A Novel Method of Development of Extinction Imagers Using Both a CCD and DMD for Efficient Determination of Atmospheric Optical Extinction Over a Large Range of Wavelengths.

Sumer Sao

PI: Larry Andrews, Ronald Phillips

Townes Institute Science and Technology Experimentation Facility, CREOL, College of Optics, KSC NASA

Overview

This project was based upon the work done by the Scripps Institution of Oceanography in partnership with the University of California San Diego, the Lincoln Laboratory at MIT, and the Naval Postgraduate School. The goal was to replicate the success of the previous extinction imagers created, and then continue development of an Infrared EI using a DMD for more efficient and cheaper processing.

Extinction Imagers are unique instruments that can measure absolute extinction over an extended path. There are several advantages to using Extinction Imagers which make them ideal for use in extreme environments, and motivate the further development of them. These reasons include the fact that it's single ended, which allows for extinction to be measured over a path where an instrument can't be put at the end of the path. Extinction imagers are passive systems, meaning they don't use lasers. Thus, it is both safer and easier to operate since alignment of beams isn't an issue. Extinction imagers are also robust and low cost compared to other instruments, making them highly desirable.

The properties of Extinction Imagers make them very useful for groups such as the Navy to be used for developmental and operational support of laser weapons. The concepts of Extinction imagers apply to all wavelengths and all locations which makes them very versatile. One of the goals of this project was to use a DMD (Digital Micromirror Device) in addition to a CCD (Charge Coupled Device) to efficiently measure Extinction in infrared wavelengths at a low cost. A wide range of wavelengths, especially in the infrared, is helpful in computing information more related to the laser weapons' wavelength.

Theory:
The group from the University of California San Diego had derived the following equations to calculate the Atmospheric Transmittance, coefficient of extinction, and the visibility.

$$C_r = \frac{{}_bT_r - {}_bL_r}{{}_bL_r} \qquad C_0 = \frac{{}_bT_0 - {}_bL_0}{{}_bL_0}$$

$$T_r = \frac{C_r}{C_0} \qquad \alpha = \frac{-1}{r}\ln\left(\frac{C_r}{C_0}\right) \qquad V = r\ln\epsilon/\ln\left(\frac{C_r}{C_0}\right), \epsilon = 0.05$$

However, we can further manipulate these equations to show that the internal noise of the CCD device doesn't affect the values of Atmospheric Transmittance, coefficient of extinction, and visibility. This is an important observation, because it reduces complications with correcting the CCD in software.
Now let x be the internal noise in the CCD. Thus if it is not removed from the values we can write C_r and C_0 as follows:

$$C_r = \frac{({}_bT_r + x) - ({}_bL_r + x)}{{}_bL_r + x} \qquad C_0 = \frac{({}_bT_0 + x) - ({}_bL_0 + x)}{{}_bL_0 + x}$$

Now simplifying,

$$C_r = \frac{{}_bT_r + x - {}_bL_r - x}{{}_bL_r + x} \qquad C_0 = \frac{{}_bT_0 + x - {}_bL_0 - x}{{}_bL_0 + x}$$

Thus,

$$C_r = \frac{{}_bT_r - {}_bL_r}{{}_bL_r + x} \qquad C_0 = \frac{{}_bT_0 - {}_bL_0}{{}_bL_0 + x}$$

It is clear that the values for C_r and C_0 will not be the same as when the noise was removed, however these aren't the values that we truly care about. Plugging into the formula for T_r

$$T_r = \frac{\dfrac{{}_bT_r - {}_bL_r}{{}_bL_r + x}}{\dfrac{{}_bT_0 - {}_bL_0}{{}_bL_0 + x}}$$

One important thing to note is that ${}_bL_r = {}_bL_0$ because we are using the same region of the horizon for both range and reference. Thus our ratio simplifies to

$$T_r = \frac{{}_bT_r - {}_bL_r}{{}_bT_0 - {}_bL_0} = \frac{C_r}{C_0}$$

Hence the values of the internal noise do not matter in computing the quantities we are interested in.

Extinction Imager Phase 1 Calibration:

This program takes in 3 different sets of fits files. 1 file of dark (background noise) frames, 25 files of linearity frames, and 1 file of uniformity frames. This program computes the corrections that need to be applied to the camera. It first finds the fluctuation size of the internal noise of the camera to be used as the size of the buckets of pixels to be used in later processing, it next determines the linearity correction needed per intensity value, and finally computes the uniformity correction needed per pixel location.

In order to determine the size of fluctuations, we use the dark fits file, which was obtained by covering the aperture of the system and recording a video. We iterate over all of the frames in the dark file and for each pixel location determine the lowest and highest value. We find the differences of all of these values and choose the maximum difference as our bucket size. It is important to note that this is the only use of these dark frames. We do not subtract out the noise from our readings as this only raises more problems. It can be shown that the dark noise values cancel out in the calculations we care about.

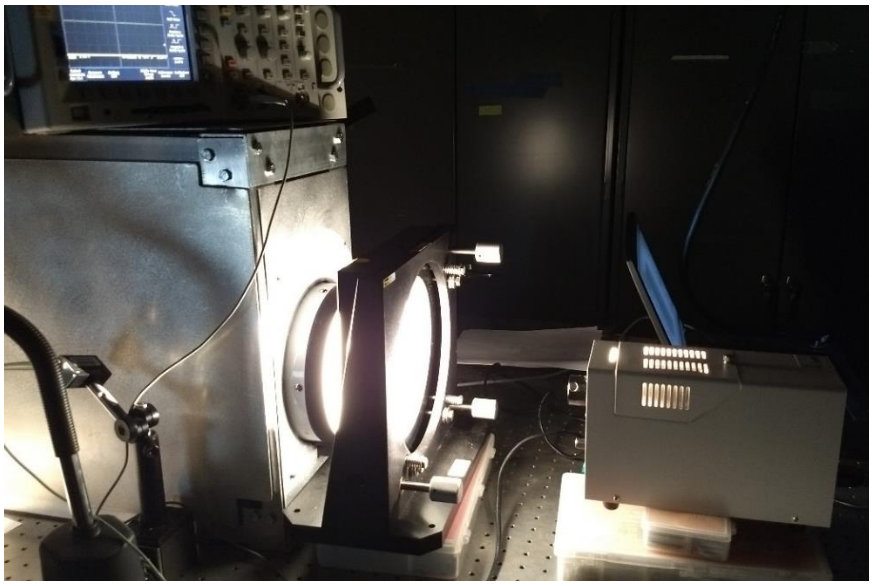
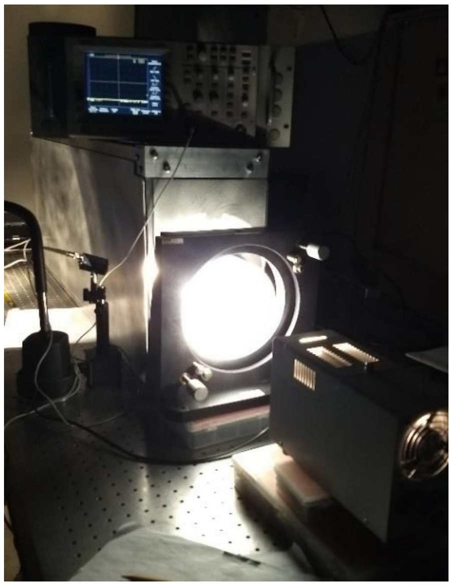
Refer to “Method 1” and “Method 2” to find the steps for the different linearity corrections. For both of these methods however, the fits files are the same. They are a series of 25 files, where each file is of a flat light source with a controlled intensity. A photodetector in a constant location was measuring the relative flux of the light. Each file was associated with a voltage from 2 to 14 volts in half volt increments. The file was then compressed into 1 number after doing a temporal and spatial average. Thus, the computations we began to do were on a set of 25 data points: (voltage, camera intensity).

The uniformity correction is calculated in order to account for any innate spatial bias the CCD camera has i.e. a pixel at position (1,10) being 2 percent higher than a pixel at position (700, 900). Thus, a video was taken with the aperture flooded with a flat light source of intensity values midrange on the 12-bit scale of the camera. Next a temporal average on the cube of data from the uniformity fits file was computed. Then the average value of this frame was computed as the metric for each pixel to be corrected to. Thus, this average value was divided by the intensity of each pixel, and that number was set to be the uniformity correction for that pixel. For example: if the average intensity of the frame was 5, and the was 7 for the pixel at position (10, 12), then the correction for that pixel would be 5/7 or approximately 0.7142. Likewise, if the intensity for the pixel at position (900,50) was 4, then the correction for that pixel would be 5/4 or 1.25. Each of these pixels are multiplied by their correction factors in order to perform the uniformity correction.

Finally, the correction, uniformity frame, and bucket size value are saved as csv files to their respective directories named by gain value, integration time, and the method of calibration. These files with then be input to the processing script according to the setting at which the data was taken.

N.B this calibration process must be done for each distinct pair of Gain and Integration time settings.

Set up of the calibration process.

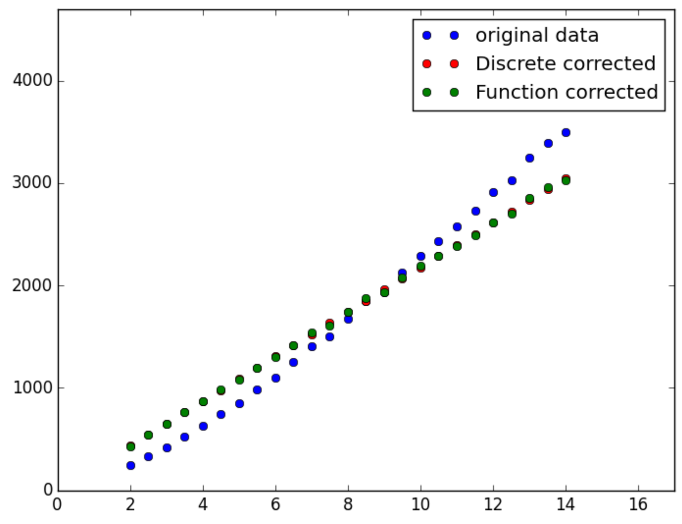
Method 1:

This method will generate a function that takes a camera intensity as input and outputs the linearity correction factor. We begin by normalizing all of the voltages to the smallest voltage value. This step isn't necessary, but makes the numbers nicer to look at. The next step is to find the camera value (out of the 25 values) that is closest to the "normval" variable. We are essentially setting this value to be a value on the camera that is truly linear. We are essentially going to be trusting this to be correct, and will correct all other values to its standard. Let's call the index of this value in our list of intensities "i". We will now normalize all of the camera values to this i^{th} camera values (divide each camera value by this value). Likewise, we will normalize all of the voltages to the i^{th} voltage. At this point all of our voltage values are essentially the values that we expect our camera values to be at if the CCD was linear. Thus the ratio of current camera values to voltage values is how "off" the CCD is at that intensity. Thus we divide all of the camera values by their corresponding voltage values. These new values that are formed as a result of this division are the correction values that each intensity should be divided by. However, we only have 25 of these corrections, so we perform another polynomial fit on the data set of original camera values and their corresponding corrections. The function we obtain therefore will take an intensity as input and output the linearity correction value which will correct that intensity by dividing it.

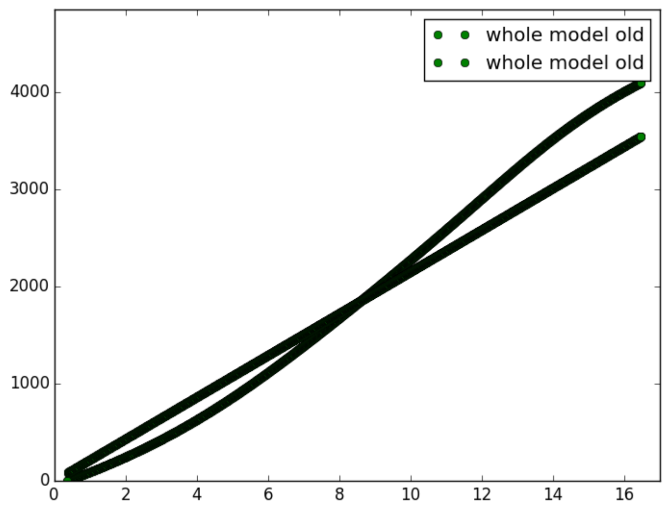
Method 2:

This method is similar to the first method with the only difference being that instead of generating a function that provides the correction, we have a lookup table of correction values. We begin this process by performing a polynomial fit on the set of 25 data points so that we have the ability to enter a camera intensity into the function, and receive an associated "voltage" as output. We then generate a set of 4096 points using this function where one axis are integers in the set [0,4095] and the other axis are the corresponding voltages. From here on out the process we follow is the same as in method 1. Namely, find how far off each camera intensity ratio is from the expected flux ratio and store the correcting factors in a table. The table is in the format where the index is the camera intensity and the value is the correction to be divided by.

Below are examples of the original camera values' non linearity and the effects of the linearity correction. These plots were done with the Gain equal to 10 and the Integration time 25000 with functions of degree 5 and a "normval" variable of 1850



Method 1



Method 2

Thoughts and Suggestions:

There are several things that still should be experimented with before choosing a method as "the" calibration method. Different function fitting techniques should be explored, perhaps even using a piecewise function. This may be unnecessarily complicated method, but using an Artificial Neural Network to build a custom function could be investigated further. How the layers of the network would be built needs to be explored further. Also, it is possible that manipulating the value of the "normval" variable can have an impact on the integrity of the model.

Extinction Imager Phase 1 Processing:

Both of these programs behave in the exact same way except for applying the linearity correction. The different ways of applying the linearity correction are described below under "Method 1" and "Method 2". The processing allows for region of interests to be chosen at any locations, and of variable sizes.

The program begins by reading in either 2 or 3 fits files. It always will require a file that contains the target at range, and a separate file that contains the target at reference. The option for the third file is if the horizon can't be fit in the same frame as the range target. However, if the horizon is in the range file, then this option is just left blank. In addition, 3 csv files are read in with information relevant to the calibration corrections. These are the linearity correction (either coefficients of a function or a lookup table), the uniformity correction (table of numbers), and the bucket size (just one number). These specific frames should be the ones that correlate to the gain and integration settings that the fits videos were taken with. The next step is to hardcode the center coordinates, and sizes of each region of interest. These numbers can be discovered by looking at images of the scenes in either "Spydr" or "fv fits editor". Based on this information, we calculate the coordinates of the four corners of the rectangle, and now are ready to being calculations. I will discuss the process for evaluating one region of interest- all three regions of interest follow the same process. We loop through every frame in time for a certain fits file. We correct the frame for linearity and then correct it for uniformity. The uniformity correction is simply multiplying the frame by the coefficient table that was inputted. Finally, we iterate through all of the pixels in the region of interest for this frame and add them to a histogram for this region. The histogram for each region of interest is set up so that the x-axis isn't just every intensity value we have, but rather buckets of size "bucket size" of a range of intensities. After adding all pixels to the histogram, namely an (area of interest)*(number of frames) amount of them, we choose the bucket with the most pixels in it to be the "true bucket of values". We then average the pixel values in that bucket to come up with our t_l_0, t_l_r, b_l_0, and b_l_r values. This method greatly helps to reduce the effects of scintillation on our region of interest. Next, we calculate C_0, C_R, T_R, alpha, and visibility using the formulas derived by Janet Shields' group. Note that we use the same value for the background (horizon both at range and reference)

Method 1:

The Linearity array is an array of the coefficients of the correction polynomial calculated in the calibration process. To generate the correction needed, we use a "for" loop to evaluate the function for each x value (camera intensity). Every iteration of the "for" loop represents stepping through one monomial in the function. For example: if we have a function in the form ax^2+bx+c, the first iteration computes "ax^2" for every pixel. It then adds that value to "bx" for each pixel etc. This allows us to quickly calculate the correction for each pixel and then correct by dividing each intensity by its computed correction factor.

Method 2:

The linearity array is simply a lookup table of 4096 corrections, where the index of the array refers to the camera intensity and the value at that position is the correction. Whatever the intensity value is, is then divided by the correction value to be corrected. We iterate through the entire frame, correcting in this fashion. Since the average of camera values don't necessarily have to be integer, when looking up the correction we choose the index of the lookup table that is closest to the camera value.

Thoughts and suggestions:

Throughout processing, on average the transmittance and visibility numbers were lower than expected. This was explained by the fact that the computed values for C_0 and C_R were too far apart. In order to help figure out why there was such a disparity in values, it was necessary to attempt to determine if there was some innate inaccuracy in the camera, or if this disparity was a result of the atmospheric conditions i.e. too much sunlight angled into the box at range. Thus, 3 data files were processed, which were taken at the APSHEL testing session, all on 08012017 and at the 5 Gain 25000 Integration time setting: The first file taken at time 1407 UTC, the second at time 1700 UTC, and the third at 2205 UTC. When performing the following test, other atmospheric conditions were not taken into effect, such as clouds, so further thorough testing should be done with regards to those aspects at a later time.

Time/Method	Range T	Ref T	Background	C_0	C_R	Transmittance	Ext Coeff	Visibility
1407 M1	671.263	285.006	2837.705	-0.899	-0.763	0.849	0.164	18.259
1407 M2	666.149	298.266	3155.276	-0.905	-0.789	0.871	0.138	21.732
1700 M1	715.650	296.557	2625.685	-0.887	-0.727	0.820	0.198	15.102
1700 M2	710.789	306.874	2887.599	-0.894	-0.754	0.843	0.170	17.600
2205 M1	663.645	223.922	2094.047	-0.893	-0.683	0.765	0.268	11.176
2205 M2	565.232	245.359	2145.422	-0.886	-0.737	0.832	0.184	16.251

The next step in refining this process is to try and figure out why the "Range T" and "Ref T" values are so far apart when they probably should be closer. Thoughts for this include performing some controlled experiment where the transmittance, extinction coefficient, and visibility of the path are predetermined. Then the problem can be debugged by walking through each step in our process.

Another place where improvement can be made is in the runtime of these python programs. Instead of performing the CCD corrections to the entire image, if only the intensities in the region of interest are corrected, the number of operations that have to be done can be reduced by 4 orders of magnitude. Namely, from 10^8 operations to 10^4. This can have a massive impact, especially in python. The python scripts following this idea have been written, and can be used if the entire corrected image isn’t necessary to see. The processing time decreased from approximately 30 minutes to just a few seconds after making this change.

Extinction Imager Phase 2 Calibration:

Phase 2 of this project has to do with implementing the DMD into the system. The beginning of this process was implemented. An important part in this step was calibrating the image formed by the DMD to the image recorded by the CCD, since each device had a difficult resolution. A raster scan was implemented in C, and run on the system to build the image captured by the DMD. Then that image was sent to the python calibration script. This script takes a small region—call it the key—with distinct features, from the DMD image and moves it around the image produced by the CCD, placing it at every possible spot. An initial thought on how to check if the two regions correlated was to sum the product of the intensities of each corresponding location in the key and the region it intersects with. More formally, let the result be $\sum_{i,j}^{m,n} key_{ij} * CCD_{ij}$, and choose the largest sum as the correct location for the correlation. However, a counterexample was presented where the highest region of correlation would always be the brightest part of the image. Thus, a different method was presented where instead of the product of intensities, the binary not xor was computed for each pixel in the region. This other method performed well while testing it on small premade data sets. It should be robust enough to get the job done for now, but a more detailed method should be pursued further on in the process if necessary.