2000

Rendering multispectral data as useful 'Super-Visual' images

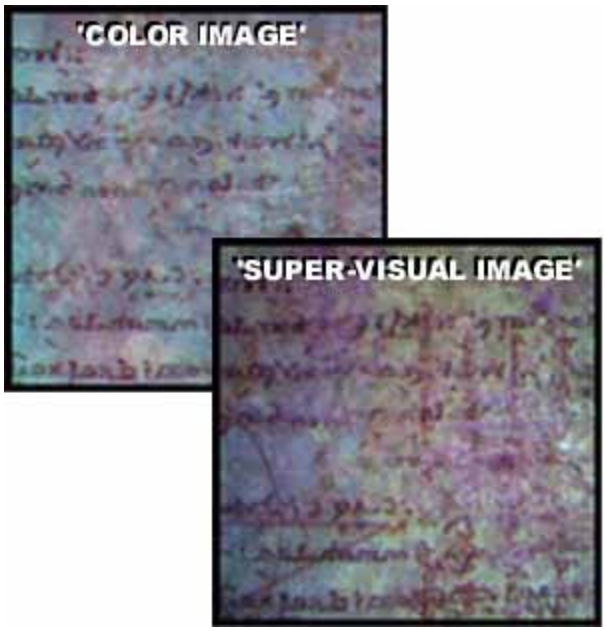
Matthew Gypson

SIMG-503

Senior Research

Rendering Multispectral Data as useful 'Super-Visual' Images

Final Report



Matthew Gypson
Center for Imaging Science
Rochester Institute of Technology
May 2000

Table of Contents

Rendering Multispectral data as useful 'Super-Visual' images

Matthew Gypson

Table of Contents

Abstract

Copyright

Introduction

Background

Theory

Methods

Results

Discussion

Conclusions

IDL code

References

Title Page

Rendering Multispectral data as useful 'Super-Visual' images

Matthew Gypson

Abstract

Due to the nature of mulitspectral images, it is difficult to present all available information in a single image. This study addresses the need for data compression, to enable faster and easier evaluation of a subject containing important details in a wide spectral range. The specific aim of this research is to construct algorithms to in effect compress sampled wide-band electromagnetic data into the visible range. This will present maximum image details within a final color image. The developed compression scheme aims to achieve the high efficiency while keeping the best quality of image subjects. Ancient documents (from the Dead Sea Scrolls) were used as test subjects because certain regions of the document contain important characters detectable only in the infrared region of the spectrum, while other characters are only in ultraviolet. Algorithms for translation, flat fielding, interpolation, incorporation of human response curves and scaling were among those used. The theory and application of this algorithm to multispectral images will be presented.

Back (Table of Contents) Forward (Copyright)

Center for Imaging Science
Rochester Institute of Technology
Rochester, NY 14623-5604

This report is accepted in partial fulfillment of the requirements of the course SIMG-503 Senior Research.

Title: Rendering Multispectral data as useful 'Super-Visual' images.
Author: Matthew Gypson
Project Advisor: Roger Easton
SIMG 503 Instructor: Joseph P. Hornak

Table of Contents

Back (Abstract) Forward (Introduction)

Rendering Multispectral data as useful 'Super-Visual' images

Matthew Gypson

Introduction

To fulfill a desire to better evaluate and understand the world around us, we often need tools that reach beyond the capabilities of human perception. Our visual system provides one means of perception that has a limited window on the world around us, 400-700nm (wavelength range) of the electromagnetic spectrum. Ancient documents such as the Dead Sea Scrolls
sometimes contain important text that is outside our visual range. To evaluate this information is necessary to have a means to visually inspect it. Image information is typically captured through a bandpass filter that isolates the spectral range of interest and records it as grayscale values corresponding to the intensity of a scene. To render information on a wide range of wavelengths, multiple grayscale images are captured through different filters. These multiple images are traditionally compared with each other to look for object details. This process is time consuming and requires an eye trained to look for important details. Labor-intensive preprocessing is required to extract useful information band by band. A particular method may be developed that is specific to the particular subject. It would be advantageous to have an algorithm that is fast and robust and that generates useful results from many different data sets and different types of subjects.

A color image can be generated by overlaying images captured with red, green and blue filters. These are the three basic components or groups of color for human perception. The spectral information we perceive is limited to combinations of these three color bands. Much like the transmissive filters placed in front of a digital camera, the cones in our retina isolate specific spectral regions. Through we cannot manipulate the biology of the eye, we can manipulate multispectral image data to construct an image that conveys information that the unaided eye cannot see. All bands must be manipulated to generate a perceptible "super-visual" image. It is useful to avoid image specific preprocessing as much as possible, to quicken the rendering of an image for display. The method aims to compress the most important details of wide specta into the confines of human perception. The task is to implement data and spectral compression.

Background

Several methods are possible for converting four or more images into the three RGB images. This research explored only that deemed most likely to give useful results. The first conception of this project was the idea of compressing a wide spectral range into a narrower one. This is a type of "data fusion". Munechika (1990) distinguishes three classes of fusion algorithms.

1. "Fusion for visual display" intends to produce images of human interpretation designed to look good to a human interpreter. Histogram manipulation, contrast stretching, and scaling transformations are all examples of such algorithms. This is the method used to develop the algorithm implemented in this project.

2. "Fusion by separate manipulation of the spatial information" is a "component-substitution" (COS) algorithm. It is based on the fact that areas that are bright or dark in one band tend to be bright or dark in at least some of the other bands, According to Schott (1997). The digital counts of two bands are plotted on a 2-D histogram. The data are distributed in an elongated band; in other words digital numbers tend to increase in one both bands. If we know the gray level in one band, we can predict the approximate value in the adjacent band. This correlation means there is redundancy in information in a multispectral information set. COS algorithms create a new coordinate system by creating linear combinations of pixels from the original coordinate system. This method can be used for mulitispectral data sets with any number of bands.

"Component Substitution"

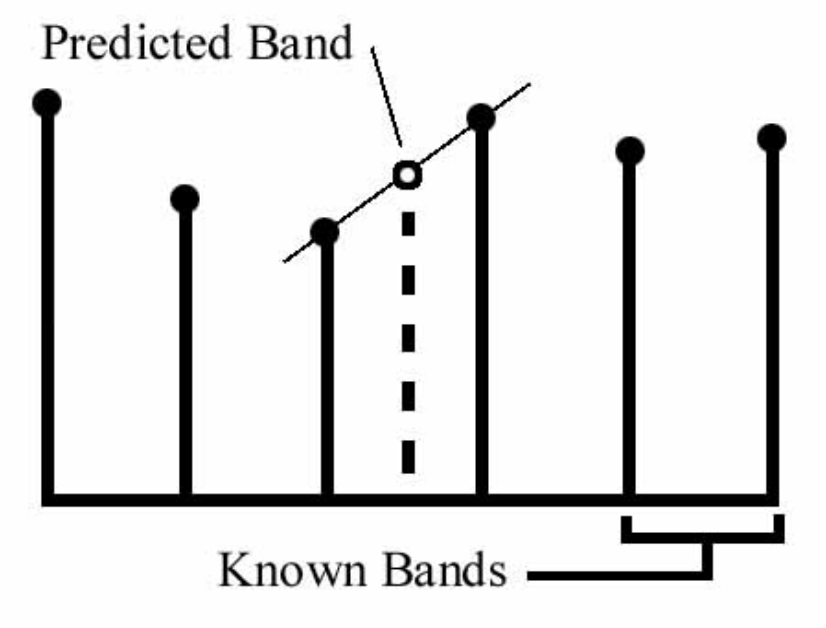


Fig. 1: Gray level in one band can predict approximate value in adjacent band

3. "Fusion for radiometric integrity" creates ratio images by dividing the digital count (DC) in one band by the corresponding DC value in another band for each pixel. The resulting ratios are plotted as an image as demonstrated by Chavez (1991). In a ratio image, the black and white extremes of the gray scale represent pixels having the greatest difference in reflectivity between the two spectral bands. The darkest signatures are areas where the denominator of the ratio is greater than the numerator. Conversely, the numerator is greater than the denominator for the bright signatures. Where the denominator and numerator are equal, there is no difference between the two bands and a midgray is created.

Theory

The human visual system has a limited range of spectral sensitivity (wavelengths from 400nm to 700nm) and three receptors that roughly correspond to three spectral bands: red, green, and blue. These receptors (cones) sample a spectral range with the detectors at three different peaks. In order to be perceived, the data of a color scene must be transformed for display into a tricolor system. If you were to look closely at a computer monitor or television screen you would see that each picture element or pixel contains three colors. The combination of the red areas make up the "red image"; the same is true for the green and blue. Each image consists of three spectral samples of the object/scene at each pixel. Red, green, and blue values are integrated over bands of wavelengths centered at approximately 400nm, 555nm, and 700nm. This transformation is a routine procedure for an image taken within the visual spectrum, but this is not the case in this system.

Imaging systems that have sample spectral bands outside the visual range require more complex transformations. These "multiple spectra" images include separate wavelength intervals, or bands. A multispectral image differs from conventional color photographs, which combine three overlapping spectral ranges in the visual region. Multispectral images contain more than three bands and several of those bands are usually outside the visual range. Only three images can be displayed as RGB on a monitor or television, so additional processing is required to convert the multispectral images for display. Spectral reflectance curves, or reflectance spectra, record the percentage of incident energy that is reflected by a material as a function of wavelength of the energy. Dips in the spectral reflectance curve are called absorption features because they represent absorption of incident energy; peaks represent

reflection of incident energy. These spectral features are clues for recognizing materials and details in images.

Gross (1996) has developed an image-fusion algorithm for remote sensing that combines images with low spatial and high spectral resolution with images that have good spatial resolution but poor spectral resolution. It uses the spectral signature to estimate the percentage of each material within each low-resolution pixel. To achieve high spectral resolution, a narrow-band filter is used to restrict the range of wavelengths. However the detector size must increase to compensate for the reduced irradiances, thus decreasing the spatial resolution. A filter with a wider bandwidth allows more light to pass onto the detector but also decreases the spectral resolution. Gross's algorithm combines an image made with high spatial resolution with wide-spectrum data from an image cube with low spatial resolution. This algorithm requires images from two sensor arrays. Most, if not all, remote sensing applications benefit from high-resolution images. To obtain high spatial resolution, spectral data is sacrificed except where a object is detected twice: a fusion algorithm is used to combine them. Most, if not all, image fusion methods are designed for multispectral remote sensing data. Their goals are to distinguish between different materials within a scene and assign a false color to each. For example, carbon materials have a different spectral reflectance than copper materials and would be assigned a different false color. The 'super-visual' algorithm has a fixed high resolution. Gross's algorithm gets at the goal of a super-visual image in that it distinguishes certain features from others by interpretation of features within the wide spectral range. There is no need to use two resolutions since high resolution is possible both spatially and spectrally with the data used for super-visual images.

Methods

Super-visual images may be used to evaluate the presence of information across a wide spectrum. When compared to a color image, any additional details in a super-visual image would indicate information present in bands outside the visible spectrum. The developed compression algorithm is designed for six input images. The test subjects used for the algorithm development contained important image details in (UV) and (IR) wavelengths. Six filters (UV, Blue, Green, Red, 2 IR) were used to give a spectral range from 300nm to 1000nm, which is thought to contain many important image details. Not all subjects contain important information in all six bands, but this would not be known until after each band was evaluated. It was known that important information is present in the ultraviolet portion of the spectrum for images of ancient documents. When compared to a color image, any additional details in a super-visual image would indicate information present in bands outside the visible spectrum. Six bands were chosen for consistency and to avoid further need for image capture. The algorithm is easily adaptable to incorporate more or fewer bands if necessary.

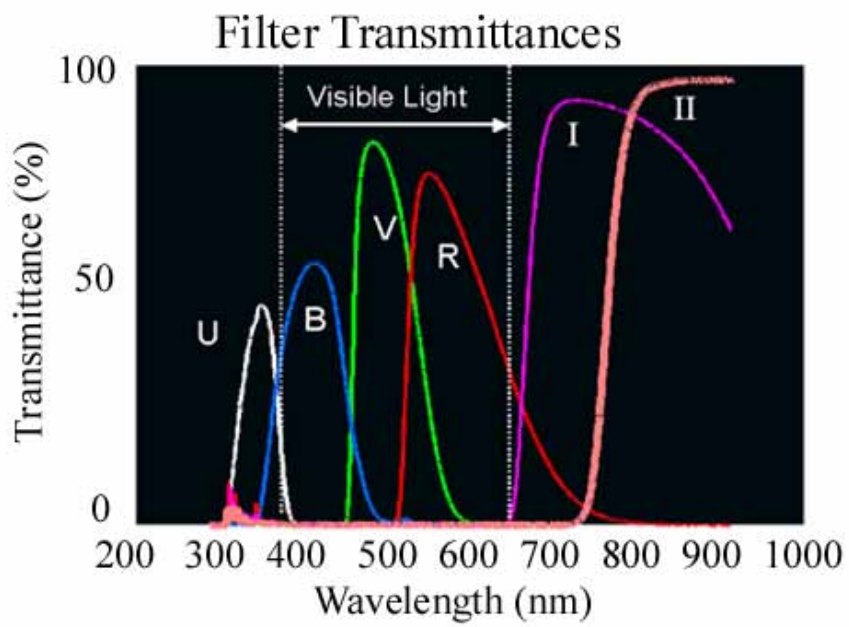


Fig. 2: Transmittance of filters used with test data

six bands captured with six filters (UV, Blue, Green, Red, 2 Infrared)

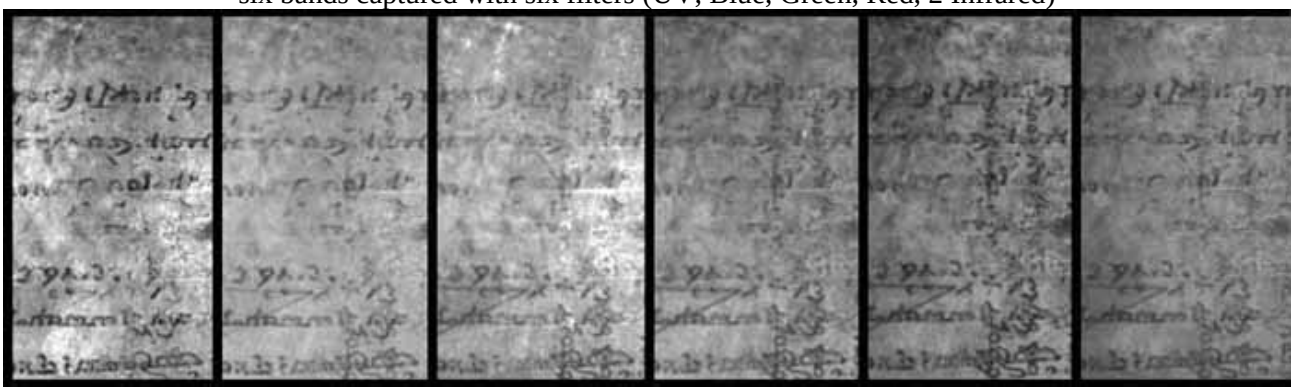


Fig. 3: Six images of the same subject through six different filters

There are four free parameters to these digital color images, spatial resolution and the three available image bands. The same scene is captured with a SENSYS camera (Photometrics, Munich, Germany) through six filters (or bands) so that the same pixel location in all six images corresponds to the same location in the scene imaged. Each pixel has six associated gray values, one for each of the six filters used to capture a different portion of the spectrum. To generate a spectrum in the visible range that approximates a wide spectrum, an interpolation function is created. The original spectrum cannot be fully recovered because it is sampled; the ideal interpolation function would exactly duplicate the spectrum emitted from the subject. The interpolation function used is only a good estimation of the amplitude integrated over each filters transmittance range. The value of the amplitude is assigned at the peak wavelength of each band. The amplitude of the spectrum between the band values is unknown. To estimate the unknown values, an approximation of the original spectrum in created by using the bands as guides. The number of values to be estimated in the spectrum determines the size of a new array. I chose to use sixty values. The pixel filter values are separated in the array by the same incremental spacing. For a faster calculation the interpolation function array is converted into frequency space. The spatial points are treated as discrete delta functions. To generate a smooth continuous curve, the delta functions are convolved with Gaussian functions. The width of the Gaussian was set to the spacing between bands in order to avoid excess dip or rise between bands.

Interpolation function

The interpolation created for each pixel is described by three functions used to produce all colors for visual

display. These functions weight the interpolated spectrum by different amounts. They were designed to represent the entire spectrum equally, i.e. the weights sum to unity at each sample at every sample. For calculation, the weighting functions obviously must have the same number of samples as the interpolated spectrum. Each pixel in the super-visual image contains information from three channels. The value of each channel comes from the sum of the corresponding weighting function (red channel, red weighting function) multiplied by the interpolation function at each sample. The interpolation function is also by the green and blue weighting functions. The sum of all samples is the amplitude of the appropriate channel in the pixel. The sum value is sometimes out of scale for an 8-bit image so all the three channels are scaled to have their min and max be 0 and 255.

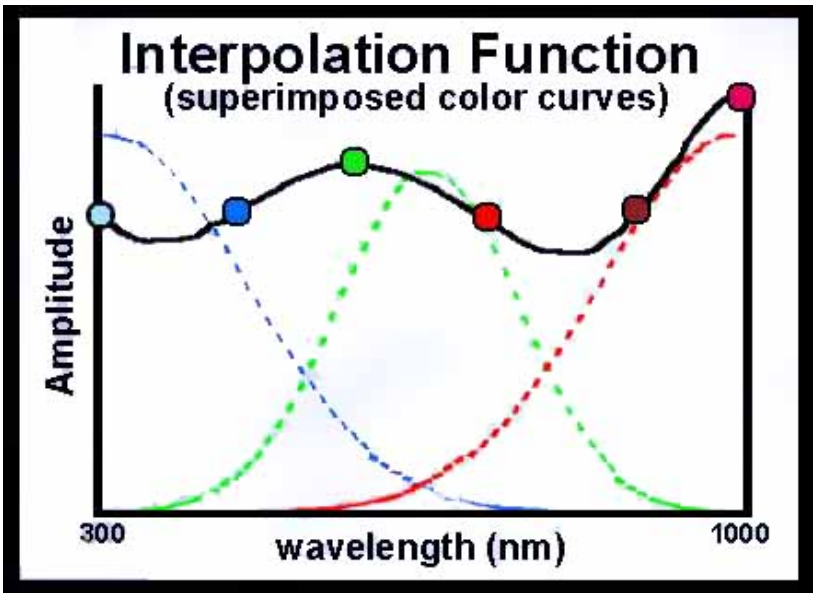


Fig. 4: Example of the interpolation function with the three color weighting functions

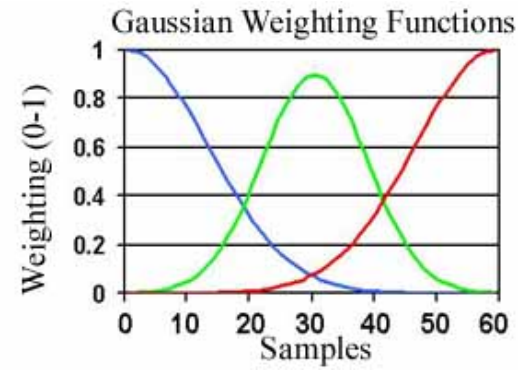


Fig. 5: Gaussian weighting functions used in calculation to generate RGB image

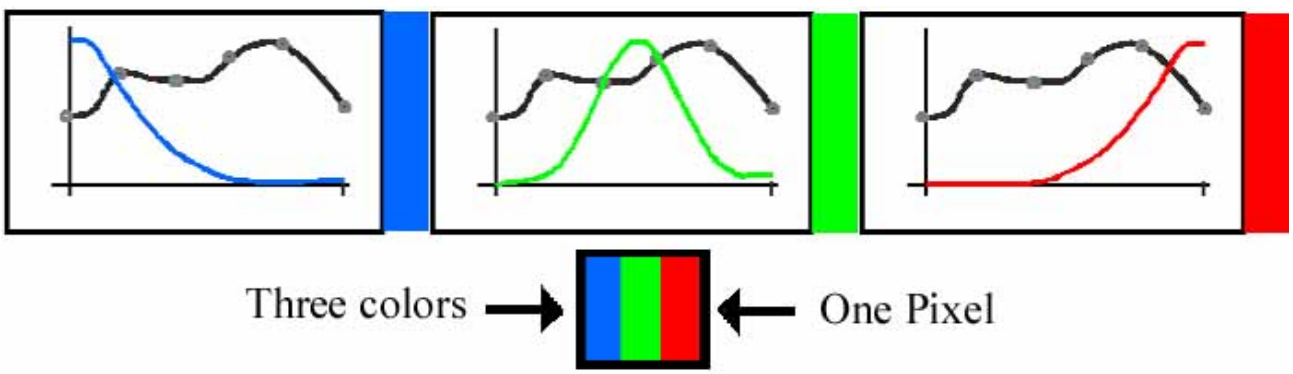


Fig. 6: Each RGB color image is calculated separatly and then combined

Flat fielding

Not all input data is in a form that is useable for a useful output. This was evident in the scroll test data which had two main problems. Algorithms for flat fielding and translation were developed to correct these problems. Some images contain a contrast from very dark to very light portions of the scene. The 'super-visual' algorithm was not useful for perceiving details contained in the darkest portions of the scene. Flat fielding eliminates slowly varying changes in brightness. To achieve an "uniform" scene, the original image is blurred to reveal only the coarse variation in brightness. Calculating the blurred image in the spatial domain is time consuming. Frequency-domain processing is much quicker. This lowpass filter kernal is a 5x5 array of units is zero valued every where except a 5x5 area in the center equal to one. Depending on the subject and the effect of the lighting, the center area could be a 3x3 or 5x5 with a 3x3 area producing the most blurred image. Fourier transforms have low-slowly varying frequencies in the center of the image so when multiplied by a lowpass filter all the low frequencies are obviously passed.

Flat Fielding

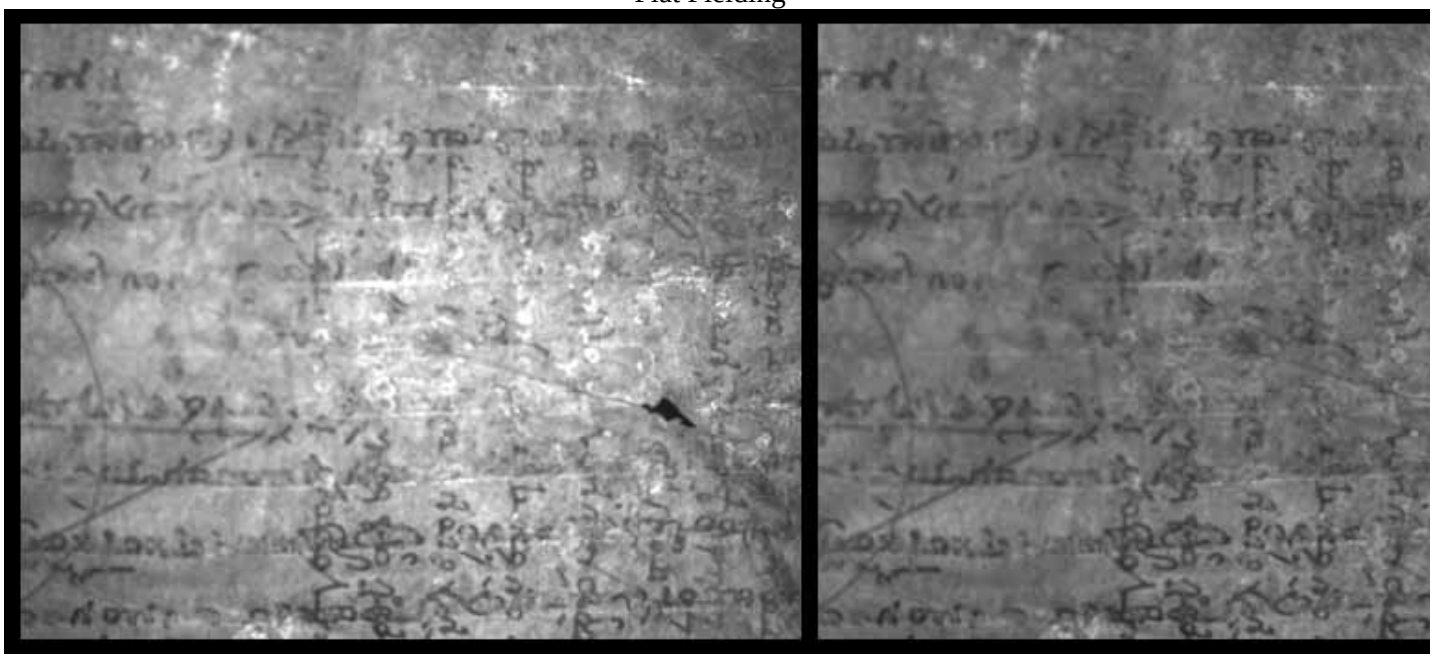


Fig. 7: Images of test data without Flat Fielding (left), and with Flat Fielding (right)

Translation

Sometimes the six input images are not all registered, i.e., the pixels do not "line up". This is usually due to movement of the camera while filters are changed. This means that the camera had a different field of view in at least one image. The result is an unregistered or "ghost" image can be seen in the 'super-visual' image. Many techniques have been developed to automatically register images. The speed of the algorithm usually depends on the desired accuracy. It is much faster to process a translation in frequency space. The shifted image can be thought of as the original image convolved by some delta function shifted from the origin (center of image). (translation formulas written out) If the images were identical, we could simply divide the shifted image by the unshifted to leave the delta function. All six images have the same main features but are certainly not identical. When trying to divide two different images in frequency space, the result looks like noise and has no distinguishable delta function. The differences between the images get highlighted as well as the shift and the two delta functions are indistinguishable from each other, which is ineffective for the purpose of this project. The effective technique treated the first image as a "reference" and compare the other five to it. Multiplying the reference image by the complex conjugate of the shifted image results in information about magnitude and phase. The phase contains the distance and direction of the shift. Since both images are similar their main difference comes from the phase. The shift in phase does not appear as a "delta" but rather as the peak of a function with finite width. The resulting function containing the shift is calculated by multiplying the magnitude terms, the image phases almost cancel out and the remaining delta shifts the multiplied magnitudes. This is not entirely accurate but is an effective indicator of the shift and is enough to get a useful result.

Steps required to translate image #2 to closely match the position of reference image #1

Task: Find x_o such that $g_2(x) = f_2(x) + \delta(x - x_o)$ most closely approximates $f_1[x]$ (reference image #1)

Ideal Case: The two images are identical but for unknown translation x_o

$$\mathcal{F}\{g_2[x]\} = G_2[\xi] = F_2[\xi]e^{-2\pi i x_0 \xi} = (|F_2[\xi]|e^{i\Phi\{F_2[\xi]\}})e^{-2\pi i x_0 \xi}$$

$$\text{If } |F_1[\xi]| = |F_2[\xi]| \text{ and } \Phi\{F_1[\xi]\} = \Phi\{F_2[\xi]\}$$

$$\text{then } \mathcal{F}\{f_1[x]\} = F_1[\xi] = |F_1[\xi]|e^{+i\Phi\{F_1[\xi]\}}$$

$$\text{So that } \frac{G_2[\xi]}{F_1[\xi]} = \frac{|F_2[\xi]|e^{+i\Phi\{F_2[\xi]\}} \cdot e^{-2\pi i x_0 \xi}}{|F_1[\xi]|e^{+i\Phi\{F_1[\xi]\}}} = e^{-2\pi i x_0 \xi}$$

$$\text{Essentially: } \mathcal{F}^{-1}\left\{\frac{G_2[\xi]}{F_1[\xi]}\right\} = \delta[x - x_0], \text{ an unknown translation}$$

Realistic Case: The image to be registered differs from the reference image, so that:

$$\text{If } |F_1[\xi]| \neq |F_2[\xi]| \text{ or } \Phi\{F_1[\xi]\} \neq \Phi\{F_2[\xi]\}$$

$$G_2[\xi] \cdot F\dagger[\xi] = (F_2[\xi] \cdot e^{+i\Phi\{F_2[x]\}})\ e^{-2\pi i x_0 \xi}\ (F_1[\xi]e^{+i\Phi\{F_1[\xi]\}})$$

$$\text{So that } \mathcal{F}^{-1}\{G_2[\xi] \cdot F\dagger[\xi]\} = g_2[x] \bigstar f_1[x] \simeq \delta[x - x_0]$$

* The star indicates an auto-correlation *

The end terms combine and the middle term $e^{-2\pi i x_0 \xi}$ indicates the shifted position of $g_2[x]$ (image # 2) compared with $f_1[x]$ (the reference image # 1)

Results

This research succeeded in presenting important spectral details out of the visual range in the final image. This was the first and most important step toward useful results. Success of this first goal indicates that uses of the final image can be explored. The scroll test data contains high contrast text as well as subtler (but equally important) image features. This test data worked well to demonstrate where details are present in the final image. The nature of the test data made evaluation of the result fast and easy. It was easy to see if important details were visible from all 6 bands.

The first algorithm was written in the space domain using loops. This required the program to process multiple time-consuming computations at each pixel individually. It took approximately 50 minutes to process a series of six 1200 x 1200 images. Without changing the algorithm, I rewrote similar code to handle everything in the frequency domain, which reduced the computation time to about 15 minutes or less for images of the same-size.

Details from bands outside of the visual range were visible in the final image. Future modifications may serve to better understand what algorithms are most effective in creating super-visual images. The weighting functions were created to separate the spectral range into three distinct groups. It was thought to be important to have each wavelength equally represented so that the sum of the three weighting functions added up equally across the spectral range. The sum of the Gaussian functions used in the weighting function algorithm closely approximated this result. There may be advantages to an unequal sum, but they were not explored. Linear weighting functions were also applied to the algorithm. There was little to no perceptible difference between 'super-visual' images that used Gaussian or linear weighting functions. Upon closer examination and with quantitative image measurements this should prove not be entirely true. The construction of weighting functions determines the output and so whether two different weighting functions are perceptibly different or not they do have different values at each pixel. These differences would be perceived if the difference were sufficiently large.

The 'super-visual' algorithm is not designed to be radiometrically accurate. Each spectral band is spaced equally in the algorithm and does not correspond to its real spectral spacing. If radiometric accuracy required bands to be spaced in proportion to the true spectrum, the 'super-visual' image would look different. To maintain proportionality, the interpolation function would certainly be needed. Potencially more accurate interpolators can be found in (Castleman, 1996)
and could be tested but would most likely take longer to process. Gaussian functions with different widths would be required to create the interpolation function. The weighting functions could stay the same. Without an interpolation function to fill in the gaps between bands, the weighting functions would be biased towards certain bands. This means that the weighting functions would incorporate an unequal number of bands. Those bands spaced further from the others would get weighted more thus displayed with greater amplitude. Since this algorithm is designed for human perception and we are not adapted to perceiving or understanding a scene with a wide spectral range; radiometric or spectral accuracy should not be an important issue for most applications.

Visual versus 'Super-Visual' image

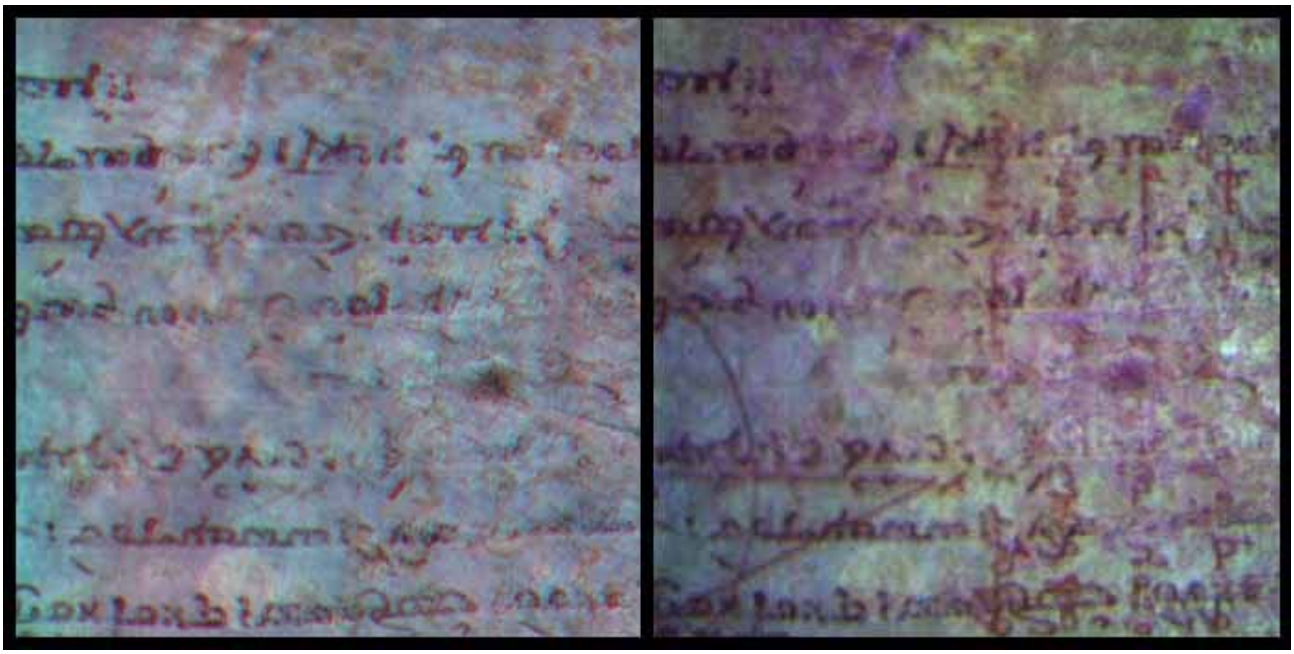


Fig. 8: Image of 'visual' color image (left), and 'Super-Visual' color image (right) with added details from additional bands

Discussion

The success of a 'super-visual' image is determined by its usefulness, which is based on the perception of details otherwise imperceptible in a typical RGB color image. If the algorithm displays those details in a way that is easy to interpret, then the 'super-visual' image is useful. In order to test its usefulness the 'super-visual' image is compared to each individual band confirming that important detail was not lost during processing. Fortunately a useful 'super-visual' image doesn't need to be compared with any bands; it contains all information (details) contained in the processed bands. The ease of interpretation is subject dependent and relies on knowledge of the subject. Since the 'super-visual' image isn't radiometrically accurate colors don't give much of an indication of the wide band spectrum. Without knowledge of the subject the colors generated may seem distracting. False color in the 'super-visual' image may give some indication of band amplitude though. In the case of the test data, red indicated higher infrared amplitude. Comparing the 'super-visual' image with an RGB image of a subject gave a helpful indication what spectral regions contained certain information.

Conclusions

This research addressed a need for data compression. The desired result was to be a faster and easier evaluation of a subject containing important details in a wide spectral range. The effectiveness of the compression method is demonstrated by the visibility of important characters in the Scroll test data from bands outside of the visible spectral range. The constructed algorithms successfully compressed sampled wide-band electromagnetic radiation data within the visible range. Important image details were present within a final color image indicating that this method generally yields useful results. Other methods may yield useful results as well and further development and testing of the compression method developed in this project may enhance the results.

Table of Contents

Back (Copyright) Forward (IDL code)

References:

Chavez, P.S., Jr., Sides, S.C., Anderson, J.A., "Comparison of Three Different Methods to Merge Multiresolution and Multispectral Data: Landsat TM and SPOT Panchromatic." Photogrammetric Engineering and Remote Sensing, 57, 3(March), 295-303, 1991.

Castleman, K., "Digital Image Processing", Prentice Hall, Upper Saddle River, 1996.

Gross, H.N., "An Image Fusion Algorithm for Specially Enhancing Spectral Mixture Maps" Dissertation, RIT, 1996

Munechika, C.K. "Merging Panchromatic and Multispectral Images for Enhanced Image Analysis." Master's Thesis, RIT, 1990

Schott, J.R., "Remote Sensing: The image chain approach", Oxford University press, New York, 1997.

Dead Sea Scrolls. URL: http://www.usc.edu/dept/LAS/wsrp/educational_site/dead_sea_scrolls/ Ellis Horowitz, Marilyn Lundberg, Jay Weaver, and Bruce Zuckerman. West Semitic Research Project, The University of Southern California. 1997-2000.

Table of Contents

Back (Thesis)

Rendering Multispectral data as useful 'Super-Visual' images

Matthew Gypson

IDL code for compression algorithm

Code (Flat Fielding and Translation)
Code (Flat Fielding, No Translation)
Code (No Flat Fielding, No Translation)

Back (Table of Contents) Forward (references)

```
;Matthew Gypson
;Senior Research
;MULTISPECTRAL DATA INTO USEFUL 'SUPER-VISUAL' IMAGES
;Compression method: takes the corresponding pixel on
;6 bands of multispectral data and by using
;gaussian curves, estimating human response to the spectrum.
;Red, green and blue images are created
;and combined into a full color image.
;Flat Fielding
;Translation
pro compression
;read in 6 images
text=dialog_pickfile(/read)
print,text
image1= read_tiff (text)
help,image1
text=dialog_pickfile(/read)
print,text
image2= read_tiff(text)
help,image2
text=dialog_pickfile(/read)
print,text
image3= read_tiff(text)
help,image3
text=dialog_pickfile(/read)
print,text
image4= read_tiff(text)
help,image4
text=dialog_pickfile(/read)
print,text
image5= read_tiff(text)
help,image5
```

```
text=dialog_pickfile(/read)

print,text

image6= read_tiff(text)

;input image size

imagesize= size(image1)

imagesizex= imagesize[1]

imagesizey= imagesize[2]

xsize= imagesizex

ysize= imagesizey

;;;

;TRANSLATION

;need to take first image as reference image, take its Fourier transform

; take the Fourier transform of each other image and divide into reference image

; take the inverse Fourier transform, find the peak by finding the Max pixels
location move the

; image appropriate amount and then repeat for each image

hxsize = xsize/2

hysize = ysize/2

hxsize = ROUND(hxsize)

hysize = ROUND(hysize)

;take fourier transform of image

Fimage1 = SHIFT(FFT(SHIFT(image1, hxsize, hysize)), -hxsize, -hysize)

Fimage_us2 = SHIFT(FFT(SHIFT(image2, hxsize, hysize)), -hxsize, -hysize)

Fimage_us2 = CONJ(Fimage_us2)

;Dividing Fourier transform of a reference image and unregistered image

FMoved2 = Fimage1*Fimage_us2

FMoved2 = Fmoved2/(ABS(Fimage1))

;Fourier transform of division to reveal location of shifted delta function

Moved2 = SHIFT(FFT(SHIFT(FMoved2, hxsize, hysize), 1), -hxsize, -hysize)

Real_Moved2 = FLOAT(Moved2)

G = MAX(Real_Moved2, I)

IX = I MOD xsize

IY = I/ysize
```

```
driftx = hxsize - IX
drifty = hysize - IY
if driftx GT 0 then signx = -1 else signx = 1
if drifty GT 0 then signy = -1 else signy = 1
image2 = SHIFT(image2, signx*driftx, signy*drifty)
Fimage_us3 = SHIFT(FFT(SHIFT(image3, hxsize, hysize)), -hxsize, -hysize)
Fimage_us3 = CONJ(Fimage_us3)
FMoved3 = Fimage1*Fimage_us3
FMoved3 = Fmoved3/(ABS(Fimage1))
Moved3 = SHIFT(FFT(SHIFT(FMoved3, hxsize, hysize), 1), -hxsize, -hysize)
Real_Moved3 = FLOAT(Moved3)
G = MAX(Real_Moved3, I)
IX = I MOD xsize
IY = I/ysize
driftx = hxsize - IX
drifty = hysize - IY
if driftx GT 0 then signx = -1 else signx = 1
if drifty GT 0 then signy = -1 else signy = 1
image3 = SHIFT(image3, signx*driftx, signy*drifty)
Fimage_us4 = SHIFT(FFT(SHIFT(image4, hxsize, hysize)), -hxsize, -hysize)
Fimage_us4 = CONJ(Fimage_us4)
FMoved4 = Fimage1*Fimage_us4
FMoved4 = Fmoved4/(ABS(Fimage1))
Moved4 = SHIFT(FFT(SHIFT(FMoved4, hxsize, hysize), 1), -hxsize, -hysize)
Real_Moved4 = FLOAT(Moved4)
G = MAX(Real_Moved4, I)
IX = I MOD xsize
IY = I/ysize
driftx = hxsize - IX
drifty = hysize - IY
if driftx GT 0 then signx = -1 else signx = 1
if drifty GT 0 then signy = -1 else signy = 1
image4 = SHIFT(image4, signx*driftx, signy*drifty)
```

```
Fimage_us5 = SHIFT(FFT(SHIFT(image5, hxsize, hysize)), -hxsize, -hysize)
Fimage_us5 = CONJ(Fimage_us5)
FMoved5 = Fimage1*Fimage_us5
FMoved5 = Fmoved5/(ABS(Fimage1))
Moved5 = SHIFT(FFT(SHIFT(FMoved5, hxsize, hysize), 1), -hxsize, -hysize)
Real_Moved5 = FLOAT(Moved5)
G = MAX(Real_Moved5, I)
IX = I MOD xsize
IY = I/ysize
driftx = hxsize - IX
drifty = hysize - IY
if driftx GT 0 then signx = -1 else signx = 1
if drifty GT 0 then signy = -1 else signy = 1
image5 = SHIFT(image5, signx*driftx, signy*drifty)
Fimage_us6 = SHIFT(FFT(SHIFT(image6, hxsize, hysize)), -hxsize, -hysize)
Fimage_us6 = CONJ(Fimage_us6)
FMoved6 = Fimage1*Fimage_us6
FMoved6 = Fmoved6/(ABS(Fimage1))
Moved6 = SHIFT(FFT(SHIFT(FMoved6, hxsize, hysize), 1), -hxsize, -hysize)
Real_Moved6 = FLOAT(Moved6)
G = MAX(Real_Moved6, I)
IX = I MOD xsize
IY = I/ysize
driftx = hxsize - IX
drifty = hysize - IY
if driftx GT 0 then signx = -1 else signx = 1
if drifty GT 0 then signy = -1 else signy = 1
image6 = SHIFT(image6, signx*driftx, signy*drifty)
;shifted fourier transforms
Fimage2 = SHIFT(FFT(SHIFT(image2, hxsize, hysize)), -hxsize, -hysize)
Fimage3 = SHIFT(FFT(SHIFT(image3, hxsize, hysize)), -hxsize, -hysize)
Fimage4 = SHIFT(FFT(SHIFT(image4, hxsize, hysize)), -hxsize, -hysize)
Fimage5 = SHIFT(FFT(SHIFT(image5, hxsize, hysize)), -hxsize, -hysize)
```

```
Fimage6 = SHIFT(FFT(SHIFT(image6, hxsize, hysize)), -hxsize, -hysize)

;BLURRING

;need to create a filter that cuts out high frequency

;need to put images into frequency space and multiply by filter, inverse Fourier
transform


exsize = 2

eysize = 2

exsize = ROUND(exsize)

eysize = ROUND(eysize)

;create filter

filter = fltarr(xsize, ysize)

filter[hxsize - exsize:hxsize + exsize, hysize - eysize:hysize + eysize] = 1


;blur image

FBlur1 = filter* Fimage1

Blur1 = SHIFT(FFT(SHIFT(FBlur1, hxsize, hysize), 1), -hxsize, -hysize)

FBlur2 = filter* Fimage2

Blur2 = SHIFT(FFT(SHIFT(FBlur2, hxsize, hysize), 1), -hxsize, -hysize)

FBlur3 = filter* Fimage3

Blur3 = SHIFT(FFT(SHIFT(FBlur3, hxsize, hysize), 1), -hxsize, -hysize)

FBlur4 = filter* Fimage4

Blur4 = SHIFT(FFT(SHIFT(FBlur4, hxsize, hysize), 1), -hxsize, -hysize)

FBlur5 = filter* Fimage5

Blur5 = SHIFT(FFT(SHIFT(FBlur5, hxsize, hysize), 1), -hxsize, -hysize)

FBlur6 = filter* Fimage6

Blur6 = SHIFT(FFT(SHIFT(FBlur6, hxsize, hysize), 1), -hxsize, -hysize)

;;;;

;Image is divided by the blurred image

;making sure there are no zero values

Blur1 = Blur1 + 0.0001

FFimage1 = (image1/Blur1)

Blur2 = Blur2 + 0.0001
```

```
FFimage2 = (image2/Blur2)
Blur3 = Blur3 + 0.0001
FFimage3 = (image3/Blur3)
Blur4 = Blur4 + 0.0001
FFimage4 = (image4/Blur4)
Blur5 = Blur5 + 0.0001
FFimage5 = (image3/Blur5)
Blur6 = Blur6 + 0.0001
FFimage6 = (image6/Blur6)
;scaleing flat fielded image1 0 to 255
max_B = MAX(FFimage1)
min_B = MIN(FFimage1)
FFimage1 = 256 * ((FFimage1[*,*]- min_B)/(max_B - min_B))
;scaleing flat fielded image2
max_B = MAX(FFimage2)
min_B = MIN(FFimage2)
FFimage2 = 256 * ((FFimage2[*,*]- min_B)/(max_B - min_B))
;scaleing flat fielded image3
max_B = MAX(FFimage3)
min_B = MIN(FFimage3)
FFimage3 = 256 * ((FFimage3[*,*]- min_B)/(max_B - min_B))
;scaleing flat fielded image4
max_B = MAX(FFimage4)
min_B = MIN(FFimage4)
FFimage4 = 256 * ((FFimage4[*,*]- min_B)/(max_B - min_B))
;scaleing flat fielded image5
max_B = MAX(FFimage5)
min_B = MIN(FFimage5)
FFimage5 = 256 * ((FFimage5[*,*]- min_B)/(max_B - min_B))
;scaleing flat fielded image6
max_B = MAX(FFimage6)
min_B = MIN(FFimage6)
FFimage6 = 256 * ((FFimage6[*,*]- min_B)/(max_B - min_B))
```

```
;RGB functions are imported
close, 1
close, 2
close, 3
OPENR, 1, '/cis/ugrad/mlg6909/SeniorResearch/rgb/rgb_b.txt'
RGBB= FLTARR(60)
READF, 1, RGBB
OPENR, 2, '/cis/ugrad/mlg6909/SeniorResearch/rgb/rgb_g.txt'
RGBG= FLTARR(60)
READF, 2, RGBG
OPENR, 3, '/cis/ugrad/mlg6909/SeniorResearch/rgb/rgb_r.txt'
RGBR= FLTARR(60)
READF, 3, RGBR
;Blue, Green and Red arrays created
B = fltarr(imagesizex,imagesizey)
G = fltarr(imagesizex,imagesizey)
R = fltarr(imagesizex,imagesizey)
intpl = fltarr(60)
intpl1 = fltarr(72)
delt = fltarr(72)
delt1 = fltarr(72)
k = 14.0
p = indgen(72)-36
gaus = (exp(-!pi*(p/k)^2))
Fqgaus = FFT(gaus)
for x = 0, imagesizex-1 do begin ;each pixel x
for y = 0, imagesizey-1 do begin ;each pixel y
;takes the value for each location
band1=image1[x,y]
band2=image2[x,y]
band3=image3[x,y]
band4=image4[x,y]
```

```
band5=image5[x,y]

band6=image6[x,y]

;for a frequency range of 300-1000 w = wavelength

delt[6] = band4

;*1.563-10

delt[18] = band5

;-10

delt[30] = band6

;*1.324-10

delt[42] = band1

;*1.891-10

delt[54] = band2

;*1.763-10

delt[66] = band3

;*1.688-10

Fqdelt = FFT(delt)

intpl1 = (FFT(Fqdelt*Fqgaus, /inverse)) * 75

;chops off the extra part of the gaus function

intpl = intpl1[6:66]

;multiply the interpolation by the XYZ curves for a R G and B value

prod1= RGBB * intpl

prod2= RGBG * intpl

prod3= RGBR * intpl

;evaluate the function at three locations and store that value in corresponding
red, green and blue images

B[x,y]= Total(prod1)

G[x,y]= Total(prod2)

R[x,y]= Total(prod3)

endfor

endfor

;get value from each image at the same pixel location

;create an array of data values

;interpolate the data values
```

```
;result = INTERPOLATE(P,X,Y,Z)

;sample new interpolated data at three locations corresponding to red, green and
blue

;an average may want to be taken over sampled area

;the resulting three values can be stored in corresponding red, green and blue
images



;scaling from 0 to 255

max_B = MAX(B)

min_B = MIN(B)

B = 256 * ((B[*,*]- min_B)/(max_B - min_B))

max_G = MAX(G)

min_G = MIN(G)

G = 256 * ((G[*,*]- min_G)/(max_G - min_G))

max_R = MAX(R)

min_R = MIN(R)

R = 256 * ((R[*,*]- min_R)/(max_R - min_R))

window, 0

tvscl, B

window, 1

tvscl, G

window, 2

tvscl, R

;WRITE_TIFF: Image array argument required.

WRITE_TIFF, '/cis/ugrad/mlg6909/SeniorResearch/rgb/TFFfinal_1748_1786.tif',
PLANARCONFIG= 2, BLUE = B, GREEN = G, RED= R



end
```

```
;Matthew Gypson
;Senior Research
;MULTISPECTRAL DATA INTO USEFUL 'SUPER-VISUAL' IMAGES
;Compression method: takes the corresponding pixel on
;6 bands of multispectral data and by using
;gaussian curves, estimates human response to the spectrum.
;Red, green and blue images are created
;and combined into a full color image using TIFF color format.
;Flat Fielding
;No Translations
pro compression
;read in 6 images
text=dialog_pickfile(/read)
print,text
image1= read_tiff (text)
help,image1
text=dialog_pickfile(/read)
print,text
image2= read_tiff(text)
help,image2
text=dialog_pickfile(/read)
print,text
image3= read_tiff(text)
help,image3
text=dialog_pickfile(/read)
print,text
image4= read_tiff(text)
help,image4
text=dialog_pickfile(/read)
print,text
```

```
image5= read_tiff(text)

help,image5

text=dialog_pickfile(/read)

print,text

image6= read_tiff(text)

;input image size

imagesize= size(image1)

imagesizex= imagesize[1]

imagesizey= imagesize[2]

xsize= imagesizex

ysize= imagesizey


hxsize = xsize/2

hysize = ysize/2

hxsize = ROUND(hxsize)

hysize = ROUND(hysize)

Fimage1 = SHIFT(FFT(SHIFT(image1, hxsize, hysize)), -hxsize, -hysize)

Fimage2 = SHIFT(FFT(SHIFT(image2, hxsize, hysize)), -hxsize, -hysize)

Fimage3 = SHIFT(FFT(SHIFT(image3, hxsize, hysize)), -hxsize, -hysize)

Fimage4 = SHIFT(FFT(SHIFT(image4, hxsize, hysize)), -hxsize, -hysize)

Fimage5 = SHIFT(FFT(SHIFT(image5, hxsize, hysize)), -hxsize, -hysize)

Fimage6 = SHIFT(FFT(SHIFT(image6, hxsize, hysize)), -hxsize, -hysize)

;BLURRING

;need to create a filter that cuts out high frequency

;need to put images into frequency space and multiply by filter, inverse Fourier
transform


exsize = 2

eysize = 2

exsize = ROUND(exsize)

eysize = ROUND(eysize)
```

```
;create filter
filter = fltarr(xsize, ysize)
filter[hxsize - exsize:hxsize + exsize, hysize - eysize:hysize + eysize] = 1

;blur image
FBlur1 = filter* Fimage1
Blur1 = SHIFT(FFT(SHIFT(FBlur1, hxsize, hysize), 1), -hxsize, -hysize)
FBlur2 = filter* Fimage2
Blur2 = SHIFT(FFT(SHIFT(FBlur2, hxsize, hysize), 1), -hxsize, -hysize)
FBlur3 = filter* Fimage3
Blur3 = SHIFT(FFT(SHIFT(FBlur3, hxsize, hysize), 1), -hxsize, -hysize)
FBlur4 = filter* Fimage4
Blur4 = SHIFT(FFT(SHIFT(FBlur4, hxsize, hysize), 1), -hxsize, -hysize)
FBlur5 = filter* Fimage5
Blur5 = SHIFT(FFT(SHIFT(FBlur5, hxsize, hysize), 1), -hxsize, -hysize)
FBlur6 = filter* Fimage6
Blur6 = SHIFT(FFT(SHIFT(FBlur6, hxsize, hysize), 1), -hxsize, -hysize)
;;;;
;Image is divided by the blurred image
;making sure there are no zero values
Blur1 = Blur1 + 0.0001
FFimage1 = (image1/Blur1)
Blur2 = Blur2 + 0.0001
FFimage2 = (image2/Blur2)
Blur3 = Blur3 + 0.0001
FFimage3 = (image3/Blur3)
Blur4 = Blur4 + 0.0001
FFimage4 = (image4/Blur4)
Blur5 = Blur5 + 0.0001
FFimage5 = (image3/Blur5)
Blur6 = Blur6 + 0.0001
```

```
FFimage6 = (image6/Blur6)
;scaleing flat fielded image1 0 to 255
max_B = MAX(FFimage1)
min_B = MIN(FFimage1)
FFimage1 = 256 * ((FFimage1[*,*]- min_B)/(max_B - min_B))
;scaleing flat fielded image2
max_B = MAX(FFimage2)
min_B = MIN(FFimage2)
FFimage2 = 256 * ((FFimage2[*,*]- min_B)/(max_B - min_B))
;scaleing flat fielded image3
max_B = MAX(FFimage3)
min_B = MIN(FFimage3)
FFimage3 = 256 * ((FFimage3[*,*]- min_B)/(max_B - min_B))
;scaleing flat fielded image4
max_B = MAX(FFimage4)
min_B = MIN(FFimage4)
FFimage4 = 256 * ((FFimage4[*,*]- min_B)/(max_B - min_B))
;scaleing flat fielded image5
max_B = MAX(FFimage5)
min_B = MIN(FFimage5)
FFimage5 = 256 * ((FFimage5[*,*]- min_B)/(max_B - min_B))
;scaleing flat fielded image6
max_B = MAX(FFimage6)
min_B = MIN(FFimage6)
FFimage6 = 256 * ((FFimage6[*,*]- min_B)/(max_B - min_B))

;RGB functions are imported
close, 1
close, 2
close, 3
OPENR, 1, '/cis/ugrad/mlg6909/SeniorResearch/rgb/rgb_b.txt'
```

```
RGBB= FLTARR(60)
READF, 1, RGBB
OPENR, 2, '/cis/ugrad/mlg6909/SeniorResearch/rgb/rgb_g.txt'
RGBG= FLTARR(60)
READF, 2, RGBG
OPENR, 3, '/cis/ugrad/mlg6909/SeniorResearch/rgb/rgb_r.txt'
RGBR= FLTARR(60)
READF, 3, RGBR
;Blue, Green and Red arrays created
B = fltarr(imagesizex,imagesizey)
G = fltarr(imagesizex,imagesizey)
R = fltarr(imagesizex,imagesizey)
intpl = fltarr(60)
intpl1 = fltarr(72)
delt = fltarr(72)
delt1 = fltarr(72)
k = 14.0
p = indgen(72)-36
gaus = (exp(-!pi*(p/k)^2))
Fqgaus = FFT(gaus)
for x = 0, imagesizex-1 do begin ;each pixel x
for y = 0, imagesizey-1 do begin ;each pixel y
;takes the value for each location
band1=image1[x,y]
band2=image2[x,y]
band3=image3[x,y]
band4=image4[x,y]
band5=image5[x,y]
band6=image6[x,y]
;for a frequency range of 300-1000 w = wavelength
delt[6] = band4
```

```
;*1.563-10

delt[18] = band5

;-10

delt[30] = band6

;*1.324-10

delt[42] = band1

;*1.891-10

delt[54] = band2

;*1.763-10

delt[66] = band3

;*1.688-10

Fqdelt = FFT(delt)

intpl1 = (FFT(Fqdelt*Fqgaus, /inverse)) * 75

;chops off the extra part of the gaus function

intpl = intpl1[6:66]

;multiply the interpolation by the XYZ curves for a R G and B value

prod1= RGBB * intpl

prod2= RGBG * intpl

prod3= RGBR * intpl

;evaluate the function at three locations and store that value in corresponding
red, green and blue images

B[x,y]= Total(prod1)

G[x,y]= Total(prod2)

R[x,y]= Total(prod3)

endfor

endfor

;get value from each image at the same pixel location

;create an array of data values

;interpolate the data values

;result = INTERPOLATE(P,X,Y,Z)

;sample new interpolated data at three locations corresponding to red, green and
blue
```

```
;an average may want to be taken over sampled area

;the resulting three values can be stored in corresponding red, green and blue
images



;scaling from 0 to 255

max_B = MAX(B)

min_B = MIN(B)

B = 256 * ((B[*,*]- min_B)/(max_B - min_B))

max_G = MAX(G)

min_G = MIN(G)

G = 256 * ((G[*,*]- min_G)/(max_G - min_G))

max_R = MAX(R)

min_R = MIN(R)

R = 256 * ((R[*,*]- min_R)/(max_R - min_R))

window, 0

tvscl, B

window, 1

tvscl, G

window, 2

tvscl, R

;WRITE_TIFF: Image array argument required.

WRITE_TIFF, '/cis/ugrad/mlg6909/SeniorResearch/rgb/FFfinal_1780_1789.tif',
PLANARCONFIG= 2, BLUE = B, GREEN = G, RED= R



end
```

```
;Matthew Gypson
;Senior Research
;Non translating Flat Fielding RGB algorithms
;MULTISPECTRAL DATA INTO USEFUL 'SUPER-VISUAL' IMAGES
;Compression method: takes the corresponding pixel on
;6 bands of multispectral data and by using
;gaussian curves, estimates human response to the spectrum
;Red, green and blue images are created
;and combined into a full color image.
;No Flat Fielding
;No Translations
pro compression
;read in 6 images
text=dialog_pickfile(/read)
print,text
image1= read_tiff (text)
help,image1
text=dialog_pickfile(/read)
print,text
image2= read_tiff(text)
help,image2
text=dialog_pickfile(/read)
print,text
image3= read_tiff(text)
help,image3
text=dialog_pickfile(/read)
print,text
image4= read_tiff(text)
help,image4
text=dialog_pickfile(/read)
```

```
print,text
image5= read_tiff(text)
help,image5
text=dialog_pickfile(/read)
print,text
image6= read_tiff(text)
;input image size
imagesize= size(image1)
imagesizex= imagesize[1]
imagesizey= imagesize[2]
xsize= imagesizex
ysize= imagesizey
hxsize = xsize/2
hysize = ysize/2
hxsize = ROUND(hxsize)
hysize = ROUND(hysize)

;RGB functions are imported
close, 1
close, 2
close, 3
OPENR, 1, '/cis/ugrad/mlg6909/SeniorResearch/rgb/rgb_b.txt'
RGBB= FLTARR(60)
READF, 1, RGBB
OPENR, 2, '/cis/ugrad/mlg6909/SeniorResearch/rgb/rgb_g.txt'
RGBG= FLTARR(60)
READF, 2, RGBG
OPENR, 3, '/cis/ugrad/mlg6909/SeniorResearch/rgb/rgb_r.txt'
RGBR= FLTARR(60)
READF, 3, RGBR
;Blue, Green and Red arrays created
```

```
B = fltarr(imagesizex,imagesizey)
G = fltarr(imagesizex,imagesizey)
R = fltarr(imagesizex,imagesizey)
intpl = fltarr(60)
intpl1 = fltarr(72)
delt = fltarr(72)
delt1 = fltarr(72)
k = 14.0
p = indgen(72)-36
gaus = (exp(-!pi*(p/k)^2))
Fqgaus = FFT(gaus)
for x = 0, imagesizex-1 do begin ;each pixel x
for y = 0, imagesizey-1 do begin ;each pixel y
;takes the value for each location
band1=image1[x,y]
band2=image2[x,y]
band3=image3[x,y]
band4=image4[x,y]
band5=image5[x,y]
band6=image6[x,y]
;for a frequency range of 300-1000 w = wavelength
delt[6] = band4
;*1.563-10
delt[18] = band5
;-10
delt[30] = band6
;*1.324-10
delt[42] = band1
;*1.891-10
delt[54] = band2
;*1.763-10
```

```
delt[66] = band3

;*1.688-10

Fqdelt = FFT(delt)

intpl1 = (FFT(Fqdelt*Fqgaus, /inverse)) * 75

;chops off the extra part of the gaus function

intpl = intpl1[6:66]

;multiply the interpolation by the XYZ curves for a R G and B value

prod1= RGBB * intpl

prod2= RGBG * intpl

prod3= RGBR * intpl

;evaluate the function at three locations and store that value in corresponding
red, green and blue images

B[x,y]= Total(prod1)

G[x,y]= Total(prod2)

R[x,y]= Total(prod3)

endfor

endfor

;get value from each image at the same pixel location

;create an array of data values

;interpolate the data values

;result = INTERPOLATE(P,X,Y,Z)

;sample new interpolated data at three locations corresponding to red, green and
blue

;an average may want to be taken over sampled area

;the resulting three values can be stored in corresponding red, green and blue
images



;scaling from 0 to 255

max_B = MAX(B)

min_B = MIN(B)

B = 256 * ((B[*,*]- min_B)/(max_B - min_B))

max_G = MAX(G)
```

```
min_G = MIN(G)

G = 256 * ((G[*,*]- min_G)/(max_G - min_G))

max_R = MAX(R)

min_R = MIN(R)

R = 256 * ((R[*,*]- min_R)/(max_R - min_R))

window, 0

tvscl, B

window, 1

tvscl, G

window, 2

tvscl, R

;WRITE_TIFF: Image array argument required.

WRITE_TIFF, '/cis/ugrad/mlg6909/SeniorResearch/Flowers_Finals_i.tiff',
PLANARCONFIG= 2, BLUE = B, GREEN = G, RED= R


end
```