

Introduction to Scientific Visualization



Foreword

It has been my experience that there has always been some kind of visualization aspect to scientific data e.g. 2D plots, pie charts or bar graphs. Even in the Humanities, data or models are rendered into visualizations such as geopolitical maps, personality charts, or timelines (e.g. Charles Minard's Napoleon's March).

This iBook is a compilation of different learning exercises that have or will be used in a course called "Introduction to Scientific Visualization". It is a unique collection of topics and exercises that brings together skills from Arts, Computer Science, Math, and Sciences. The topics are generally brief and will demonstrate skill levels from basic to low-moderate.

Various freeware (and O/S-ware) will be examined to demonstrate styles of image renderings from data, terminology, and the basic math manipulations required to generate the images.

This iBook is a work in progress with Version 1 discussing Excel and ImageJ software in some detail as applicable to the Introduction to Scientific Visualization course. Later versions of the iBook will add more exercises with Excel and ImageJ as well as add newer software

demonstrations with VisIt, VMD and ParaView software. It is hoped that KiwiViewer, 123D, and augmented reality can be included sometime in the near future.

Links are provided in the text to direct the reader to a few websites where files, software, and documentation can be downloaded. A PDF version of the iBook will be made available for those readers that don't have an iPad. The interactive widgets won't be active, but in the case of the Gallery widgets, short Powerpoint presentations will be available to show step by step instructions on how to perform the described task.

Although the word "Scientific" is used in the course title, by no means is that a restriction as to the origins of the data, but rather the method in which it was collected, i.e. applying the Scientific Method. For example, in the Excel Chapter, a "4D" graph will be created using a song playlist.

After browsing the iBook, it is hoped that the reader will at least have an appreciation for Scientific Visualization, the math and art that goes along with it, and

perhaps inspire them to pursue it as part of their profession.

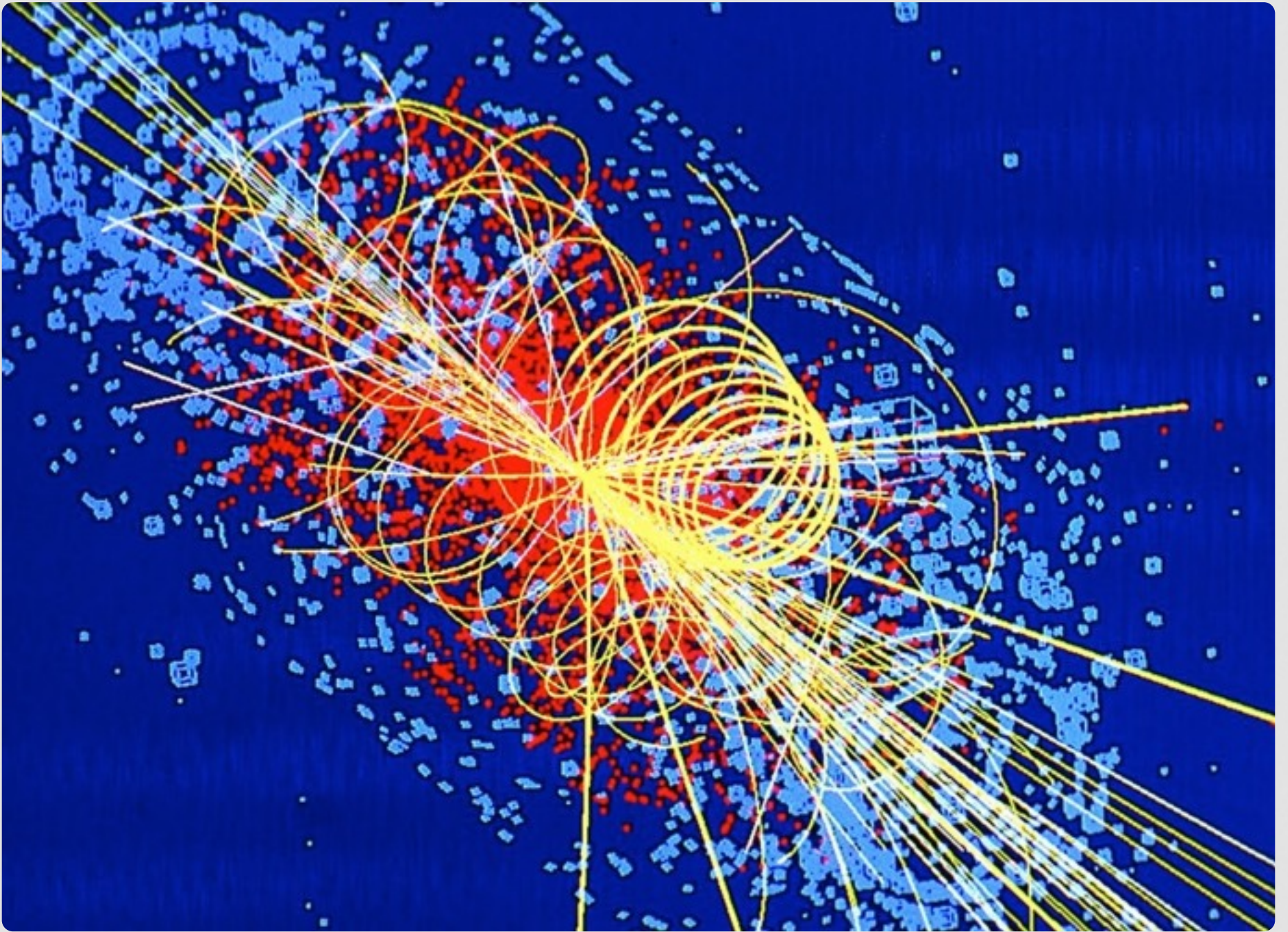
The mathematician/scientist will exercise their artistic skill. A humanities researcher can convert qualitative information into quantitative information and thus communicate their discoveries via a visualization. The artist can discover the art hidden within data and how basic math can significantly alter the art.

Virtual reality and reality can be combined, for example I have seen an awesome animation that combined a virtual image of a sun with real coronal data from our sun and thus a virtual/real hybrid image was created.

1

Visualization

Image 1.1 Visualizing the Higgs Boson



Shapes, colors, glyphs, plots, and illustrations (i.e. visualizations) are used extensively for representing objects some of which are real and tangible, real but intangible, or completely abstract. Take for

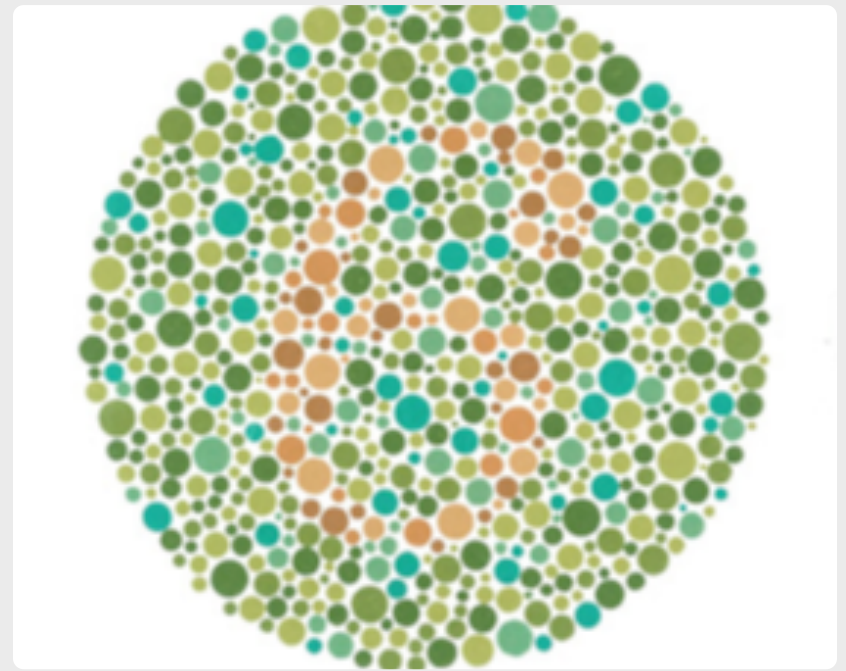
example Image 1.1. It is a computer simulation of the particle traces that would be made when a Higgs Boson is produced at the LHC at CERN. The Higgs Boson concept, its visualization, and the data

required to generate the visualization are quite complex. You'll never see the particle directly, only indirect evidence that it was produced. Furthermore, the image is a simulation, will it match up well with data taken from a real Higgs Boson producing event?

Visualizations can also communicate ideas, emotions or actions. There are 1000's of languages that exist in the world. Can you assume that everyone knows how to read, write, or speak YOUR language? It might be that a person never learned to read or write at all or perhaps can't speak because of a medical condition. I've put together Gallery 1.1 as a fun exercise on thinking what different symbols stand for and the significance of color. Each Gallery symbol probably inspires a thought, action and/or challenge. Each image in the Gallery communicates a "message", shouldn't you illustrate your data to give as clear of a message as possible and NOT be misleading?

Here is another tangible perspective on visualization: Let's assume that you've invested a specific amount of money in a particular stock. Your stocker broker looks you straight in the eye and truthfully says, "Your stock is worth more now than when you started." Great!!! Look at the four

Gallery 1.1 Symbols and colors, what do they communicate to you?

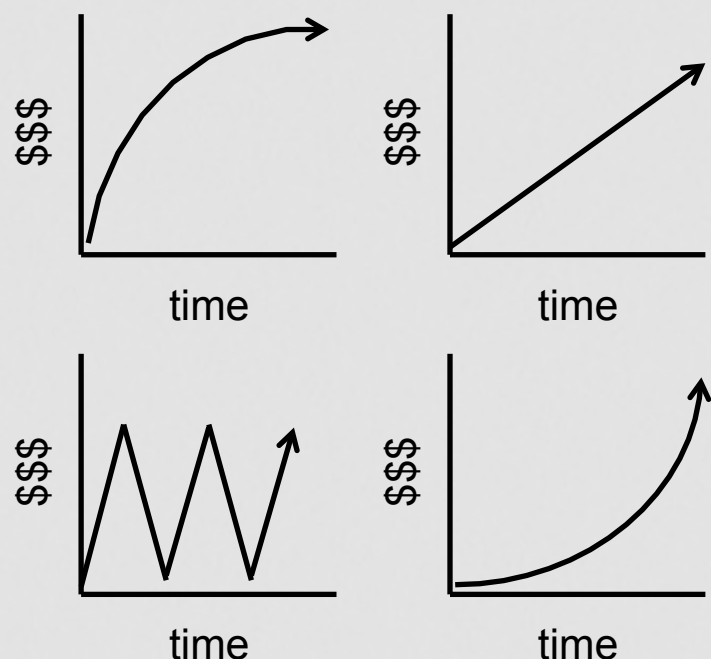


The next 5 slides are a color blind test. Do you know if you are colorblind? What do you see in this one? When creating data images, be mindful of what color combinations you use.

22 of 26

examples in Image 1.2 that fit the statement. Which one would be the best pathway for future earnings?

Image 1.2 Stock worth versus time.



Transforming data into a meaningful picture is what visualization is all about. If you simply look at sets of numbers, you might not see a story or get the whole story line.

Another example I find interesting that illustrates the significance of transforming numbers into a picture is called Anscombe’s Quartet, Table 1.1. When you look at the four data sets, what do you “see”? At least a couple of things might be obvious. Three of the x data sets are the same and the fourth is different, the fourth x data set has all “8.0” except for one “19.0”. The y data sets don’t seem to have any similarities, perhaps just random.

Down load the file [Anscombe’s Quartet](#). The link should take you to a website that contains several folders one of which is labeled Excel. In that folder you should find the desired file. After downloading, open it in Excel, use the spread sheet functions to find the average and standard deviation for the x and y data columns for each data set. When you are finished, two statistical observations can be made: all of the x-data set columns have the same average and standard deviation, “9.0” and “3.30” respectively. All of the y-data set columns have the same average and standard deviation, “7.5” and “2.03”

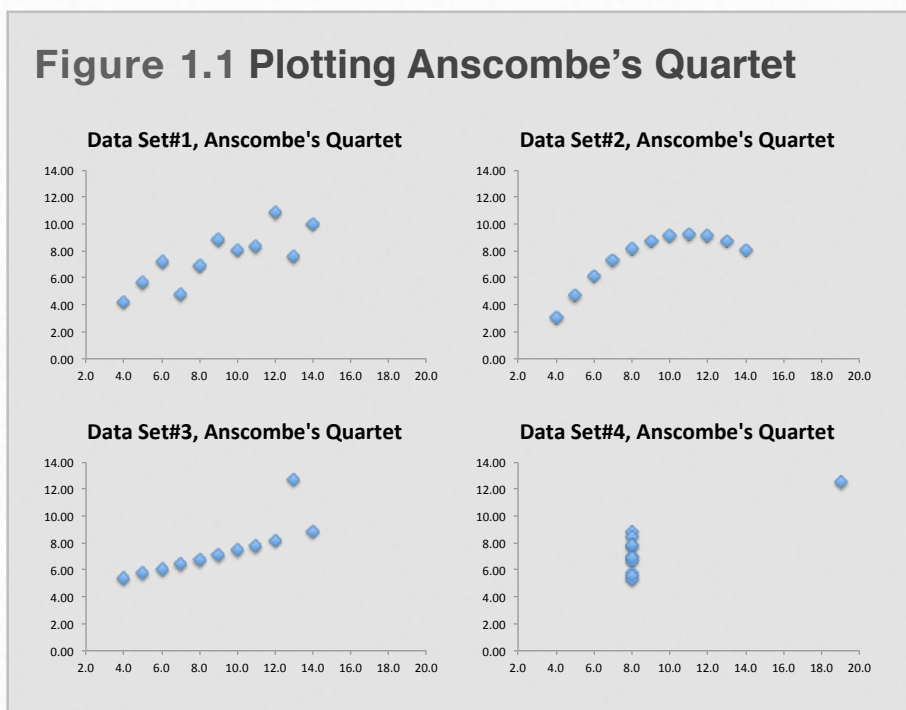
Table 1.1 Anscombe’s Quartet

| x1 | y1 | | x2 | y2 |
|------|-------|--|------|-------|
| 10.0 | 8.04 | | 10.0 | 9.14 |
| 8.0 | 6.95 | | 8.0 | 8.14 |
| 13.0 | 7.58 | | 13.0 | 8.74 |
| 9.0 | 8.81 | | 9.0 | 8.77 |
| 11.0 | 8.33 | | 11.0 | 9.26 |
| 14.0 | 9.96 | | 14.0 | 8.10 |
| 6.0 | 7.24 | | 6.0 | 6.13 |
| 4.0 | 4.26 | | 4.0 | 3.10 |
| 12.0 | 10.84 | | 12.0 | 9.13 |
| 7.0 | 4.82 | | 7.0 | 7.26 |
| 5.0 | 5.68 | | 5.0 | 4.74 |
| | | | | |
| x3 | y3 | | x4 | y4 |
| 10.0 | 7.46 | | 8.0 | 6.58 |
| 8.0 | 6.77 | | 8.0 | 5.76 |
| 13.0 | 12.74 | | 8.0 | 7.71 |
| 9.0 | 7.11 | | 8.0 | 8.84 |
| 11.0 | 7.81 | | 8.0 | 8.47 |
| 14.0 | 8.84 | | 8.0 | 7.04 |
| 6.0 | 6.08 | | 8.0 | 5.25 |
| 4.0 | 5.39 | | 19.0 | 12.50 |
| 12.0 | 8.15 | | 8.0 | 5.56 |
| 7.0 | 6.42 | | 8.0 | 7.91 |
| 5.0 | 5.73 | | 8.0 | 6.89 |

respectively. So from a statistical stand point, one could say they are all similar, at

least from an average and standard deviation point of view.

What is the correlation between x and y in each of the sets? Let's visualize each set by plotting each of them in Excel. You



should get results similar to Figure 1.1. Are you surprised at what you see? How would you describe the correlation between x and y in each plot? Hopefully you can see how important a role visualization can play in “telling a story”.

The remaining chapters will exam different aspects of visualization and softwares (most of which are free) that can be used. Each software has it's unique purpose and has specific exercises and data sets to work with.

2

Beginning Visualization of Data - Plotting in Excel

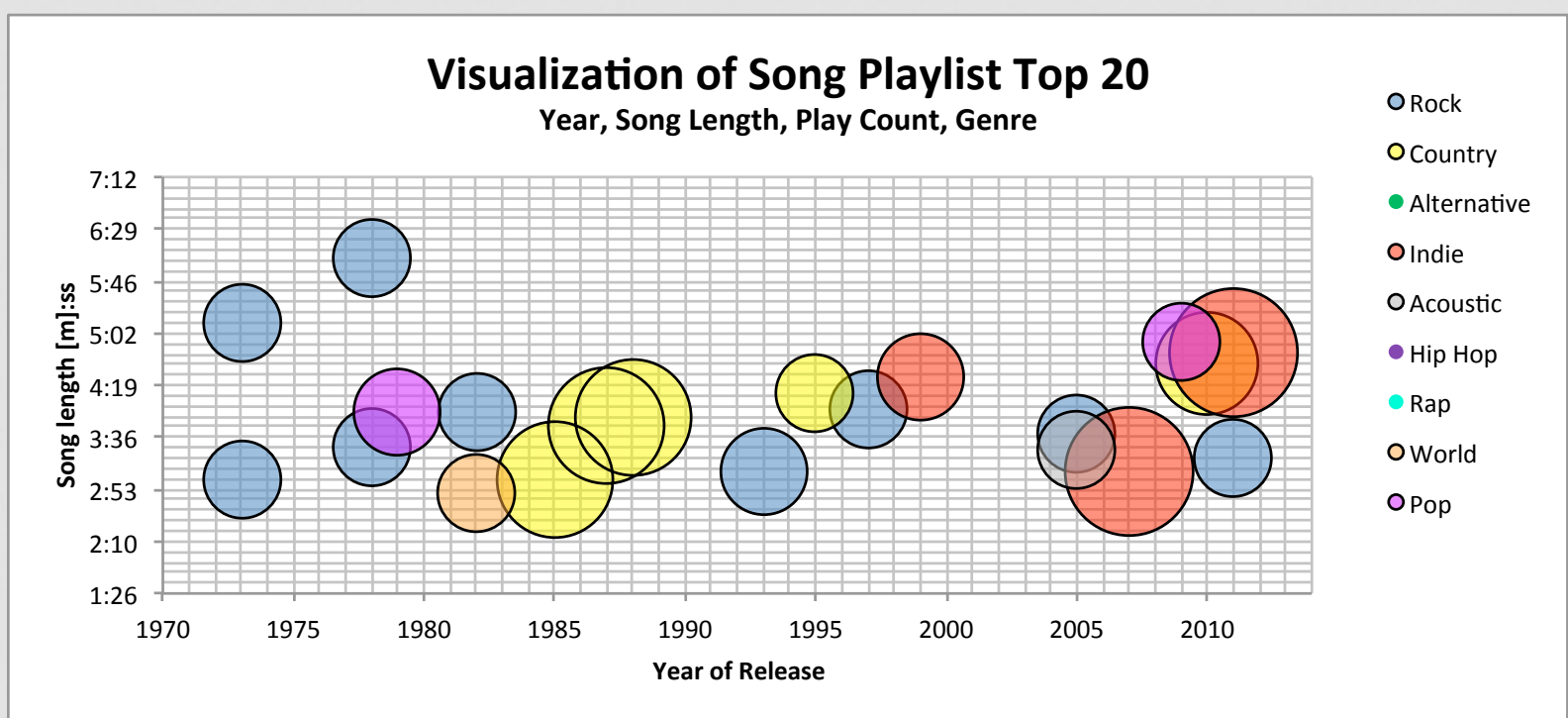


Figure 2.1 Four dimensions of information are represented in an Excel plot: length of song, year of song, color represents genre, and size of the bubble represents frequency of song play. See Gallery 2.3.

The idea behind Scientific Visualization is to represent data in some sort of meaningful illustrative form so that the viewer can interpret the data in hopefully a quick and easy manner. One of the most basic methods of rendering simple 2-dimensional data is an X/Y graph.

Excel is a fairly common program for performing this task. It is assumed that the

reader has the basic skills for creating a 2 dimensional plot of data in Excel along with adding Chart title, axis labels, and legend labels. This chapter will show you a few additional Excel calculation and graphing tweaks you can perform to add more dimensions to your traditional 2-D plot such as overlaying other data sets, choosing different colors, shapes, sizes, and alternative axis formatting.

Although most of the data used in this chapter is of a scientific nature, it doesn't have to be. It can be of most any type as long as the measurements can be transformed into some kind of numerical value. Instead of exhaustive lists of numbers and letters which are hypnotizing, they can be transformed into colors, shapes and overlays that are quicker to interpret. Take for example the figure at the beginning of this chapter. It presents the Data (Table 2.1) obtained from one of the authors' music playing devices into a more illustrative manner. The Data focused on what were the Top-20 most listened to songs. Most of the data was numerical in nature, but genre (which is qualitative) was transformed into numbers by categorization strictly for ease of organization.

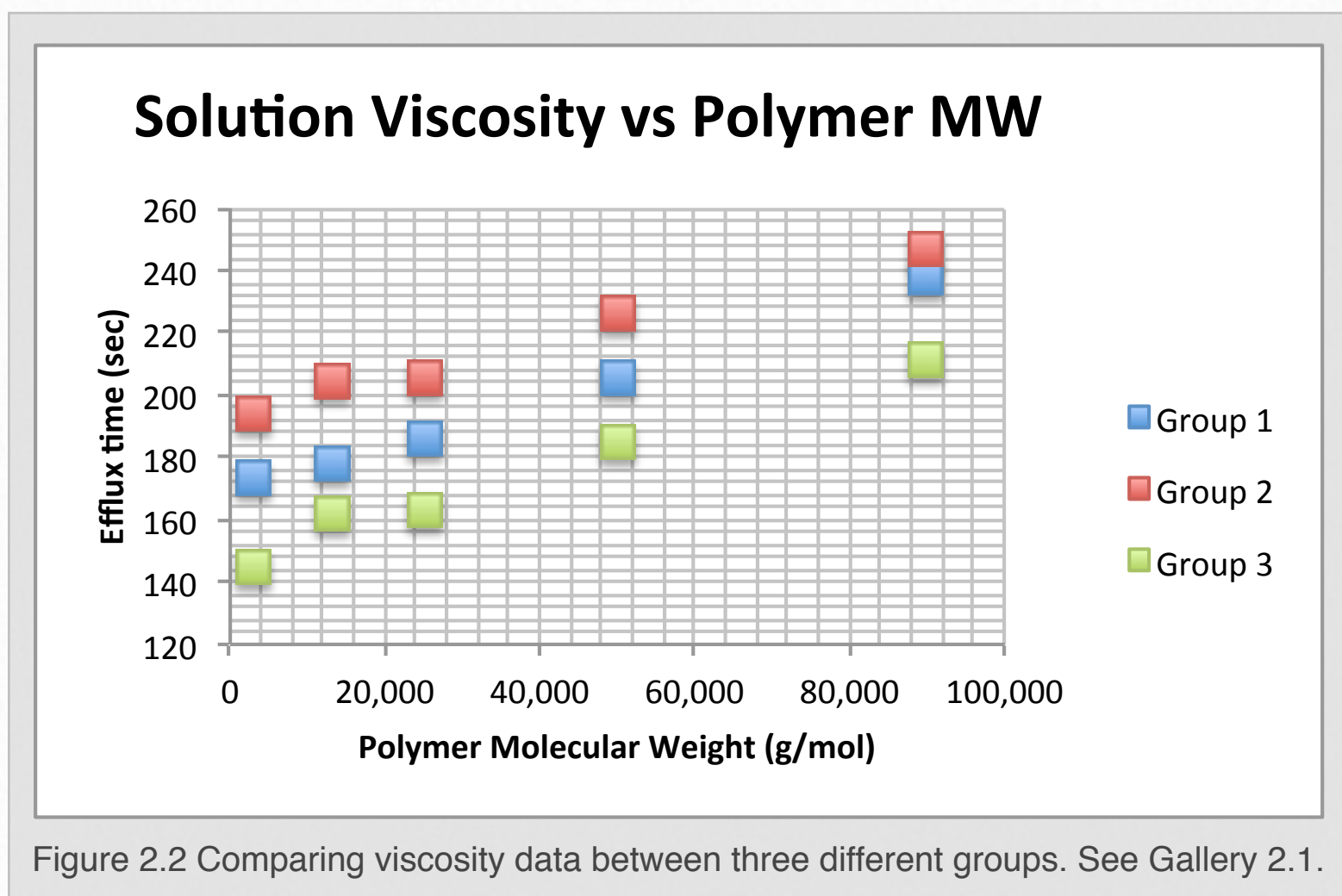
All in all 4 What are some conclusions from looking at this plot? Here are some possibilities: 1) the songs are somewhat evenly scattered from early 70's to early 2010's, 2) there are more blue circles than any other color, 3) yellow seems to be the larger circles followed closely by red, 4) other than blue, yellow and maybe red, very few other types of colors are represented. This might give rise to other questions that are worthy of pursuing: just what was that favorite "pop" (purple) song

Table 2.1 Top 20 Song Playlist

| year | length | frequency | genre |
|------|--------|-----------|-------|
| | [m]:ss | | |
| 1978 | 3:28 | 4 | 1 |
| 1993 | 3:08 | 5 | 1 |
| 1982 | 3:57 | 4 | 1 |
| 1973 | 3:00 | 4 | 1 |
| 1978 | 6:05 | 4 | 1 |
| 1973 | 5:11 | 4 | 1 |
| 2011 | 3:19 | 4 | 1 |
| 1997 | 4:00 | 4 | 1 |
| 2005 | 3:39 | 4 | 1 |
| 1995 | 4:13 | 4 | 2 |
| 1985 | 3:00 | 9 | 2 |
| 2010 | 4:38 | 7 | 2 |
| 1987 | 3:45 | 9 | 2 |
| 1988 | 3:53 | 9 | 2 |
| 1999 | 4:26 | 5 | 4 |
| 2011 | 4:47 | 11 | 4 |
| 2007 | 3:07 | 11 | 4 |
| 2005 | 3:26 | 4 | 5 |
| 1982 | 2:50 | 4 | 8 |
| 1979 | 3:56 | 5 | 9 |
| 2009 | 4:55 | 4 | 9 |

back in 1979? Is there a reason that there is a small cluster of country songs at the mid 80's? If you were to randomly pick a song for this person to listen to, what would you pick and have a high probability that they would like it?

Overlays, Marker colors, & Marker glyph style



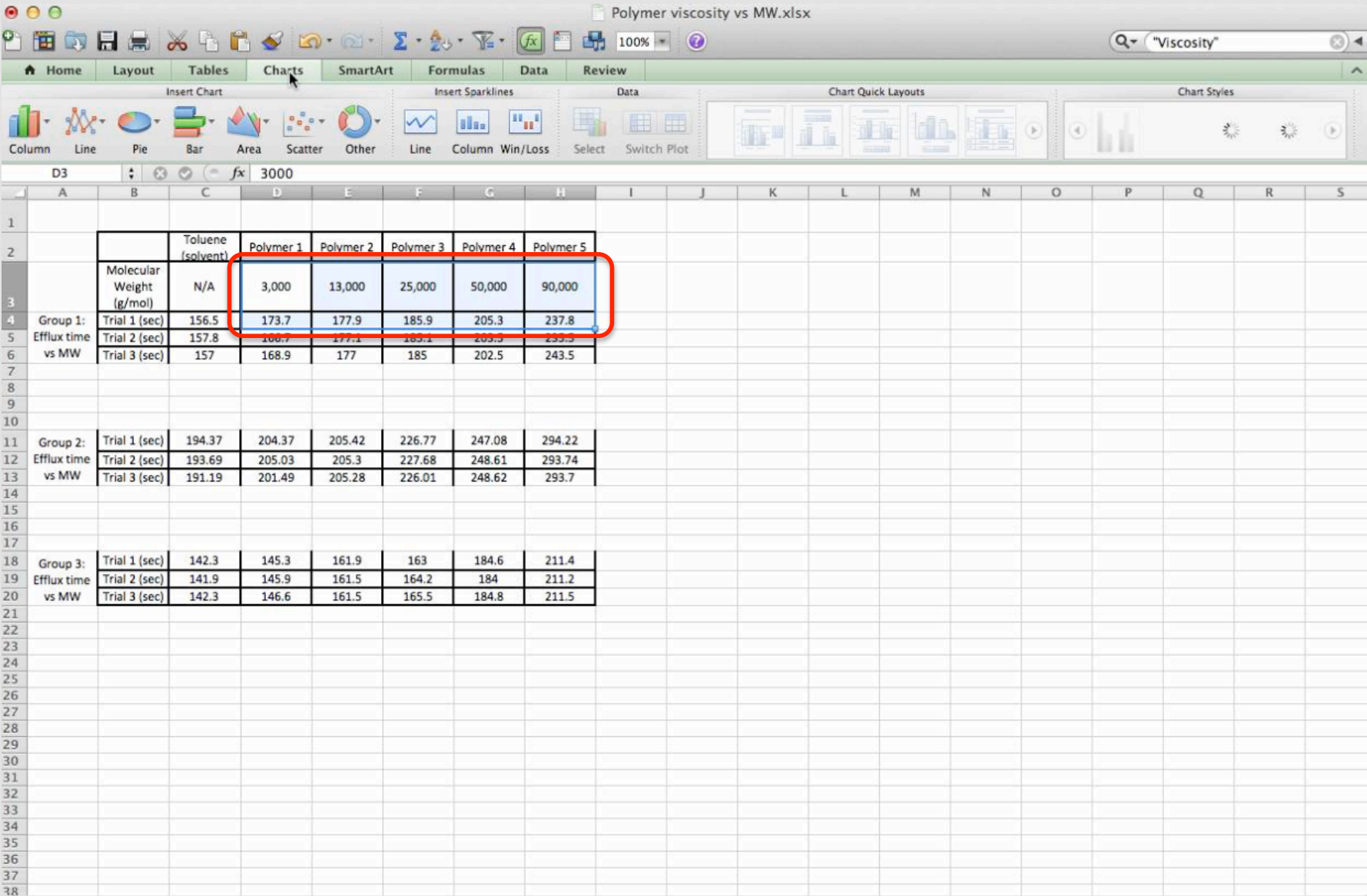
The main focus of this section will demonstrate how to overlay several data sets on the same plot within Excel. Secondly it will demonstrate how to change the marker color and glyph style. Re-sizing the Y axis will also be reviewed.

Now would be an appropriate time for the Reader to download the Excel files that will be used for the step-by-step Gallery 2.1 ([Polymer viscosity](#)) and Gallery 2.2 ([IR Spectra](#)). Click on the links and it should take you to a website where the data files

are stored. Download them (REMEMBER WHERE SAVED THEM). For additional practice exercises [Cyclic Voltammetry](#) and [Gas Chromatography](#) can be downloaded.

Let's begin with looking at Gallery 2.1. Open the Excel file **Polymer viscosity** and you should see something that resembles the first Gallery slide. Follow the slides at your own pace and perform each step. A red box highlights the portion of the Excel spreadsheet that is your current task.

Gallery 2.1 Overlay, Trim axis, Marker color, Marker style in Excel



Highlight the cells for Group 1 - Molecular Weights & Trial 1

1 of 22

After creating overlays and changing marker colors and styles, it is left up to the reader as to what style they want to make their plot. The markers can all be the same color but different style, all the same style but different colors, or different colors and different styles. Whatever you choose, I wouldn't recommend making it too foo-foo

looking. Keep it professional. Also consider whether you have a color printer or not. If you don't have a color printer, all of your colors may end up being indiscernible shades of gray. You should change your marker styles to "X", "O" (open circle), shaded circle, open/shaded boxes and/or triangles. The reader

should also add at a minimum, a Chart title, and axes labels with units. Gridlines may or may not be appropriate. If the plot is just for show, no gridlines should be used. If the plot is used for calibration purposes, then major and/or minor gridlines should be added.

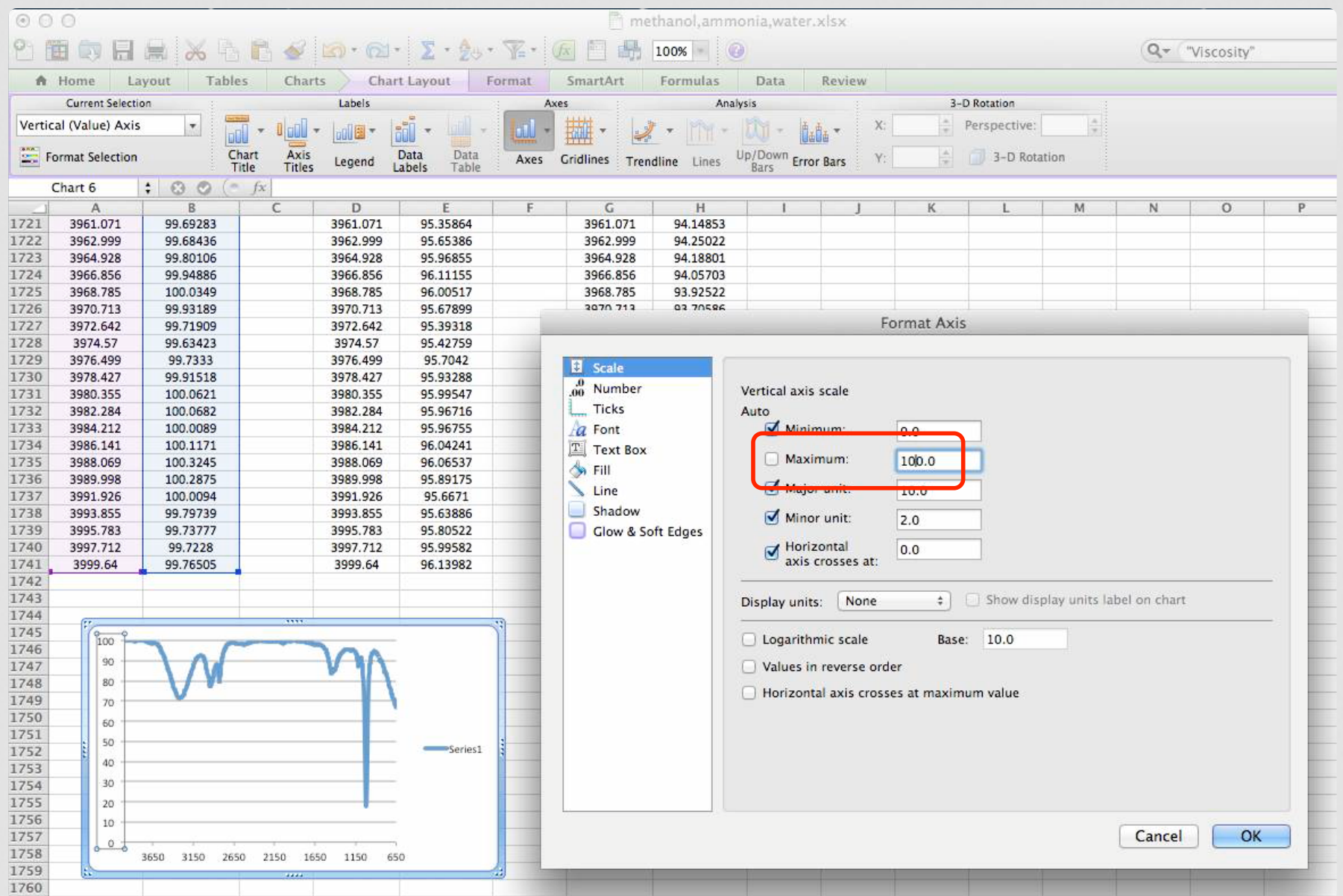
Let's move on to Gallery 2.2, an IR Spectra data set. The techniques demonstrated are very similar to Gallery 2.1 in that axes trimming will be done, BUT a couple of new techniques will be demonstrated: 1) there are over 1700 x/y data points and thus marker style is NOT appropriate. Instead we'll use "smooth lined scatter". 2) Instead of the x-axis going from small to big, we'll make it go from big to small. So, let's open up the excel file **IR Spectra** and proceed with the steps in Gallery 2.2.

After finishing Gallery 2.2, it is recommended that the Reader practice overlay and axes trimming skills with the files: [CyclicVoltammetry](#) and [GasChromatography](#) which can be downloaded from the web page that their links direct you to.

The next Section 2.2, Bubble Graphs will show you how the size of the markers can be used as an extra dimension on your Excel plots. It also introduces the idea of

converting qualitative data into quantitative data along with appropriate organization of your data columns to yield meaningful bubble graphs.

Gallery 2.2 Smooth Lined Scatter, Trim axes, Reverse x-axis in Excel

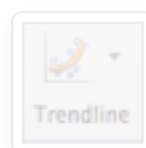
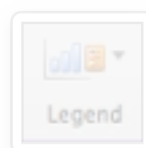
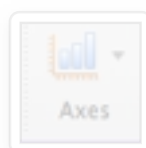
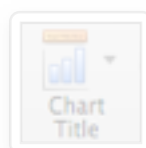
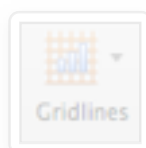
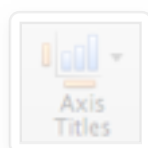
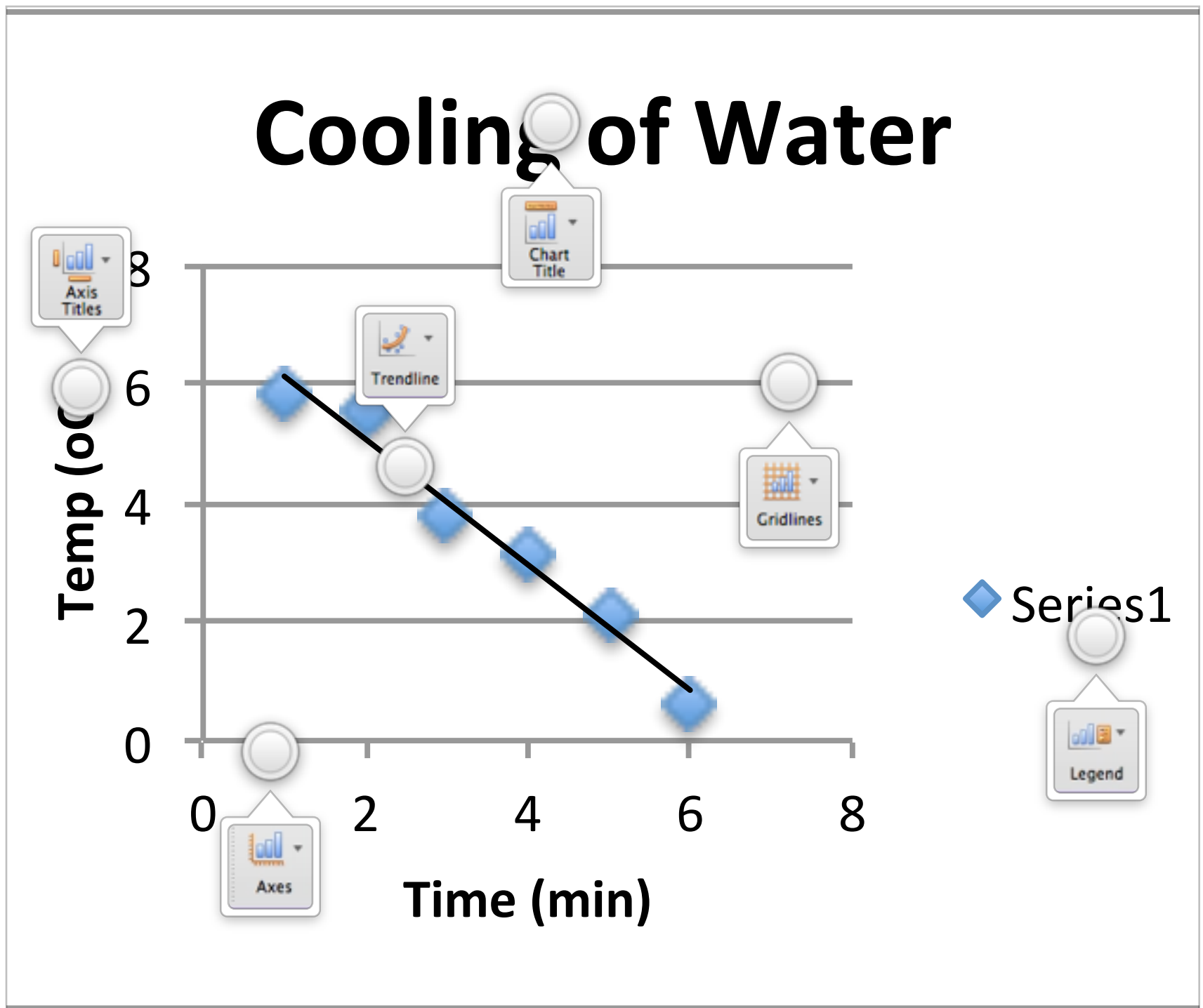


Looking at our y-data, the maximum is about 100%. Make sure you have “Scale” highlighted, change the maximum to 100.

Review 2.1 Excel Overlay Review

Question 1 of 5

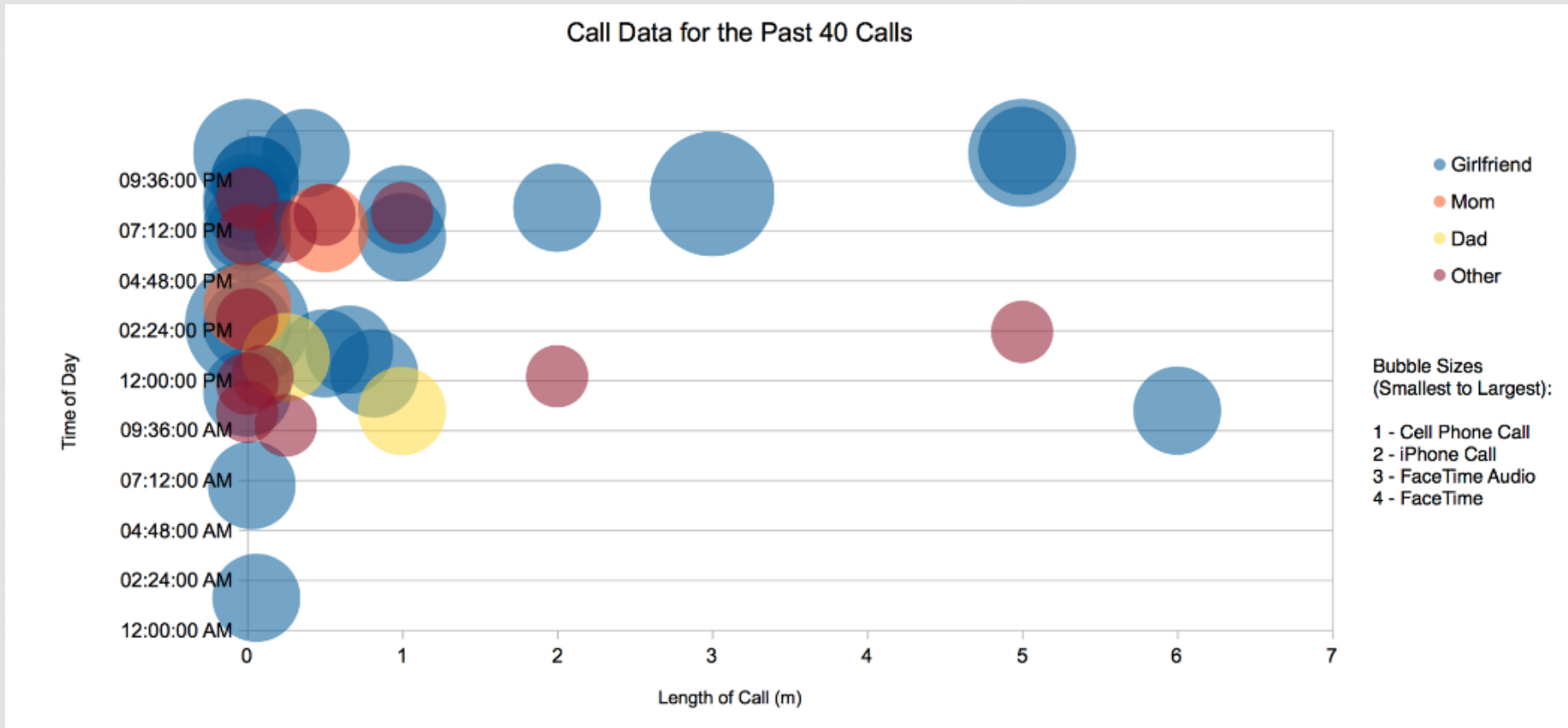
When formatting a plot, different menus are required to add, remove, or alter an item. Match the following thumbnails to the location on the plot that it modifies.



Check Answer

Bubble Graphs

Figure 2.1 Student's Call Data



From the previous Overlay Section the addition of overlays, color and marker style adds more dimensions to the typical x/y graph. Another interesting plotting style that Excel has is something called Bubble Graph. Basically the SIZE of the marker is related to some type of information. In Figure 2.1, you have four dimensions of information represented, quantitative data graphed on the x/y axes and qualitative information given by color and bubble size.

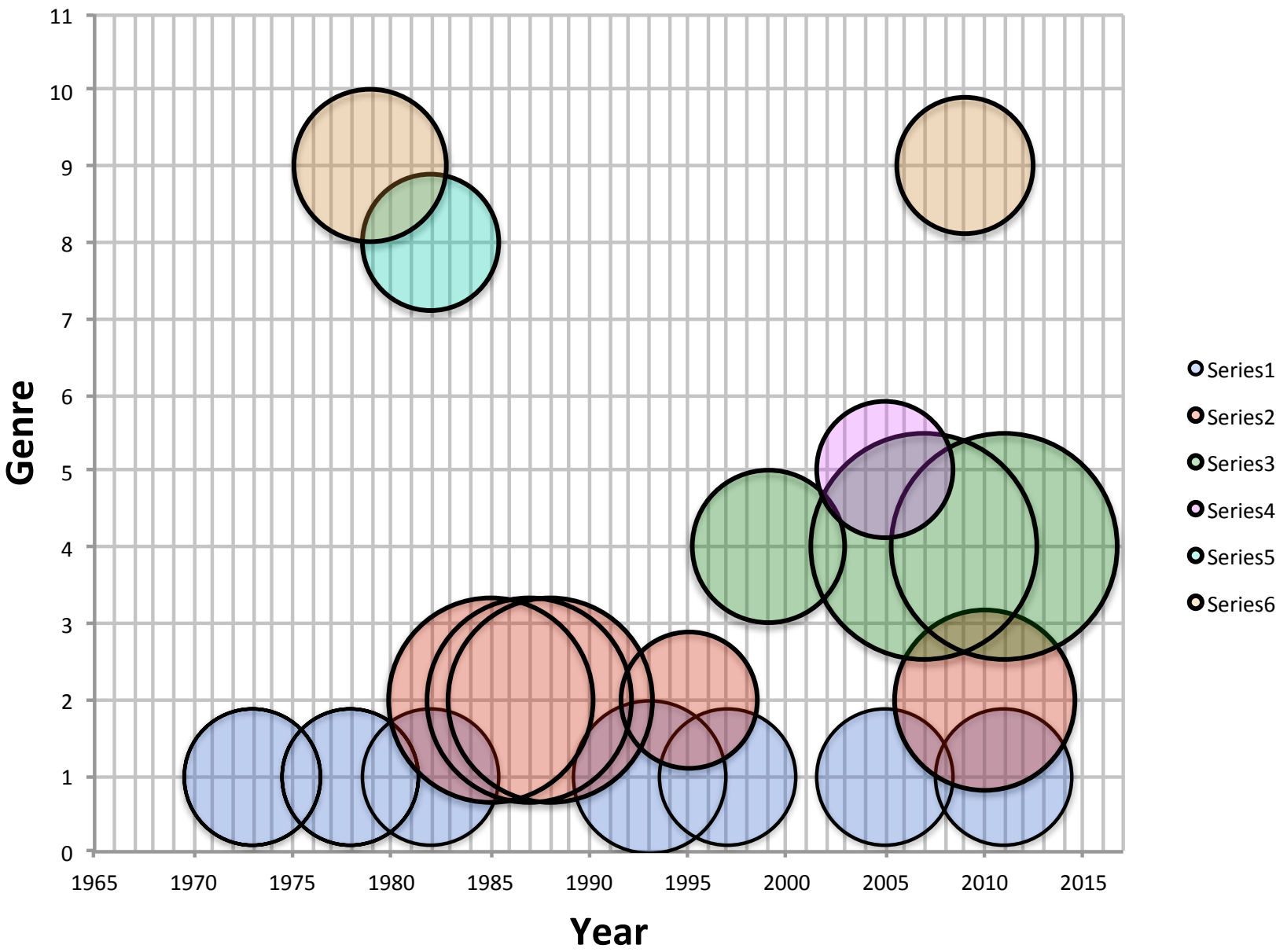
A large component of Visualization is what conclusions can you quickly interpret from the data. In Figure 2.1, the overwhelming majority of phone calls are less than a minute and a majority of them were made to his girlfriend. With a little more effort, maybe some correlations between size, color, time of day or length of call can be made.

In this section you will be shown how to make bubble graphs in Excel, which isn't

so hard, it's just a series of overlays of bubble graphs each one having it's own color. The time consuming part is generally obtaining the data and organizing it, which is true for most data sets. Take for example the Top 20 Song Playlist. What would make more sense, 1 day's worth of listening, 1 week, 1 month, 1 year? That is left up to the investigator. How do you get the numbers? Scroll through 100's of songs trying to find the ones of highest play frequency. That might take a while by hand but someone with basic computer science skills could write a short computer program that could select and organize the data in a matter of seconds.

In any event, the reader can download the Excel file [playlist](#) to work with for Gallery 2.3. The reader is encouraged to create their own data set of Top 20 Song Playlist and thus compare one person's music interest versus another. If comparing playlists, be sure to use predefined numbers and colors so that everything is consistent, you can match color for color and still be referring to the same genre. If your list does not contain a specific genre, add it as an overlay anyway by highlighting empty data cells for your X value, Y value, and Size. The genre will still be displayed in your legend, there just won't be any genre plotted.

Gallery 2.3 Bubble Graph in Excel



In this result, song length was deleted. Genre becomes the value for the Y axis. Frequency still remains the Size dimension. Each type of genre was done as a separate overlay so you get a different color for each genre. Is this representation better than Slide 21?

3

ImageJ & Mathematica: Images as Data

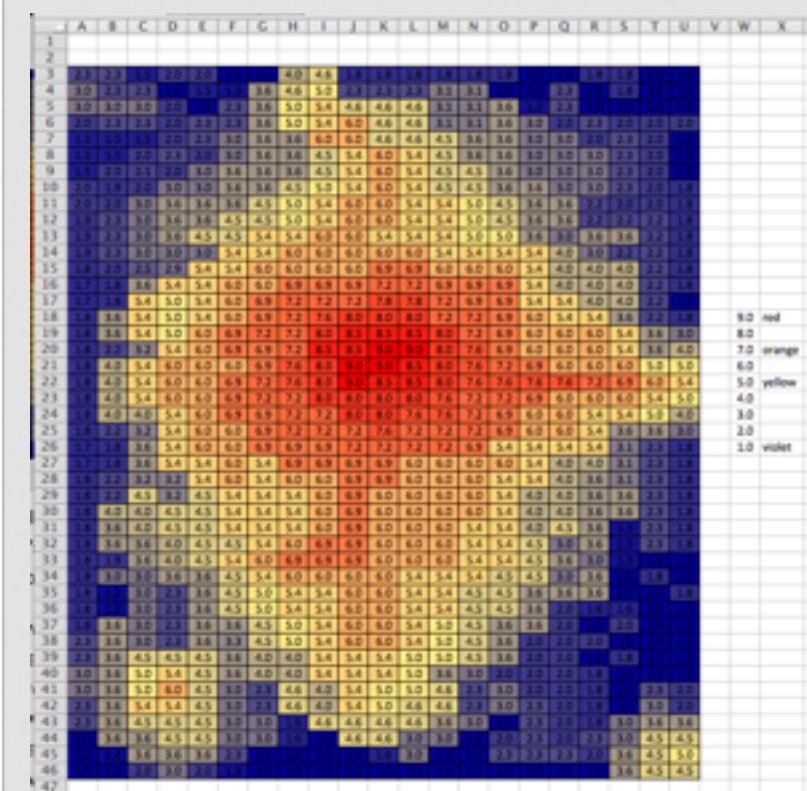
Image 3.1 Excel grid with numbers simulating an image

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X |
|----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|--------|---|
| 1 | | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | 2.3 | 2.3 | 1.5 | 2.0 | 2.0 | 1.0 | 1.0 | 4.0 | 4.6 | 1.8 | 1.8 | 1.8 | 1.8 | 1.8 | 1.0 | 1.0 | 1.8 | 1.8 | 1.0 | 1.0 | | | | |
| 4 | 3.0 | 2.3 | 2.3 | 1.0 | 1.5 | 1.3 | 3.6 | 4.6 | 5.0 | 2.3 | 2.3 | 2.3 | 3.1 | 3.1 | 1.0 | 1.0 | 2.3 | 1.0 | 1.8 | 1.0 | 1.0 | | | |
| 5 | 3.0 | 3.0 | 3.0 | 2.0 | 1.0 | 2.3 | 3.6 | 5.0 | 5.4 | 4.6 | 4.6 | 4.6 | 3.1 | 3.1 | 3.6 | 1.3 | 2.3 | 1.0 | 1.0 | 1.0 | 1.0 | | | |
| 6 | 2.0 | 2.3 | 2.3 | 2.0 | 2.3 | 2.3 | 3.6 | 5.0 | 5.4 | 6.0 | 4.6 | 4.6 | 3.1 | 3.1 | 3.6 | 3.0 | 2.0 | 2.3 | 2.0 | 2.0 | 2.0 | | | |
| 7 | 1.5 | 1.5 | 1.5 | 2.0 | 2.3 | 3.0 | 3.6 | 3.6 | 6.0 | 6.0 | 4.6 | 4.6 | 4.5 | 3.6 | 3.6 | 3.0 | 3.0 | 2.0 | 2.3 | 2.0 | 1.0 | | | |
| 8 | 1.5 | 1.5 | 2.0 | 2.3 | 2.0 | 3.0 | 3.6 | 3.6 | 4.5 | 5.4 | 6.0 | 5.4 | 4.5 | 3.6 | 3.6 | 3.0 | 3.0 | 3.0 | 2.3 | 2.0 | 1.0 | | | |
| 9 | 1.5 | 2.0 | 2.1 | 2.0 | 3.0 | 3.6 | 3.6 | 3.6 | 4.5 | 5.4 | 6.0 | 5.4 | 4.5 | 4.5 | 3.6 | 3.0 | 3.0 | 3.0 | 2.3 | 2.0 | 1.0 | | | |
| 10 | 2.0 | 1.9 | 2.0 | 3.0 | 3.0 | 3.6 | 3.6 | 4.5 | 5.0 | 5.4 | 6.0 | 5.4 | 4.5 | 4.5 | 3.6 | 3.6 | 3.0 | 3.0 | 2.3 | 2.0 | 1.8 | | | |
| 11 | 2.0 | 2.0 | 3.0 | 3.6 | 3.6 | 3.6 | 4.5 | 5.0 | 5.4 | 6.0 | 6.0 | 5.4 | 5.4 | 5.0 | 4.5 | 3.6 | 3.6 | 2.2 | 2.0 | 2.0 | 1.8 | | | |
| 12 | 1.9 | 2.3 | 3.0 | 3.6 | 3.6 | 4.5 | 4.5 | 5.0 | 5.4 | 6.0 | 6.0 | 5.4 | 5.4 | 5.0 | 4.5 | 3.6 | 3.6 | 2.2 | 2.2 | 2.2 | 1.8 | | | |
| 13 | 1.9 | 2.3 | 3.0 | 3.6 | 4.5 | 4.5 | 5.4 | 5.4 | 6.0 | 6.0 | 5.4 | 5.4 | 5.4 | 5.0 | 5.0 | 3.6 | 3.0 | 3.6 | 3.6 | 2.2 | 1.8 | | | |
| 14 | 1.9 | 2.3 | 3.0 | 3.0 | 3.0 | 5.4 | 5.4 | 6.0 | 6.0 | 6.0 | 6.0 | 6.0 | 5.4 | 5.4 | 5.4 | 5.4 | 4.0 | 3.0 | 2.2 | 2.2 | 1.8 | | | |
| 15 | 1.8 | 2.0 | 2.5 | 2.9 | 5.4 | 5.4 | 6.0 | 6.0 | 6.0 | 6.0 | 6.9 | 6.9 | 6.0 | 6.0 | 6.0 | 5.4 | 4.0 | 4.0 | 4.0 | 2.2 | 1.8 | | | |
| 16 | 1.7 | 1.8 | 3.6 | 5.4 | 5.4 | 6.0 | 6.0 | 6.9 | 6.9 | 6.9 | 7.2 | 7.2 | 6.9 | 6.9 | 6.9 | 5.4 | 4.0 | 4.0 | 4.0 | 2.2 | 1.8 | | | |
| 17 | 1.7 | 1.8 | 5.4 | 5.0 | 5.4 | 6.0 | 6.9 | 7.2 | 7.2 | 7.2 | 7.8 | 7.8 | 7.2 | 6.9 | 6.9 | 5.4 | 5.4 | 4.0 | 4.0 | 2.2 | 1.0 | | | |
| 18 | 1.8 | 3.6 | 5.4 | 5.0 | 5.4 | 6.0 | 6.9 | 7.2 | 7.6 | 8.0 | 8.0 | 8.0 | 7.2 | 7.2 | 6.9 | 6.0 | 5.4 | 5.4 | 3.6 | 2.2 | 1.8 | 9.0 | red | |
| 19 | 1.8 | 3.6 | 5.4 | 5.0 | 6.0 | 6.9 | 7.2 | 7.2 | 8.0 | 8.5 | 8.5 | 8.5 | 8.0 | 7.2 | 6.9 | 6.0 | 6.0 | 6.0 | 5.4 | 3.6 | 3.0 | 8.0 | | |
| 20 | 1.8 | 2.2 | 3.2 | 5.4 | 6.0 | 6.9 | 6.9 | 7.2 | 8.5 | 8.5 | 9.0 | 9.0 | 8.0 | 7.6 | 6.9 | 6.0 | 6.0 | 6.0 | 5.4 | 3.6 | 4.0 | 7.0 | orange | |
| 21 | 1.8 | 4.0 | 5.4 | 6.0 | 6.0 | 6.0 | 6.9 | 7.6 | 8.0 | 9.0 | 9.0 | 8.5 | 8.0 | 7.6 | 7.2 | 6.9 | 6.0 | 6.0 | 6.0 | 5.0 | 5.0 | 6.0 | | |
| 22 | 1.8 | 4.0 | 5.4 | 6.0 | 6.0 | 6.9 | 7.2 | 7.6 | 8.0 | 9.0 | 8.5 | 8.5 | 8.0 | 7.6 | 7.6 | 7.6 | 7.6 | 7.2 | 6.9 | 6.0 | 5.4 | 5.0 | 5.0 | |
| 23 | 1.8 | 4.0 | 5.4 | 6.0 | 6.0 | 6.9 | 7.2 | 7.2 | 8.0 | 8.0 | 8.0 | 8.0 | 7.6 | 7.2 | 7.2 | 6.9 | 6.0 | 6.0 | 6.0 | 5.4 | 5.0 | 4.0 | | |
| 24 | 1.8 | 4.0 | 4.0 | 5.4 | 6.0 | 6.9 | 6.9 | 7.2 | 7.2 | 8.0 | 8.0 | 7.6 | 7.6 | 7.2 | 6.9 | 6.0 | 6.0 | 5.4 | 5.4 | 5.0 | 4.0 | 3.0 | | |
| 25 | 1.8 | 2.2 | 3.2 | 5.4 | 6.0 | 6.0 | 6.9 | 7.2 | 7.2 | 7.2 | 7.6 | 7.2 | 7.2 | 7.2 | 6.9 | 6.0 | 6.0 | 5.4 | 3.6 | 3.6 | 3.0 | 2.0 | | |
| 26 | 1.7 | 1.8 | 3.6 | 5.4 | 6.0 | 6.0 | 6.9 | 6.9 | 6.9 | 7.2 | 7.2 | 7.2 | 7.2 | 6.9 | 5.4 | 5.4 | 5.4 | 5.4 | 3.1 | 2.3 | 1.8 | 1.0 | violet | |
| 27 | 1.7 | 1.8 | 3.6 | 5.4 | 5.4 | 6.0 | 5.4 | 6.9 | 6.9 | 6.9 | 6.9 | 6.0 | 6.0 | 6.0 | 6.0 | 5.4 | 4.0 | 4.0 | 3.1 | 2.3 | 1.8 | | | |
| 28 | 1.8 | 2.2 | 3.2 | 3.2 | 5.4 | 6.0 | 5.4 | 6.0 | 6.0 | 6.9 | 6.9 | 6.0 | 6.0 | 6.0 | 5.4 | 5.4 | 4.0 | 3.6 | 3.1 | 2.3 | 1.8 | | | |
| 29 | 1.8 | 2.2 | 4.5 | 3.2 | 4.5 | 5.4 | 5.4 | 5.4 | 6.0 | 6.9 | 6.0 | 6.0 | 6.0 | 6.0 | 5.4 | 4.0 | 4.0 | 3.6 | 3.6 | 2.3 | 1.8 | | | |
| 30 | 1.8 | 4.0 | 4.0 | 4.5 | 4.5 | 5.4 | 5.4 | 5.4 | 6.0 | 6.9 | 6.0 | 6.0 | 6.0 | 6.0 | 5.4 | 4.0 | 4.0 | 3.6 | 3.6 | 2.3 | 1.8 | | | |
| 31 | 1.8 | 3.6 | 4.0 | 4.5 | 4.5 | 5.4 | 5.4 | 5.4 | 6.0 | 6.9 | 6.0 | 6.0 | 6.0 | 5.4 | 4.0 | 4.5 | 3.6 | 1.3 | 2.3 | 1.8 | | | | |
| 32 | 1.8 | 3.6 | 3.6 | 4.0 | 4.5 | 4.5 | 5.4 | 6.0 | 6.9 | 6.9 | 6.0 | 6.0 | 6.0 | 5.4 | 5.4 | 4.5 | 3.0 | 3.6 | 1.3 | 2.3 | 1.8 | | | |
| 33 | 1.8 | 1.9 | 3.6 | 4.0 | 4.5 | 5.4 | 6.0 | 6.9 | 6.9 | 6.9 | 6.0 | 6.0 | 6.0 | 5.4 | 5.4 | 4.5 | 3.6 | 3.0 | 1.3 | 1.0 | 1.0 | | | |
| 34 | 1.8 | 3.0 | 3.0 | 3.6 | 3.6 | 4.5 | 5.4 | 6.0 | 6.0 | 6.0 | 6.0 | 5.4 | 5.4 | 5.4 | 4.5 | 4.5 | 3.0 | 3.6 | 1.0 | 1.8 | 1.0 | | | |
| 35 | 1.8 | 1.2 | 3.0 | 2.3 | 3.6 | 4.5 | 5.0 | 5.4 | 5.4 | 6.0 | 6.0 | 5.4 | 5.4 | 4.5 | 4.5 | 3.6 | 3.6 | 3.6 | 1.0 | 1.0 | 1.8 | | | |
| 36 | 1.8 | 1.2 | 3.0 | 2.3 | 3.6 | 4.5 | 5.0 | 5.4 | 5.4 | 6.0 | 6.0 | 5.4 | 5.4 | 4.5 | 4.5 | 3.6 | 2.0 | 1.6 | 1.6 | 1.0 | 1.0 | | | |
| 37 | 1.8 | 3.6 | 3.0 | 2.3 | 3.6 | 3.6 | 4.5 | 5.0 | 5.4 | 6.0 | 6.0 | 5.4 | 5.0 | 4.5 | 3.6 | 3.6 | 2.0 | 1.0 | 2.0 | 1.0 | 1.0 | | | |
| 38 | 2.3 | 3.6 | 3.0 | 2.3 | 3.6 | 3.3 | 4.5 | 5.0 | 5.4 | 6.0 | 6.0 | 5.4 | 5.0 | 4.5 | 3.6 | 2.0 | 2.0 | 2.0 | 1.0 | 1.0 | 1.0 | | | |
| 39 | 2.3 | 3.6 | 4.5 | 4.5 | 4.5 | 3.6 | 4.0 | 4.0 | 5.4 | 5.4 | 5.4 | 5.0 | 5.0 | 4.5 | 3.6 | 2.0 | 2.0 | 1.0 | 1.8 | 1.0 | 1.0 | | | |
| 40 | 3.0 | 3.6 | 5.0 | 5.4 | 4.5 | 3.6 | 4.0 | 4.0 | 5.4 | 5.4 | 5.0 | 3.6 | 3.0 | 2.0 | 2.0 | 2.0 | 1.8 | 1.0 | 1.0 | 1.0 | 1.0 | | | |
| 41 | 3.0 | 3.6 | 5.0 | 6.0 | 4.5 | 3.0 | 2.3 | 4.6 | 4.0 | 5.4 | 5.0 | 5.0 | 4.6 | 2.3 | 3.0 | 2.0 | 2.0 | 1.8 | 1.0 | 2.3 | 2.0 | | | |
| 42 | 2.3 | 3.6 | 5.4 | 5.4 | 4.5 | 3.0 | 2.3 | 4.6 | 4.0 | 5.4 | 5.0 | 4.6 | 4.6 | 2.3 | 3.0 | 2.3 | 2.0 | 1.8 | 1.0 | 3.0 | 2.0 | | | |
| 43 | 2.3 | 3.6 | 4.5 | 4.5 | 4.5 | 3.0 | 3.0 | 1.6 | 4.6 | 4.6 | 4.6 | 4.6 | 3.6 | 3.0 | 1.0 | 2.3 | 2.0 | 1.8 | 3.0 | 3.6 | 3.6 | | | |
| 44 | 1.0 | 3.6 | 3.6 | 4.5 | 4.5 | 3.0 | 3.0 | 1.6 | 1.0 | 4.6 | 4.6 | 3.0 | 3.0 | 1.0 | 2.0 | 2.3 | 2.0 | 2.3 | 3.0 | 4.5 | 4.5 | | | |
| 45 | 1.0 | 1.2 | 3.6 | 3.6 | 3.6 | 2.3 | 1.0 | 1.0 | 1.0 | 1.0 | 1.6 | 3.0 | 1.0 | 1.0 | 2.3 | 2.3 | 2.3 | 2.0 | 3.6 | 4.5 | 5.0 | | | |
| 46 | 1.0 | 1.0 | 2.0 | 3.0 | 2.0 | 1.3 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 3.6 | 4.5 | 4.5 | | | |
| 47 | | | | | | | | | | | | | | | | | | | | | | | | |

Images themselves are just a collection of numbers derived from pixel intensities. These numerical values are then assigned a color. So in theory, any set of numbers could be displayed as an image. As a quick and “simple” example, the Excel grid

in Image 3.1 is colored to yield the image in Image 3.2. Given that these pixels have numerical values and x,y coordinates within the image, they can be subjected to mathematical operations (counted,

Image 3.2 Colorized grid of Image 3.1



process, I would also create a convenient desktop icon for ImageJ.

located, add/subtracted, multiply/divided, logic operations, to name a few).

This chapter will devote itself to the many ways that ImageJ can be applied to images. I have found it to be a really handy piece of software (freeware) that lets you perform tasks of a numerical nature upon images. You will need ImageJ to perform several tasks in later Sections, you may want to download and install it on your computer now. The easiest way I have found to do it is to use a search engine for “[ImageJ](#)”. At their homepage there are a variety of versions, one of which should match your O/S. They also have a lot of useful documentation that you might find handy. In the download and installation

Dissecting an Image




Image
Types

If you have some interest in video games you probably have heard that back in the early days of Pacman and Space Invaders the games were “8-bit” or “16-bit”. And unless you have a very old computer, you are running a “32-bit” or “64-bit” operating system on your computer. So what are all these “bits” and why are there 16 or 32 of them? At some point you have probably wondered what exactly a bit is, and in this section I am going to explain that as well

as what the heck it has to do with visualization.

Bit by Bit

Let’s start off with defining the word bit. A bit is the smallest unit of digital information, and it can have a value of either “0” or “1”. This means that the bit is binary, or can have only two values. So how do computers get larger values?

Computers combine bits together into bytes to get more than just the two values

a single bit provides. If you combine two bits you then get four possible values: “00”, “01”, “10”, and “11”. If you continue to add more bits you get exponentially more possible values. Three bits provides you with eight values: “000”, ”001”, ”010”, ”011”, ”100”, ”101”, “110”, and ”111”. Table 3.1 shows the possible values with standard byte sizes.

| Table 3.1 Bit Sizes and Numerical Values | | |
|--|---------------------------|---------------------------------|
| Size of Bit | Number of Possible Values | Maximum Value (zero is a value) |
| 1 | 2 | 1 |
| 2 | 4 | 3 |
| 4 | 16 | 15 |
| 8 | 256 | 255 |
| 16 | 65536 | 65535 |
| 32 | 4294967296 | 4294967295 |
| 64 | 2^64 | 2^64 - 1 |

This means that different numbers, often referred to as values in the world of data, require different amounts of bits to process. For example, the number sixteen would need an 8-bit byte because it is greater than 15 (the maximum value possible with only 4 bits). As another example using the same reasoning and provided table, we can determine the

number 1,764,567 would need a 32-bit byte.

Pixels

Pixel is a commonly used in relation to computer monitors and image sizes, but what a pixel is does not get much thought. A pixel, by definition, is a single point in an image that has a value and an address. The value of the pixel is what a program uses to determine how to display the pixel (color and opacity) and the address tells the computer where to display the pixel inside the image. When the computer puts a bunch of pixels together with certain values it makes the images that we are accustomed to seeing on computers.

Image Types

The type of an image is determined by the maximum bit size of a pixel value within an image. For example, an 8-bit image means that the greatest possible value of any pixel within the image is 255 assuming the lowest number is 0 (meaning there are 256 total different values). You can also have an 8-bit binary image, meaning that there are only minimum 0-values and maximum 255-values within the image.

In some imaging processing programs a 32-bit value is known as a “float” value. This means that instead of going from 0 to

the maximum 32-bit value each pixel can have a value on a scale from -1 to 1 with decimal values in-between.

You can convert between image types to achieve various results. If you convert from a smaller bit size to a greater bit size (i.e. 8-bit to 32-bit) you will not see any difference in an image. That is because a value within that fits within 8-bits will easily fit within 32-bits, so no change in values is required. If you do the opposite and convert a greater bit-size to a smaller bit-size (i.e. 32-bit to 8-bit) you will lose some image quality. This is because the values in a 32-bit image can have 4294967296 different values while an 8-bit image can only have 256 different values. Some contrast will be lost because the differences between values will be compressed due to having a relatively smaller range.

RGB Images

An Red-Green-Blue (RGB) image is an image **type** where each pixel actually has three values: a red value, a green value, and a blue value. In the standard format each of the color values is an 8-bit value, making the RGB image type 24-bit ($8 \cdot 3 = 24$). This allows for a multitude of combinations of red, green, and blue that

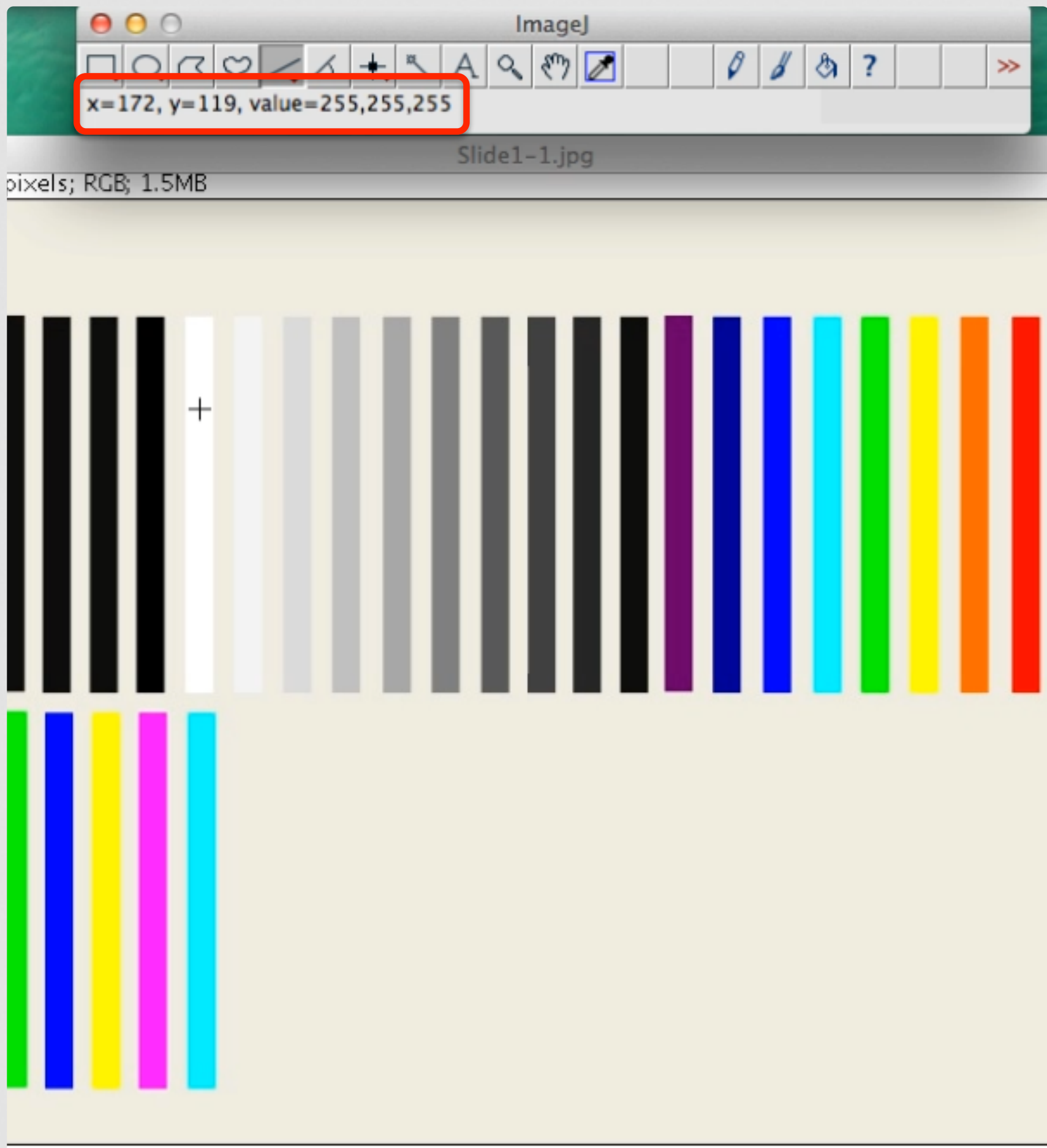
can produce most colors. In fact it provides 2^{24} (or roughly 16.8million) colors.

Another important aspect of RGB images is that you can separate the red, green, and blue pixel values for each pixel and get three different 8-bit images. Each of these images are called *channels*.

Separating RGB images into channels is a useful tool for image processing that many people use.

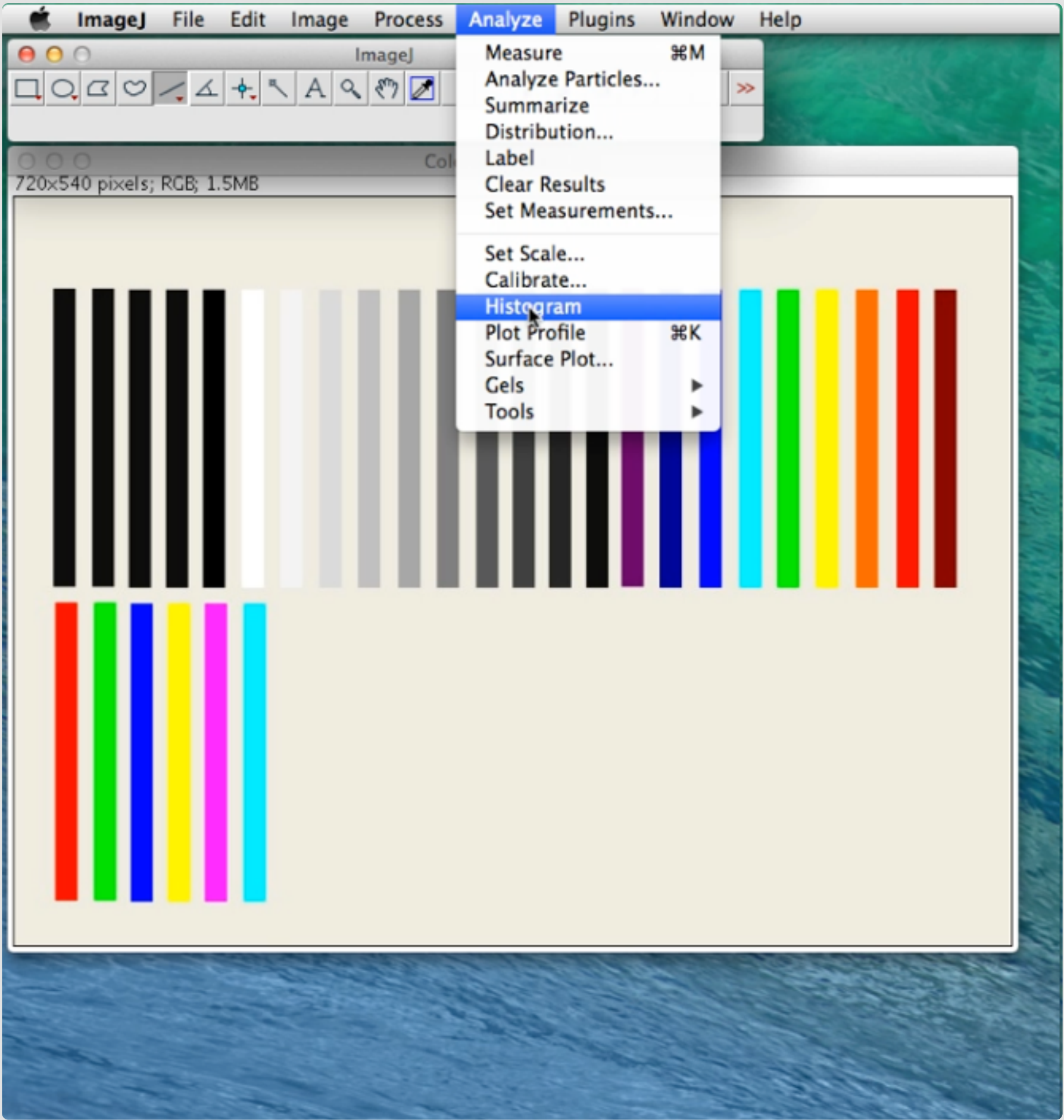
Having downloaded and installed ImageJ, download the file called [Colors&Gradient](#). REMEMBER WHERE YOU SAVED IT. Use ImageJ to open the file. Placing your cursor on various locations of the image, you'll see some numbers change on the ImageJ tool bar. Each pixel has an x,y coordinate and RGB value. Knowing where to find these values and what they are will be helpful in future exercises. Gallery 3.1 shows you different x, y, R, G, and B values for different colors on the image. For additional knowledge you should determine where in the image the (0,0) coordinate and maximum (x,y) coordinates are. Also, determine RGB values for various other colors and shades of gray. In Gallery 3.2, you will be shown how to do grayscale, R, G, and B histograms for the overall image and for a subset area of the image.

Gallery 3.1 X,Y coordinates and RGB in ImageJ.



Using the “Colors&Gradient” image, the cursor is placed on the white color bar. Notice the x, coordinate and the RGB values that are highlighted by the red box. Being white, R, G, and B should be at their maximum value of 255.

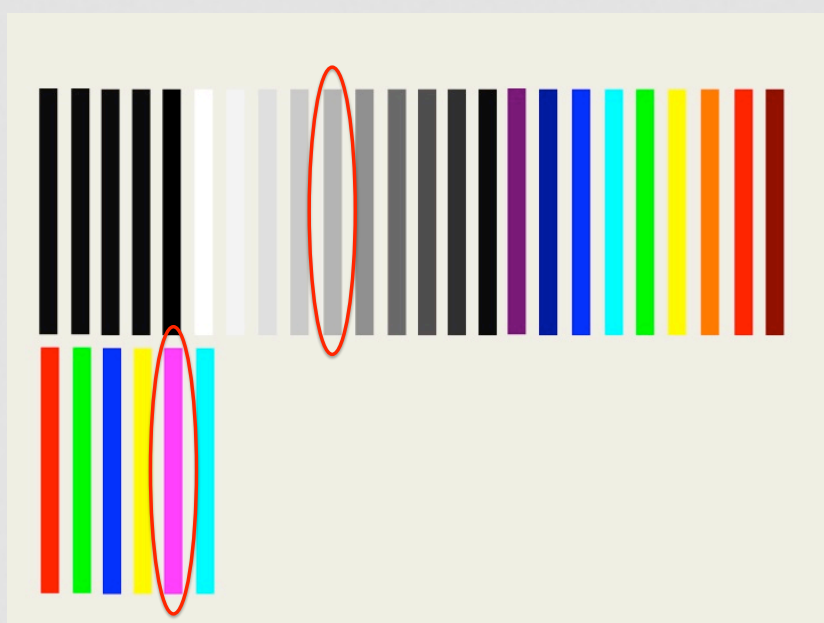
Gallery 3.2 Histograms in ImageJ.



Using the “Colors&Gradient image, go to the Analyze menu and from that select Histogram.

At this time we should define what grayscale means. This will be important for Section 3.2. Beyond the fact that the image will appear gray, what will happen is that the R, G, and B values will be averaged together. The result should fall between 0-255 with 0 being black and 255 being white. Anything in between will appear as varying shades of gray. Take for example the magenta colored strip in Image 3.3 from the Colors&Gradient image. ImageJ tells us that $R = 255$, $G = 0$, and $B = 252$. The average is 169. This is almost an exact RGB match with the gray strip circled in Image 3.3. ImageJ measures the gray strip as R, G, and B = 166.

Image 3.3 If magenta were converted to grayscale.



Once you obtain a Histogram and the corresponding List, many things can be calculated. An example might be, what % of a Monarch butterfly's wing is orange? You could simply take the number of orange pixels and divide it by the number of pixels of the overall butterfly.

This brings this Section to a close. Future Sections will make use of definitions and menus demonstrated thus far. As a short review of some of the concepts covered in this Section, answer the questions given in Review 3.1.

Review 3.1 Checkpoint

Question 1 of 6

Using the Table 3.1, determine the bit-size of the number 40,637

- ☐ A. 2-bits
- ☐ B. 8-bits
- ☒ C. 16-bits
- ☐ D. 32-bits

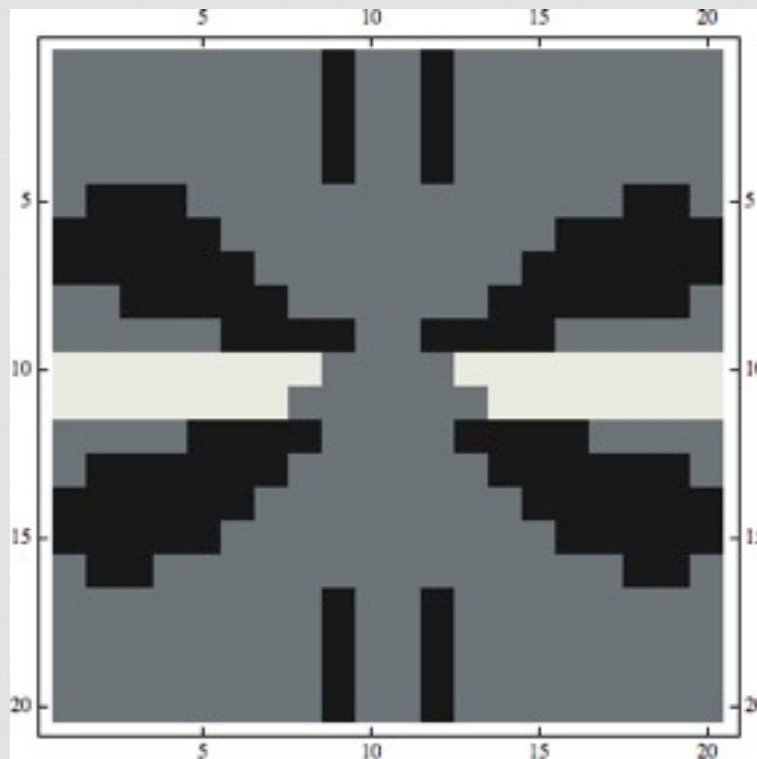


Check Answer



ImageJ Subtraction

Image 3.4 The difference between an asterisk image and a plus sign image



While waiting in the Doctor's office, I would amuse myself by working visual puzzles. One of my favorites was finding differences between two pictures. I would systematically go back and forth between both pictures, area by area and circle the differences as I found them. It would take quite a while.

Finding differences between images has real life applications. Examples such as observing changes in a person's face and

extrapolating to an older age or tracking a storm using images. Changes in images may be due to a natural process or artificially induced. Many professions routinely alter pictures to make images that look "prettier".

This section will explore the ability to treat images as sets of numbers. Using ImageJ and Mathematica, you will manipulate these numbers to see how an image will change.

Although the data sets employed in this chapter are in the form of images, the fundamental concepts can be applied to any collection of numerical data generated from a scientific experiment. An experimentalist would typically change an independent variable and see what its effect it has upon the dependent variable. Since data sets are typically composed of numbers, it wouldn't be uncommon to "subtract" two data sets from each other to see how the outcome has changed by altering a variable.

To help you see these changes, images (data sets) are quite convenient to use.

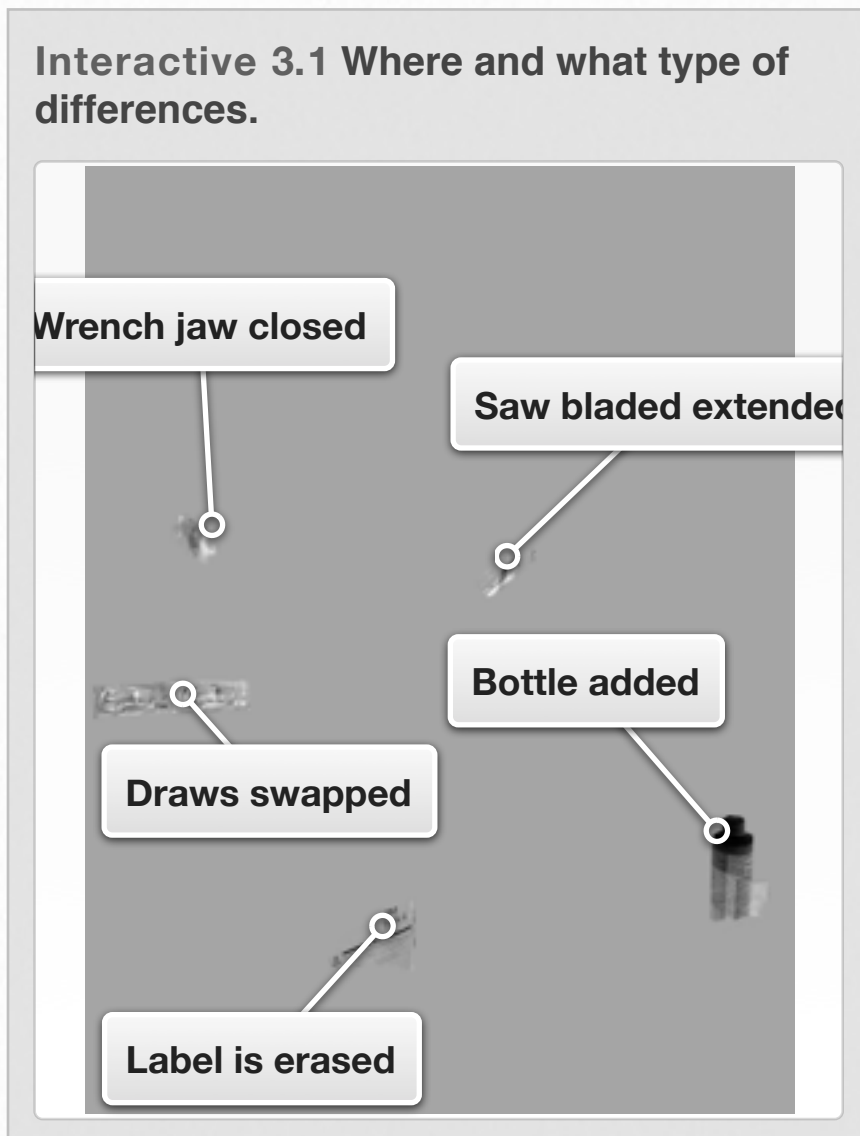
Look at two pictures in Image 3.5 and find as many differences as possible. Give yourself 20 seconds, no cheating!!! Hint: the alterations can take on many forms: add, remove, change color, change size/shape, switch, and slide left/right/up/down.

20 seconds was not much time. How many did you find? There are 6 changes. How long do you think it would have taken you to find all 6? Within 20 seconds, software such as ImageJ or Mathematica would allow us to find these subtle differences easily. Applying ImageJ subtract operation with gray scale result

Image 3.5 Messy Garage before and after alterations.



yields Interactive 3.1. There are 5 changes easily seen. Using some extended techniques a 6th one is more readily seen and highlighted in Image 3.6.



The 6th change doesn't show up in the gray scale result very well.....unless you expand the image on the iPad (and hold it in the expanded view with two fingers) and wiggle the gray image back and forth rapidly. You might then notice a faint ghost image wiggling that hasn't been pinpointed. It's just to the left of the "Saw blade extended" bubble. In the color image, it's above the saw blade and just to

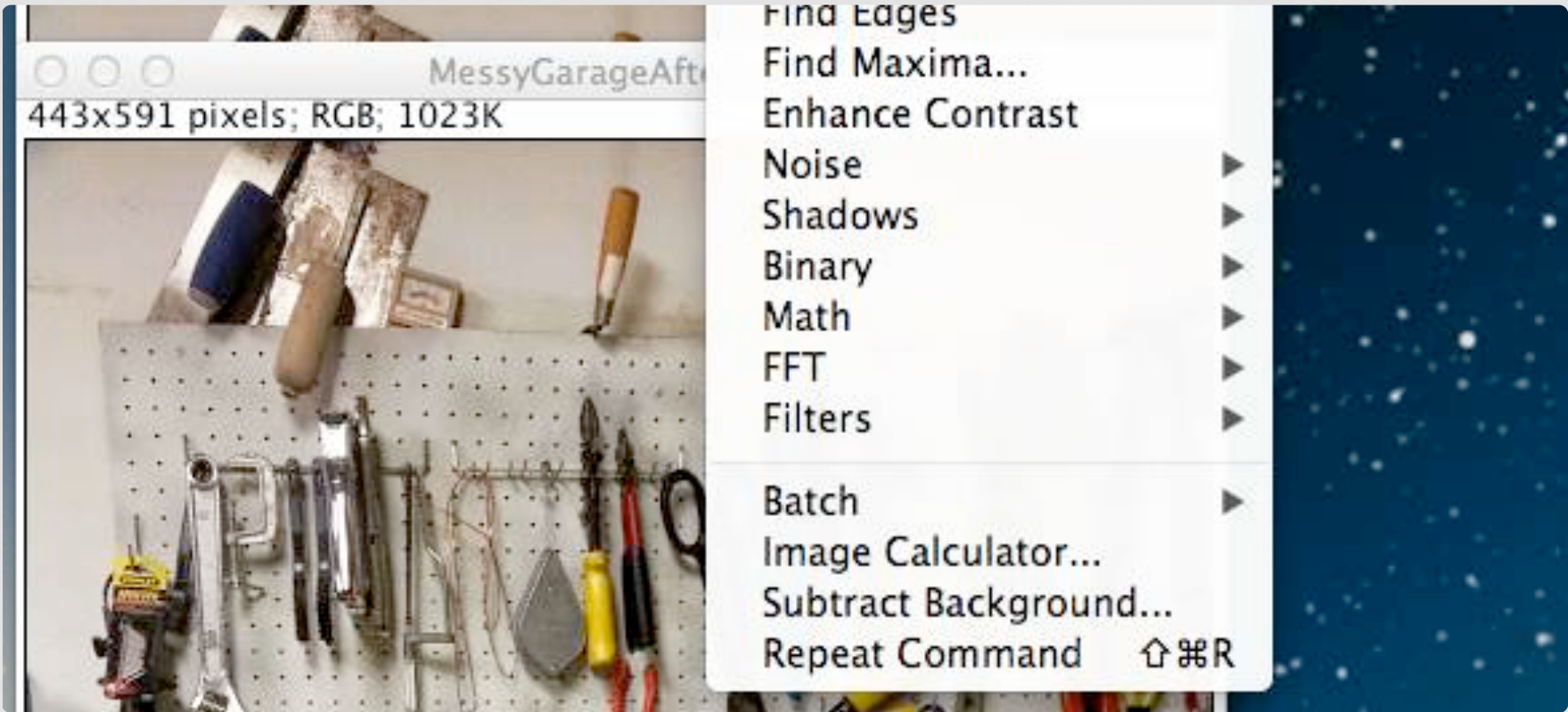
the right of the blue-handled tool. The bike lock has been changed from green to red.

Image 3.6 Messy Garage with outlines of alterations.

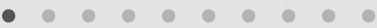


But, let's show you the step by step method of creating the gray scale resultant shown in Interactive 3.1 using ImageJ by going through Gallery 3.3 (or Movie 3.1). At this time you should download and install ImageJ onto your computer. It would be convenient to create a desktop icon for it. You should also download the [MessyGarage Before and After](#) files. REMEMBER WHERE YOU SAVE THEM!! After installing ImageJ, you should use ImageJ to open both the before and after MessyGarage files. An animated version of the technique is shown in Movie 3.1.

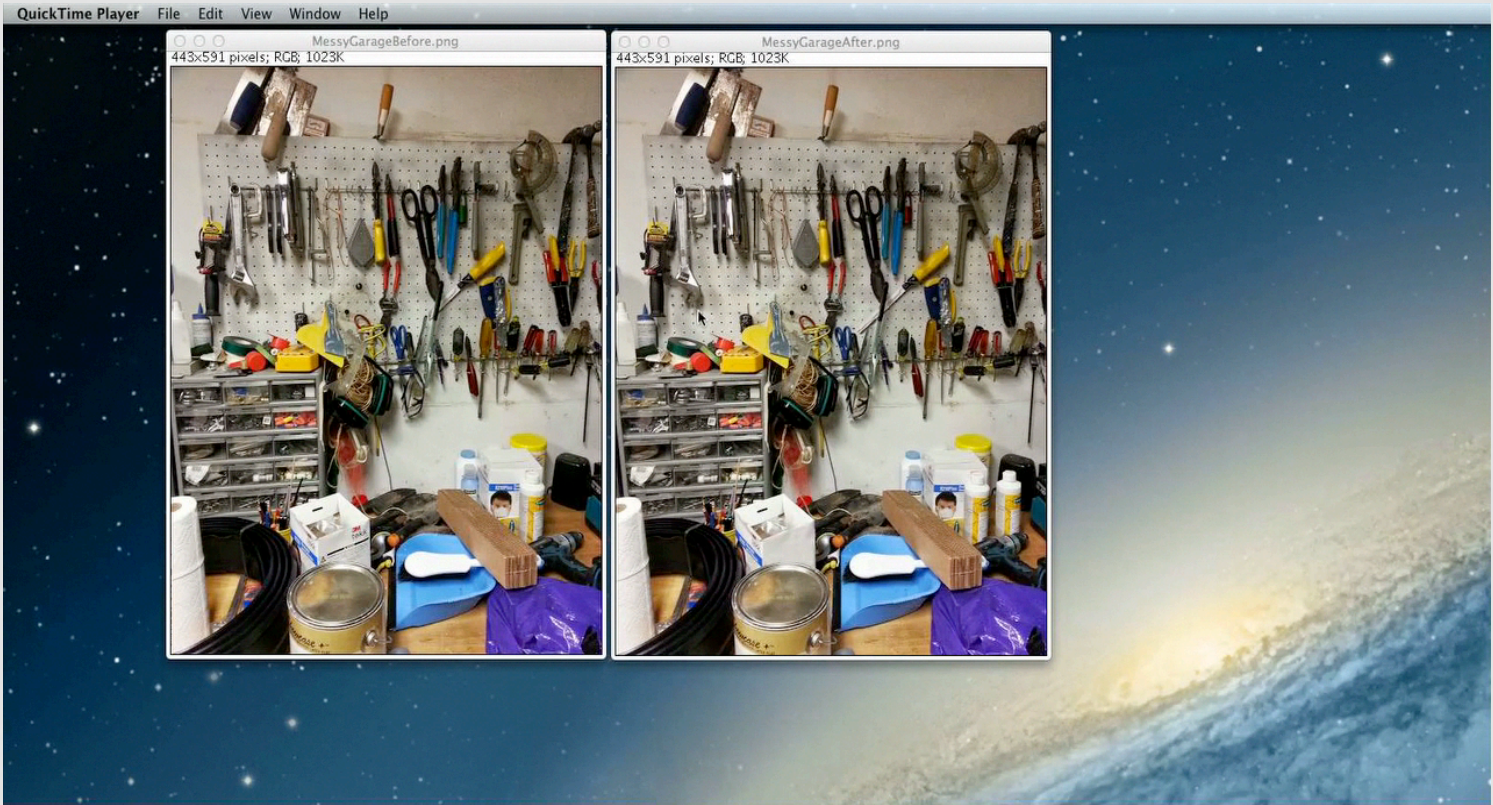
Gallery 3.3 Self-paced step by step image subtraction



1 of 10 - Go to Process in the Main Menu Bar.



Movie 3.1 Image Differencing step by step in ImageJ



This movie shows the step by step process of differencing two images using the Image Calculator within ImageJ. Tap the start button, then expand the screen, then rotate the iPad to Landscape.

For further practice in subtracting images, download the following pairs of files: [HelicopteroverCitybefore & HelicopteroverCityafter](#), [ScienceClassroombefore & ScienceClassroomafter](#), and [Waterfrontbefore & Waterfrontafter](#). You are encouraged to try to find the differences manually. Give yourself 5 minutes. Then use ImageJ to find the differences to see if you found them all.

In MessyGarage, you worked with a second image that was derived from a first image. The Before image is altered and then saved as an After image. So, other than the changes that are made, the remaining unaltered pixels will be exactly identical. Thus subtraction will yield 0 for the unchanged pixels and the image will be homogeneously gray in those areas. In the real world, you would acquire a replicate data set, not simply a derivative made from the original. Just how identical are two pictures one taken right after the other?

In general, when you perform an experiment twice under supposedly identical conditions, do you get EXACTLY the same set of numerical data? Ideally you would want an exact duplicate from experiment to experiment, but in reality

this is not the case. To test this idea, TWO photographs are taken under supposedly identical conditions (Image 3.7). We can then use subtraction with ImageJ to see how reproducible the photographic technique is.

Your next assignment is to compare [TacsinBin](#) image against [TacsinBinReplicate](#) image using ImageJ. Download these two files. REMEMBER WHERE YOU SAVED THEM!!! Use ImageJ to open them up and apply the image subtract technique shown in Gallery 3.4. Be ready to explain your resultant image. A follow up assignment is to use the subtraction function in ImageJ and compare TacsinBin against [TacsinBinShaken](#) (download!). How does the resultant image in this assignment compare to the resultant image in the previous assignment?

Looking at Gallery 3.8, hopefully you can see the more pronounced (real) differences compared to the differences created in small imperfections (not real) in the photographing technique. As an experimentalist, YOU will have to determine which changes are significant and which ones are not.

Image 3.7 Two subsequent photographs of tacs in a bin, supposedly under identical conditions.



Gallery 3.4 Reproducibility and Real change



This is the resultant image of subtracting TacsinBinReplicate from TacsinBin



To briefly review the tasks you have performed, a Review widget has been provided to self-evaluate your knowledge. Select the correct multiple choice answer or match up the items correctly. You can check your answer for immediate feedback.

Review 3.2 Let's review some of the techniques you should have learned in this Section:

Question 1 of 4

In ImageJ, the Image Calculator function is found within:

- ☐ **A.** Edit
- ☐ **B.** Image
- ☒ **C.** Process
- ☐ **D.** Analyze



Check Answer



Mathematica: Subtracting Images

The Math Behind ImageJ

We've let ImageJ do our image processing so far. However, ImageJ is really just hiding the math from you. In this lesson we'll take a look at the math behind the graphics processing.

Image subtraction:

In ImageJ, image subtraction was just a few clicks away. However, what the program is doing is actually hiding the math behind the image subtractions. Movie 3.2 shows you the math behind the image subtraction by using a couple of very simple images.

In Movie 3.2:

- We Create two simple pictures. (A plus sign and an asterisk)
- We Subtract those two images to highlight the differences in the two pictures.
- *The three images to the right are from the process itself. Image 3.8 shows the plus sign that was created, Image 3.9 shows the asterisk and Image 3.10 is the difference between the images.*
- *You may notice that the subtraction has three tones: white, gray, and black. In this case the gray tone is zero, which results when two pixels of the same value are subtracted (white - white or black - black). The white tone is 255, a positive value that results when a black = 0 is subtracted from a white = 255. Black tone is a value of -255. This results when we take a black = 0 and subtract a white = 255.*

Image 3.8 Plus sign Image

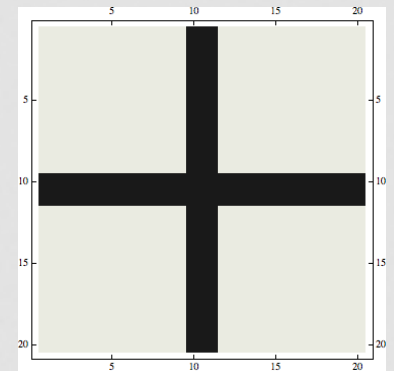


Image 3.9 Asterisk Image

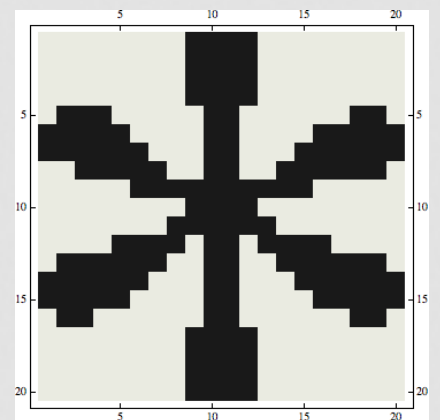
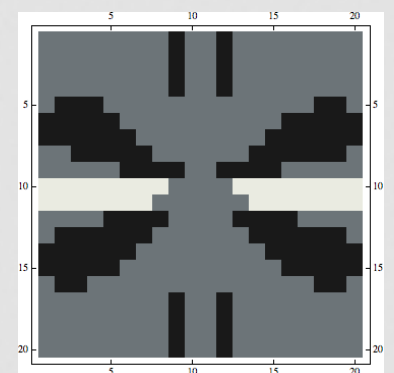
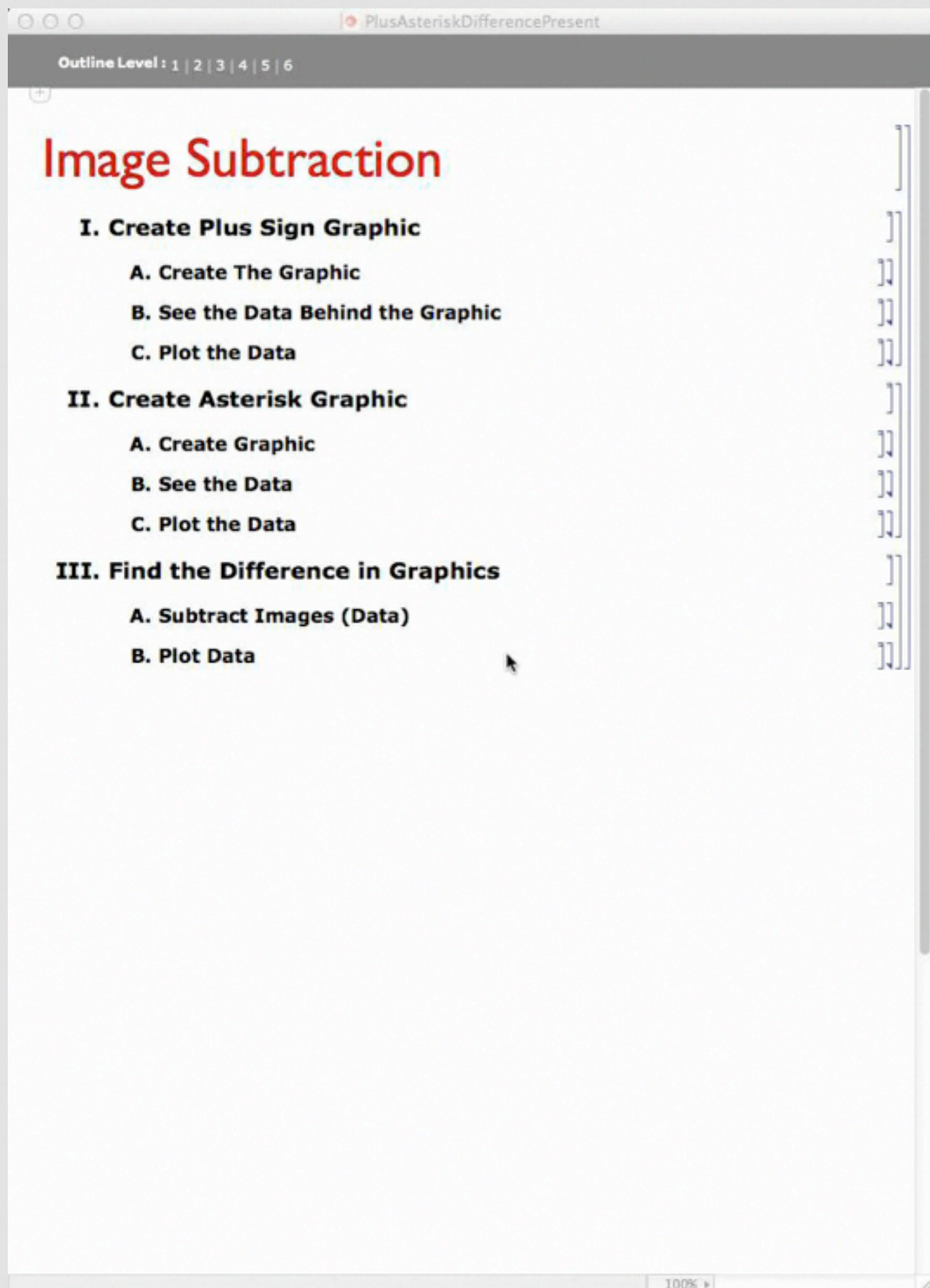


Image 3.10 Subtraction of Plus sign from the Asterisk



Movie 3.2 Simple Image Subtraction in Mathematica



A demonstration of image subtraction using simple 40pixel images

Adjusting the Contrast of an Image

Image Contrast Adjustment

Most graphics programs make it easy to adjust the contrast of an image. However, adjusting the contrast of an image is really just multiplying the image by a constant.

In *Movie 3.3*:

- We create a simple 16pixel grayscale image and adjust its contrast *Image 3.11*
- We introduce the idea of clipping *Image 3.12*
- We introduce a ~1kilo-pixel grayscale image and adjust its contrast
- We import the “*MessyGarageBefore*” image from earlier and adjust its image with a slider via the ‘*manipulate*’ command in *Mathematica*.

Image 3.11 Simple 16 pixel Grayscale

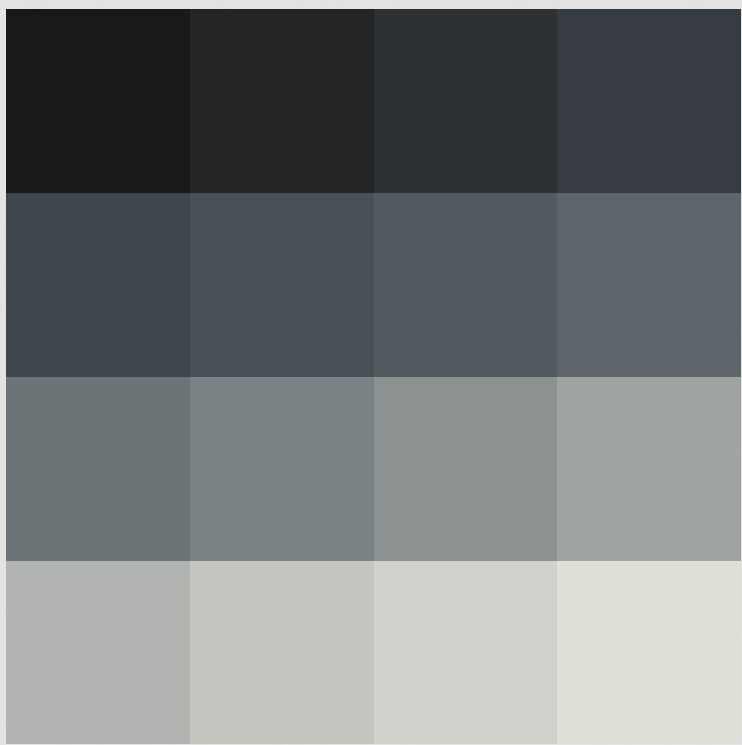
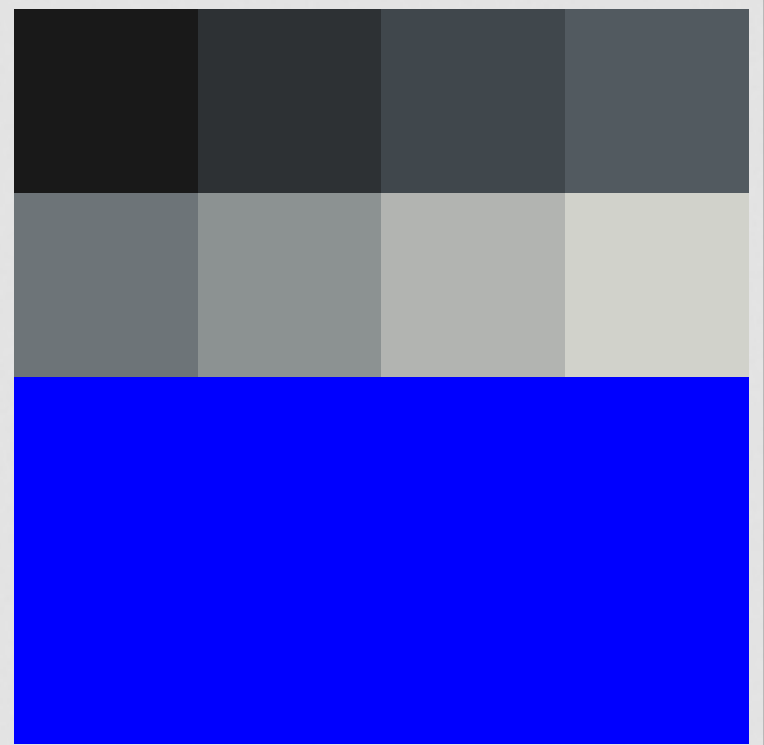


Image 3.12 Raised Contrast Grayscale



Movie 3.3

16pixelGrayScaleMultiply

Contrast function

First Let's Create a Simple 16 pixel image to manipulate

For this example we'll create a 4x4 table ranging in value from 0 to 255. Then display that table as a graphic using ArrayPlot.

```
In[310]:= TableForm[greyl = Partition[Table[x*16, {x, 0, 255/16}], 4]]
```

```
In[311]:= ArrayPlot[greyl, PlotRange -> {0, 255}, ColorFunction -> "GrayTones"]
```

Now Let's Increase the Contrast

Contrast control is done by multiplying our image array by a constant. If the constant is greater than "1" (one) then we will increase the contrast. That is the dark pixels will stay dark, but the bright pixels will get brighter. However, if the constant is less than one, you will decrease the contrast. The Blue you see is clipping. this happens because multiplying 255 times 2 gives a value that is greater than the maximum allowed value of 255. In this example I've set the clipping color to blue. Most graphics programs will just have the clipped cells to default to 255, and thus appear white.

```
In[312]:= TableForm[greyl = Partition[Table[x*16, {x, 0, 255/16}], 4]]
```

```
In[312]:= ArrayPlot[greyl, PlotRange -> {0, 255}, ClippingStyle -> Blue, ColorFunction -> ColorData["GrayTones"]]
```

```
In[315]:= Manipulate[ArrayPlot[greyl*contrast, PlotRange -> {0, 255}, ClippingStyle -> {Black, White}, ColorFunction -> ColorData["GrayTones"]], {{contrast, 1}, 0, 16, 0.1}]
```

A more Complex Image

In this case we'll create a much larger table of values. In this case a random 1 kilopixel image.

```
In[316]:= TableForm[BigPict = RandomInteger[{0, 255}, {32, 32}]]
```

```
In[317]:= Manipulate[ArrayPlot[BigPict*contrast, PlotRange -> {0, 255}, ClippingStyle -> White, ColorFunction -> ColorData["GrayTones"]], {{contrast, 1}, 0, 32, .1}]
```

A Real Picture

Now, let's use this math concept on a real picture. Below you'll see a manipulate similar to what we did earlier. However, now we are using the "Messy Garage" picture from earlier.

```
In[318]:= g = Import[NotebookDirectory[], "MessyGarageBefore.png"];
```

```
In[318]:= Manipulate[ImageMultiply[g, contrast], {{contrast, 1}, 0, 32, .1}]
```

100%

Adjusting the Brightness of an Image

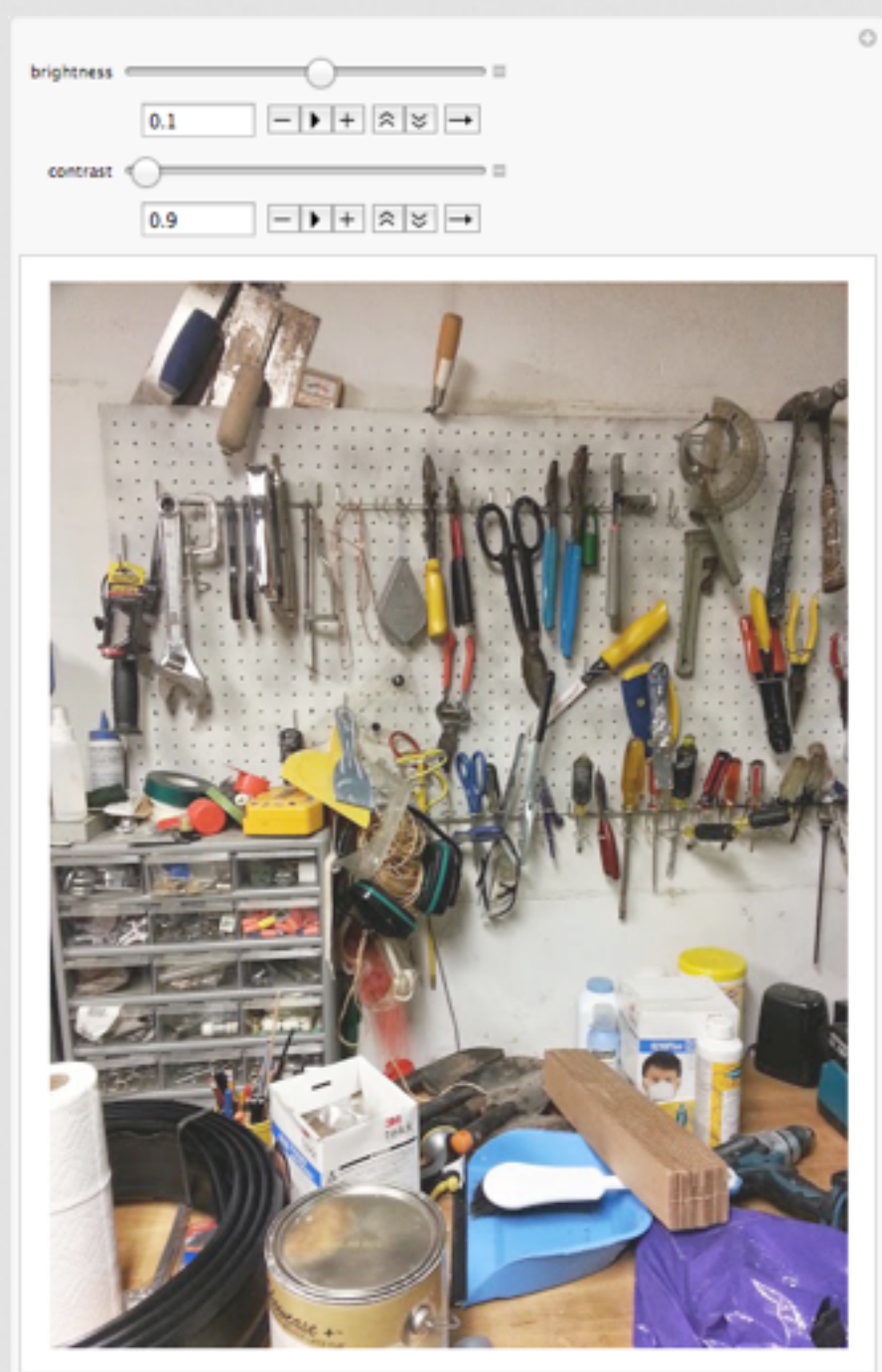
Image Brightness Adjustment

Most graphics programs make it easy to adjust the brightness of an image. However, adjusting the brightness of an image is really just adding a constant to the image.

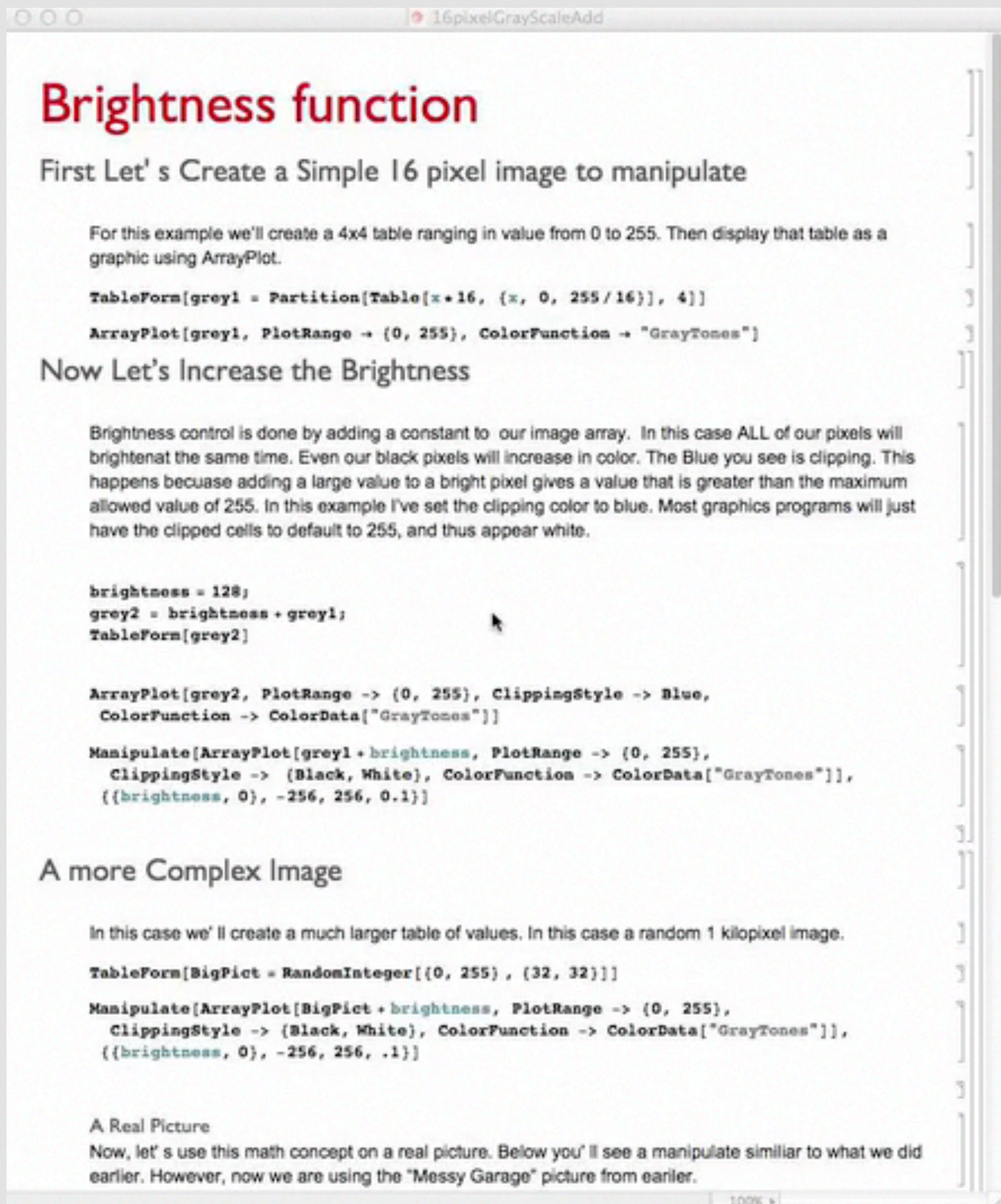
In *Movie 3.4*:

- We create a simple 16pixel grayscale image and adjust its brightness *Image 3.13*
- We create a ~1 kilo-pixel grayscale image and adjust its brightness
- We import the “Messy GarageBefore” image from earlier and adjust its image with a slider via the ‘manipulate’ command in *Mathematica* *Image 3.13*

Image 3.13 Brightness and Contrast Sliders



Movie 3.4 Brightness with Mathematica



Brightness function

First Let's Create a Simple 16 pixel image to manipulate

For this example we'll create a 4x4 table ranging in value from 0 to 255. Then display that table as a graphic using ArrayPlot.

```
TableForm[greyl = Partition[Table[x+16, {x, 0, 255/16}], 4]]
ArrayPlot[greyl, PlotRange -> {0, 255}, ColorFunction -> "GrayTones"]
```

Now Let's Increase the Brightness

Brightness control is done by adding a constant to our image array. In this case ALL of our pixels will brighten at the same time. Even our black pixels will increase in color. The Blue you see is clipping. This happens because adding a large value to a bright pixel gives a value that is greater than the maximum allowed value of 255. In this example I've set the clipping color to blue. Most graphics programs will just have the clipped cells to default to 255, and thus appear white.

```
brightness = 128;
grey2 = brightness + greyl;
TableForm[grey2]

ArrayPlot[grey2, PlotRange -> {0, 255}, ClippingStyle -> Blue,
  ColorFunction -> ColorData["GrayTones"]]

Manipulate[ArrayPlot[greyl + brightness, PlotRange -> {0, 255},
  ClippingStyle -> {Black, White}, ColorFunction -> ColorData["GrayTones"]],
  {{brightness, 0}, -256, 256, 0.1}]
```

A more Complex Image

In this case we'll create a much larger table of values. In this case a random 1 kilopixel image.

```
TableForm[BigPict = RandomInteger[{0, 255}, {32, 32}]]
Manipulate[ArrayPlot[BigPict + brightness, PlotRange -> {0, 255},
  ClippingStyle -> {Black, White}, ColorFunction -> ColorData["GrayTones"]],
  {{brightness, 0}, -256, 256, .1}]
```

A Real Picture

Now, let's use this math concept on a real picture. Below you'll see a manipulate similar to what we did earlier. However, now we are using the "Messy Garage" picture from earlier.

Review 3.3 Section 3 review

In the contrast process:

- ☒ **A.** all pixels are multiplied by the same constant. Dark remains dark but bright gets brighter.
- ☐ **B.** all pixels are multiplied by the same constant. Dark and bright get brighter.
- ☐ **C.** all pixels have the same constant added to them. Dark remains dark but bright gets brighter.
- ☐ **D.** all pixels have the same constant added to them. Dark and bright get brighter.

Check Answer

Highlighting the differences

Introduction

In the Section 3.2 you used the image calculator in ImageJ to subtract one image from another. This produced a grayscale image where you could easily see some of the differences between the pictures.

However, as you will see in this section, simply subtracting one image from another is not a sure-fire way to catch all of the differences.

Let us take another look at the image we got from the image calculator. We can clearly see the locations of five differences

within the picture. However, unless you have exceptionally good eyes or you already knew that there was a difference in that spot, you probably did not catch the sixth difference in Interactive 3.2. This shows a common problem in computer visualization - **you can often miss something if you only look at the image or dataset in one way**. So, in this section we will explore some more options that you have at your disposal within ImageJ to process images so that we are able to see our data in different ways.

Interactive 3.2 More than Meets the Eye



A sixth difference?

Analyzing the Image

The goal of analyzing the two original two images in Image 3.5 can be essentially broken down into two tasks: **1)** count how many differences there were between the pictures and **2)** find where the differences are within the pictures. There are many different ways to go about completing these tasks, but only a few of the techniques to do so will be covered in this section. If you care to challenge yourself then continue exploring ImageJ to find other methods to complete those tasks.

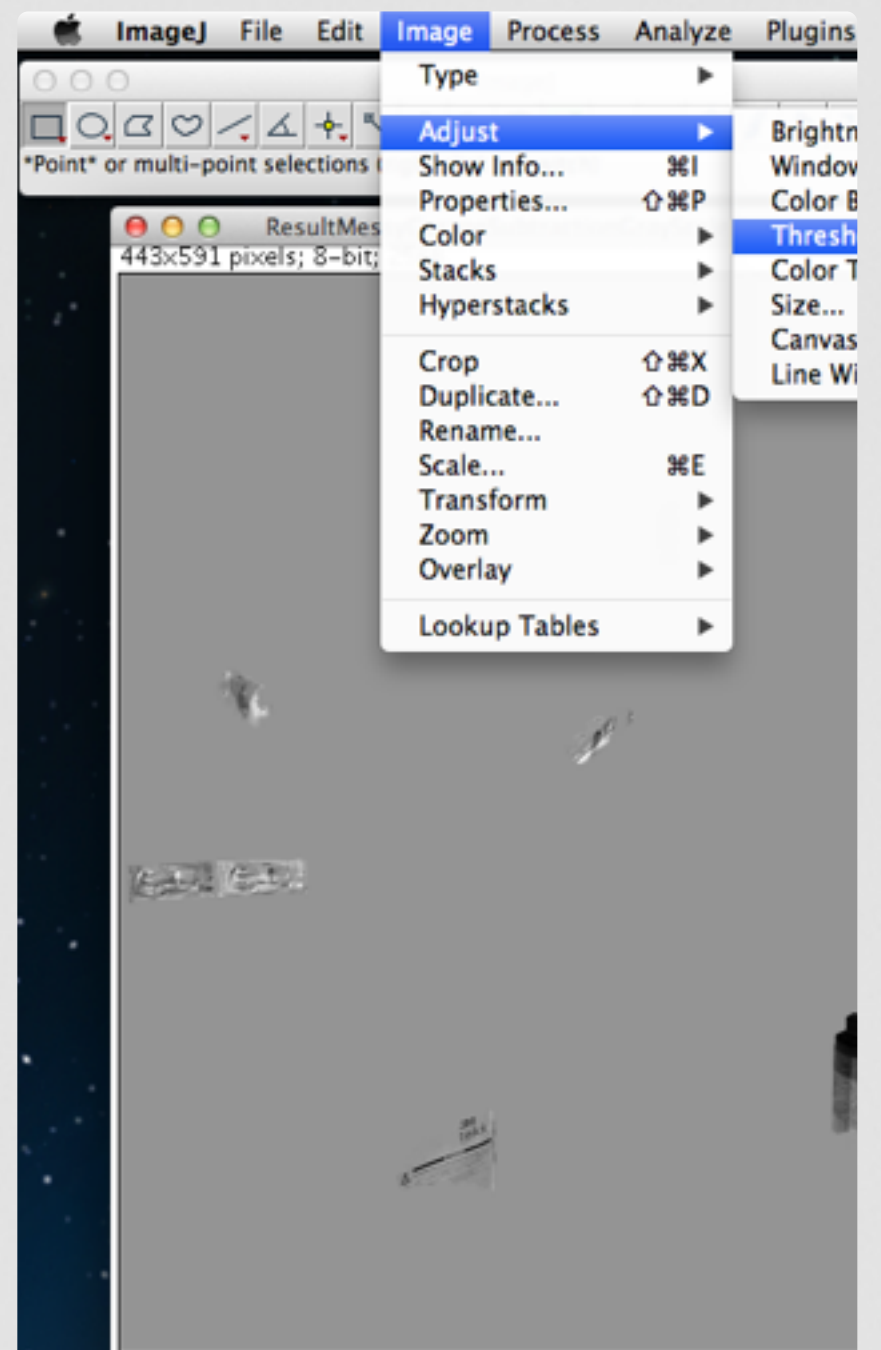
Finding the Differences by using Threshold

On the last page we saw that there may be a sixth difference in the ImageJ that was very hard to see. We are going to make it easier to see by adjusting the *Threshold* of the image. A threshold defines a range of values and changes any value within that range in some way. Any values outside of the defined range are changed in a different way. This makes it easier to tell the difference between the value within and outside the defined range.

In the example in Gallery 3.5 I had the threshold type set to black and white with the *Dark Background* option checked. As shown in the third image of the gallery, this set any values between 0 and 147 as white

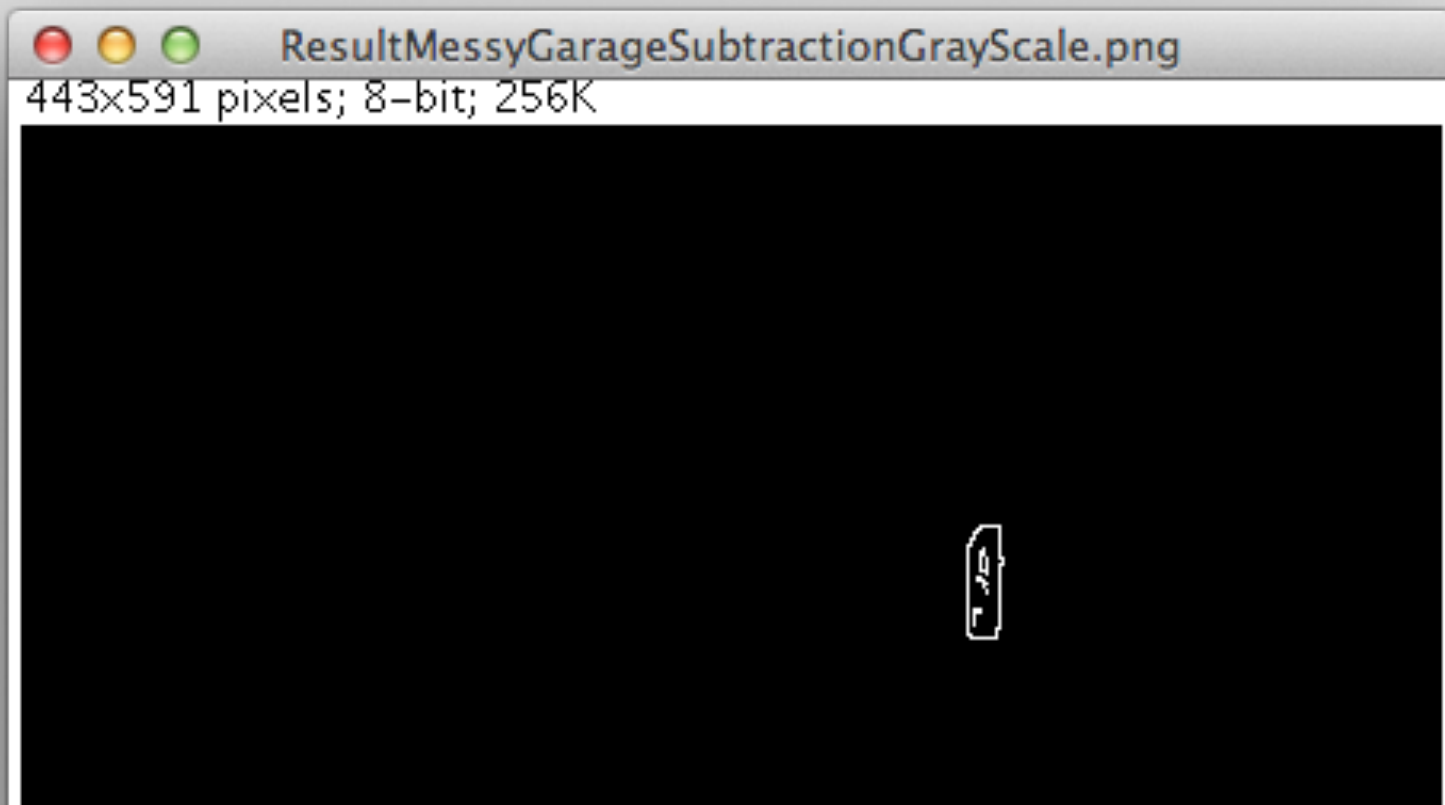
(pixel value = 255) and anything outside of that range as black (pixel value = 0).

Gallery 3.5 Adjusting the Threshold (highlighting the differences)



This is the menu location for Threshold.

Gallery 3.6 Dilate and Outline



The result of only using the *Outline* tool found in the *Binary* menu

• • •

Binary functions

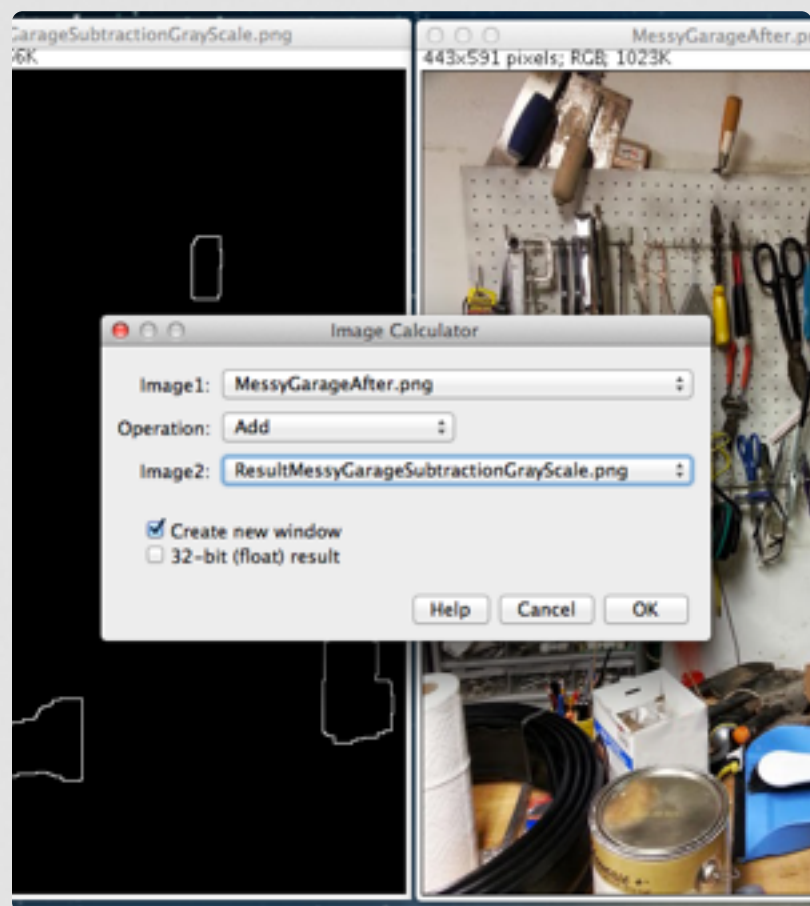
Earlier we converted the image from an RGB color image to a 32-bit grayscale image through the image calculator. When we set the threshold values with the B&W setting we turned the image into an 8-bit binary image. You can see the image type in the top left corner of any image opened in ImageJ. While it may seem like we are losing information in the image through all this processing - we went from having hundreds of colors to having just two colors - there is actually a very useful set

of tools that is opened up to us once we have a binary image.

While our eyes can see there are six objects on the screen, if you asked a computer to outline the objects you would get a result no prettier than the original binary image (see *Gallery 3.6 - Image 2*). So we have to fill in some of the gaps. The tool that we will use here is called **Dilate**. In laymen's terms Dilate makes the white spots bigger. To give a more technical definition, Dilate takes all of the pixels with non-zero values (remember white = pixel

value of 255 and black = pixel value of 0) and makes its neighboring pixels have the same value (of white). So if we repeat using this function a few times we can get six solid white objects that are much more easily outlined (see *Gallery 3.6 - Images 3 & 4*).

Gallery 3.7 Adding Outlines to Resultant



Remember to uncheck the “32-bit (float) result” box so that your resulting image comes out in color.

Add the Outlines to the Changed Image

Though we used the image calculator to subtract before, you can also do a variety of other math operations between two images. Now we will use the add command to add the outlines onto the changed image so we can easily see what the differences are. This is demonstrated in Gallery 3.7.

Even though this process was more involved than the simple image subtraction technique used in earlier sections it produces a more accurate and clearer result. In the field of visualization the goal is usually to display your results to others in an easy-to-interpret way. We must always ask ourselves if what we are displaying is accurate and easy to interpret.

Visual Trick

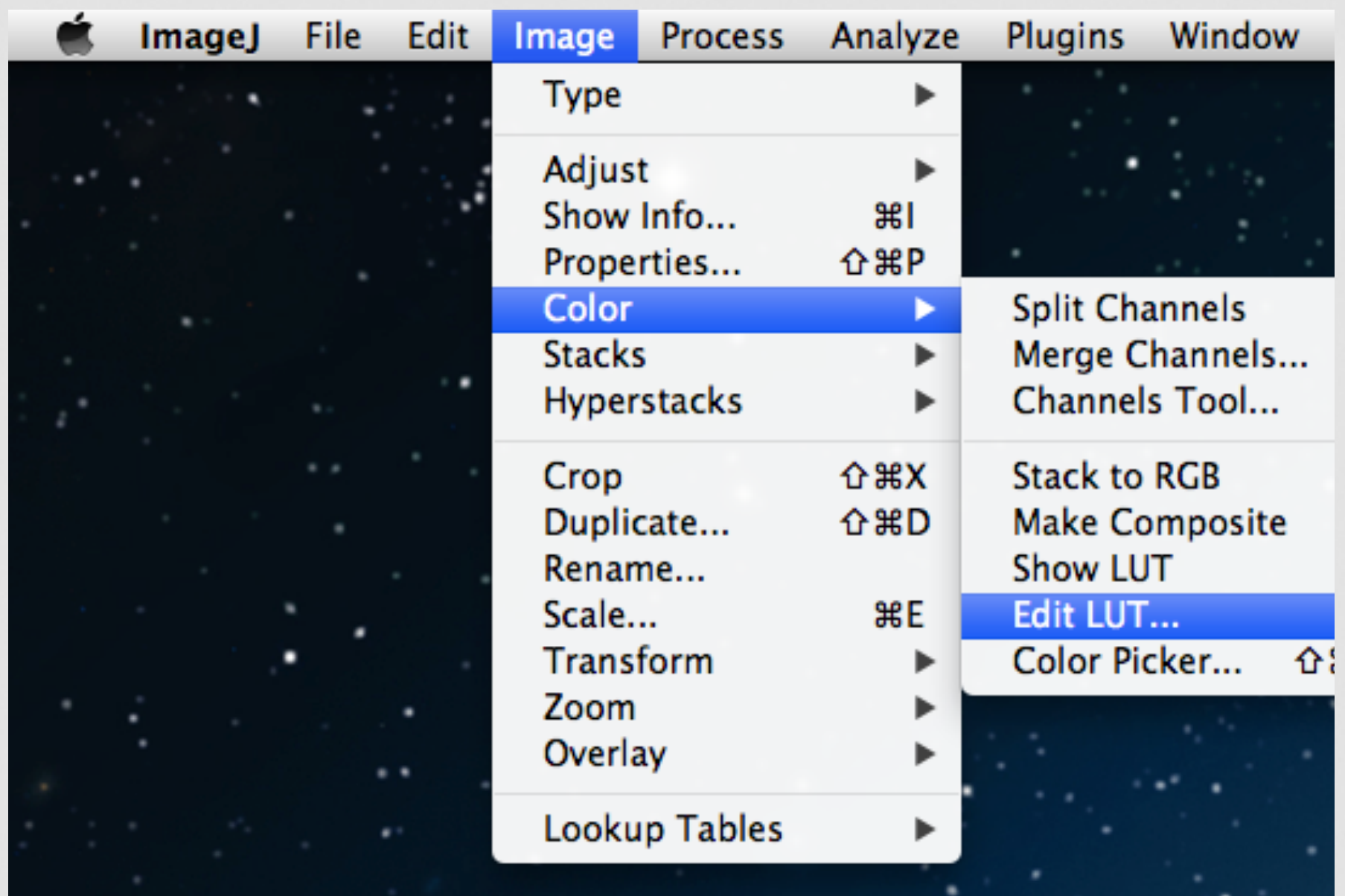
Though we just went over the procedure for image processing there is a trick in ImageJ that can make the visualization better with no manipulation of the actual data.

Look-up Tables (LUT)

The look-up tables, more commonly referred to as LUT, are what ImageJ uses to determine what color to display pixels in

On the next page we will use the Image Calculator again to make use of these outlines that we now have.

Gallery 3.8 Changing color of Outlines to green



Edit LUT menu location

• • • • •

32-bit, 16-bit and 8-bit images. For each pixel ImageJ refers to the LUT to determine which color to display. In the image gallery you can see the grid layout of the LUT. By default it is grayscale but in ImageJ you can edit the LUT to display any colors that you want.

This is a useful ability to have. In many cases black and white may not be the best way to visualize something. For example, lets take the outlines that we just created on the previous page. If there were a different color, say bright green, they would stand out much better. See Gallery 3.8 for a guide on how to edit the LUT in ImageJ.

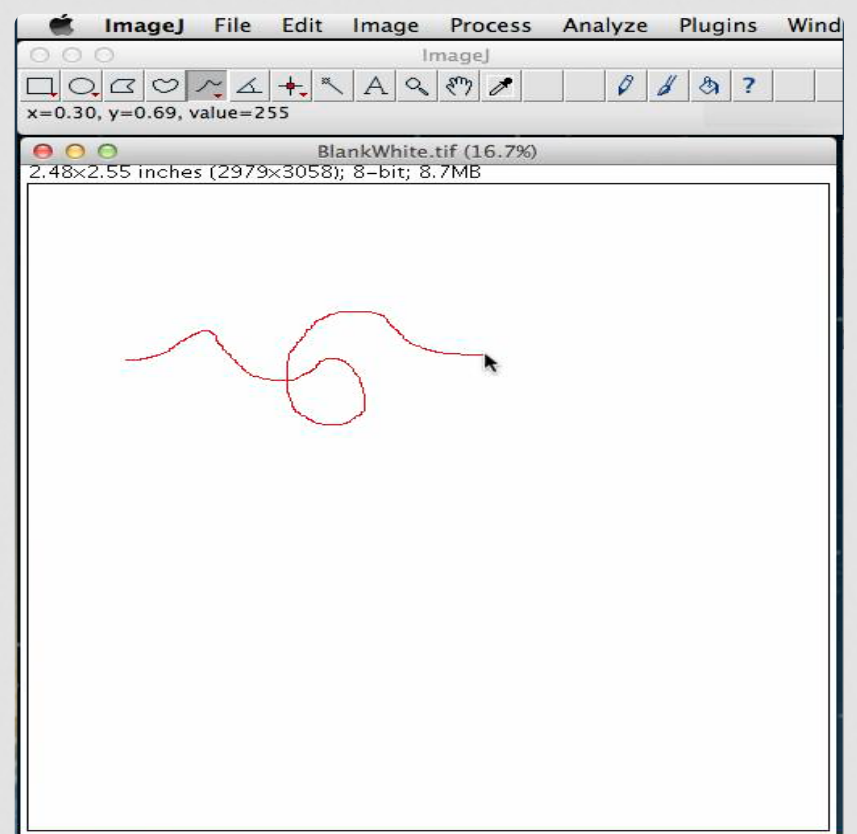
Line Draw in ImageJ and its Applications: Histograms, Lengths, Areas, and Angles

LineS

This section starts out with a very short and straight forward task. You will be shown how to draw lines of various styles upon your image. Although a fairly simple task, its importance is significant. Drawing lines on an image can be used to focus the viewer's attention to a specific place on the image. The line can also be used as a measuring stick of features within your image. And once you have a measuring stick, you can then measure areas. The line can also be used to define a row of pixel values from which you can create a histogram.

Gallery 3.9 will show you the step by step techniques in line drawing. You should download the file [BlankWhite](#) and open it up in ImageJ to give you a "canvass" to play on.

Gallery 3.9 Drawing different line styles in ImageJ.



Put your cursor where you want to start, click and hold, draw as you see fit. Release the click and the line is done. You will not be able to move line around.

Once you feel comfortable with the different modes of Line Draw, you should practice on a real image (like MessyGarage or an image of your own) by drawing lines, line segments, and freehand. The Arrow tool is left for you to play with to make different styles of arrows. You should also practice making these line drawings as a “permanent” overlay on your image.

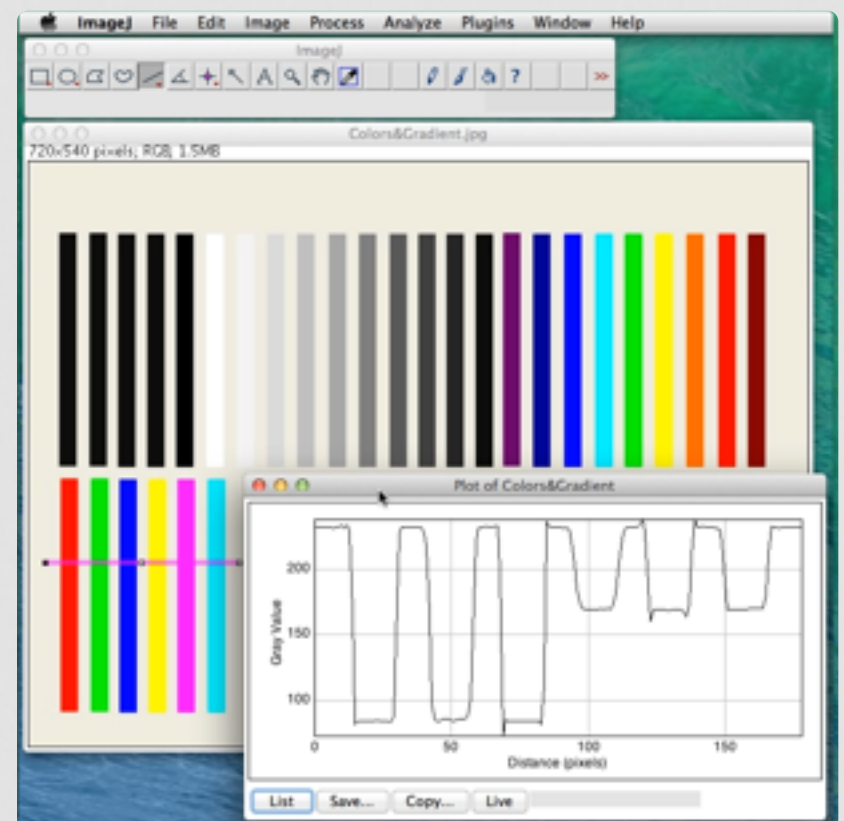
Gallery 3.10 will show you how to evaluate grayscale pixel intensities (i.e. histogram) on the image as defined by the line drawn. This will require the use of Plot Profile, NOT Histogram. This is one of many ways to extract secondary numerical information from your image. You can also obtain x,y coordinate values of these pixels as learned in Section 3.1.

A line drawn on an image can have its length defined. You can then use that definition to measure the length and area of different structures on your image. This is demonstrated in Gallery 3.11.

And finally, lines will be used to create an angle that can then be measured as shown in Gallery 3.12.

OK, let's get started with Gallery 3.10 applying line draw. Open up [Colors&Gradient](#) in ImageJ.

Gallery 3.10 Obtaining a Plot Profile in ImageJ.



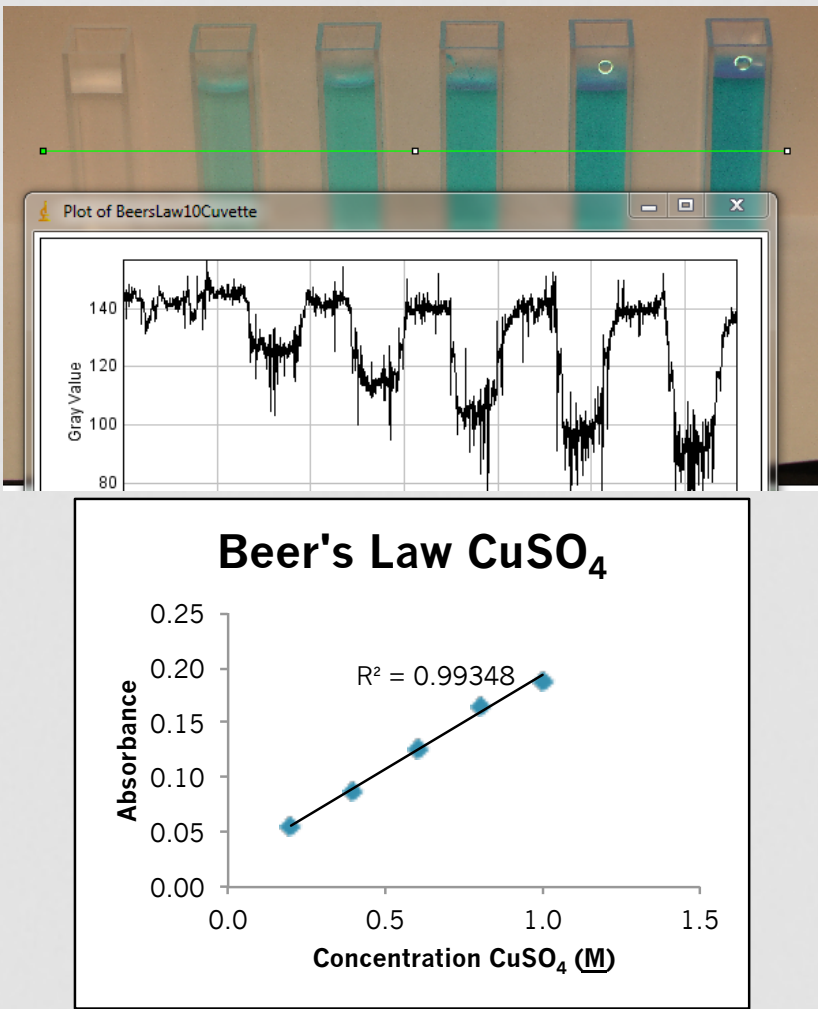
The colors are converted to grayscale for plotting purposes. Remember in Section 3.1 you were shown how grayscale is calculated from RGB. Red would be calculated as $(255 + 0 + 0)/3 = 85$. The same holds true for G & B. Yellow, magenta, and cyan would be more like $(255 + 255 + 0)/3 = 170$

After you've completed Gallery 3.10, I want to stress that you use “Plot Profile”, NOT “Histogram”. Another thing to point out is that the results are grayscale, the individual R, G, or B values are not provided.

Applications of this? Measuring how much a paint has faded with time or how well a laundry detergent has cleaned a stain to mention a couple. Any experiment in which

intensity is an important variable. Image 3.14 shows a Beer's Law plot for a CuSO_4 solution. Beer's Law assumes that absorbance and concentration are directly proportional. As the solution becomes more concentrated, it gets darker, absorbance is increasing.

Image 3.14 Using a camera, ImageJ, and Excel to create a Beer's Law plot.

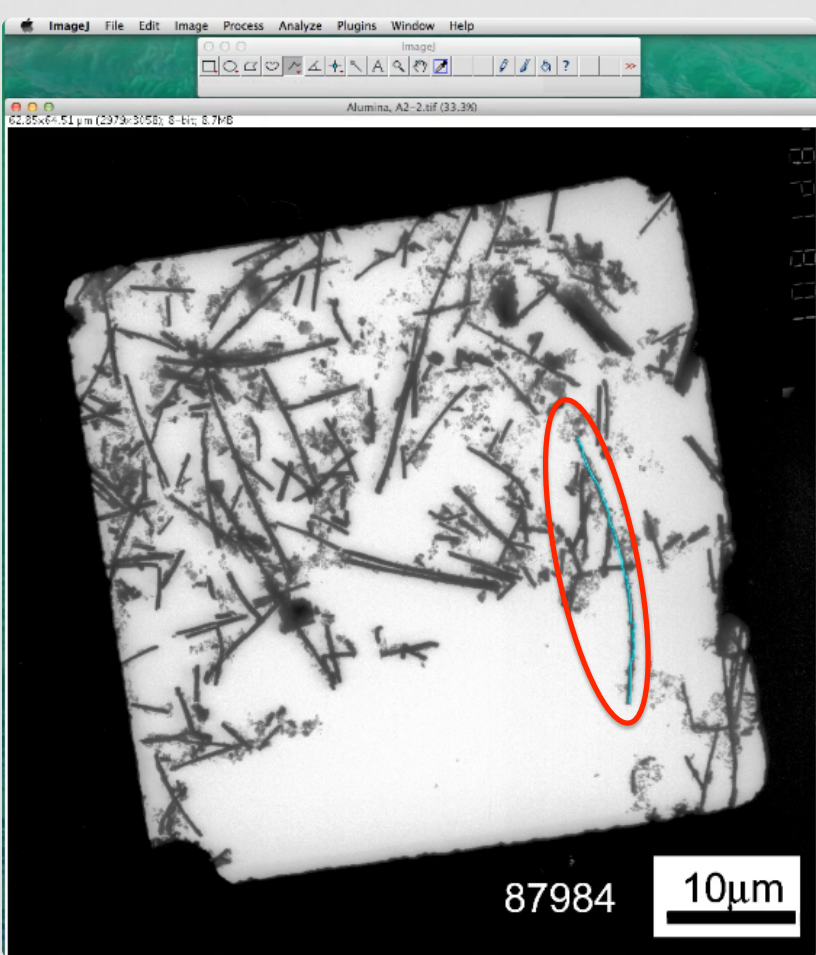


A photo of different concentrations of solutions is taken. Line draw and Plot Profile is performed. The Gray Value intensity of each solution is estimated and converted to an Absorbance which is then inputted into Excel along with concentration values. The Absorbance versus concentration graph generated

from the data set has a strong linear correlation as noted by the R^2 value. What would normally take a \$2000 spectrophotometer to generate the absorbance data can now be done with a \$30 camera!!!

Now we'll move on to Gallery 3.11 that demonstrates how to define a line length and measure the lengths of items. You should download the file [Alumina, A2-2](#) and then open it up using ImageJ. View

Gallery 3.11 Measuring Length in images using ImageJ.



Segmented line is changed to Freehand line (Gallery 3.9, Images 14 & 15). A curved wire is traced and can be measured.

the step by step process of defining line length and measuring in Gallery 3.11.

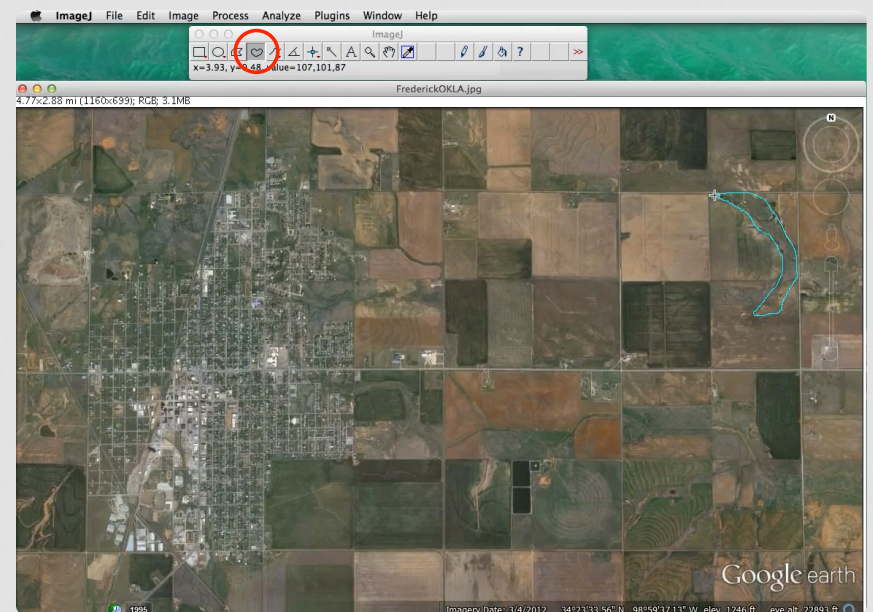
FYI, “Alumina, A2-2” is a micrograph of metal nano-wires grown via electrochemical deposition. A TEM was used to view the structures. To practice more with defining line lengths and measuring, you can download and use [Alumina, A-6](#) and [Alumina, B-3](#). Three more Galleries to go showing you how to make measurements upon images using ImageJ!!

You should download the file called [FrederickOKLA](#) which is a satellite image of my hometown as seen by Google earth. Open it up in ImageJ. Gallery 3.12 starts out reviewing the line measuring setup (images 1-8) followed by demonstrating area measurements using rectangular, circular, polygon, and freehand drawing styles (images 9-15).

Gallery 3.13 will show you an additional feature that is handy when defining an area to measure. A paint brush tool can erase unwanted area.

Practice your Plot Profile, length, area and paint brush skills by downloading and opening [MercuryCraters](#) and [GiantSwallowtailButterfly](#).

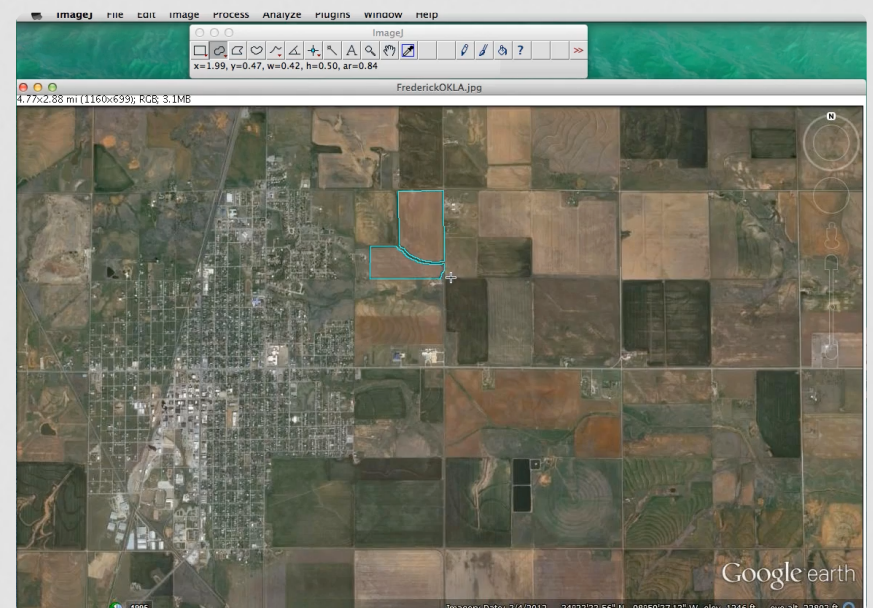
Gallery 3.12 Measuring Areas in images using ImageJ.



Freehand tool is selected. Click and hold your mouse at a point of origin and trace around the object while you are continuing to hold the mouse-click. Once you get back to the origin, release the mouse-click and your end will be connected to the beginning.

15 of 15

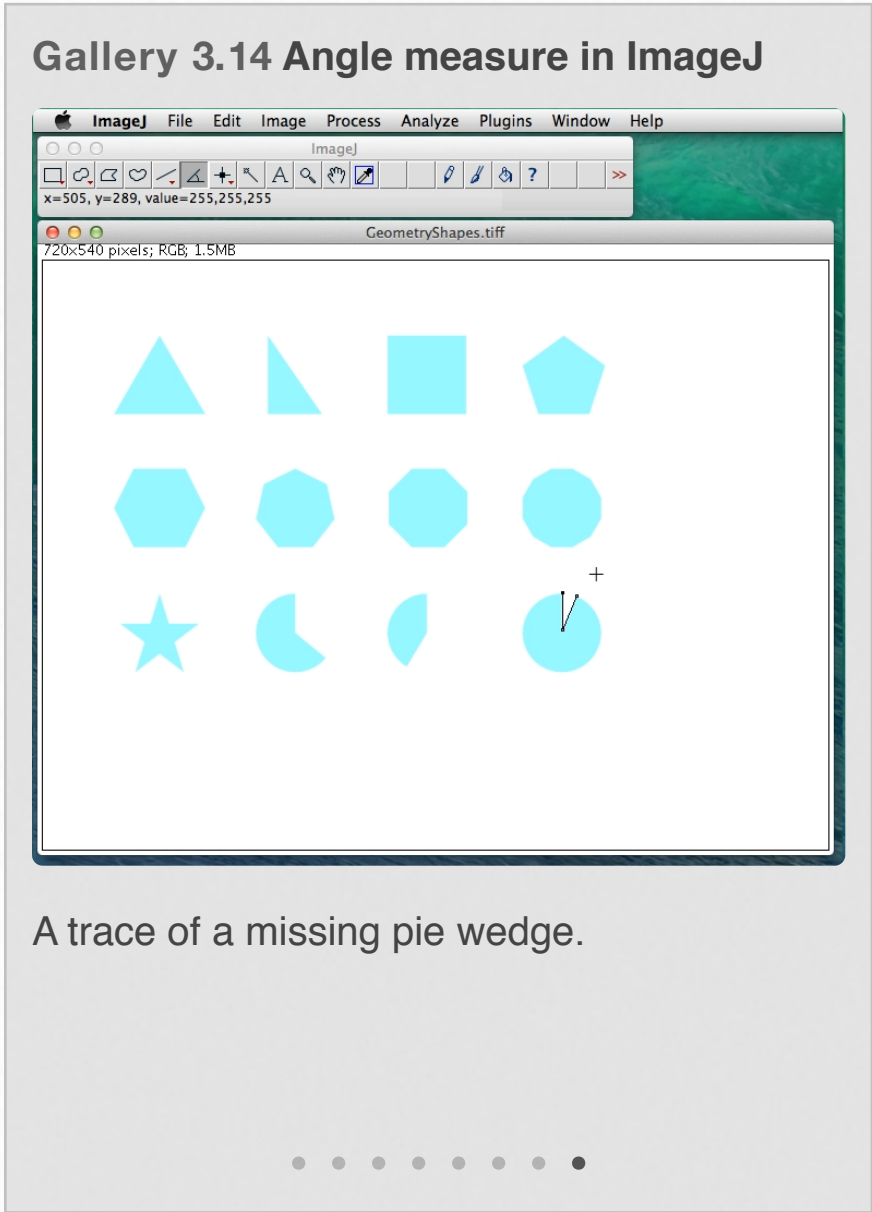
Gallery 3.13 Paint brush with Areas in ImageJ



Here, a polygon was traced and then Paint Brush was used to remove a strip and round the corner of the elbow.

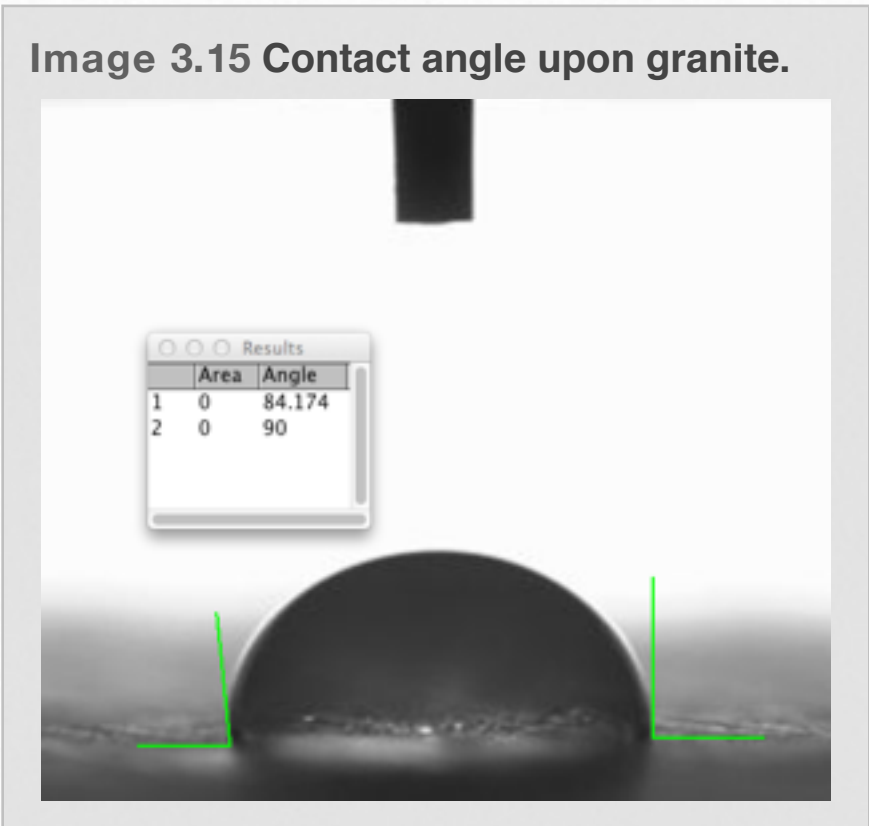
● ● ● ● ●

One more Gallery to go and then you'll have a lot of tools to play with in ImageJ. The last Gallery 3.14 shows you how to draw angles and measure them. Download [GeometryShapes](#) and open it up in ImageJ.



You can practice your angle measure skills on the geometry shapes in the file. You should check the accuracy and reproducibility of your drawing ability. For a real life application, download the files: [ContactangleGraniteTW](#), [ContactangleLimestoneTW](#), and [ContactangleMarbleTW](#). These pictures,

courtesy of the National Center for Preservation Training and Technology (NCPTT), represents a technique used to measure surface treatment. A drop of water is placed on the surface and the angle between the surface the leading edge of the droplet is measured. An example of what this could look like is given in Image 3.15.



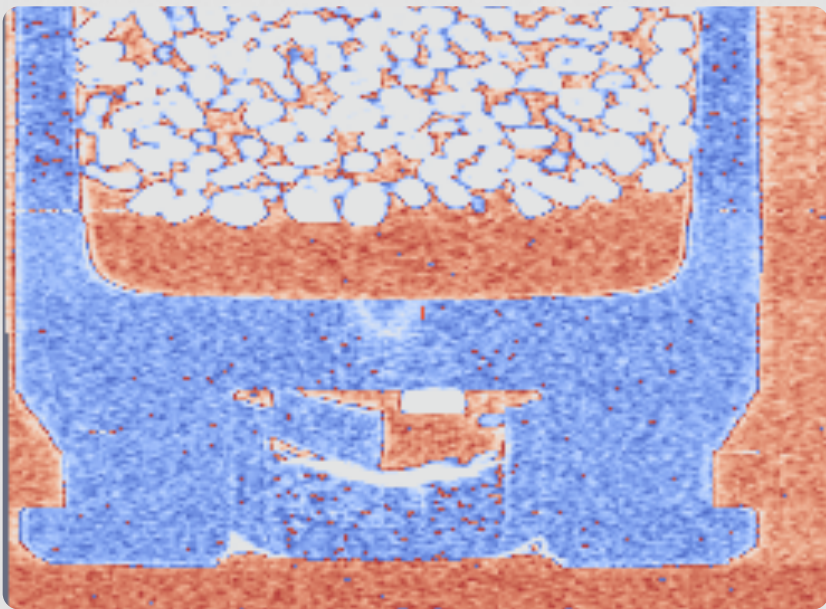
More applications in ImageJ will be given in future editions of this visualization iBook, hopefully very soon. Things like counting and number particles and viewing 3D images.

4

X-Ray and 3D Imaging

Cross sections of a bullet casing made with Neutron Tomography.
!!! citation needed

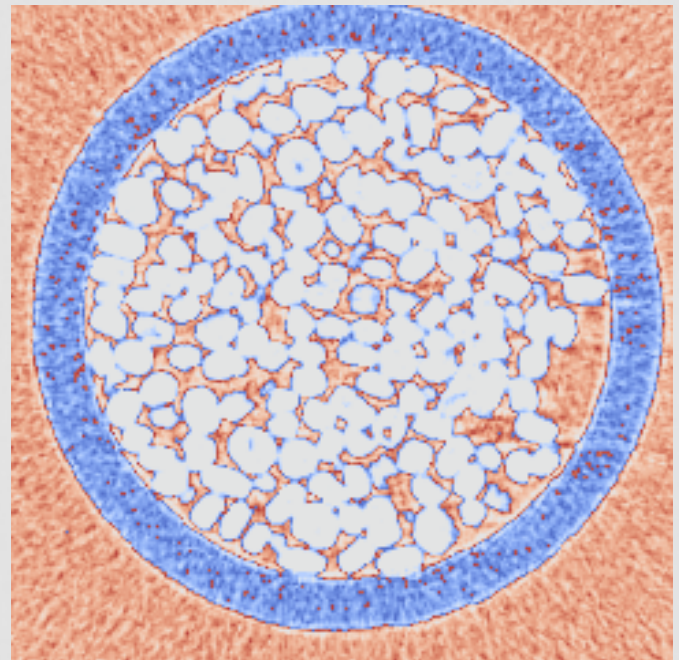
Image 4.1 Vertical Cross Section of Bullet



In this chapter, we look at how we actually get these cool 3D images. Most people have heard of a CT (or CaT) scan. A CT scan is a Computed Tomography scan most famously used in the medical field to get a good look inside of a body without having to do invasive cuts.

The idea of **tomography** is that we use some type of beam (sound, light, x-rays, or even subatomic particles) that can pass through the material of the object we are scanning to different degrees. The most

Image 4.2 Horizontal Cross Section of Bullet

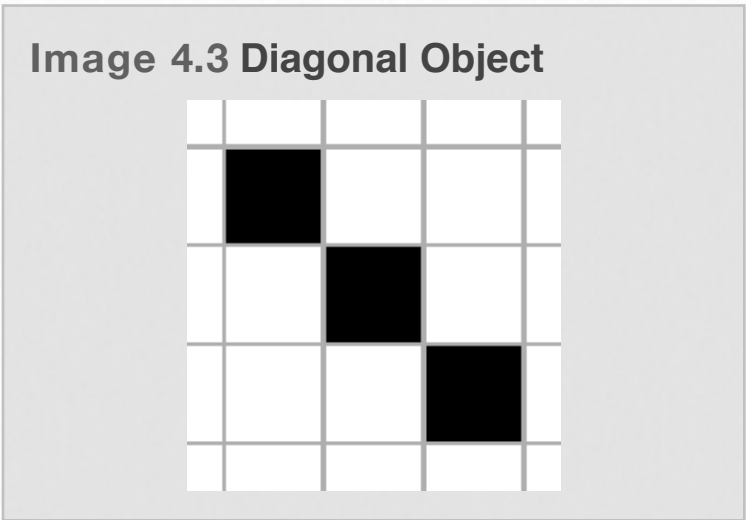


common beam used is X-rays. However, other beams including light, neutrons, and even sound waves. In the example above our object is a cartridge (or bullet) and our beam is a beam of neutrons.

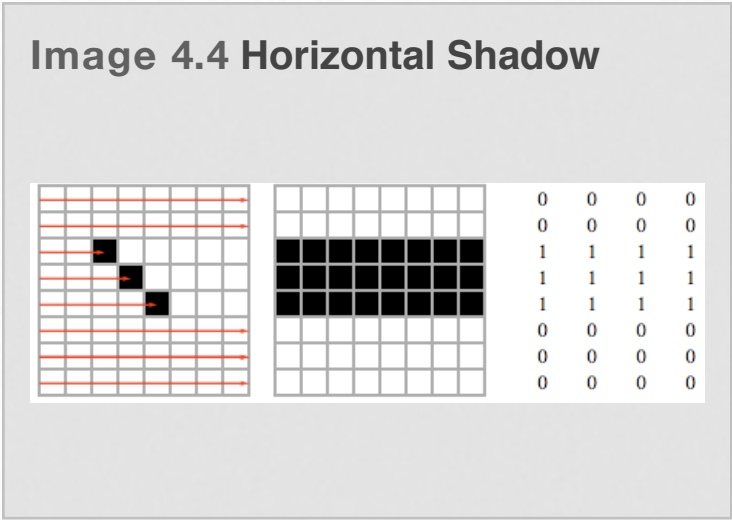
The neutrons can easily pass through the air between the grains of powder, but are very much affected by the metal housing and the grains of gun powder. The **pseudocolor** here shows pink as low interference with the beam, blue as high interference, and white in between.

Shadow Lesson

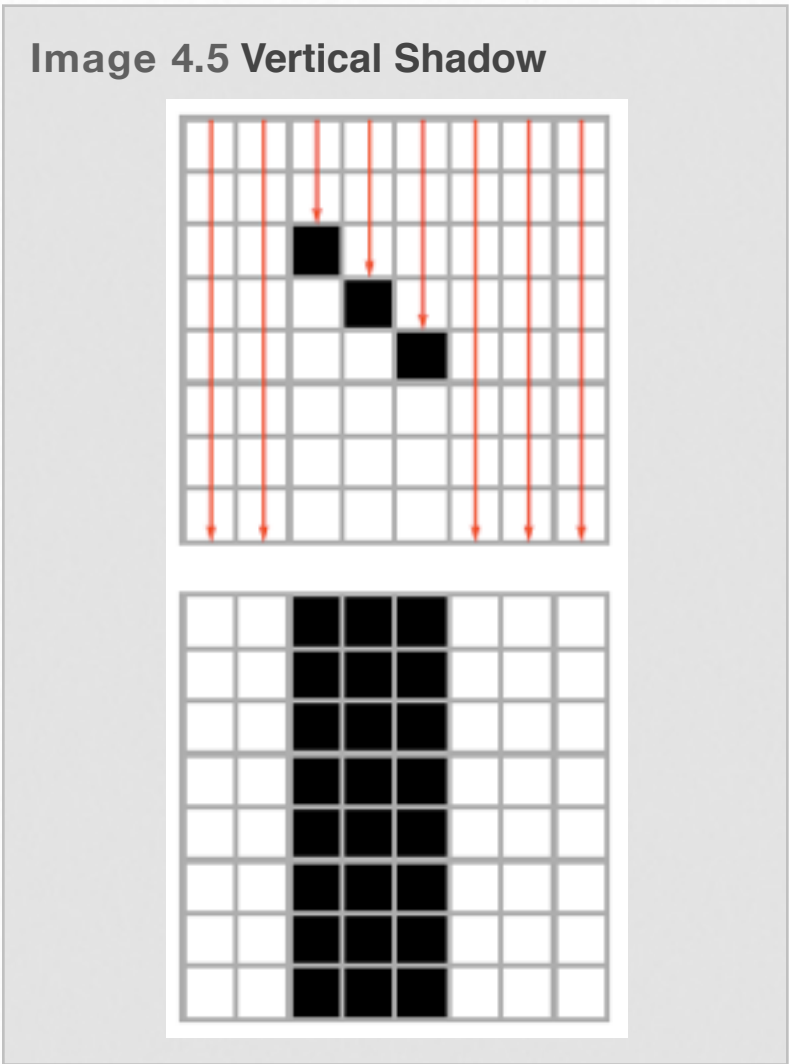
For this example we will stick to a very simple, fictitious object Image 4.3 and try to find it using shadows. In this case our beam will be “visible light” and our beam will either pass through or not. While this shadow creation is a much simplified version of what is happening, it will suffice for an introduction.



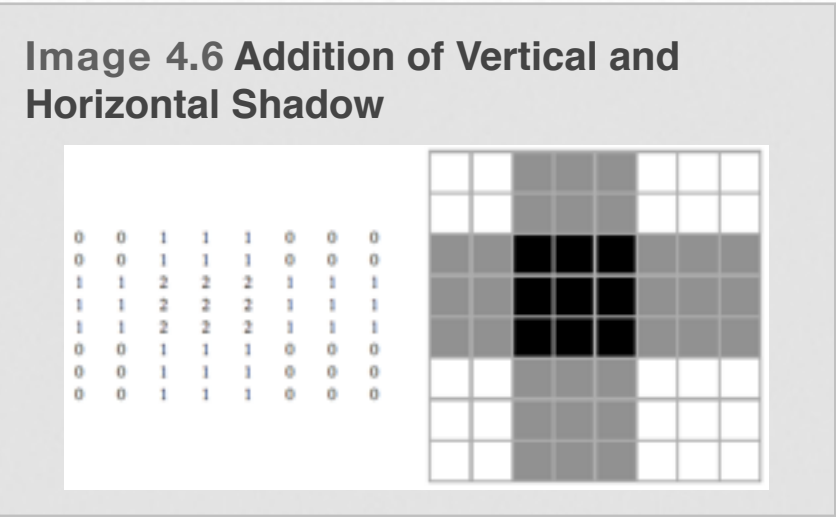
We create the horizontal shadow Image 4.4 by imagining a horizontally aimed fan of light rays. Every a light ray hits part of our object it is blocked and creates a shadow. However, our shadow array (middle image) must shadow in all places where the object might be. That is, any filled in block on that row could create a shadow at that point. So at this time we know our object is fully contained somewhere within the three rows of our platform.



This doesn't give us much information. So, we need another scan. This time we'll do a vertical shadow. Image 4.5. Our vertical shadow tells us that the object is contained in three columns of our platform. By themselves, these shadows do not tell us much.

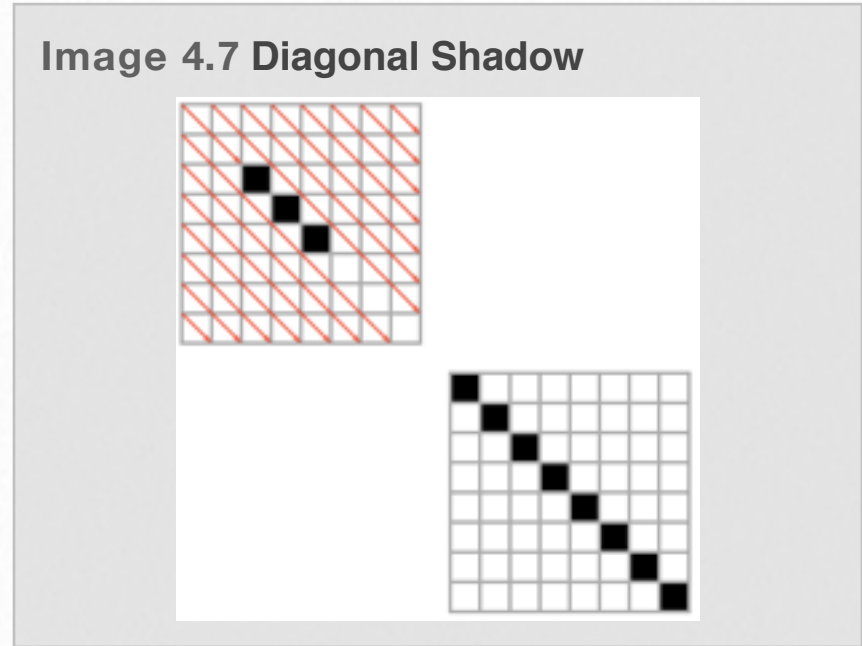


However, we've got a couple of steps left. Let's start by adding our two shadow arrays to create Image 4.6. From the



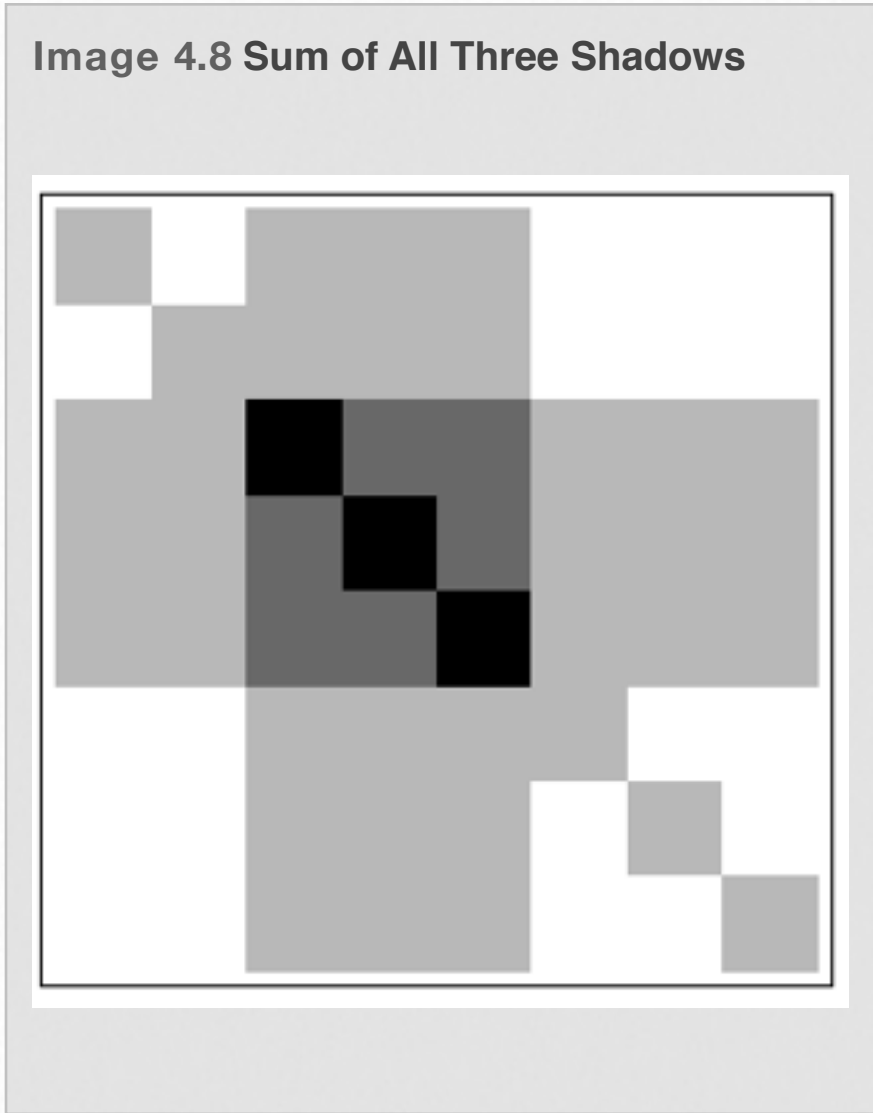
addition of our two shadows we get the idea that our object is mostly likely somewhere in the nine blocks just off center of our platform. Looking back at our object Image 4.4, we can see that this is true. However, those nine shaded blocks really don't give us a very good idea of what our object looks like.

It seems we will need (at least) one more scan. Let's try an oblique (The diagonal) Shadow Image 4.7. This Shadow tells us a



lot more all by itself than either of the other two. However, that's only because the object was diagonal to begin with. Even though this shadow tells us more, it still only narrows down where the object could be to eight blocks running diagonally across the screen.

Our final step is to add all three shadows Image 4.8. This summation gives a pretty good idea of where our object is. There's some guess work there. The darkest spots on the image (three diagonal squares) are where there was a shadow on all three shadow arrays. So, we are pretty sure our object is contained in those three spots.



The Shadow Lesson was performed in Mathematica and the step by step process is shown in Movie 4.1.

Movie 4.1 Video of Lesson Shadow

```

XrayLessonShadowOnly

1. First: We create a simple 2D "Object" to scan:

tblank = Table[0, {i, 0, 7}, {j, 0, 7}];
image = {{0, 0, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 0, 0, 0, 0, 0}, {0, 0, 1, 0, 0, 0, 0, 0},
{0, 0, 0, 1, 0, 0, 0, 0}, {0, 0, 0, 0, 1, 0, 0, 0}, {0, 0, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 0, 0, 0, 0, 0},
{0, 0, 0, 0, 0, 0, 0, 0}};
ArrayPlot[image, Mesh -> True]

2. Now lets find the horizontal shadow of the object.

horShadow = {{0, 0, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 0, 0, 0, 0, 0}, {1, 1, 1, 1, 1, 1, 1, 1}, {1, 1, 1, 1, 1, 1, 1, 1},
{1, 1, 1, 1, 1, 1, 1, 1}, {0, 0, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 0, 0, 0, 0, 0}};
GraphicsRow[
{ArrayPlot[image, Mesh -> True,
Epilog -> {{Red, Arrow[{{0, 0.5}, {8, .5}}]}, {Red, Arrow[{{0, 1.5}, {8, 1.5}}]}, {Red, Arrow[{{0, 1.5}, {8, 1.5}}]},
{Red, Arrow[{{0, 2.5}, {8, 2.5}}]}, {Red, Arrow[{{0, 3.5}, {4.5, 3.5}}]}, {Red, Arrow[{{0, 4.5}, {3.5, 4.5}}]},
{Red, Arrow[{{0, 5.5}, {2.5, 5.5}}]}, {Red, Arrow[{{0, 6.5}, {8, 6.5}}]}, {Red, Arrow[{{0, 7.5}, {8, 7.5}}]}]},
ArrayPlot[horShadow, Mesh -> True], TableForm[horShadow]]

3. Now Let's find the vertical Shadow of the object

vertShadow = {{0, 0, 1, 1, 1, 0, 0, 0}, {0, 0, 1, 1, 1, 0, 0, 0}, {0, 0, 1, 1, 1, 0, 0, 0}, {0, 0, 1, 1, 1, 0, 0, 0},
{0, 0, 1, 1, 1, 0, 0, 0}, {0, 0, 1, 1, 1, 0, 0, 0}, {0, 0, 1, 1, 1, 0, 0, 0}, {0, 0, 1, 1, 1, 0, 0, 0}};
GraphicsColumn[
{ArrayPlot[image, Mesh -> True,
Epilog -> {{Red, Arrow[{{0.5, 8}, {.5, 0}}]}, {Red, Arrow[{{1.5, 8}, {1.5, 0}}]}, {Red, Arrow[{{2.5, 8}, {2.5, 6}}]},
{Red, Arrow[{{3.5, 8}, {3.5, 5}}]}, {Red, Arrow[{{4.5, 8}, {4.5, 4}}]}, {Red, Arrow[{{5.5, 8}, {5.5, 0}}]},
{Red, Arrow[{{6.5, 8}, {6.5, 0}}]}, {Red, Arrow[{{7.5, 8}, {7.5, 0}}]}]}, ArrayPlot[vertShadow, Mesh -> True],
TableForm[vertShadow]]

4. We add those two images together to find our shadow array.

GraphicsRow[{TableForm[Ttot = horShadow + vertShadow],
Manipulate[ArrayPlot[Threshold[Ttot, x], Mesh -> True], {x, 0, 10, 1}]}]

5. The shadow array from above doesn't define the object very well. So, we find we must add another angle to our scan.

diagShadow = {{1, 0, 0, 0, 0, 0, 0, 0}, {0, 1, 0, 0, 0, 0, 0, 0}, {0, 0, 1, 0, 0, 0, 0, 0}, {0, 0, 0, 1, 0, 0, 0, 0},
{0, 0, 0, 0, 1, 0, 0, 0}, {0, 0, 0, 0, 0, 1, 0, 0}, {0, 0, 0, 0, 0, 0, 1, 0}, {0, 0, 0, 0, 0, 0, 0, 1}};
GraphicsGrid[
{{ArrayPlot[image, Mesh -> True,
Epilog -> {{Red, Arrow[{{0, 1}, {1, 0}}]}, {Red, Arrow[{{0, 2}, {2, 0}}]}, {Red, Arrow[{{0, 3}, {3, 0}}]},
{Red, Arrow[{{0, 4}, {4, 0}}]}, {Red, Arrow[{{0, 5}, {5, 0}}]}, {Red, Arrow[{{0, 6}, {6, 0}}]},
{Red, Arrow[{{0, 7}, {7, 0}}]}, {Red, Arrow[{{0, 8}, {2, 6}}]}, {Red, Arrow[{{1, 8}, {8, 1}}]},
{Red, Arrow[{{2, 8}, {8, 2}}]}, {Red, Arrow[{{3, 8}, {8, 3}}]}, {Red, Arrow[{{4, 8}, {8, 4}}]},
{Red, Arrow[{{5, 8}, {8, 5}}]}, {Red, Arrow[{{6, 8}, {8, 6}}]}, {Red, Arrow[{{7, 8}, {8, 7}}]}]},
ArrayPlot[tblank, Frame -> False], ArrayPlot[tblank, Frame -> False]],
{ArrayPlot[tblank, Frame -> False], ArrayPlot[diagShadow, Mesh -> True], ArrayPlot[tblank, Frame -> False]},
{ArrayPlot[tblank, Frame -> False], ArrayPlot[tblank, Frame -> False], ArrayPlot[diagShadow]}]

Now we add all three shadow arrays to produce a new total shadow that includes all three scans.

GraphicsRow[{tTot = horShadow + vertShadow + diagShadow, Manipulate[ArrayPlot[Threshold[tTot, x]], {x, 0, 5, 1}],
ArrayPlot[image]}]

```

This video walks you through the Shadow Lesson step-by-step including the Mathematica code. Tap on it for full screen view

3D Object From Shadows

In this section we are going to apply our previous lessons to create a 3D model. we will be using multiple shadow arrays to create a 3D model of a water bottle. While the steps are simple the execution of the steps can get complicated. As such, we will only look at the individual steps superficially.

The first step was to take a picture of the object we want to image. In this case it was a water bottle that was handy in the office Image 4.9.

Image 4.9 Water Bottle



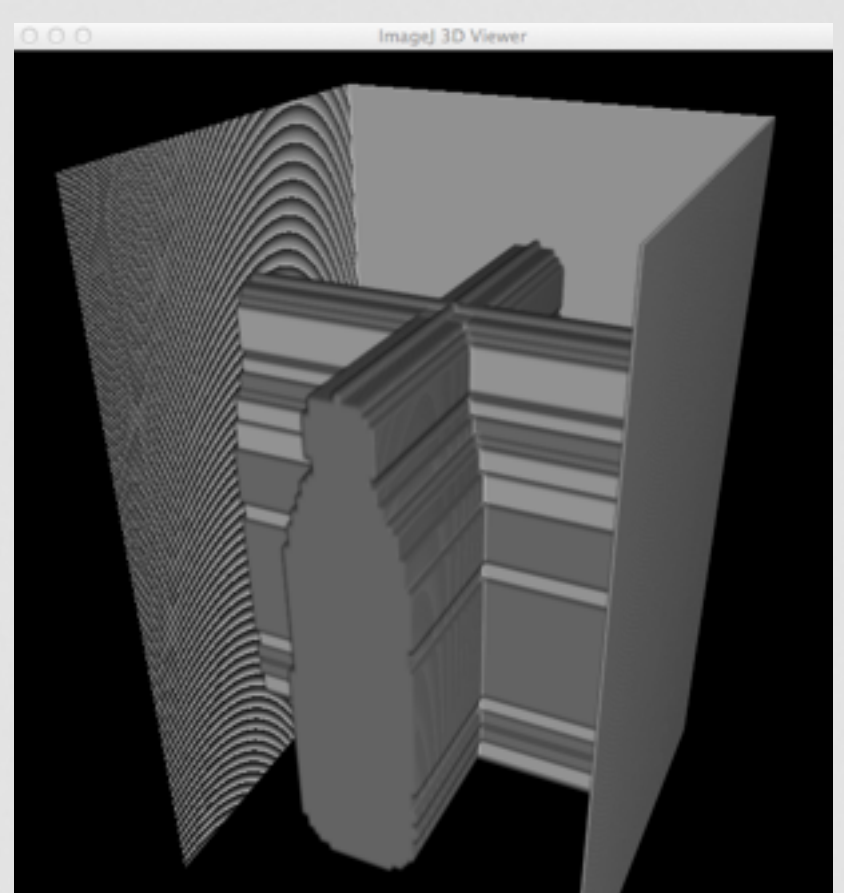
This picture was then binarized as seen in Image 4.10. This was actually done twice, once for the front of the bottle and again for the side of the bottle. However, in the case of a round bottle, these images look identical.

Image 4.11 Binary of Bottle



Next we create shadow arrays for each view of the bottle and then add them. Remember we are adding 3D arrays in this case.

Image 4.10 Added 3D arrays for bottle



Much like our 2D example from above we get an idea of what the bottle should look

like. However, we know that the bottle is not 3D cross shape. This is where a little bit of artistry comes into play on our tomography.

We apply a threshold. We do this by adjusting the transparency of the low numbers. Remember, some of the values in our 3D array only have a one because that location only showed up in one of our shadow arrays. However, some of the cells showed up in both arrays and therefore have a value of two. By adjusting the low value arrays to appear transparent, we see a much better representation of our bottle Image 4.12.



However, this bottle sort of reminds me of my youth playing eight bit video games. It does resemble the water bottle. But where the water bottle is a cylinder this model shows a square base to it. How can we improve our model? The answer is, of course, “Take more scans from more angles”. Like our previous exercise in two dimensions, adding an oblique would shave some of the squareness off of our bottle.

To see the Mathematica steps to create Image 4.12, play the Movie 4.2 “Building a 3D Model”

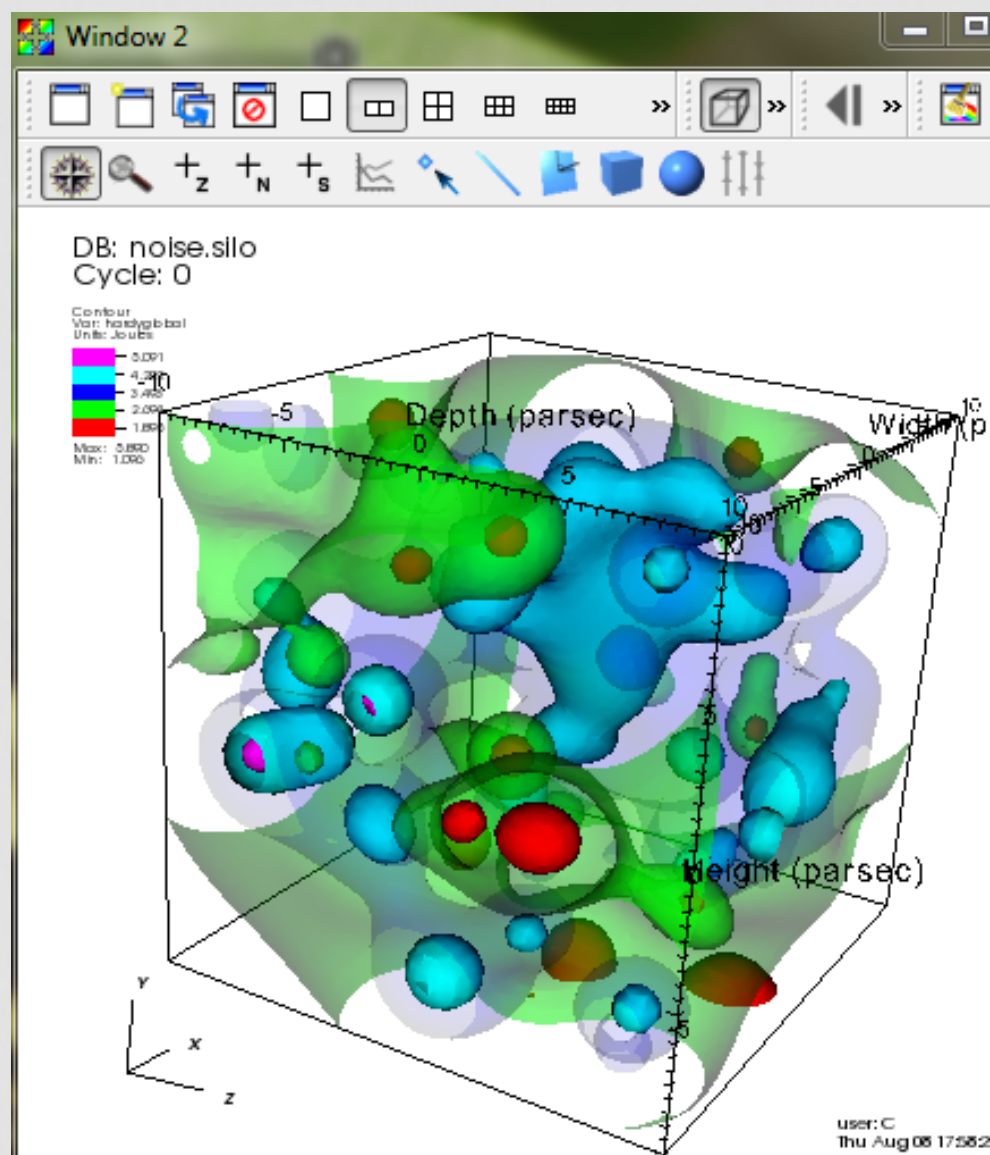


In this video we build a 3D model of a water bottle. (tap on the video to access full screen.)

5

VisIt - work in progress!

Image 5.1 Iso-surface rendering with opacity adjustments of the noise.silo file supplied by VisIt.



This Chapter will be devoted to demonstrating how to do a multitude of VisIt renderings with 2-D and 3-D data sets. It will include Contours, Iso-surface, Pseudocolor, Truecolor, Histogram,

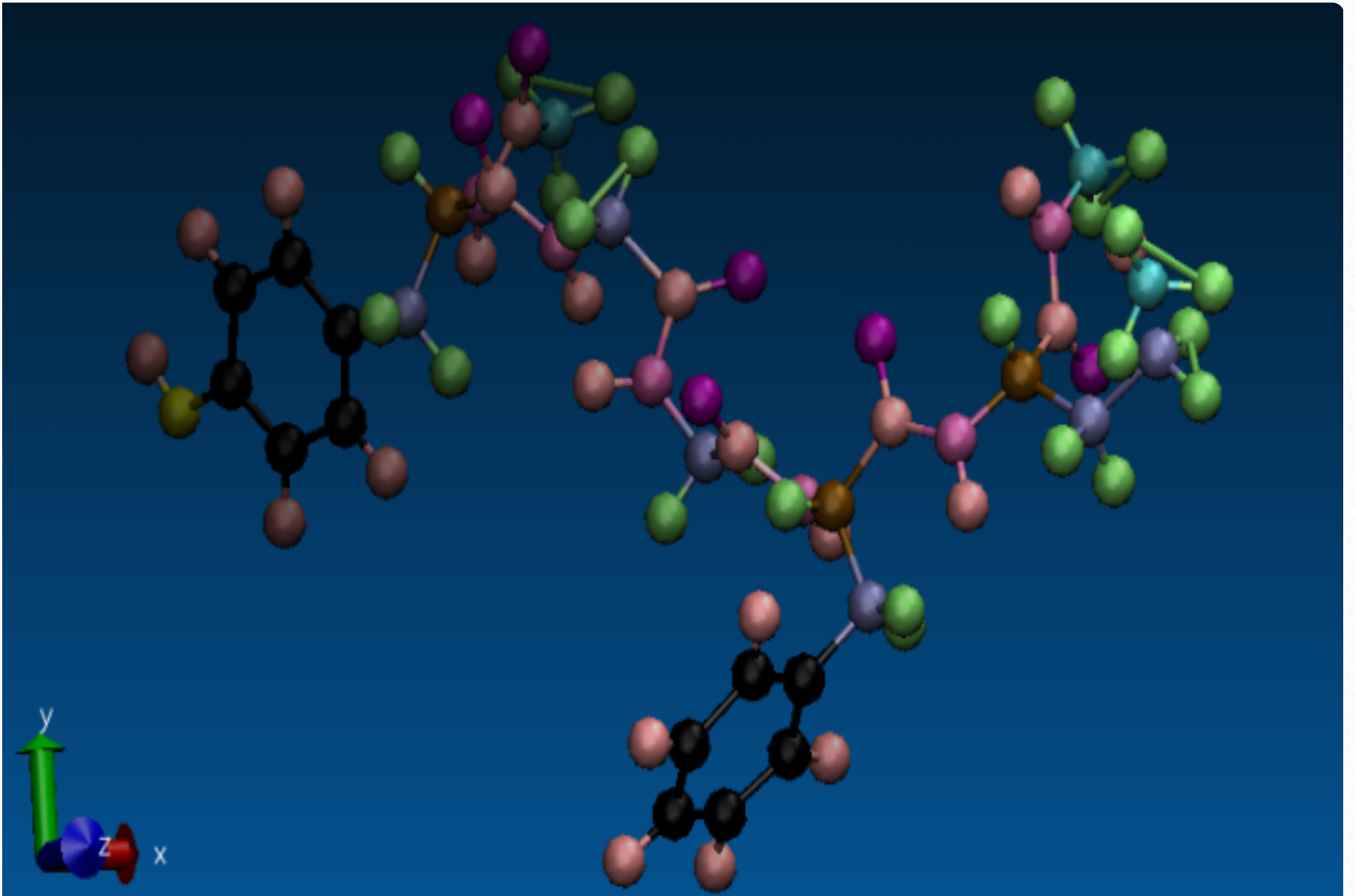
Orthoslicing, and Volume rendering along with sub-parameters to enhance visual effects. The reader should be forewarned that as the file size gets larger, the processing speed dramatically drops and

it's not uncommon that their computer might crash. Hence, the need for super computers that can handle the data size and number of calculations that have to be performed.

At this time the reader should download the appropriate version of [VisIt](#) for their operating system and then install it so that an icon appears on their desktop. The software will come with some data files that we can use.

6

VMD - Coming Soon!



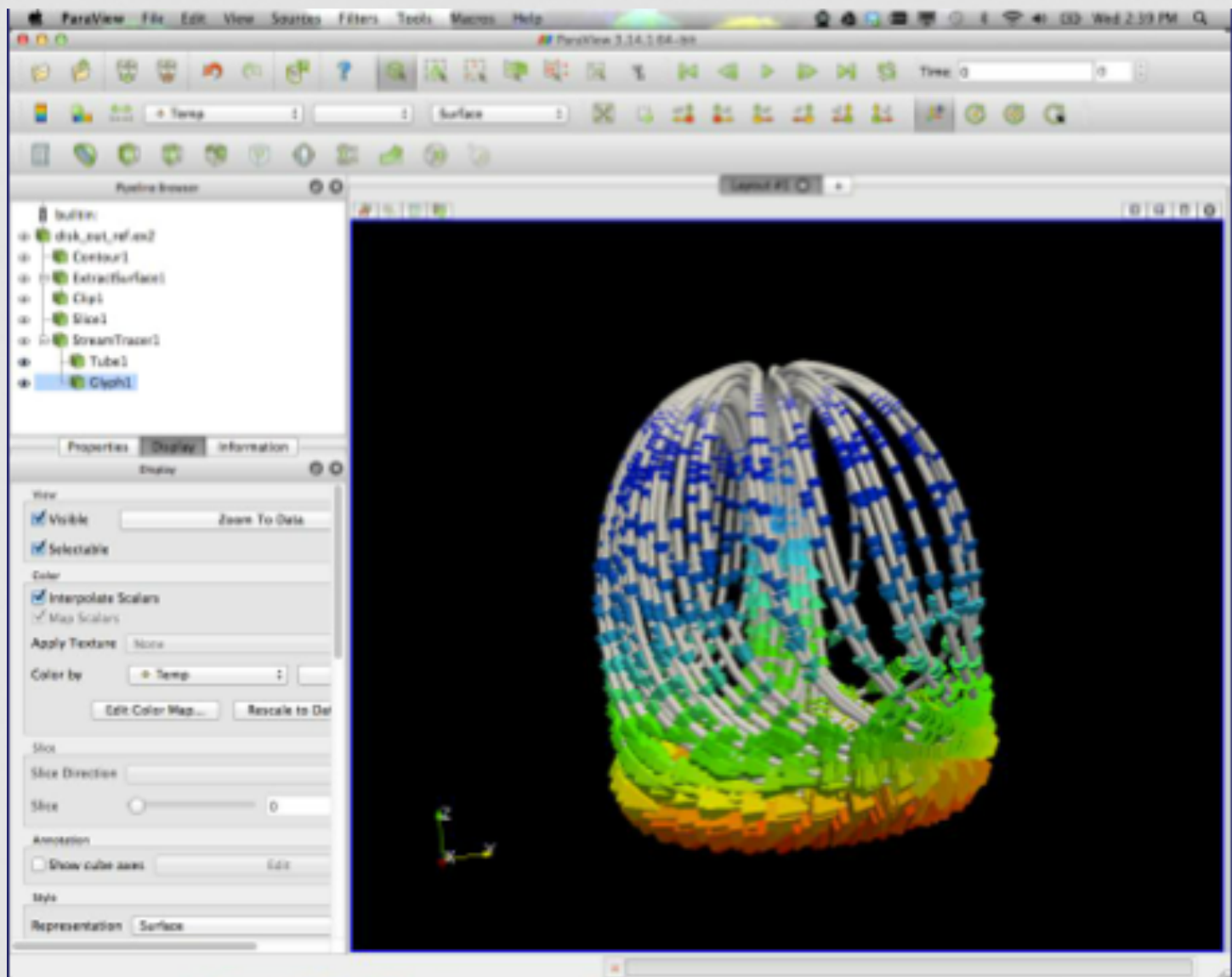
This Chapter is devoted to learning some basic visual effects in the Visual Molecular Dynamics (VMD) software. The files used are the result of molecular dynamic simulations that required the use of high

performance computing due to the sheer number of calculations. After the resultant files are created, we will animate them in VMD.

7

Paraview - Coming Soon!

Image 7.1 This image is courtesy of Texas Advanced Computing Center (TACC)



This image is from a Visualization with ParaView pdf tutorial supplied by TACC and can be found at <https://www.tacc.utexas.edu/user-services/training/course-materials> under the sub-heading of “Introduction to Scientific Visualization on Longhorn - 09/06/2012”

This Chapter focuses on ParaView which is a counter part to VisIt. Several of the rendering styles done in Visit will be demonstrated in ParaView. The purpose is

to contrast two different softwares that essentially do the same thing. The basic principles are the same but the terminology, menu locations, and

sequence of tasks will be different. The reader may find one software may be easier to use than another. The only reason VisIt was demonstrated before ParaView is because that is what the author first learned and has the most experience with.

8

Other cool stuff - Coming Soon -



Kiwi viewer, Augmented reality, 123D

Kiwi Viewer(?)



Add Kiwi Viewer stuff

Augmented Reality(?)



Augmented reality stuff

123D



Add 123D stuff

Binarize

to convert an image from having a string of possible values to having possible values. Usually some threshold is set and all values above the threshold are changed to one and all values below that threshold are changed to zero.

Related Glossary Terms

Drag related terms here

Index

Find Term

Bit

A single on/off switch of a computer. Also, the smallest digit of a binary number.

Related Glossary Terms

Drag related terms here

Dilate

To increase in average radius. Generally, dilate is used to increase the objects in a picture. These objects grow outward from a center point v
dilation.

Related Glossary Terms

Drag related terms here

Pixel

The smallest part of a picture. It stems from the term “Picture Element” as a picture is made of pixels. Imagine that all digital pictures are made of tiny dots. The pixels would be those tiny dots.

Related Glossary Terms

Drag related terms here

Pseudocolor

The process of representing values

Related Glossary Terms

Drag related terms here

Tomography

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam
nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo co

Related Glossary Terms

Drag related terms here