Zihan (Anna) Wu
Prof. Mokshay Madiman
MATH 349-080
19 November 2018

Linear Algebra in Colorimetry and Digital Imaging

Introduction

Linear algebra - the study of linear sets of equations under closed systems - is not something that everyone knows of or even learns of in school. This branch of mathematics deals mostly with vectors, matrices, dimensions, and the manipulation of those three items. Its definition may sound only specifically applicable and marginally useful, but linear algebra is actually incredibly prevalent in everyday life and technology. Unbeknownst to us, its mechanics are working everyday behind the colorful screens of our phones and computers. Everything from the instagram filters, photo editing, printing, and color computer graphics, all could not exist without some understanding of linear algebra.

Image and Color Basics

Hundreds of thousands of images run by us daily on screens, but how do those images work behind the scenes and what part does mathematics play in this? It may not be obvious at first, but math and linear algebra take a big part in the representation of images and color, especially when one thinks of digital imaging in terms of matrices.

Starting off simple there are many types of images, the most basic of which would be the binary image - a picture of black and white pixels (Abadpour). If we were to think of this image digitally, one could break it down into a grid of binary numbers or a matrix of 0s for black and 1s for white. For grayscale images, the range of integers is extended to 0 - 255 (the range of numbers that can be defined by one byte of information in a computer). Right now, any value on this grayscale can be thought of as one number on a number line, or in other words, in one dimension. A point on a line. This is not limited to noir tones though, as other colors can be represented similarly. A greyscale is only a gradient between two binary colors: black and white. In actuality, we can represent any combination of two colors in one dimension. Consider a line with yellow on one end and red on the other. Integer values on this line could describe yellow to red colors and everything in between.

This however isn't a particularly useful way of representing colors. We want to be able to represent images with all colors of the rainbow, not just monochromatic or analogous color images. This is why most digital colors are defined in three dimensions. The most widely used color system in web, movies, and any photo editing software is RGB. This scheme consists of a red, green, and blue axis oriented perpendicular to each other. Each axis, again has only positive values of 0 to 255. In this case, black is the origin and - in accordance with the additive color model - white is the combination of all three saturated colors (Doughty). Now each individual

color can be written as three integers pertaining to the amount of red, green, or blue it contains. For example (0,0,0) for black. A point in a three dimensional area.

Color Spaces to Vector Spaces

This 3D model of color can be referred to as a color space and has an interesting relationship to vector spaces in linear algebra. A color space is a model for representing colors in regards to three or more numbers in a coordinate (Stokes). Meanwhile, a vector space is defined as a set of vectors (over a field) that is closed under addition and scalar multiplication (Poole). In other words, for a vector space, any linear combination of two vectors in the set will produce another vector in that set. If the RGB values are thought of as vectors in R^3 instead of points - with the red, green, and blue axes acting as the basis and each number representing the scaling of an axis vector - it becomes obvious that our current set of RGB vectors is not a vector space. Case in point, white added to white is (255, 255, 255) + (255, 255, 255) which produces a vector outside our set (510, 510, 510).

This result presents a variety of issues. The most blatant problem is that we want to be able to mix colors and transform colors for such programs like Paint and Photoshop. Logically, the values for red + yellow should result in orange and white + white should result in white. But this does not always hold true for our current model; white + white results in "not a valid color." Finding the inverse of a color is not easy either. Subtracting 255 from each value may find the inverse of white, but it won't always find the inverse of every color, same goes for multiplication. So there needs to be a way to add two colors and still have the result be a color. And there must be a way to perform operations on a single color to always get a similar result, like an opposite color.

Thankfully, this color model can be mapped onto all of R^3 to form a vector space. First each vector is divided by 255, so as to work in a range of 0 to 1. Now there are plenty of ways to map from (0,1) to the set of real numbers, it's just a matter of picking one that's intuitive for working with colors. How about f(x) = (x - ½) / (½ - |x - ½|) where x does not equal 0 or 1? If we use this function, we get vectors in the form of:

$$(r^*, g^*, b^*) = \left(\frac{r - 1/2}{1/2 - |r - 1/2|}, \frac{g - 1/2}{1/2 - |g - 1/2|}, \frac{b - 1/2}{1/2 - |b - 1/2|} \right)$$ (Seacreast)

and can transform back and forth using a little bit of algebra. The visual interpretation of this is three axes with red, green, blue on the positive ends and their respective opposites cyan, magenta, yellow on the negative ends. (See Appendix: fig 1)

This particular mathematical and visual representation of color makes a lot of sense in that it is consistent with known color theory. The center or origin represents the mixing of opposing colors, which nets a grey tone. The solely positive values of rgb behave accordingly with additive color theory, with maxed out red green and blue values producing colors close to white. The soley negative values align with subtractive color theory, with maxed cyan, magenta, and yellow creating colors close to black (Doughty).

Color Manipulation

Now, since we're working in all of R^3, addition and scalar multiplication are closed and there's no need to worry about results ending up as invalid colors. We can go ahead and transform these vectors however we like, plus operations on this particular color model are finally intuitive and will produce a consistent and useful result every time. For instance, inverting colors is just takes multiplication now. Multiplication by -1 will turn reds to cyan, greens to magenta, blues to yellow, and whites to black. Furthermore adjusting color contrast is fairly straightforward. Multiplying by a scalar less than 1 brings all color values closer to the center grey - which reduces the contrast. Similarly, multiplying by anything greater than one increases the distance between positive values (light) and negative values (dark) - which increases contrast (Seacrest).

Going back to the idea of an image being a matrix of color pixels, assume matrix M is an image in which every ijth pixel is represented by the color vector $\mathbf{v}_{ij}$ and A is some matrix representing a transformation. Then we can apply a linear transformation T to the image such that $T(M) = \{A\mathbf{v}_{ij} \mid \mathbf{v}_{ij} \in M\}$. In other words, we could use any 3x3 matrix A to multiply all of our image's 3x1 color vectors, and that would result in some sort of image-color manipulation. Some really interesting concepts can be defined visually this way, like a "rotation of color" or a "reflection of color" if A were to be some 3x3 orthogonal matrix. With this, we start to get a small idea of how the numerous photo filters and photo adjustment layers in applications work.

CIE Standard Color Spaces

Of course, the example above is only a simplified version of how color spaces work and gives a general idea of how image editing works. In practice there are several, much more complicated, approaches to representing colors via spaces and no concrete "best" way to think about them in terms of math. The most established color space history dates all the way back to work done in 1931 by the French International Commission on Illumination (CIE). They fabricated the famous horseshoe-shaped CIE XY chromaticity diagram in which many color spaces today are based around (Sawh). (See figure 4)

Through scientific research on color matching and how the eye works, the CIE defined a standard for color representation. They chose the wavelengths 700 nm for red, 546.1 nm for green, and 435.8 nm for blue as the best wavelengths for which to represent the visible colors. They then proceeded to find and graph how every wavelength on the visible spectrum of light could be made using a combination of these three selected wavelengths. Through this process, the CIE discovered that even the best three wavelengths could not possibly combine in an additive way to make all visible colors. They found that bright cyan, could not be made without a negative amount (tristimulus value) of red. (See figure 2, 3)

This again created many inconveniences at the time, one being that negative values were not ideal for performing many calculations with the extra need of subtractions in a time before computers (Higham). The other issue touches back on color spaces. It became difficult to represent the negative values - and thus full visible color spectrum - in a compact 3D and 2D space. Starting with a RGB colorspace in which 0-1 meant least to most saturated and the R, G, and B vectors (1,0,0) (0,1,0) and (0,0,1) formed a basis, the CIE wanted to map this 3D space onto the 2D triangular unit plane that went through each positive side of the RGB vectors (Blackwell). Each color vector's length on this plane would be one, hence "unit plane."

Furthermore, since vectors with length one correspond with fully saturated colors, this would mean that each fully saturated color could be represented by only two coordinates on the unit plane (See figure 5). It was a simple, efficient, and ingenious way to depict color spaces in 2D, however, the idea could not work with negative red values. Colors made from negative amounts of red were not in the positive octant of the color space, and thus did not land on the unit plane.

To solve this problem, the CIE invented a set of new imaginary colors: X, Y, and Z, where Y stood for perceived luminance and X and Z corresponded with perceived color. The combination of these imaginary colors would form the XYZ color space, which was defined as the following linear transformation of the RGB color space:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.49 & 0.31 & 0.20 \\ 0.17697 & 0.81240 & 0.01063 \\ 0 & 0.01 & 0.99 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}.$$ (Higham)

So the XYZ color space is just the set of all vectors resulting from a matrix multiplication of all RGB vectors. These new imaginary XYZ colors guaranteed that any visible color could be represented using only positive amounts. Now they could represent all saturated colors on the 2D unit plane in XYZ colorspace. They used another transformation to go from the 3 dimensional XYZ space, to the 2 dimensional area of their chromaticity diagram:

$$x = \frac{X}{X+Y+Z}, \qquad y = \frac{Y}{X+Y+Z} \qquad (z = 1 - x - y)$$ (Higham)

Basically a simple projection of vectors onto a plane, another fundamental linear algebra concept. Mapping the vectors for the visible colors onto the unit plane traces out a horseshoe shape that the CIE appropriately dubbed the “spectral locus” in their chromaticity diagram (See figure 4, 5).

Chromaticity Diagram

Color manipulation on CIE’s chromaticity diagram is very intuitive. Since we have a way to transform XYZ color space coordinates to (x,y) chromaticity diagram coordinates and vice versa, this makes color manipulation in XYZ easier to visualize and understand as a consequence. To mix equal amounts of two colors in the chromaticity diagram, take the coordinates of each color and find the midpoint. To find the complementary color, take the color coordinate and draw a line through white, defined as (0.33, 0.33). The color coordinate on the other side of white, with the same distance away from white as the current color should be its complement. In other words, in line with additive color theory, adding two complementary colors together creates white.

Keeping this in mind, if we find the wavelengths (700nm, 546.1nm, 435.8nm) of the previous RGB primaries on the spectral locus and connect their coordinates, we create a triangle inside the locus. (See figure 4) The area inside this triangle, called the color gamut, represents all the colors that can be successfully made with mixing the specified RGB primaries. The large portion of blue-green area that lies outside of the triangle are colors that require negative amounts of red to create (Abraham). Picking different wavelengths for the RGB primaries, or

using entirely different colors as primaries, will result in a completely different area of representable colors.

Moving Between Color Spaces

Virtual colors on monitors do not have a specific defined standard. The color coordinates seen in web development, like rgb(255, 255, 255), do not always use the RGB wavelengths specified by the CIE for their primaries. In fact, the coordinates can refer to any wavelength of red, green, or blue depending on the software, use, company, etc. The most common color space is the sRGB color space created by Microsoft and the Adobe RGB color spaces. Like the CIE RGB space, they're obtained from some linear transformation of the XYZ color space and can be mapped onto the chromaticity diagram (See figure 6).

Since all the common color spaces are derivatives of the CIE's color spaces, there are ways of translating between them via linear algebra. Take for the sRGB and Apple RGB spaces written here based on CIE's XYZ color space (Lindbloom).

XYZ to sRGB

$$\begin{bmatrix} Rs \\ Gs \\ Bs \end{bmatrix} = M \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} .4125 & .3576 & .1804 \\ .2127 & .7152 & .0722 \\ .0193 & .1192 & .9503 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

XYZ to Apple RGB

$$\begin{bmatrix} Ra \\ Ga \\ Ba \end{bmatrix} = A \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} .4497 & .3162 & .1845 \\ .2447 & .672 & .0833 \\ .0252 & .1412 & .9225 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

To go from Apple RGB vectors to sRGB vectors, one only needs to find a way to write XYZ in terms of sRGB and then transform from Apple RGB to XYZ to sRGB. Linear algebra can easily solve this with inverses of matrices and some matrix-vector multiplication:

$$\begin{bmatrix} Rs \\ Gs \\ Bs \end{bmatrix} = M \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \rightarrow M^{-1} \begin{bmatrix} Rs \\ Gs \\ Bs \end{bmatrix} = M^{-1} M \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \rightarrow \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = M^{-1} \begin{bmatrix} Rs \\ Gs \\ Bs \end{bmatrix}$$

$$\begin{bmatrix} Ra \\ Ga \\ Ba \end{bmatrix} = A \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = A\, M^{-1} \begin{bmatrix} Rs \\ Gs \\ Bs \end{bmatrix}$$

Currently, there is no accurate and convenient way to encompass the full spectrum of visible colors in just three real primary colors. So each colorspace, depending on the different primaries used, will have a different gamut within the bounds of the chromaticity diagram (See figure 6). This is why no virtual image can truly capture all the colors seen in real life and why colors can appear different when seen from a photo or camera vs from the real world. If a photograph is taken of something with a color outside the camera's color space gamut, the color will usually get approximated to the closest expressible color (Abraham). Same goes for transferring images from programs with one color space to programs using a different color space. Say an image had a cyan color within the gamut of the Adobe color space. The cyan looks nice and bright in Adobe editing software, like Photoshop and Illustrator, but appears much more dull and blue-ish when the image is viewed through an application like Windows Photos. This would be because Windows Photos is using the microsoft sRGB color space, which has a gamut that does not include bright cyan, so the software does the next best thing and approximates the color to something within reach.

This kind of "color rounding" is not limited to translating between software, but can also apply to sending digital images to other devices and to the analog world. It can occur across monitors, as different manufacturing companies may use different RGB primaries as color pixels

in screens. So, screens may also have different color gamuts to work within. Furthermore, color approximation happens a lot in printing. Printers use the CMYK model instead of the RGB one because colored inks behave very differently from colored lights and pixels when it comes to mixing. The CMYK model has a gamut that is quite different from the RGB one, and designers and artists know that they have to be especially careful when printing images because of this fact. All-in-all, colors can look quite different from device to device, depending on the display monitor, viewing application, recording device, and color spaces used in each.

Conclusion

It's interesting to note how virtual colors and ways of manipulating images are mostly based around color spaces, linear transformations, and the idea of colors as vectors. Surprisingly, there are a lot of technical mathematical concepts working behind all these simple pretty pictures. This just goes to show how truly ingrained and useful linear algebra can be in our everyday lives, without us even knowing.

Appendix

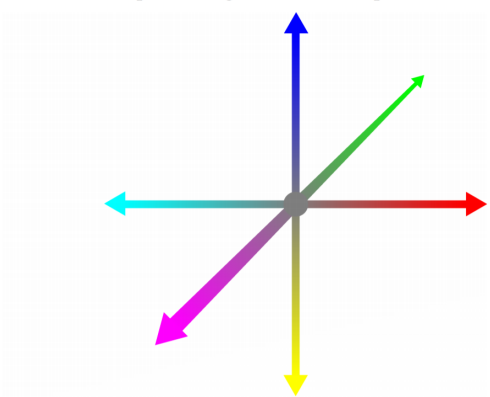

Fig 1: Intuitive RGB colorspace
(Seacrest)

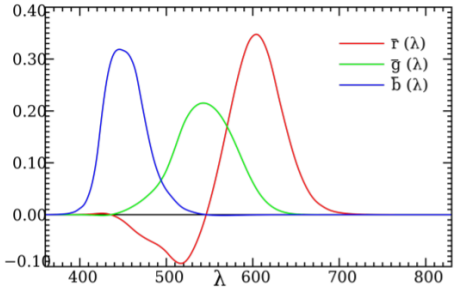

Fig 2: CIE RGB color matching functions
(Higham)

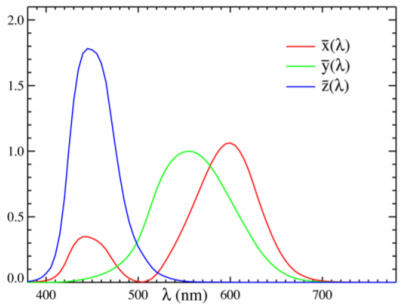

Fig 3: CIE XYZ color matching functions

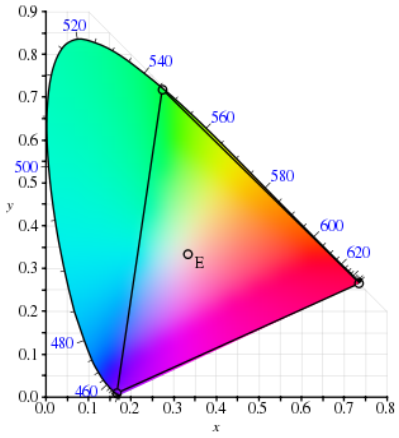

Fig 4: CIE XY Chromaticity Diagram
("CIE 1931 Color Space")

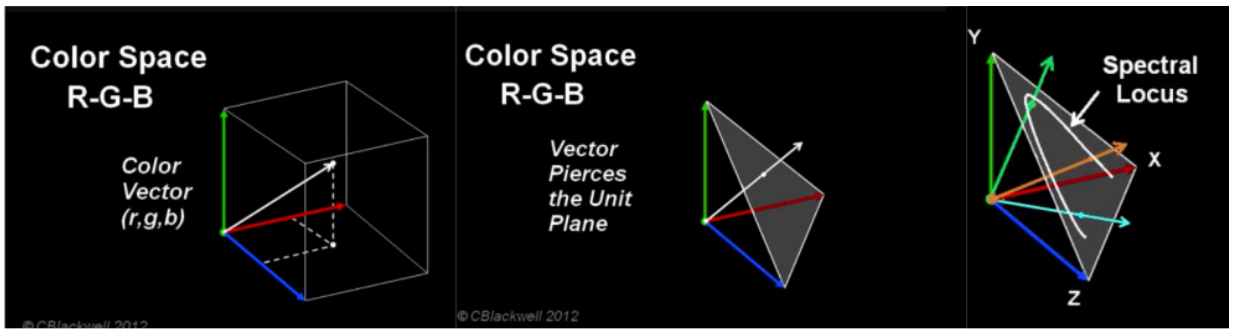

Fig 5: Color mapping onto the unit plane and the spectral locus.
(Blackwell)

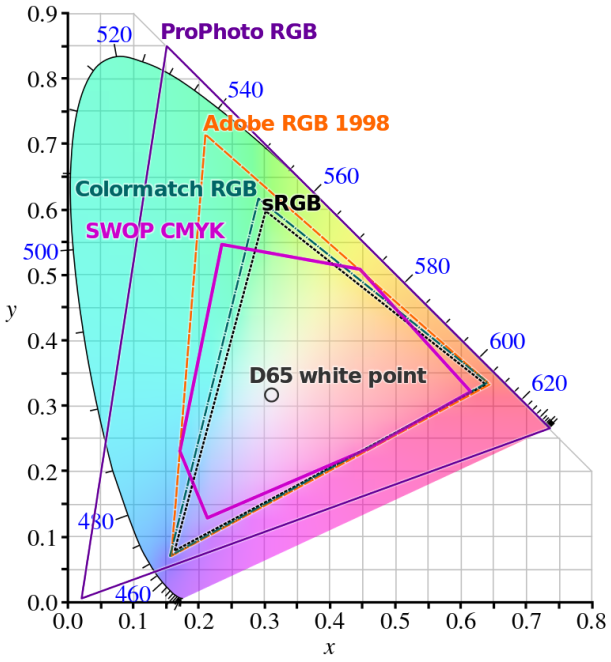

Fig 6: Different color space gamuts on the CIE chromaticity diagram.
("CIE 1931 Color Space")

Works Cited

Abadpour, Arash. "Color Image Processing Using Principal Component Analysis." Sharif University of Technology, July 2005. Web. 26 Nov. 2018. <http://abadpour.com/files/pdf/ethesisl.pdf>.

Abraham, Chandler. "A Beginner's Guide to (CIE) Colorimetry." Medium.com. Medium, 10 Sept. 2016. Web. 10 Dec. 2018.<https://medium.com/hipster-color-science/a-beginners-guide-to-colorimetry-401f1830b65a>

Blackwell, Craig. "Color Vision 2: Color Matching." YouTube. YouTube, 14 Jan. 2013. Web. 10 Dec. 2018.<https://www.youtube.com/watch?v=82ItpxqPP4I>

"CIE 1931 Color Space." Wikipedia. Wikimedia Foundation, 26 Nov. 2018. Web. 10 Dec. 2018.<https://en.wikipedia.org/wiki/CIE_1931_color_space>.

Doughty M. "Color Models." N.p., 27 Mar. 2009. Web. 10 Dec. 2018. <http://acsweb.ucsd.edu/~pwhuynh/CSE3/Lab2/ColorModels.htm>.

Higham, Nicholas J. "Color Spaces and Digital Imaging." University of Manchester, 2015. Web. 7 Dec. 2018. <http://eprints.maths.manchester.ac.uk/2380/1/color.pdf>.

Lindbloom, Bruce Justin. "RGB/XYZ Matrices." Bruce Lindbloom. N.p., 7 Apr. 2017. Web. 10 Dec. 2018. <http://brucelindbloom.com/index.html?Eqn_RGB_XYZ_Matrix.html>.

Poole, David. Linear Algebra: A Modern Introduction. Pacific Grove, Calif. U.a.: Brooks/Cole/Cengage Learning, 2011. Print.

Sawh, Kishore. "A Breakdown Of Color Spaces." SLR Lounge. SLR Lounge, 29 Aug. 2015. Web. 7 Dec. 2018. <https://www.slrlounge.com/breakdown-color-spaces-really-grasp/>.

Seacrest, Tyler. "Coloring Linear Algebra." Harvey Mudd College, May 2006. Web. 26 Nov. 2018.<https://www.math.hmc.edu/seniorthesis/archives/2006/tseacres/tseacres-2006-thesis-poster.pdf>.