 267-276.

[6] R. Fisher, S. Perkins, A. Walker, and E. Wolfart. (2003). Thinning. *Hypermedia Image Processing Reference* [Online]. Available: http://homepages.inf.ed.ac.uk/rbf/HIPR2/thin.htm

[7] R. Fisher, S. Perkins, A. Walker, and E. Wolfart. (2003). Histogram Equalization. *Hypermedia Image Processing Reference* [Online]. Available: http://homepages.inf.ed.ac.uk/rbf/HIPR2/histeq.htm

[8] R. Fisher, S. Perkins, A. Walker, and E. Wolfart. (2003). Hit-and-Miss Transform. *Hypermedia Image Processing Reference* [Online]. Available: http://homepages.inf.ed.ac.uk/rbf/HIPR2/hitmiss.htm

[9] R. Fisher, S. Perkins, A. Walker, and E. Wolfart. (2003). Hough Transform. *Hypermedia Image Processing Reference* [Online]. Available: http://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm

[10] R. S. Furuya, ed., *COMICS Data Reduction Manual*, ver. 2.1.1, Subaru Telescope Facility, 2008.

[11] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Addison-Wesley Publishing Company, 1992, ch. 7.

[12] H. L. Grimes, "The Extension of the Data Reduction Manager (DRM) to Support Reduction of Planetary Spectra," unpublished report, 2008.

[13] H. L. Grimes, "The Extension of the Data Reduction Manager (DRM) to Support the Reduction of Planetary Spectra," poster presented at the 2008 Oregon NASA Space Grant Consortium Student Symposium, Corvallis, OR.

[14] IDL Reference Guide, IDL Version 7.0, November 2007 ed. [Online], ITT Visual Information Solutions, 2007, pp. 2626-2628. Available: http://www.ittvis.com/portals/0/pdfs/idl/refguide.pdf

[15] P. Irwin, *Giant Planets of Our Solar System: Atmospheres, Composition, and Structure*. Chichester, UK: Springer-Praxis Books in Geophysical Sciences, Praxis Publishing Ltd. 2003.

[16] V.G. Kunde et al, "Jupiter's Atmospheric Composition from the Cassini Thermal Infrared Spectroscopy Experiment," *Science*, vol. 305, pp. 1582-1586, 2004.

[17] J. Mallon and P. F. Whelan. (2005). Projective Rectification from the Fundamental Matrix. *Image and Vision Computing* [Online]. Available: http://www.vsg.dcu.ie/papers/ivc_2005_jm.pdf

[18] G.S. Orton et al, “Evidence for Methane Escape and Strong Seasonal and Dynamical Perturbations of Neptune’s Atmospheric Temperatures,” *Astronomy & Astrophysics*, vol. 473, pp. L5-L8, 2007.

[19] G. S. Orton et al, “Semi-Annual Oscillations in Saturn’s Low-Latitude Stratospheric Temperatures,” *Nature*, vol. 453, pp. 196-199, 2008.

[20] M. Pollefeys, R. Koch, and L. Van Gool, “A Simple and Efficient Rectification Method for General Motion,” in *Proceedings of the International Conference on Computer Vision*, vol. 1, pp. 496-501, 1999.

[21] W. H. Wang, SIMPLE: An IDL Based Data Reduction Pipeline for Wide-Field Near-Infrared Imaging [Online]. Available: http://www.aoc.nrao.edu/~whwang/idl/SIMPLE/simple.pdf, November 2008.

Appendix A: The Hough Transform

The Hough transform is a useful tool for identifying regular curves, such as lines or circles, in an image. This appendix, however, will only discuss the detection of straight lines. The basic idea underlying this line identification technique is as follows: each non-zero pixel, or point, in an image may be part of one or more straight lines in the image. Knowing that two or more points lie along the same line provides evidence that a line connecting those points is present in the image; the greater the number of points, the greater the evidence for a line connecting those points.[9][11]

As we collect evidence for lines, it will be useful to describe the lines in parametric, or normal, form:

$$x \cos\theta + y \sin\theta = r \qquad \text{(A.1)}$$

In this equation, r is the perpendicular distance from the origin to the line, and θ is the angle between r and the x-axis (see Figure A-1). Each line is described uniquely by an (r, θ) pair using this representation.

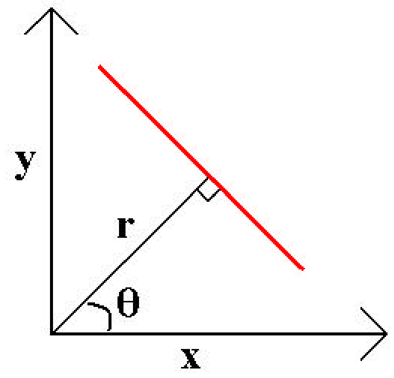
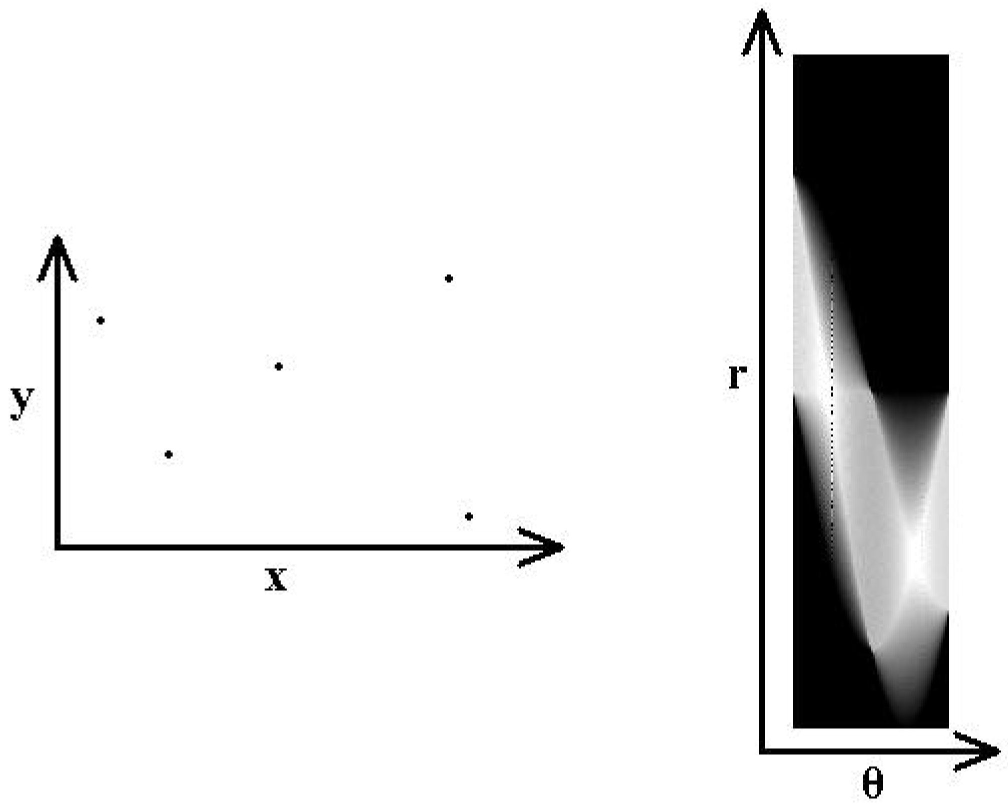


Figure A-1. The parametric representation of a line.

The Hough transform collects evidence for lines by changing the representation of the image: Instead of depicting the image as a set of points in Cartesian space, the Hough transform represents the image as a set of sinusoidal curves in the (r, θ) parameter space. To calculate the image's representation in parameter space, the following procedure is performed: For each point in the image as it is depicted in Cartesian space, plot the (r, θ) pairs that represent lines which include the point. After this process is complete, each point in the Cartesian space is represented as a sinusoid in the parameter space (see Figure A-2). If two points lie on the same line in Cartesian image space, then the corresponding sinusoids intersect in parameter space. Thus, the number of sinusoids that intersect at point (r, θ) in the parameter space is equal to the number of points that lie on the corresponding line in Cartesian space.[9][11]



(a) (b)

Figure A-2. (a) A Cartesian-space image that consists of five points, and (b) the Hough transform of that image.

The Hough transform is implemented by dividing the parameter space into finite-sized accumulator cells. Each point (x, y) in the Cartesian image space is transformed into a sinusoid in parameter space, and the accumulator cells that lie along this curve are incremented. The values in the accumulator cells at the end of the transform reveal the number of sinusoids that intersect within each cell.[9]

As just described, the Hough transform may be used to represent an image in a parameter space that facilitates the identification of straight lines. It is also possible to take an image that is represented in the Hough parameter space and calculate the image's representation in Cartesian space. This inverse of the Hough transform is known as the Hough backprojection and is computed by mapping each point in the parameter space to a line in Cartesian space.[9]

Appendix B: Histogram Equalization

Histogram equalization is a technique that can be used to enhance the contrast and dynamic range of an image; here, the dynamic range is the range of pixel values that are found in the image. This technique works by using a monotonic, nonlinear mapping function to change the values of the pixels in the image based on the structure of the image's histogram[7].

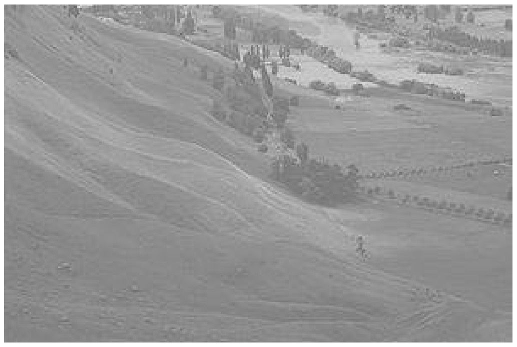
The following discussion is based on Gonzalez and Wood's treatment of the discrete form of histogram equalization[11]. Consider a gray-level image I, and let L be the number of gray levels. Let r be a variable that represents the gray levels in I, and let r_k, $0 \le k \le L-1$, be a variable that represents the kth gray level. Further, let n be the total number of pixels in I and let n_k be the number of times that the kth gray level appears in I. Then,

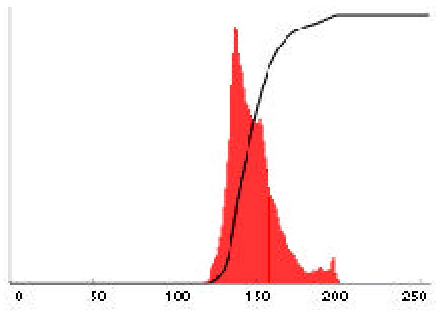
$$p_r(r_k) = \frac{n_k}{n}, \quad k = 0,1,\ldots, L-1 \qquad \text{(B.1)}$$

is the probability of the kth gray level. The histogram of image I is then a plot of $p_r(r_k)$ versus r_k, and the mapping function used for histogram equalization is

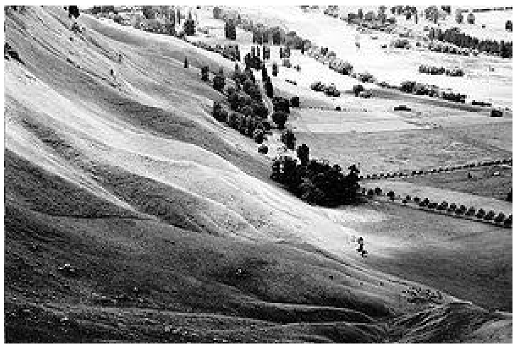
$$s_k = \sum_{j=0}^{k} p_r(r_j) \qquad \text{(B.2)}$$

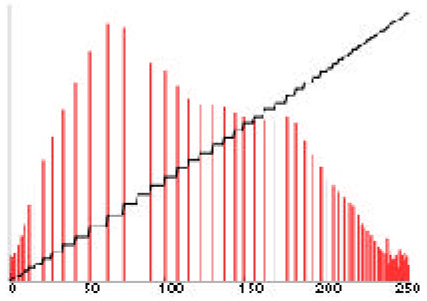
where s_k is a variable that represents the kth gray level in the image produced by histogram equalization. This mapping function is the cumulative distribution function of the variable r and will produce an output image whose dynamic range and contrast are greater than those of I (see Figure B-1 for an example).


(a)



(b)


(c)



(d)

Figure B-1. (a) An unequalized image along with (b) its histogram (in red) and cumulative distribution function (in black). (c) The image from (a) after histogram equalization along with (d) its histogram (in red) and cumulative distribution function (in black). These images and their histograms were obtained online at http://en.wikipedia.org/wiki/Histogram_equalization.

Appendix C: Thinning

Thinning is an operation from mathematical morphology that is often used to erode lines in binary images until those lines are only one pixel thick. This operation is defined in terms of the hit-and-miss transform described below[6].

The hit-and-miss transform takes as its input a binary image I and a structuring element S, and produces a new binary image as output. A structuring element is usually a square array with an odd number of entries that contains ones and zeros (see Figure C-1 for an example); the contents of this element determine the effect that the hit-and-miss transform has on the image I. The center pixel (colored in red Figure C-1) of the structuring element is called the origin. This hit-and-miss transform works as follows: for each p pixel in image I, the structuring element is laid over the image so that the origin of the structuring element is aligned with pixel p. Then, the pixels in the structuring element are compared with the underlying pixels in image I. If the pixels in the structuring element exactly match the pixels in the image, then pixel p is set to 1; otherwise, pixel p is set to 0[8].

0	0	0
0	1	0
1	1	1

Figure C-1. A structuring element for the hit-and-miss transform. The center pixel (colored in red) is the origin of the structuring element.

In some implementations of the hit-and-miss transform, such as the implementation provided by the Interactive Data Language (IDL), the transform takes three inputs: a binary image I, a hit structural element, and a miss structural element. The hit structural element is translated over all of the pixels in image I as described in the last paragraph to produce a new binary image A. Then, the miss structural element is translated over all of the pixels in the complement of image I to produce a binary image B. The hit-and-miss transform then outputs the image $A \wedge B$.

Now that we understand how the hit-or-miss transform works, we can define thinning as follows:

$$\text{thin}(I,H,M) = I - \text{hit\_and\_miss}(I,H,M). \qquad \text{(C.1)}$$

Here, I is a binary image, H is a hit structural element, M is a miss structural element, and the subtraction operation is logical subtraction defined by the equation $X - Y = X \wedge \neg Y$. This thinning operation is normally applied iteratively until convergence (that is, until the procedure no longer causes the image to change)[6].

It is sometimes useful to apply more than one pair of hit and miss structural elements to an image during thinning. In this case, a sequence of calls to the thinning operation is performed where each call uses a different pair of structural elements; the output of one call to the thinning operation is used as the input for the next call. This sequence of thinning operations is usually applied iteratively until convergence[6].

Appendix D: Bilinear Interpolation

Bilinear interpolation is a technique that may be used during geometric transformations, such as the image transformations discussed in section 4.3.3 of this thesis. This technique is used to calculate the values that should be assigned to each pixel in a transformed image.[11]

Suppose we have a geometric transformation described by the equations

$$\begin{aligned}\hat{x} &= r(x, y)\\ \hat{y} &= s(x, y)\end{aligned} \qquad \text{(D.1)}$$

where $(\hat{x}, \hat{y})$ are the coordinates of the point in the original image whose value should be given to the point at the coordinates (x, y) in the transformed image, and where $r(x, y)$ and $s(x, y)$ are the transformation functions. The calculated coordinates $\hat{x}$ and $\hat{y}$ may have non-integer values, while all of the pixels in the original image have integer-valued coordinates. Therefore, interpolation must be used to compute the value that should be assigned to pixel (x, y) in the transformed image.[11]

For non-integer coordinates $(\hat{x}, \hat{y})$, bilinear interpolation uses the known pixel values of the four nearest neighbors to calculate the value at $(\hat{x}, \hat{y})$, denoted $v(\hat{x}, \hat{y})$, in the original image (see Figure D-1). These four nearest neighbors all have integer-valued coordinates. The equation

$$v(\hat{x}, \hat{y}) = a\hat{x} + b\hat{y} + c\hat{x}\hat{y} + d \qquad \text{(D.2)}$$

is used to perform the interpolation. Here, the coefficients a, b, c, and d are determined using the four nearest neighbors. Once these coefficients have been computed, $v(\hat{x}, \hat{y})$ is calculated using equation (D.2), and the resulting value is assigned to pixel (x, y) in the transformed image.[11]

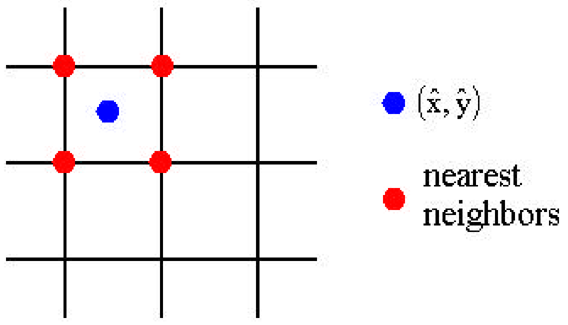


Figure D-1. A point $(\hat{x}, \hat{y})$ with non-integer coordinates and the four nearest neighbors that will be used to compute the value at $(\hat{x}, \hat{y})$.