Lab 1 – Basics of Image Processing

Goals for this lab:

- To introduce ENVI Software.
- To learn what a digital image is vs. how it is displayed.
- To learn the basics of image manipulation (i.e., image math).
- To introduce the image histogram and basic contrast stretching.

Launch ENVI and then, using the drop-down "Help" menu, begin by working through ENVI Tutorials 1 & 2 ("A Quick Start to ENVI" and "Introduction to ENVI") until you feel comfortable with these functions of the software.

What is a digital image?

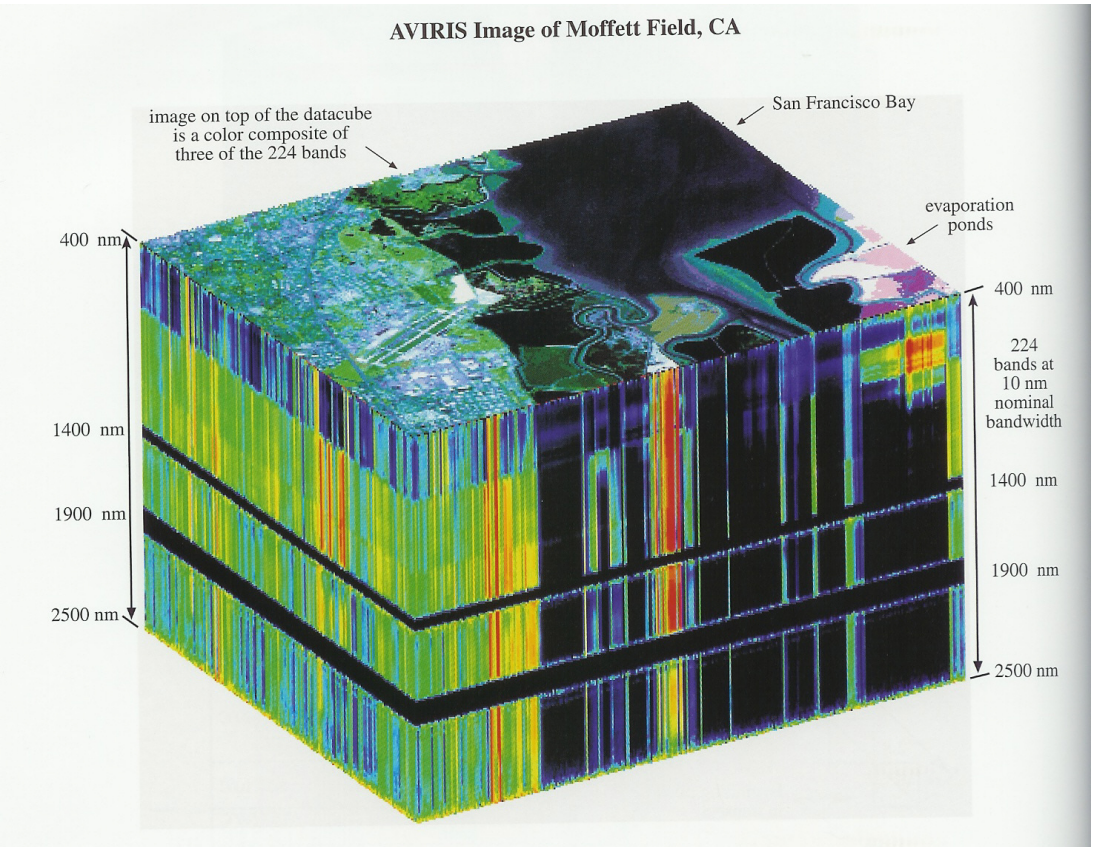
In this class the basic dataset for any and all projects we do is going to be the digital image. Fortunately, most of us have seen plenty of digital images, either from web pages, or email attachments, or any number of other sources. Many of us have even "touched up" digital images with programs like Photoshop or Paint Shop Pro. What makes this possible? Yes, many of us have told computers to manipulate an image for us, but how does the computer do it? The answer lies with understanding what a digital image is.

A digital image is a grid of picture elements, called *pixels*. Pixels are familiar to most, as when we zoom in on an image on the computer screen enough times the image becomes very blocky, or *pixelated*. Each pixel represents a certain color, but it is important to realize that pixels are not simply colors – **they are numbers called *brightness values* or, sometimes, *data numbers (DN).*** In other words, a digital image is really a grid or array of numbers stored in a computer file. Whatever program used to display the image turns the number into a color and displays that color on the screen. Pixel values are called brightness values because the higher the number, the brighter the color that is assigned to them. And because digital images are grids of numbers, we can manipulate them mathematically. This is the basis of image processing.

It's also important to remember what each of the numbers in an image represents. Let's consider using a digital camera. In a digital camera, a grid of detectors is used in place of film. If you were to take a picture with a digital camera, the numbers for each pixel in the image would be proportional to the number of photons that each of the detector cells detected. In a typical remote sensing image, the pixel values represent the number of photons *of a certain wavelength* (or range of wavelengths) that are recorded by the detector. Actually, it is more true to say that most remote sensing images merely start off this way. In most cases calibrations and corrections are used to convert the photon count images to more useful quantities (like reflectance, as we'll see later).

Of course, there are many image files that are made of more than one grid of numbers. Color images, for example, are made of three stacked grids. It is perfectly appropriate to think of these kinds of image files as being made of 3 separate images. We call each of the separate images a *band* (or a *channel*). In remote sensing, we can be using single band images, like many radar images, images with several bands (Landsat images have 7), and even images with hundreds of bands (like AVIRIS images, which

have 224 bands). The complete image file, with all bands considered together, is often called an *image cube*. The figure below is an example of an AVIRIS image cube. We will now look at some basic processing techniques, starting, of course, with the simpler cases – single band images.



Single-band images

Single band images are usually called *grayscale images* because they are usually not displayed in color (though this is possible). Instead they are usually displayed with the lowest numbers having darkest shades of gray, and the highest numbers becoming closest to white. The numbers stored in the image can actually have any range, but this range is usually scaled to fall into the range 0-255 (called the *full dynamic range*) before the image is displayed, which we call the screen value. So a screen value of 0 will have the color black, and a screen value of 255 will be white when displayed on screen.

As we said before, images are grids of numbers. And, as with any set of numbers, statistics can be used to describe them. We can plot a histogram of the brightness values, and we can also calculate an average and a standard deviation. We can also consider the image to be a matrix, and perform matrix algebra on the image. In fact, many of the higher-order processing techniques we'll learn about later in the semester will use matrix algebra. The matrix representing an image will have as many columns and rows as there are pixels in the x and y dimensions of the image. The values in each cell of the matrix will be equivalent to the brightness values of the corresponding pixel.

To illustrate, let's consider two basic properties of images, *brightness* and *contrast*. Brightness, as its name implies, refers to how bright or dark the overall image is. Contrast refers to how much definition the image has – in images with low contrast, the colors all look the same and it becomes difficult to make out the objects in the image (refer to the figure below for examples). We can begin to quantify these properties (though this is by no means perfect) by realizing that the higher the average of the brightness values, the higher the overall brightness of the image. And the higher the standard deviation of the brightness values, the more contrast there is.

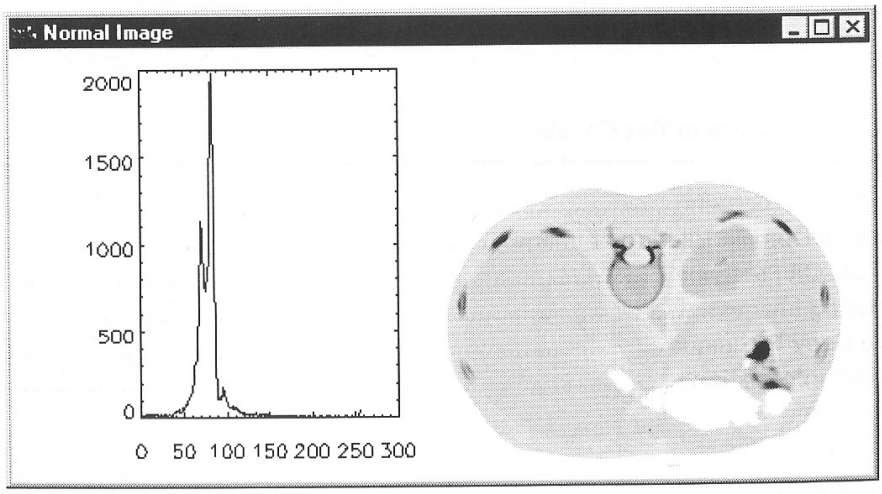
Examples of brightness/contrast variation.

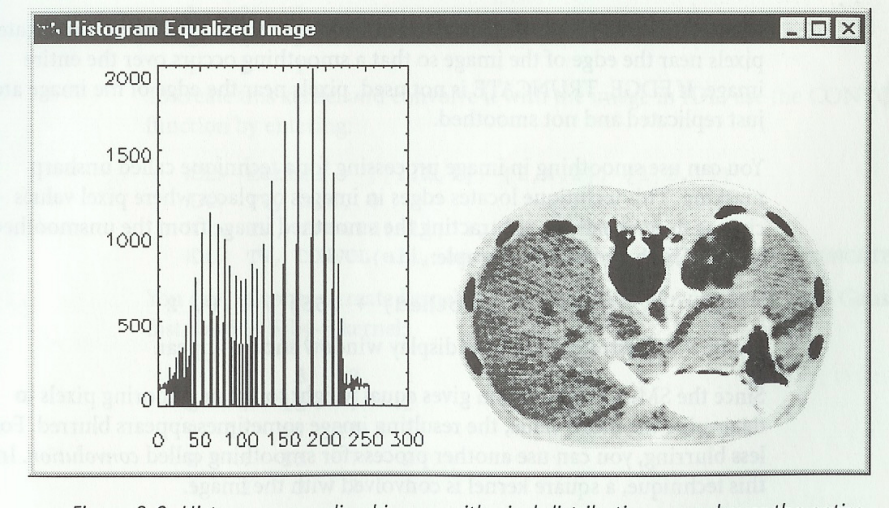
What if we want to adjust the brightness or contrast of an image? This is where image math comes in. Recall that images can be thought of as matrices. So let any image be represented by the matrix **A**. We can create a new image **B** by setting **B** = **A** + 2. This is equivalent to adding the number 2 to every brightness value in the image (and subtraction works the same way). If we do this, which property of the image have we changed? We haven't changed the contrast because adding the same number to every pixel does not spread individual brightness values farther apart - in other words, it does not increase the standard deviation of the population of all the brightness values. What we have done, however, is make every pixel's brightness value higher, so we've increased the brightness of the image. To change the contrast, we would need to multiply every pixel's value by 2 (or some other value), since that would cause the numbers to be spread apart (increase the standard deviation).

These basic techniques are not the only way contrast is adjusted in an image, nor are they necessarily the best way. This is simply to introduce you to the basics of image math. In practice, adjusting the contrast of an image is usually done using computer routines, and the technique is usually referred to as *contrast stretching*. ENVI, the software program you will use, has powerful contrast stretching abilities. Note that contrast stretching is, in essence, a manipulation of the image histogram. One very common contrast stretching technique is called *histogram equalization*. This method is good for images with DNs that fall into a narrow range (i.e., have narrow, tall histograms,

giving low image contrast). It manipulates the histogram such that it tries to give each pixel value (each bin in the histogram) an equal number of pixels with that value. The two figures below illustrate this. The image on top is a CAT scan of part of a human body, and its histogram is to the left of it. You can see the large spikes in the histogram, and that the range in pixel values is about 50-150, about half of the total available range of brightness values (0-255, the full *dynamic range* of DN). The image on the bottom is the histogram-equalized version of the original, along with the equalized histogram. The range is 0-255 now, and each pixel value occurs more frequently (note the y-axis). The histogram equalization has clearly brought out quite a bit of detail in the CAT scan.

What we have discussed so far is manipulation of an image by adding/subtracting or multiplying/dividing an image by a single number. It is also possible to add two images together (although this gets a bit more complicated if they don't have the same pixel dimensions). In fact, any of the basic mathematical operations can be performed between two images. For example, one image can be multiplied by another, in which case each pixel's value in one image would be multiplied by the corresponding pixel's value in the other image (though note that this is not a matrix algebra operation). Why would we do this? As an example, there are special cases of single band images called *binary images*, in which pixels can have only one of two values, 0 or 1. Binary images are often used as *image masks*. Multiplying some base image by a mask has the effect of keeping some pixels in the base image at their original value (because they are being multiplied by 1), while all others are discarded (they become 0 because they are being multiplied by 0).



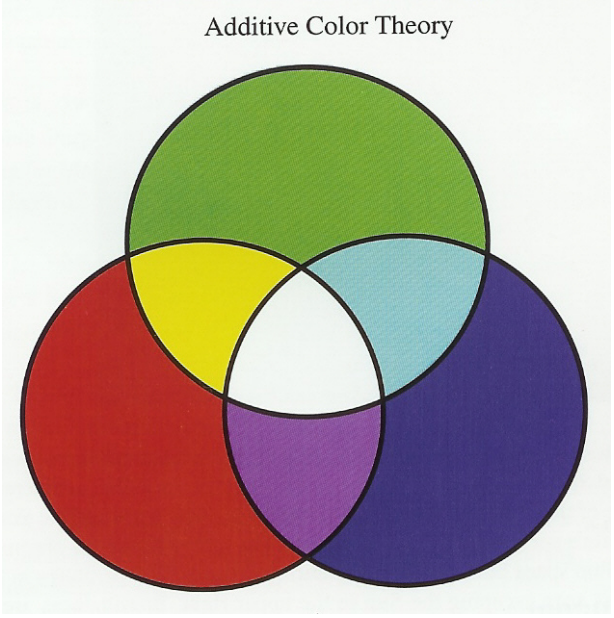


Three Band Images (Color images)

We've talked about grayscale images so far, but what about color images? Intuitively, we realize that for an image to be displayed in color, that color information has to be stored in the image file. How is this accomplished?

It turns out that, from the point of view of our human perception, all colors are really mixtures of three primary colors – red, green, and blue. (*Note*: the color mixing we are talking about here is not the same as the way colors mix when, say, you mix paints of a different color. The results are very different.) These 3 primary colors, then, are the basis for color images, since that this is the minimum information required to accurately reproduce color in an image. As you might have guessed, in digital color images a separate band is created for each of the three primary colors, thus creating a three-band image. These images are often called *RGB images* because of the three colors being represented. The program that displays the image looks at each band's value for each pixel, and, based on their relative proportions, displays the appropriate color on the screen.

It is important to develop an intuitive feel for what proportions of red, green, and blue will produce which color in an image. Recall that when displayed, whatever range the image data falls in is usually scaled to fit into the range 0-255. This is true for each band in an RGB image as well. So each pixel of an RGB image has a set of three values in this range that determines its overall colors. This set of values is often called a *color triplet*, and is always ordered R-G-B. The color black is represented by the numbers [0 0 0], and white is represented by [255 255 255]. Pure red would be [255 0 0], pure green [0 255 0], and pure blue [0 0 255]. What happens if we have equal proportions of red and green, but no blue? We get the color yellow (again, remember that this is not like mixing red and green paint). The figure below, which is a diagram illustrating *additive color theory*, shows this. Also note that if you mix all three of the primary colors you get white.



The contrast stretching that can be done on a single band image can also be done for each of the individual bands that make up the RGB image (or also for the more complex images cubes with any number of bands). This gives the image processor the power to exert a lot of control over the information the image displays. As you'll learn, very frequently noise or some object in the image (like a cloud or a shadow) will complicate your efforts. Contrast stretching is one tool that can be used to overcome the limitations of some of these impairments. Shadowed areas can be stretched, for example, to bring out detail that would otherwise be masked by the shadow. And it has the potential to let you show details that are important to you in an image, while hiding those that are not. Therefore you the student should learn to stretch the contrast in images in ways that are important. For example, if your image contains very dark portions, and a very light portion, and some parts somewhere in between, it can easily be the case that the lightest and darkest parts of the image have no detail because most of the contrast is in the areas of the image with brightness values that are not extreme. Water areas like streams can often show up very dark in images where land is also shown, and in that case it would be very difficult to pick out detail in the stream. If you want to see sediment flux in the river, for example, you should do your stretch so that the pixel values in the darker areas are spread out. In a color image, you have even greater control. If you are working on an image of a red car, the blue and green channels will likely not have a lot of detail in them. You might be able to stretch the contrast in these channels and bring out more detail in the image.

Remote Sensing Lab 1 Name _____
Image Processing Basics

This is your introduction to the basics of image processing. You'll use these basic concepts over and over during the semester, so although ***you don't need to turn in your answers***, you should still make sure you can answer all the questions below. The files you'll need for questions 4-6 are at:
http://wray.eas.gatech.edu/remotesensing2013/RS_Lab1_files.zip

1. Use the following "images" to answer the questions below:

A

1	4	2
2	4	1
3	5	1

B

7	5	6
1	2	1
8	6	9

Print **A** + 2**:**
(draw in a grid
with the answer)

A + B:

Average of **A**:

Std. Dev. Of **B**:

B * 3:

Std. Dev. Of (**B** * 3):

histogram of **A**:
(hand drawn is fine)

2. A. Give reasonable colors for these RGB triplet values:
(hint: this is one of the few times this semester when Photoshop might actually be a useful tool for you, or you can guesstimate based on the color diagram given in the text for this lab.)

[51 255 255] _____

[102 51 0] _____

[255 0 153] _____

[102 102 102] _____

B. Give reasonable color triplet values for these colors (you don't need to get them exact):

_____ orange

_____ black

_____ tan

_____ cyan

3. Draw hypothetical (but valid) histograms for images that have the following characteristics:

high contrast, low brightness:

moderate brightness, low contrast

moderate brightness, high contrast

low brightness, moderate contrast

moderate brightness, zero contrast

4. One very common application of remote sensing is called *change detection*. In this application, two (or more) images of the same area are taken at different times and compared. Can you figure out a very simple way to do this using band math techniques you've learned so far? You have two images, cover2002.tif and cover2003.tif. Produce an image that shows the amount of coast line that has eroded between 2002 and 2003. These are very simple hypothetical images. If these were real images, what would be some problems that would occur (or at least that you would have to watch out for) if you used the technique you used for the hypothetical images that would make the technique less effective?

5. The file dem.img represents a *digital elevation model*. This is an image where the pixel values represent topography, so bright colors are high elevation and dark colors are lower elevation. Somebody comes to you and says that they want to see all the places where the elevation is above 1300 ft. Make them an image in which all elevations above 1300 ft. are preserved, and all values below that are set to 0. Also, make them an image that only shows elevations between 1250 and 1275 feet. (Hint: you can make an image mask where all values above 1300 are set to 1, all values below set to 0 using the band math expression (b1 gt 1300).)

6. Using image math and the file mask.img, insert the figure of Mace Windu in mace.jpg into background.jpg. Computer graphics (CG)-generated special effects are, of course, a major modern application of image processing.