Tetrapy: A Python Package for Cleaning and Preprocessing Tetracam Multispectral Imagery

University of Minnesota
Driven to Discover℠

College of Food, Agriculture, and Natural Resource Sciences

Tyler Nigon[a], Ce Yang[ab], David Mulla[a], Daniel Kaiser[a]

[a] Department of Soil, Water, and Climate
[b] Department of Bioproducts and Biosystems Engineering

Abstract

Aerial multispectral imagery can provide valuable information for precision agriculture applications, can aid in agricultural decision-making, and provide powerful research information. However, it is often necessary to perform some level of data cleaning and image preprocessing before the data are quantitatively objective and useful for research purposes. The Tetracam Mini-Multiple Camera Array (MCA) System is a popular multispectral imager, but there are preprocessing issues that are not adequately addressed using Tetracam's included software application (*i.e.,* PixelWrench2; Tetracam Inc., Chatsworth, CA, USA). The Python package that was developed is called Tetrapy, but prior to its use, RAW images must first be converted to multipage TIFs using PixelWrench2 because of the proprietary format of Tetracam's RAW image files. Tetrapy has the following batch preprocessing functionality for multipage TIFs in the input directory: *i)* converts images from 8-bit to 10-bit pixel depth if images were captured using the 10-bit RAW setting, *ii)* writes all metadata to an independent header file (i.e., .hdr) according to ENVI standards, *iii)* allows user to export images to .dat data format as band interleaved by line, band interleaved by pixel, or band sequential, *iv)* performs flat field vignetting correction, *v)* performs radial distortion correction, and *vi)* has some functions to aid in improving band co-registration. Tetrapy greatly reduces the time that Tetracam users spend in preprocessing their imagery while maintaining data integrity and original image metadata. Tetrapy is licensed under the GNU General Public License.

Introduction

The Tetracam Multiple Camera Array (MCA) systems (Tetracam Inc., Chatsworth, CA, USA) contain either 4, 6, or 12 factory-aligned monochromatic image sensors (depending on the model). Each imager is equipped with an interchangeable band-pass filter that only allows light energy to be transmitted within a specific wavelength range to be captured by its CMOS sensor. The filter configuration and characteristics are provided in **Table 1**. See **Table 2** for further sensor and camera specifications.

Table 1: Filter configuration for the Mini-MCA 6.

Band	Center Wavelength (nm)	FWHM	Transmission (%)
b_{490}	490.0 +/- 3.0	10.0 +/- 2.0	55
b_{550}	550.0 +/- 3.0	10.0 +/- 2.0	55
b_{680}	680.0 +/- 3.0	10.0 +/- 2.0	55
b_{720}	720.0 +/- 3.0	10.0 +/- 2.0	55
b_{800}	800.0 +/- 3.0	10.0 +/- 2.0	50
b_{900}	900.0 +/- 3.0	10.0 +/- 2.0	50

*FWHM = Full width at half maximum

Table 2: Sensor and camera specifications for the Mini-MCA 6.

Specification/Feature	Description/Value
Sensor type	CMOS
Sensor size (mm)	5.32 (h) x 6.66 (w)
Sensor resolution (pixels)	1024 x 1280 (1.3 MP)
Pixel size (µm)	5.2
Lens focal length (mm)	9.6
Lens aperture	f/3.2
Horizontal FOV (degrees)	38.26
Vertical FOV (degrees)	30.97
Radiometric resolution (bit depth)	8- or 10-bit
Exposure time (ms)	Auto or 1 – 20

Objectives

The overall objective of this work was to develop a Python (2.7) module, Tetrapy, for performing fundamental image preprocessing tasks on Tetracam MCA images captured via aerial remote sensing.

The specific objectives were to:

1. Decode images from 8-bit to 10-bit pixel depth
2. Copy all metadata to an independent header file (*i.e.*, .hdr) according to ENVI standards
3. Perform flat field vignetting correction
4. Perform radial distortion correction
5. Provide some functionality in improving band co-registration (not covered in this poster)

Following image preprocessing, imagery can be analyzed in a more quantitatively objective manner and will be more suitable for research purposes. There was a focus on performing preprocessing tasks on all images within a folder directory (*i.e.,* batch processing) with minimal input from the user. It should be noted that although this module was developed for preprocessing Tetracam MCA imagery, it could very well be updated to support multispectral images from other camera sensors.

Methods

Over the past few years, an octacopter has been used as a platform to mount a Tetracam MCA and capture hundreds of aerial multispectral images for our research groups (**Figures 1** and **2**). Most images are over corn nitrogen trials, but include other crop fertility trials as well.

Figure 1: The octacopter platform used to capture our aerial multispectral images.

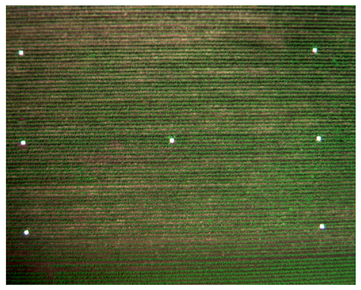

Figure 2: Original image over a corn nitrogen trial in Waseca, MN. Note the artifacts of vignetting and radial distortion.

The `Preprocess` class of Tetrapy contains methods that initialize parameters and perform preprocessing tasks on image sets. This section will be used to provide motivation for the available functionality in the `Preprocess` class and generally explain what the class methods accomplish and how they do so.

Decoding to 10-bit Pixel Depth

The PixelWrench2 software enables conversion of DCM or RAW images into single or multipage TIF or JPG formats. However, when images are captured at the 10-bit pixel depth, PixelWrench2 actually encodes the 10-bit data using three 8-bit bands when converting to the multipage TIF format. This requires the pixel values (*i.e.*, digital number; DN) from bands two and three to be decoded from the 8-bit multipage TIFs according to **Equation 1** for the 10-bit dynamic range to be realized. PixelWrench2 has a Tif Export Tool (currently version 1.0.1.1) to perform this decoding, but there are two issues with the tool: during decoding, the 10-bit data are stretched to 16 bits, which seems unnecessary, and more notably, the tool does not allow batch processing.

Equation 1: $DN_{10\text{-}bit} = (4 * DN_{b2}) + (DN_{b2} - DN_{b3})$

Vignetting Correction

Definition:
Vignetting is the effect of radial fall-off of light intensity from the center towards the periphery of an image (see **Figure 2**).

Vignetting in remotely sensed images is not ideal because it results in a misrepresentation of reflected light energy for pixels across the image **(Figure 3).** PixelWrench2 does have functionality for image vignetting correction, but the user is responsible for determining the vignetting correction coefficients for each sensor. As an alternative to the approach PixelWrench2 provides, Tetrapy allows the user to implement an image-based correction method to minimize the effect of vignetting. This method derives look-up tables (LUT) for each band from a flat field target. The LUTs are composed of correction coefficients for each image pixel, and can be used to eliminate much of the observed vignetting that exists for each spectral band/sensor.

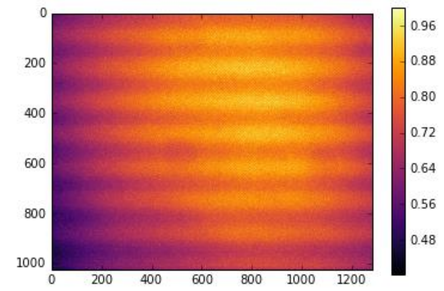

Figure 3: Relative observed image vignetting (b_{550}). This is an image of a flat field target that should be evenly illuminated.

Ideally, sensor vignetting is characterized by capturing images of a flat field panel with Lambertian properties and constant spectral characteristics that is evenly illuminated using a completely diffuse light source. If the illumination source is not completely diffuse, the flat field panel may exhibit variability in how much light energy is reflected at different locations on the panel (*i.e.,* "hotspots" may exist).

A flat field was created using a 25 cm by 25 cm Spectralon panel illuminated using four 150W wide beam halogen bulbs mounted at an equal distance away from the Spectralon (**Figure 4a**). The bulbs utilized a diffusion lens to provide more even illumination, and were oriented so the center beam from each bulb was aimed approximately two-thirds the distance across the Spectralon.

Methods

This setup was found to produce the most homogeneous illumination of the Spectralon. The variability of relative light intensity across the flat field panel was characterized by measuring the reflectance using an ASD FieldSpec4 at several locations across the Spectralon (**Figure 4b**).

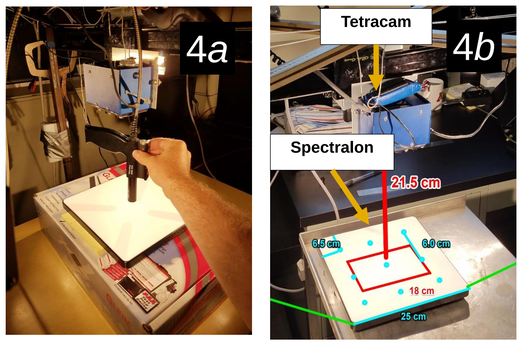

Figure 4: Setup for capturing flat field images. An ASD Fieldspec 4 was used to measure the reflectance (**Fig. 4*a***) at nine points on the Spectralon panel (labeled in **Fig. 4*b***).

Radial Distortion Correction

Definition:
Radial distortion is the symmetric deviation from a rectilinear projection that arises from the symmetry of the camera lens. A slight barrel distortion was observed in the images that were captured with our camera - straight lines appear to be curved inwards (**Figure 5**).

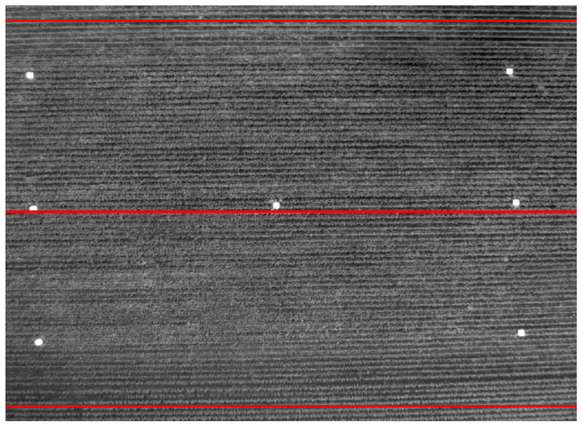

Figure 5: Aerial image of a corn field illustrating barrel distortion (note that corn rows appear curved inwards near the edges due to distortion).

Using Tetrapy

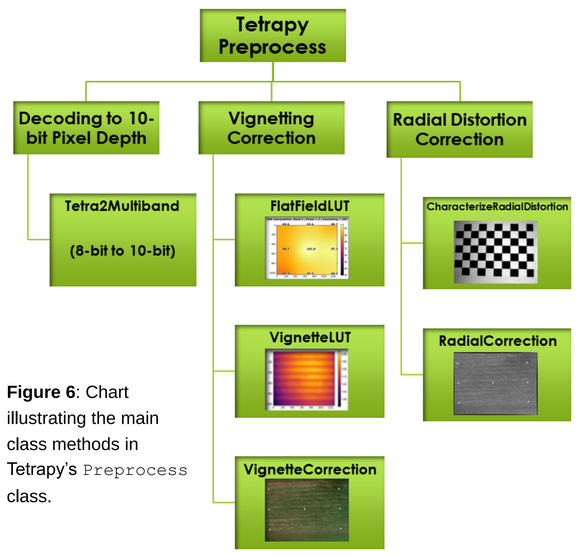

Figure 6: Chart illustrating the main class methods in Tetrapy's `Preprocess` class.

Within the `Preprocess` class, Tetrapy has class methods to achieve three main image preprocessing techniques:

1. Decode images so full 10-bit pixel depth is realized (according to **Equation 1**).
2. Remove the effect of vignetting across images
3. Remove the effect of radial distortion.

Figure 6 illustrates the relationship between these three preprocessing techniques and the main class methods that are used to achieve them within the `Preprocess` class of Tetrapy.

Results

The code for utilizing Tetrapy has been omitted from this poster for simplification. There are many optional parameters that can be used for each of the class methods, and note that the results presented here may change with different parameters. Please reference the Tetrapy documentation to see code, examples, and the optional parameters.

Vignetting Correction

To correct for the effect of vignetting using Tetrapy, the following steps should be performed:

1. Capture flat field images for each band using a setup similar to that shown in **Figure 4b**. Images should be saved to separate folders based on the band sensor that was centered over the flat field.
2. [**OPTIONAL**] Capture dark current images by covering the lenses and save them to a separate folder directory. A completely dark environment is recommended for this step.
3. [**OPTIONAL**] Use a spectrometer to measure the relative light intensity at several measured locations across the flat field (**Figures 4a** and **4b**) and save values in a *.csv file.
4. The `FlatFieldLUT` method (see **Figure 6**) utilizes an inverse distance weighting (IDW) approach to generate the measured light intensity that each pixel experienced when flat field images were captured (**Figure 7**):

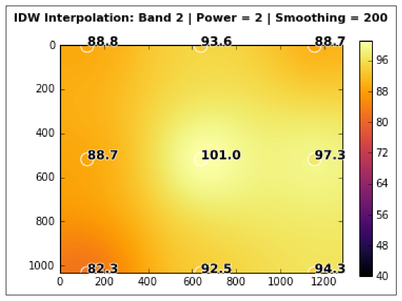

Figure 7: Variability of relative light intensity across the flat field panel generated using an inverse distance weighting approach based on measurements from nine points at different locations.

5. The `VignetteLUT` method (see **Figure 6**) calculates the lookup tables by consolidating the flat field images for each band (**Figure 3** is a vignette lookup table for the 550 nm band).
6. The `VignetteCorrection` method applies the vignetting correction for all images in a directory and saves corrected images in a new folder beside the original images (**Figure 8**).

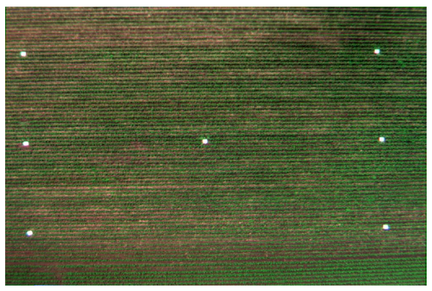

Figure 8: RGB image after vignetting correction.

Radial Distortion Correction

To correct the radial distortion, the following steps should be performed (note that specific details are omitted):

1. Capture and save several images with a chess board pattern to be used to characterize the apparent radial distortion (see **Figure 6**).
2. The `CharacterizeRadialDistortion` method characterizes the extent to which radial distortion exists in images:
3. The `RadialCorrection` method applies the distortion correction to all images in a directory and saves corrected images in a new folder beside the original images (**Figure 9**).

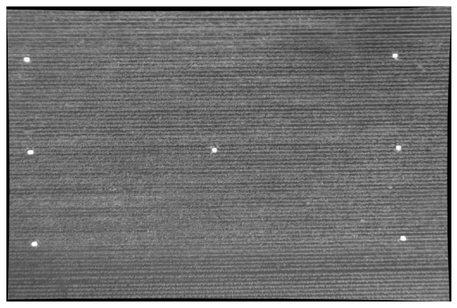

Figure 9: Image band (550 nm) after radial distortion correction.

For more information about Tetrapy, including installation requirements, optional parameters, and additional examples, please visit: https://tetrapy.readthedocs.org