H
Human Visual System

Image and video compression deal with large quantities of data, and one way to achieve good compression of images is to lose some of this data. The data loss must be done selectively and the guiding principle is to lose data for which the human visual system is not sensitive. This requires a detailed knowledge of color, its representations, and the way the eye perceives it. The hardware used to display color is also an important consideration in any computer application that uses color. The CRT (Section 6.1.1) is currently a common output device. It displays color by means of light emitting phosphors, but those cannot reproduce pure spectral colors. Color printers use various inks (in the form of liquid, wax, or powder) to mix colors on paper, and they suffer from the same shortcoming. Printers, CRTs, LCD displays, and all other output devices have their limitations and work best when using certain color representations. This appendix is an introduction to color, its representations, and the way the eye perceives color.

Some of the ideas and results presented here were developed by the International Committee on Illumination (Commission Internationale de l'Éclairage, or CIE), an international organization with a long history of illuminating various aspects of light and illumination.

H.1 Color and the Eye

We rarely find ourselves in complete darkness. In fact, most of the time we are flooded with light. This is why people get used to having light around and, as a result, they only rarely ask themselves *what is light*?

The best current explanation is that light can be understood (or interpreted) in two different ways, as a wave, or as a stream of particles. The latter interpretation sees light as a stream of massless particles, called *photons*, moving at a constant speed. The most important attribute of a photon is its frequency, since the photon's energy is proportional to it. Photons are useful in physics to explain a multitude of phenomena (such as the interaction of matter and light). In computer graphics,

the most important property of light is its color, which is why we use the former interpretation and we consider light to be a wave.

What "waves" (or undulates) in light is the electric and magnetic fields. When a region of space is flooded with light, those fields change periodically as we move from point to point in space. If we stay at one point, the fields also change periodically with time. Visible light is thus a (small) part of the electromagnetic spectrum (Figure H.1) which includes radio waves, ultraviolet, infrared, X-rays, and other types of radiation.

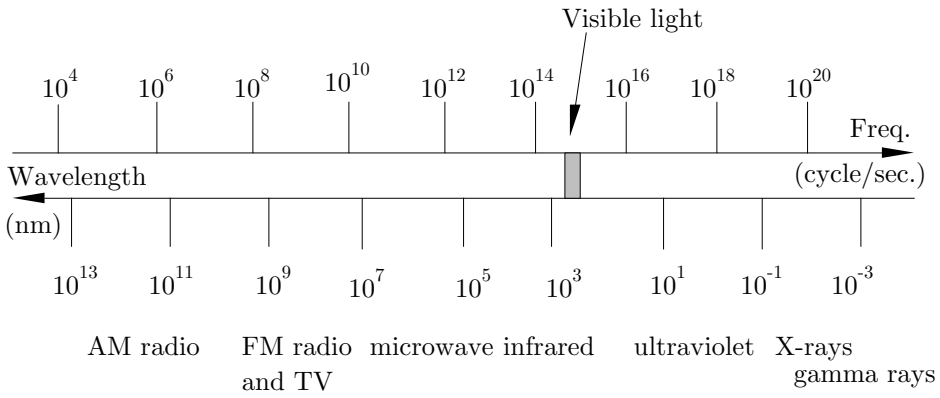


Figure H.1: The Electromagnetic Spectrum.

The most important properties of a wave are its frequency f, its wavelength λ, and its speed. Light moves, obviously, at the speed of light (in vacuum, it is $c \approx 3\times 10^{10}$cm/s). The three quantities are related by $f\lambda = c$. It is important to realize that the speed of light depends on the medium in which it moves. As light moves from vacuum to air to glass, it slows down (in glass, the speed of light is about $0.65c$). Its wavelength also decreases, but its frequency remains constant. Nevertheless, it is customary to relate colors to the wavelength and not to the frequency. Since visible light has very short wavelengths, a convenient unit is the nanometer (1 nm=10^{-9} m).

Visible light ranges from about 400 nm to about 700 nm and the color is determined by the wavelength. A wavelength of 420 nm, for example, corresponds to pure violet, while 620 nm is perceived by the eye as pure red. Using special lasers, it is possible to create pure (monochromatic) light consisting of one wavelength (Figure H.2a). Most light sources, however, output light that is a mixture of several (or even many) wavelengths, normally with one wavelength dominating (Figures H.2b and H.12).

The colors of the spectrum that are most visible to the human eye are (Figure H.7) violet (390–430), blue-violet (460–480), cyan, green (490–530), yellow (550–580), orange (590–640), and red (650–800).

White light is a mixture of all wavelengths, but what is gray light? It turns out that the wavelength of light is not its only important attribute. The *intensity*

is another attribute that should be considered. Gray light is a mixture of all wavelengths, but at a low intensity. When doing computer graphics, the main problem is how to specify the precise color of each pixel to the hardware. In many real-life situations, it is sufficient to say "I think I would like a navy blue suit," but computer hardware requires, of course, a much more precise specification. It, therefore, comes as a surprise to discover that color can be completely specified by just three parameters. Their meanings depend on the particular *color model* used. The RGB model is popular in computer graphics. In the printing industry, the CMYK model is normally used. Many artists use the HLS model. These models are discussed below.

Figure H.2b shows a simplified diagram of light smeared over the entire range of visible wavelengths, with a spike at about 620 nm (red), where it has a much higher intensity. This light can be described by specifying its *hue*, *saturation*, and *luminance*. The hue of the color is its dominant wavelength—620 nm in our example. The luminance is related to the intensity of the light. It is defined as the total power included in the spectrum and it is proportional to the area under the curve, which is $L = (700-400)A + B(D-A)$. The saturation is defined as the percentage of the luminance that resides in the dominant wavelength. In our case, it is $B(D-A)/L$. When there is no dominant wavelength (i.e., when $D = A$ or $B = 0$), the saturation is zero and the light is white. Large saturation means either large $D - A$ or small L. In either case, there is less white in the color and we see more of the red hue. Large saturation therefore corresponds to *pure color*.

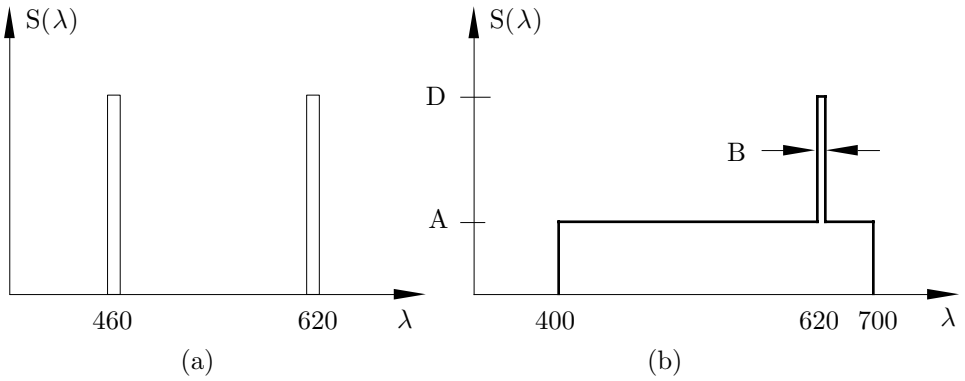


Figure H.2: (a) Pure Colors. (b) A Dominant Wavelength.

H.2 The HLS Color Model

This model was introduced in 1978 by Tektronics, aiming for an intuitive way to specify colors. The name stands for hue, lightness, and saturation. Lightness (or value) refers to the amount of black in the color. It controls the brightness of the color. Maximum lightness always creates white, regardless of the hue. Minimum lightness results in black. Saturation (or chroma) refers to the amount of white in the color. It controls the purity or vividness of the color. Low saturation means

more white in the color, resulting in a pastel color. Very low saturation results in a washed-out color. For a pure, vivid color, the saturation should be maximum. The achromatic colors black, white, and gray have zero saturation and differ in their values.

The HLS model is summarized in the double cone of Figure H.3. The vertical axis corresponds to L (lightness). It starts at zero (black) at the bottom and ends at one (white) at the top. The distance from the central axis corresponds to S (saturation). All points on the axis have zero saturation, so they correspond to shades of gray. Points farther away from the axis have more saturation; they correspond to more vivid colors. The H parameter (hue) corresponds to the hue of the color. This parameter is measured as an angle of rotation around the hexagon.

H.3 The HSV Color Model

The HSV model also uses hue, saturation, and value (lightness). It is summarized in the cone of Figure H.4. This is a single cone where the value V, which corresponds to lightness, goes from 0 (black) at the bottom, to 1 (white) at the flat top. The S and H parameters have the same meanings as in the double HLS cone.

> It's the weird color-scheme that freaks me. Every time you try to operate one of these weird black controls, which are labeled in black on a black background, a small black light lights up black to let you know you've done it!
>
> — Mark Wing-Davey (as Zaphod Beeblebrox) in *The Hitchhiker's Guide to the Galaxy* (1981).

H.4 The RGB Color Model

A *primary hue* is a color in a color model that cannot be made from the other colors used in that model. Primary hues serve as a basis for mixing and creating all other colors in the color model. Any color created by mixing *two* primary hues in a color model is a *secondary hue* in that model.

In the RGB color model, the three primaries are Red, Green, and Blue. They can be combined, two at a time to create the secondary hues. Magenta (pinkish hue) is (R+B), cyan (bluish hue) is (B+G), and yellow is (R+G). There are two reasons for using the red, green, and blue colors as primaries: (1) the cones in the eye are very sensitive to these colors (Figure H.7) and (2) adding red, green, and blue can produce many colors (although not all colors, see discussion of RGB color gamut on page 906).

The RGB color model is useful in computer graphics because of the way color CRTs work. They creates different colors by light emitted from phosphors of different types. The colors are then mixed in the eye of the observer, creating the impression of a perfect mixture. Assuming a range of $[0, 255]$ for each RGB color component, here are some examples of mixed colors:

$$\text{red} = (255, 0, 0),\ \text{magenta} = (255, 0, 255),\ \text{white} = (255, 255, 255),$$
$$50\%\ \text{gray} = (127, 127, 127),\ \text{light gray} = (25, 25, 25).$$

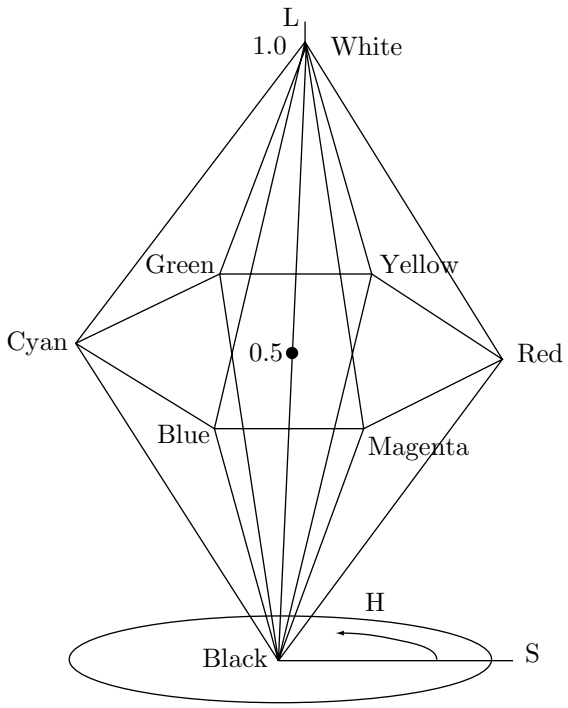


Figure H.3: The HLS Double Hexcone.

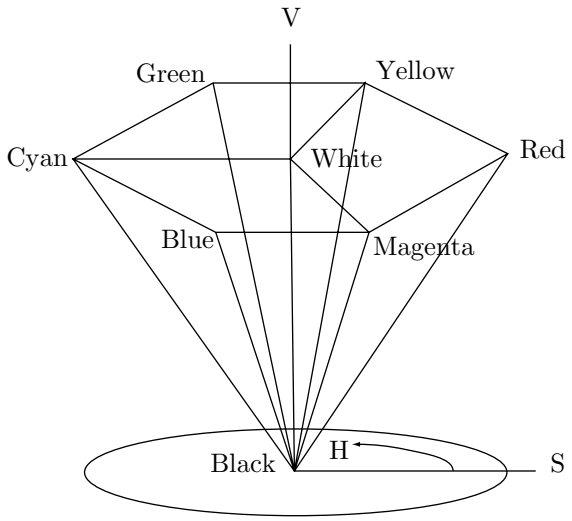


Figure H.4: The HSV Hexcone.

H.4.1 The RGB Cube

The *color gamut* of a color model is the entire range of colors that can be produced by the model. The color gamut of the RGB model can be summarized in a diagram shaped like a cube (Figures H.5 and G.3). Every point in the cube has three coordinates (r, g, b)—each in the range $[0, 1]$—which give the intensities of red, green, and blue of the point. Small values, close to $(0, 0, 0)$, mean a dark shade, whereas anything close to $(1, 1, 1)$ is very bright. Point $(1, 1, 1)$ itself corresponds to pure white. Point $(1, 0, 0)$ corresponds to red and point $(0, 1, 0)$ corresponds to green. Therefore, point $(1, 1, 0)$ describes a mixture of red and green, that is, yellow.

The RGB cube is useful because coordinates of points in it can readily be translated into values stored in the color lookup table of the computer.

◇ **Exercise H.1:** (Easy.) What colors correspond to the diagonal line connecting the black and white corners of the RGB cube?

H.5 Additive and Subtractive Colors

To create a mixture of several colors, we sometimes have to add them and sometimes have to subtract them. Imagine a white wall in a dark room. There is no light for the wall to reflect, so it looks black. We now shine red light on it. Since the wall is white (reflects all colors), it will reflect the red light and will look red. The same is true for green light. If we now shine both red and green colors on the wall, it will reflect both, which our brain interprets as yellow. We say that in this case the colors are added.

To understand the concept of subtracting colors, imagine a white sheet of paper in a bright environment. The paper reflects all colors, so it looks white. If we want to paint a certain spot red, we have to cover it with a chemical (red paint) that absorbs all colors except red. We say that the red paint *subtracts* the green and blue from the original white reflection, so the spot now reflects just red light. Similarly, if we want a yellow spot, we have to use yellow paint, which is a substance that absorbs blue and reflects red and green.

We conclude that in the case where we shine white light on a reflecting surface, we have to subtract colors in order to get the precise color that we want. In the case where we shine light of several colors on such a surface, we have to add colors to get any desired mixture.

> The various colours that may be obtained by the mixture of other colours, are innumerable. I only propose here to give the best and simplest modes of preparing those which are required for use. Compound colours, formed by the union of only two colours, are called by painters virgin tints. The smaller the number of colours of which any compound colour is composed, the purer and the richer it will be. They are prepared as follows:
>
> —Daniel Young, 1861, *Young's Translation of Scientific Secrets*.

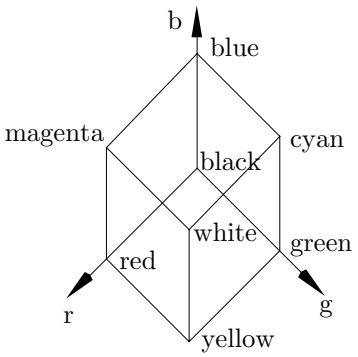


Figure H.5: The RGB Cube.

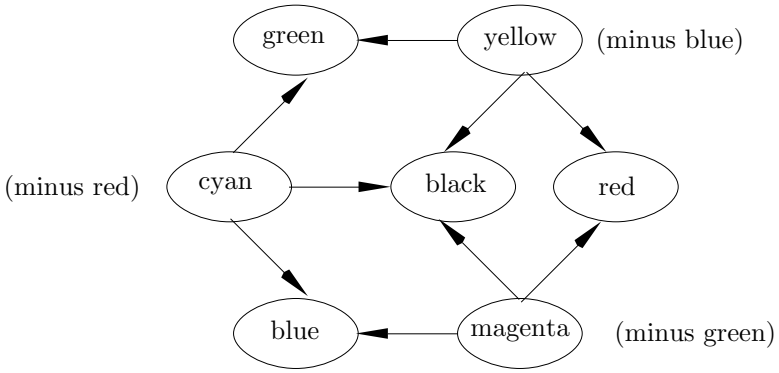


Figure H.6: RGB and CMYK Relationships.

For example, the human eye and its controlling software implicitly embody the false theory that yellow light consists of a mixture of red and green light (in the sense that yellow light gives us the same sensation as a mixture of red light and green light does). In reality, all three types of light have different frequencies and cannot be created by mixing light of other frequencies. The fact that a mixture of red and green light appears to us to be yellow light has nothing whatever to do with the properties of light, but is a property of our eyes. It is a result of a design compromise that occurred at some time during our distant ancestors' evolution.

—David Deutsch, *The Fabric Of Reality*.

H.5.1 Subtractive Color Models

There are two such models: painter's pigments and printing pigments.

Painter's Pigments: The primary colors of this color model are red, yellow, and blue. They were chosen because past artists believed that they were pure colors, containing no traces of any other colors. These three primaries can be mixed, two at a time, to produce the secondaries purple (R+B), green (B+Y), and orange (Y+R). Mixing equal amounts of all three primaries subtracts all colors and, hence, yields black.

Printing Pigments: This color model is also known as *process color* and is the result of development in color ink and printing processes. The three primaries are magenta, yellow, and cyan. The three secondaries are blue (M+C), red (M+Y), and green (C+Y). Mixing equal amounts of all three primaries should yield black, but, because of the properties of real inks, this black is normally not dark enough. In practice, true black is included in this model as an artificial fourth primary (also because black ink is cheaper). It is used when grayscale or black printing is required. The model is therefore sometimes called CMYK (K for black, to avoid confusion with blue) and color printing is known as the four-color process. Figure H.6 shows the relationships between the three CMY primaries and their secondaries.

Because of the particular primaries and secondaries of the CMY model there is a simple relationship between it and the RGB model. The relation is

$$(r, g, b) = (1, 1, 1) - (c, m, y).$$

This relationship shows that, for example, increasing the amount of cyan in a color, reduces the amount of red.

Traditional color printing uses color separation. The first step is to photograph the original image through different color filters. Each filter separates a primary color from the multi colored original. A blue filter separates the yellow parts of the original and creates a transparency with those parts printed in black and white. A red filter separates the cyan parts and a green filter separates the magenta parts. Another transparency is prepared, with the black parts. Each of the four transparencies is then converted to a halftone image (Section H.11) and the four images become masters for the final printing. They are placed in different stages of the printing machine and, as the paper moves through the machine, each stage adds halftone dots of colored ink to the paper. The result is a picture made of four halftone grids, each in one of the CMYK colors. The grids are not superimposed on the paper, but are printed offset. The eye sees dots colored in the four primaries, and the brain creates a mixed color that depends on the number of halftone dots of each primary.

When such a color print is held close to the eye, the individual dots in the four colors can be seen. Today, there are dye sublimation printers that mix wax of different dyes inside the printer to create a drop of wax of the right color which is then deposited on the paper. No halftoning is used. The result is a picture in vivid colors, but these printers are currently too expensive for home use. For more information on color printing and the CMYK model, see [Stone et al. 88].

To simplify the task of editors and graphics designers, several color standards have been developed. Instead of figuring out the ratios of CMYK, the graphics designer looks at a table that has many color samples, selects one, and uses its name to specify it to the printer. One such standard in common use today is the PANTONE matching system. It is described in [Pantone 91].

◇ **Exercise H.2:** A surface has a certain color because of its ability to absorb and reflect light. A surface that absorbs most of the light frequencies appears dark; a surface that reflects most frequencies appears bright. What colors are absorbed and what are reflected by a yellow surface?

> Blueness doth express trueness.
> — Ben Jonson

H.6 Complementary Colors

The concept of complementary colors is based on the idea that two colors appear psychologically harmonious if their mixture produces white. Imagine the entire color spectrum. The sum of all the colors produces white. If we subtract one color, say, blue, the sum of the remaining colors produces the complementary color, yellow. Blue and yellow are thus complementary colors (a dyad) in an additive color model. Other dyads are green and magenta, red and cyan, yellow-orange and cyan-blue, cyan-green and red-magenta, and yellow-green and blue-violet.

Subtractive Complementary Colors: These are based on the idea that two colors look harmonious if their mixture yields a shade of gray. The subtractive dyads are a yellow and violet, red and green, blue and orange, yellow-orange and blue-violet, blue-green and red-orange, and yellow-green and red-violet.

Complementary colors produce strong visual contrast, which creates a feeling of color vibrations or activity.

◇ **Exercise H.3:** Is there such a thing as additive color triads?

H.7 Human Vision

We see light that enters the eye and falls on the retina, where there are two types of photosensitive cells. They contain pigments that absorb visible light and hence give us the sense of vision. One type is the *rods*, which are numerous, are spread all over the retina, and respond only to light and dark. They are very sensitive and can respond to a single photon of light. There are about 110,000,000 to 125,000,000 rods in the eye [Osterberg 35]. The other type is the *cones*, located in one small area of the retina (the fovea). They number about 6,400,000, are sensitive to color, but require more intense light, in the order of hundreds of photons. Incidentally, the cones are very sensitive to red, green, and blue (Figure H.7), which is one reason why CRTs use these colors as primaries. There are three types of cones. The A cones are sensitive to red light. The B cones are sensitive to green light (a little more than the A cones), and the C cones are sensitive to blue light, but their sensitivity is about 1/30 that of the A or B cones.

The trichromatic theory of color vision says that light of a given wavelength stimulates each of the three types of cones in a different way, and it is the *ratio* of those stimuli, not their absolute values, that creates the sensation of color in the brain. As a result, any light intensity and background illumination that happen to produce the same ratio of stimuli will seem to have the same color. This theory explains why any color representation uses three parameters.

Each of the light sensors in the eye, rods and cones, sends a light sensation to the brain that is essentially a pixel, and the brain combines these pixels to a continuous image. The human eye is, thus, similar to a digital camera. Once this is realized, we naturally want to compare the resolution of the eye to that of a modern digital camera. Current digital cameras have from 300,000 sensors (for a cheap camera) to about six million sensors (for a high-quality one).

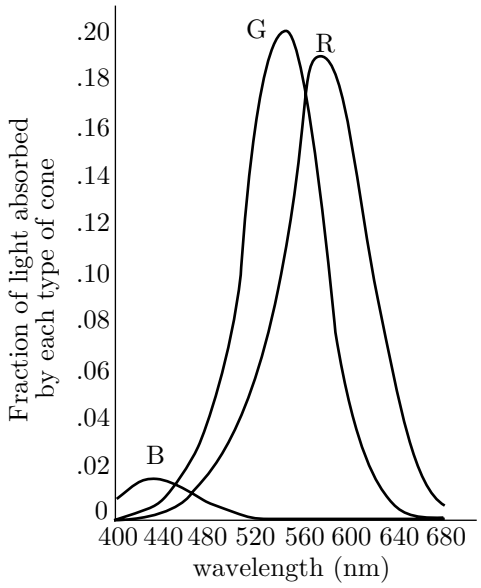


Figure H.7: Sensitivity of the Cones.

Thus, the eye features a much higher resolution, but its effective resolution is even higher when we consider that the eye can move and refocus itself about three to four times a second. This means that in a single second, the eye can sense and send to the brain about half a billion pixels. Assuming that our camera takes a snapshot once a second, the ratio of the resolutions is about 100.

Certain colors—such as red, orange, and yellow—are psychologically associated with heat. They are considered *warm* and cause a picture to appear larger and closer than it really is. Other colors—such as blue, violet, and green—are associated with cool things (air, sky, water, ice) and are therefore called *cool* colors. They cause a picture to look smaller and farther away.

Color

Colors, like features, follow the changes of the emotions.
— Pablo Picasso
There is no blue without yellow and without orange.
— Vincent Van Gogh

H.8 Luminance and Chrominance

The RGB and other color representations discussed earlier have one important drawback, they are not based on the human visual system. We know that the three types of cones in the eye are sensitive to green and (a little less) to red, and are less sensitive to blue. It is also known that the eye can best resolve small spatial details in the middle of the visible spectrum, which is green, and is poorest in this respect in the blue. This is important to an image compression algorithm, since it means that there is no need to preserve small details in blue regions of an image, but small green details are important and should be preserved.

It turns out that a color representation based on the two quantities *luminance* and *chrominance* can take advantage of these features more than the RGB representation. Luminance is a quantity that is closely related to the brightness of a light source as perceived by the eye. It is defined as proportional to the light energy emitted by the light source per unit area. Chrominance is related to the way the eye perceives hue and saturation.

One important aspect of the human visual system is that the perceptual response L^* of the eye to a light source with luminance Y is not linear but is proportional to $Y^{1/3}$. Specifically, the CIE has found that

$$L^* = \begin{cases} 116(Y/Y_n)^{1/3} - 16, & \text{if } Y/Y_n > 0.008856, \\ 903.3(Y/Y_n), & \text{otherwise,} \end{cases}$$

where the quantity L^* is called *lightness* and Y_n is the white reference luminance (see the discussion of illuminant white in Section H.10). The region $Y/Y_n \leq 0.008856$ corresponds to low values of Y (very dark) and is not important.

In addition to this feature (or problem) of the human visual system, the CRT itself is the source of another, physical, problem. It turns out that the intensity I of the light emitted by a CRT depends nonlinearly on the voltage V that is fed to the electron gun. The relation is $I = V^\gamma$, where the voltage V is assumed to be in the range $[0, 1]$, and γ (gamma) is a real number, typically in the range $[2, 3]$. Thus, the light intensity is a function somewhere between the square and the cube of the voltage.

It is one of life's many lucky coincidences that these two features cancel each other out. The two relations $L^* = 116(Y/Y_n)^{1/3} - 16$ and $I = V^\gamma$ imply that the lightness perceived by the eye is proportional to the voltage fed to the CRT if γ is 3 or close to 3. Figure H.8 shows the plots of x^2 and $x^{(1/2)}$ (and also x^3 and $x^{(1/3)}$). It is easy to see how adding two such plots produces a straight line (i.e., linear dependence).

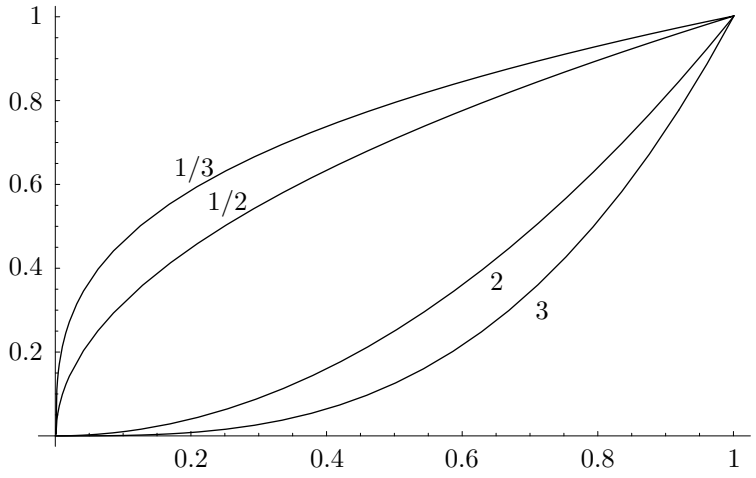


Figure H.8: Plots of x^γ and $x^{1/\gamma}$ For $\gamma = 2, 3$.

It should be stressed that this nonlinear behavior of the CRT is not caused by the phosphor but stems from the properties of the electron gun (the cathode). Since most CRTs are similar electrically, the value of gamma for a particular CRT depends mainly on its brightness and contrast settings. CRT users should therefore adjust their monitor's gamma with the intensity and contrast controls, using special test patterns. Figure H.9 is such a pattern where the goal is to find the bar where the brightnesses of the top and bottom parts match. This should be done from a distance of 6–10 feet. Some people find it more comfortable to look for the bar where the line in the middle disappears, or almost so. Once that bar is found, the gamma of the display is known and experts recommend to adjust it to between 2.2 and 2.5.

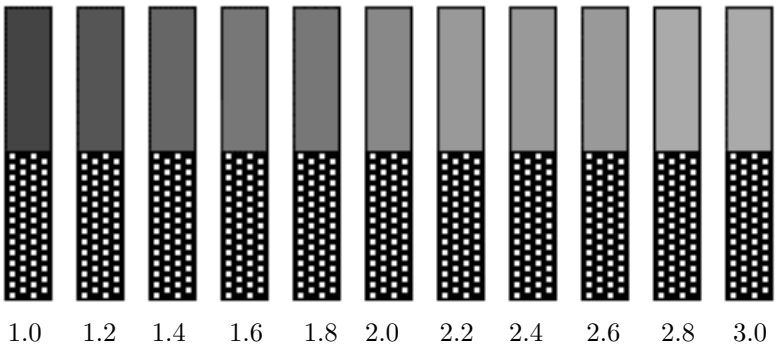


Figure H.9: A Gamma Test Pattern.

We turn now to the important (and confusing) concept of *gamma correction*. We have just stated that the lightness perceived by the eye is proportional to the

voltage fed to the CRT. Obviously we have to make sure that the voltage fed to the CRT is proportional to the light intensity seen by the video camera. The light sensor in the video camera (typically CCD) also has a gamma value and outputs a voltage V of the form $V = I^{\gamma}$ where I is the intensity "seen" by the sensor. This is why an extra circuit is built into the camera, that corrects for the nonlinear behavior of the sensor by changing $V \leftarrow V^{1/\gamma}$ and outputs a voltage V that is proportional to the intensity of the light falling on the sensor. Figure H.10 shows a block diagram of a video camera and a television receiver.

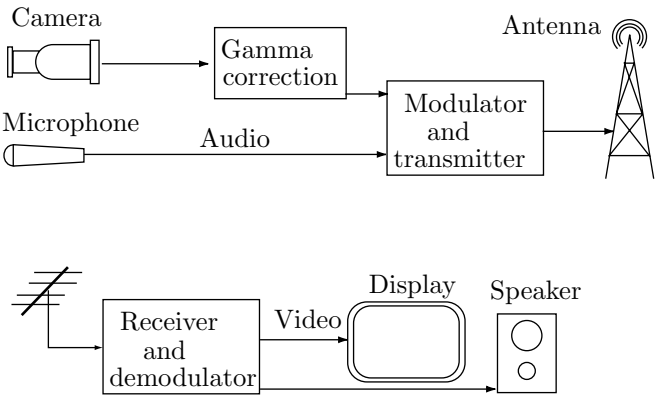


Figure H.10: Television Transmission With Gamma Correction.

In computer graphics, gamma correction is implemented in the color lookup table. To get an idea of how this is done, the reader should consider another feature of the human visual system. The eye-brain system can detect differences in light intensity, but this ability is not uniform from black to white. It is about 1% of the intensity itself. When looking at dark images, we can detect small variations in intensity. When seeing a bright image, we can only detect larger changes. With this in mind, imagine a color lookup table with 8-bit luminance values, where 0 represents black and 255 represents white. If two entries in the table have values of 25 and 26, then the difference between them is 1, but this one unit of difference is 4% of 25. Changing the value of a pixel from 25 to 26 would increase the brightness of the pixel by 4%, a large, noticeable intensity difference. Similarly, if the value of a pixel is changed from 100 to 101, then this one unit of change represents 1% of 100, and is, therefore, just barely noticeable to the eye. Changing a pixel from 200 to 201 increases the brightness of the pixel by 0.5% and would be unnoticeable. The value 201 is, in effect, wasted, and on the other hand we would like to have one or two values between 25 and 26. This suggests a gamma correction that assigns small intensity differences to small pixel values and large intensity differences to large pixel values. Ideally, adjacent values should correspond to luminance values that differ by about 1%, and one way of achieving this is to interpret pixel values as the logarithm of luminance.

Readers usually find gamma correction to be a confusing topic! Our aim, however, is to understand luminance and chrominance, and we start by stating

that the RGB components of a color are gamma corrected (in the video camera or in the lookup table), and the new, corrected values (which are also normalized in the range $[0,1]$) are denoted R', G', and B'. Each of these three components contributes differently to the brightness perceived by the eye (because of the three types of cones in the retina), so a new quantity Y', is defined as the weighted sum

$$Y' = 0.299R' + 0.587G' + 0.114B'. \tag{H.1}$$

Y' is called *luma*, but most authors call it luminance. Since the three corrected RGB components are normalized, Y' is also normalized in the range $[0,1]$. The two chrominance components U' and V' are defined as the differences $U' = B' - Y'$ and $V' = R' - Y'$. They can be interpreted as the presence or absence of blue and red in the color, respectively. Since R', G', B', and Y' all vary in the range $[0,1]$, it is easy to verify that U' varies in the range $[-0.886, +0.886]$ and V' varies in the range $[-0.701, +0.701]$, which suggests that they be normalized. Before we do that let's try to get a better understanding of the relations between the RGB and YUV color representations (the primes are omitted for convenience). Figure H.11 shows the RGB cube in dashed and the three planes $Y = 0.3$, $U = 0$, and $V = 0$. The latter two planes intersect at points that satisfy $R = G = B = Y$, so the intersection is the line that corresponds to all the gray colors.

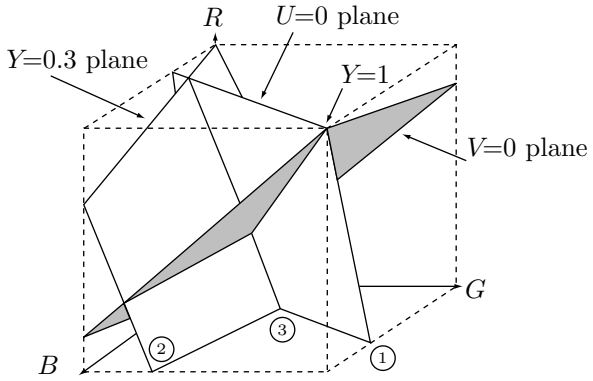


Figure H.11: Relationship Between the RGB and YUV Color Representations.

One point on the $U = 0$ plane is $(0, 0.755, .5)$. One point on the $V = 0$ plane is $(0.5, 0.5, 0.5)$. The reader should try to locate these points on the diagram.

◇ **Exercise H.4:** What are the RGB coordinates of the three points labeled ①, ②, and ③ in Figure H.11?

Next, we get to the chrominance components. Because of the ranges of variation of U' and V' they are easy to normalize. The two normalized quantities are denoted by Cb and Cr (for chrominance blue and chrominance red) and are defined by

$$Cb = (U'/2) + 0.5, \qquad Cr = (V'/1.6) + 0.5.$$

The $YCbCr$ color representation is commonly used in video and computer graphics, so image and video compression methods often assume that the images to be compressed are represented this way.

The three color representations RGB, YUV and $YCbCr$ are mathematically equivalent, since it is possible to transform each of them into the other ones. The advantage of $YCbCr$ is that the eye is more sensitive to small spatial luminance variations than to small color variations. A compression method that deals with an image in $YCbCr$ format should keep the Y component lossless (or close to lossless), but can afford to lose information in the chrominance components.

Transforming $R'G'B'$ into $Y'CbCr$ is done by

$$\begin{pmatrix} Y' \\ Cb \\ Cr \end{pmatrix} = \begin{pmatrix} 16 \\ 128 \\ 128 \end{pmatrix} + \begin{pmatrix} 65.481 & 128.553 & 24.966 \\ -37.797 & -74.203 & 112 \\ 112 & -93.786 & -18.214 \end{pmatrix} \begin{pmatrix} R' \\ G' \\ B' \end{pmatrix}.$$

The inverse transformation is

$$\begin{pmatrix} R' \\ G' \\ B' \end{pmatrix} = \begin{pmatrix} 0.00456621 & 0 & 0.00625893 \\ 0.00456621 & -0.00153632 & -0.00318811 \\ 0.00456621 & 0.00791061 & 0 \end{pmatrix} \left[\begin{pmatrix} Y' \\ Cb \\ Cr \end{pmatrix} - \begin{pmatrix} 16 \\ 128 \\ 128 \end{pmatrix} \right].$$

Note: The luminance (or luma) defined by Equation (H.1) is part of the ITU-R recommendation BT.601-4. Strictly speaking it should be denoted Y_{601} and should be called nonlinear video luma. There is also a quantity called linear luma, which is defined by means of the linear (uncorrected) RGB. It is defined by the ITU-R recommendation 709 as

$$Y_{709} = 0.2125R + 0.7154G + 0.0721B.$$

H.9 Spectral Density

A laser is capable of emitting "pure" light, i.e., just one wavelength. Most light sources, however, emit "dirty" light that is a mixture of many wavelengths, normally with one dominating. For each light source, the graph of light intensity as a function of the wavelength λ is called the *spectral density* of the light source. Figure H.12 shows the spectral densities of several typical light sources.

These simple diagrams illustrate one problem in attempting to specify color systematically and unambiguously. Several different spectral densities may be perceived by us as identical. When the colors created by these spectral densities are placed side by side, we find it impossible to distinguish between them. The first step in solving the problem is color matching. Suppose that we use a color model defined by the three primaries $A(\lambda)$, $B(\lambda)$, and $C(\lambda)$ and we have a color described by the spectral density $S(\lambda)$. How can we express $S(\lambda)$ in terms of our three primaries? One way to do this is to shine a spot of $S(\lambda)$ on a white screen and, right next to it, a spot of light $P(\lambda) = \alpha A(\lambda) + \beta B(\lambda) + \gamma C(\lambda)$ created by mixing the three primaries (where $0 \le \alpha, \beta, \gamma \le 1$). Now change the amounts of α, β, and γ until a

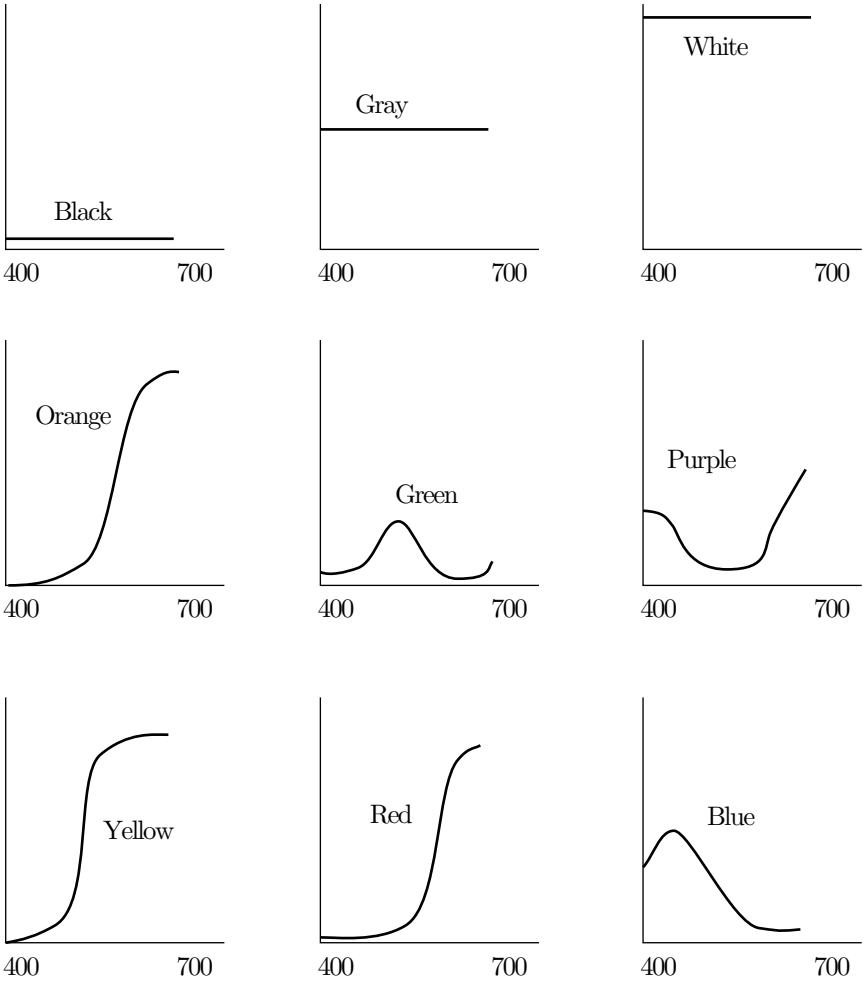


Figure H.12: Some Spectral Densities.

trained observer agrees that the spots are indistinguishable. We can now say that, in some sense, $S(\lambda)$ and $P(\lambda)$ are identical, and write $S = P$.

In what sense is the preceding true? It turns out that the above statement is meaningful because of a remarkable property of colors. Suppose that two spectral densities $S(\lambda)$ and $P(\lambda)$ have the same perceived color, so we write $S = P$. We now select another color Q and shine it on both spots S and P. We know from experience that the two new spots would also be indistinguishable. This means that we can use the symbol + for adding lights and we can describe the two spots by $S(\lambda)+Q(\lambda)$ and $P(\lambda)+Q(\lambda)$. In short, we can say "if $S = P$, then $S+Q = P+Q$." The same is true for changing intensities. If $S = P$, then $\alpha S = \alpha P$ for any intensity α. We, therefore, end up with a vector algebra for colors, where a color can be treated as a three-dimensional vector, with the usual vector operations.

Given the above, we can select a color model based on three primaries, A, B, and C, and can represent any color S as a linear combination of the primaries $S = \alpha A + \beta B + \gamma C$. We can say that the vector (α, β, γ) is the representation of S in the basis (A, B, C). Equivalently, we can say that S is represented as the point (α, β, γ) in the three-dimensional space defined by the vectors $A = (1, 0, 0)$, $B = (0, 1, 0)$, and $C = (0, 0, 1)$.

Since three-dimensional graphs are hard to draw on paper, we would like to artificially reduce the representation from three to two dimensions. This is done by realizing that the vector $(2\alpha, 2\beta, 2\gamma)$ represents the same color as (α, β, γ), only twice as bright. We, therefore, restrict ourselves to vectors (α, β, γ), where $\alpha + \beta + \gamma = 1$. These are vectors normalized to unit brightness. All vectors of unit brightness lie in the $\alpha + \beta + \gamma = 1$ plane and it is this plane (which, of course, is two-dimensional) that we plot on paper. We can specify a color by using two numbers, say, α and β and calculate γ as $1 - \alpha - \beta$.

For the RGB color model, we now select the pure spectral colors using trained human experts. The idea is to shine a spot of a pure color, say, 500 nm, on a screen and, right next to it, a spot that is a mixture $(r, g, b = 1 - r - g)$ of the three primaries of the RGB model. The values of r and g are varied until the observer judges the two spots to be indistinguishable. The point (r, g, b) is then plotted in the three-dimensional RGB color space. When all the pure colors have been plotted in this way, the points are connected to form a smooth curve, $\mathbf{P}(\lambda) = (r(\lambda), g(\lambda), b(\lambda))$. This is the *pure spectral color curve* of the RGB model (Figure H.13).

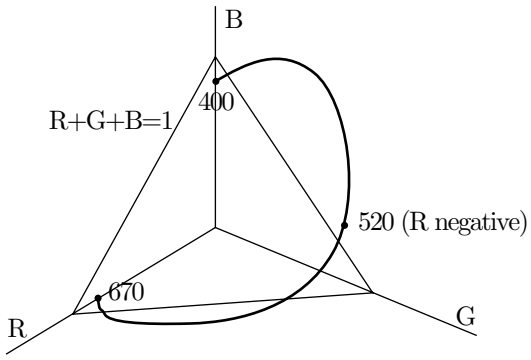


Figure H.13: Pure RGB Spectral Color Curve.

An important property of the curve is that some of r, g, and b may sometimes have to be negative. An example is $\lambda \approx 520\,\text{nm}$, where r turns out to be negative. What is the meaning of adding a negative quantity of green in a color defined by, for example, $S = 0.8R - 0.1G + 0.3B$? The way to understand this is to write the equation as $S + 0.1G = 0.8R + 0.3B$. It now means that color S cannot be constructed from the RGB primaries, but color $S + 0.1G$ can. The important conclusion is that not every color can be created in the RGB model! This is illustrated in Figure H.14. For some colors we can only create an approximation. This is true for all color models that can be created in practice.

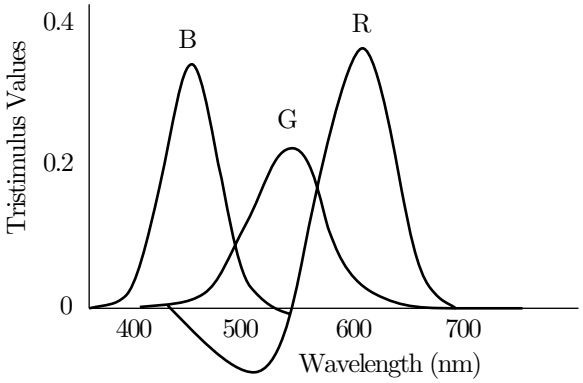


Figure H.14: RGB Color Combinations.

> Did I ever tell you my favorite color is blue?
> — Jürgen Prochnow (as Sutter Cane) in *In the Mouth of Madness*.

H.10 The CIE Standard

This standard was created in 1931 by the CIE. It is based on three carefully chosen artificial color primaries X, Y, and Z. They don't correspond to any real colors, but they have the important property that any real color can be represented as a linear combination $xX + yY + zZ$, where $x + y + z = 1$ and none of x, y, and z are negative (Figure H.15).

The plane $x + y + z = 1$ in the XYZ space is called the CIE chromaticity diagram (Figure H.16). The curve of pure spectral color in the CIE diagram covers all the pure colors, from 420 nm to 660 nm. It is shaped like a horseshoe.

Point $w = (0.310, 0.316)$ in the CIE diagram is special and is called "illuminant white." It is assumed to be the fully unsaturated white and is used in practice to match to colors that should be pure white.

◇ **Exercise H.5:** If illuminant white is pure white, why isn't it on the curve of pure spectral color in the CIE diagram?

The CIE diagram provides a standard for describing colors. There are instruments that, given a color sample, calculate the (x, y) coordinates of the color in the diagram. Also, given the CIE coordinates of a color, those instruments can generate a sample of the color. The diagram can also be used for useful color calculations. Here are some examples:

1. Given two points a and b in the diagram (Figure H.16), the line connecting them has the form $(1 - \alpha)a + \alpha b$ for $0 \leq \alpha \leq 1$. This line shows all the colors that can be created by adding varying amounts of colors a and b.

2. Imagine two points, such as c and d in Figure H.16. They are on the opposite sides of illuminant white and, therefore, correspond to complementary colors.

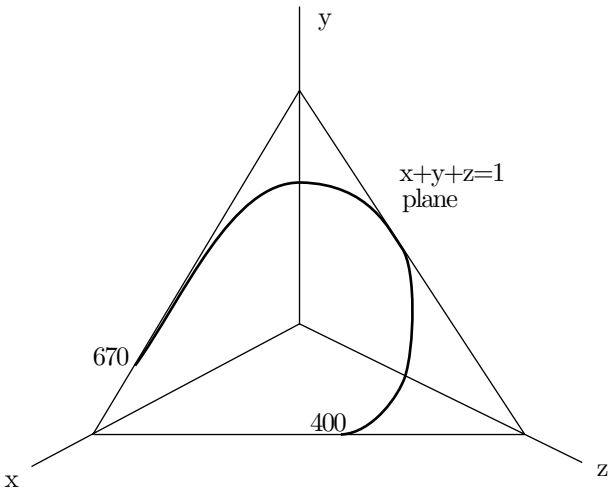


Figure H.15: Pure Spectral Color Curve.

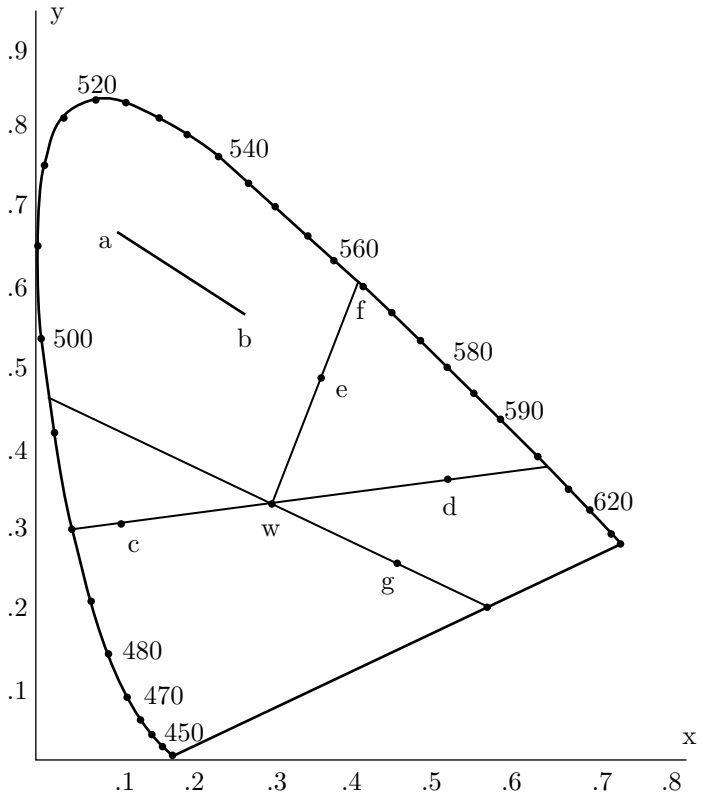


Figure H.16: The CIE Chromaticity Diagram.

◇ **Exercise H.6:** Why is that true?

3. The dominant wavelength of any color, such as e in Figure H.16 can be measured in the diagram. Just draw a straight line from illuminant white w to e and continue it until it intercepts the curve of pure spectral color. Then read the wavelength at the interception point f (564 nm in our example).

◇ **Exercise H.7:** How can the saturation of color e be calculated from the diagram?

◇ **Exercise H.8:** What is the dominant wavelength of point g in the CIE diagram?

> Have you lost your mind? What color is this bill?
> — Lori Petty (as Georgia "George" Sanders) in *Lush Life* (1996).

4. The *color gamut* of a device is the range of colors that can be displayed by the device. This can also be calculated with the CIE diagram. An RGB monitor, for example, can display combinations of red, green, and blue, but what colors are included in those combinations? To find the color gamut of an RGB monitor, we first have to find the locations of pure red, green, and blue in the diagram [these are points $(0.628, 0.330)$, $(0.285, 0.590)$, and $(0.1507, 0.060)$], then connect them with straight lines. The color gamut consists of all the colors within the resulting triangle. Each can be expressed as a linear combination of red, green, and blue, with non-negative coefficients (a convex combination).

Interestingly, because of the shape of the horseshoe, no three colors on or inside it can serve as ideal primaries. No matter what three points we select, some colors will be outside the triangle defined by them. This means that no set of three primaries can be used to create all the colors. The RGB set has the advantage that the triangle created by it is large, and thus contains many colors. The CMY triangle, for example, is much smaller by comparison. This is another reason for using red, green, and blue as the RGB primaries.

H.11 Halftoning

Color inkjet printers are common nowadays, but the inks are still expensive. Most laser printers are black and white. Halftoning is a method that makes it possible to print or display images with shades of gray on a black and white (i.e., bi-level) output device. The trade-off is loss of resolution. Instead of small, individual pixels, halftoning uses groups of pixels where only some of the pixels in a group are black. Halftoning is important, since it makes it possible to print pictures consisting of more than just black and white on a black and white printer. It is commonly used in newspapers and books. A classic reference is [Ulichney 87].

The human eye can resolve details as small as 1 minute of arc under normal light conditions. This is called the *visual acuity*. If we view a very small area from a normal viewing distance, our eyes cannot see the details in the area and end up integrating them, such that we only see an average intensity coming from the area. This property is called *spatial integration* and is very nicely demonstrated by Figure H.17. The figure consists of black circles and dots on a white background, but spatial integration creates the effect of a gray background.

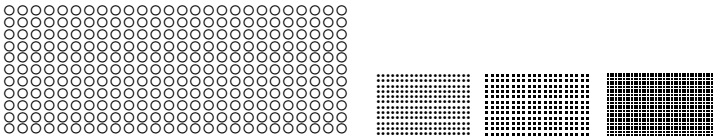


Figure H.17: Gray Backgrounds Obtained by Spatial Integration.

The principle of halftoning is to use groups of $n \times n$ pixels (with n usually in the range 2–4) and to set some of the pixels in a group to black. Depending on the black-to-white percentage of pixels in a group, the group appears to have a certain shade of gray. An $n \times n$ group of pixels contains n^2 pixels and can therefore provide $n^2 + 1$ levels of gray. The only practical problem is to find the best pattern for each of those levels. The $n^2 + 1$ pixel patterns selected must satisfy the following conditions:

1. Areas covered by copies of the same pattern should not show any textures.

2. Any pixel set to black for pattern k must also be black in all patterns of intensity levels $> k$. This is considered a good *growth sequence* and it minimizes the differences between patterns of successive intensities.

3. The patterns should grow from the center of the $n \times n$ area, to create the effect of a growing dot.

4. All the black pixels of a pattern must be adjacent to each other. This property is called *clustered dot halftoning* and is important if the output is intended for a printer (laser printers cannot always fully reproduce small isolated dots). If the output is intended for CRT only, then *dispersed dot halftoning* can be used, where the black pixels of a pattern are not adjacent.

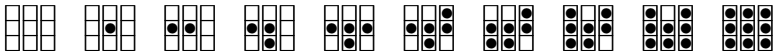
As a simple example of condition 1, a pattern such as

must be avoided, since large areas with level-3 groups would produce long horizontal lines. Other patterns may result in similar, annoying textures. With a 2×2 group, such effects may be impossible to avoid. The best that can be done is to use the patterns  .

A 3×3 group provides for more possibilities. The 10 patterns below ($= 3^2 + 1$) are the best ones possible (reflections and rotations of these patterns are considered identical) and usually avoid the problem above. They were produced by the matrix

$$\begin{bmatrix} 7 & 9 & 5 \\ 2 & 1 & 4 \\ 6 & 3 & 8 \end{bmatrix}$$

using the following rule: To create a group with intensity n, only cells with values $\leq n$ in the above matrix should be black.



The halftone method is not limited to a monochromatic display. Imagine a display with four levels of gray per pixel (2-bit pixels). Each pixel is either black or can be in one of three other levels. A 2×2 group consists of four pixels, each of which can be in one of three levels of gray or in black. The total number of levels is therefore $4\times3+1=13$. One possible set of the 13 patterns is shown below.

Bibliography

Ulichney, Robert (1987) *Digital Halftoning*, Cambridge, MA, MIT Press.

H.12 Dithering

The downside of halftoning is loss of resolution. It is also possible to display continuous-tone images (i.e., images with different shades of gray) on a bi-level device *without* losing resolution. Such methods are sometimes called *dithering* and their trade-off is loss of image detail. If the device resolution is high enough and if the image is watched from a suitable distance, then our eyes perceive an image in grayscale, but with fewer details than in the original.

The dithering problem can be phrased as follows: given an $m\times n$ array A of pixels in grayscale, calculate an array B of the same size with zeros and ones (corresponding to white and black pixels, respectively) such that for every pixel $B[i,j]$ the average value of the pixel and a group of its near neighbors will approximately equal the normalized value of $A[i,j]$. (Assume that pixel $A[i,j]$ has an integer value I in the interval $[0,a]$; then its normalized value is the fraction I/a. It is in the interval $[0,1]$.)

The simplest dithering method uses a threshold and the single test: Set $B[i,j]$ to white (0) if $A[i,j]$ is bright enough (i.e., less than the value of the threshold); otherwise, set $B[i,j]$ to black (1). This method is fast and simple, but generates very poor results, as the next example shows, so it is never used in practice. As an example, imagine a human head. The hair is generally darker than the face below it, so the simple threshold method may quantize the entire hair area to black and the entire face area to white, a very poor, unrecognizable, and unacceptable result. (It should be noted, however, that some images are instantly recognizable even in just black and white, as Figure H.18 aptly demonstrates.) This method can be improved a little by using a different, random threshold for each pixel, but even this produces low-quality results.

Four approaches to dithering, namely ordered dither, constrained average dithering, diffusion dither, and dot diffusion, are presented in this section. Another approach, called ARIES, is discussed in [Roetling 76] and [Roetling 77].

H.12.1 Ordered Dither

The principle of this method is to paint a pixel $B[i,j]$ black or leave it white, depending on the intensity of pixel $A[i,j]$ **and** on its position in the picture [i.e., on its coordinates (i,j)]. If $A[i,j]$ is a dark shade of gray, then $B[i,j]$ should ideally be dark thus, it is painted black most of the time, but sometimes it is left white. The decision whether to paint it black or white depends on its coordinates i and j. The opposite is true for a bright pixel. This method is described in [Jarvis et al. 76].

Figure H.18: A Familiar Black-and-White Image.

The giant panda resembles a bear, although anatomically it is more like a raccoon. It lives in the high bamboo forests of central China. Its body is mostly white, with black limbs, ears, and eye patches. Adults weigh 200 to 300 lb (90 to 140 kg). Low birth rates and human encroachment on its habitat have seriously endangered this species.

The method starts with an $m \times n$ dithering matrix D_{mn} which is used to determine the color (black = 1 or white = 0) of all the B pixels. In the example below we assume that the A pixels have 16 gray levels, with 0 as white and 15 as black. The dithering matrices for $n = 2$ and $n = 4$ are shown below. The idea in these matrices is to minimize the amount of texture in areas with a uniform gray level.

$$D_{22} = \begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix}, \qquad D_{44} = \begin{bmatrix} 0 & 8 & 2 & 10 \\ 12 & 4 & 14 & 6 \\ 3 & 11 & 1 & 9 \\ 15 & 7 & 13 & 5 \end{bmatrix}.$$

The rule is as follows: Given a pixel $A[x, y]$ calculate $i = x \bmod m$, $j = y \bmod n$, then select black (i.e., set $B[x, y]$ to 1) if $A[x, y] \geq D_{mn}[i, j]$ and select white otherwise.

To see how the dithering matrix is used, imagine a large, uniform area in the image A where all the pixels have a gray level of 4. Since 4 is the fifth of 16 levels, we would like to end up with 5/16 of the pixels in the area set to black (ideally they should be randomly distributed in this area). When a row of pixels is scanned in this area, y is incremented, but x does not change. Since i depends on x, and j depends on y, the pixels scanned are compared to one of the rows of matrix D_{44}. If this happens to be the first row, then we end up with the sequence 10101010 . . .

in bitmap B.

When the next line of pixels is scanned, x and, as a result, i have been incremented, so we look at the next row of D_{44}, that produces the pattern 01000100... in B. The final result is an area in B that looks like

$$\begin{array}{l} 10101010\ldots \\ 01000100\ldots \\ 10101010\ldots \\ 00000000\ldots \end{array}$$

Ten out of the 32 pixels are black, but $10/32 = 5/16$. The black pixels are not randomly distributed in the area, but their distribution does not create annoying patterns either.

◇ **Exercise H.9:** Assume that image A has three large uniform areas with gray levels 0, 1, and 15 and calculate the pixels that go into bitmap B for these areas.

Ordered dither is easy to understand if we visualize copies of the dither matrix laid next to each other on top of the bitmap. Figure H.19 shows a 6×12 bitmap with six copies of a 4×4 dither matrix laid on top of it. The threshold for dithering a pixel $A[i,j]$ is that element of the dither matrix that happens to lie on top of $A[i,j]$.

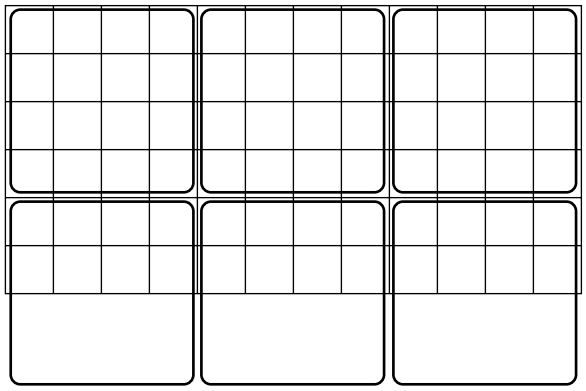


Figure H.19: Ordered Dither.

Matrix D_{44} above was created from D_{22} by the recursive rule

$$D_{nn} = \begin{pmatrix} 4D_{n/2,n/2} & 4D_{n/2,n/2} + 2U_{n/2,n/2} \\ 4D_{n/2,n/2} + 3U_{n/2,n/2} & 4D_{n/2,n/2} + U_{n/2,n/2}, \end{pmatrix}, \tag{H.2}$$

where U_{nn} is an $n\times n$ matrix with all ones. Other matrices are easy to generate with this rule.

◇ **Exercise H.10:** Use the rule of Equation (H.2) to construct D_{88}.

The basic rule of ordered dither can be generalized as follows: Given a pixel $A[x, y]$, calculate $i = x \bmod m$ and $j = y \bmod n$, then select black (i.e., assign $B[x, y] \leftarrow 1$) if $Ave[x, y] \geq D_{mn}[i, j]$, where $Ave[x, y]$ is the average of the 3×3 group of pixels centered on $A[x, y]$. This is computationally more intensive, but tends to produce better results in most cases, since it considers the average brightness of a group of pixels.

Ordered dither is a simple, fast method, but it tends to create images that have been described by various people as "computerized," "cold," or "artificial." The reason for that is probably the recursive nature of the dithering matrix.

H.12.2 Constrained Average Dithering

In cases where high speed is not important, this method [Jarvis and Roberts 76] gives good results, although it involves more computations than ordered dither. The idea is to compute, for each pixel $A[i, j]$, the average $\bar{A}[i, j]$ of the pixel and its eight near neighbors. The pixel is then compared to a threshold of the form

$$\gamma + \left(1 - \frac{2\gamma}{M}\right) \bar{A}[i, j],$$

where γ is a user-selected parameter and M is the maximum value of $A[i, j]$. Notice that the threshold can have values in the range $[\gamma, M - \gamma]$. The final step is to compare $A[i, j]$ to the threshold and set $B[i, j]$ to 1 if $A[i, j] \geq$ threshold and to 0 otherwise.

The main advantage of this method is edge enhancement. An example is Figure H.20 that shows a 6×8 bitmap A, where pixels have 4-bit values. Most pixels have a value of 0, but the bitmap also contains a thick slanted line indicated in the figure. It separates the 0 (white) pixels in the upper-left and bottom-right corners from the darker pixels in the middle.

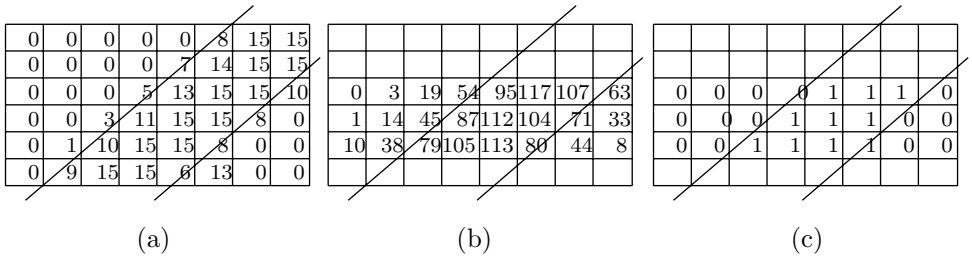


Figure H.20: Constrained Average Dithering.

Figure H.20a shows how the pixels around the line have values approaching the maximum (which is 15). Figure H.20b shows (for some pixels) the average of the pixel and its eight near neighbors (the averages are shown as integers, so an average of 54 really indicates 54/9). The result of comparing these averages to the

threshold (which in this example is 75) is shown in Figure H.20c. It is easy to see how the thick line is sharply defined in the bi-level image.

> So I'm a ditherer? Well, I'm jolly well going to dither, then!
> — Roland Young (as Cosmo Topper) in *Topper* (1937).

H.12.3 Diffusion Dither

Imagine a photograph, rich in color, being digitized by a scanner that can distinguish millions of colors. The result may be an image file where each pixel $A[i,j]$ is represented by, say, 24 bits. The pixel may have one of 2^{24} colors. Now, imagine that we want to display this file on a computer that can only display 256 colors simultaneously on the screen. A good example is a computer using a color lookup table whose size is 3×256 bytes.

We begin by loading a palette of 256 colors into the lookup table. Each pixel $A[i,j]$ of the original image will now have to be displayed on the screen as a pixel $B[i,j]$ in 1 of the 256 palette colors. The diagram below shows a pixel $A[i,j]$ with original color $(255,52,80)$. If we decide to assign pixel $B[i,j]$ the palette color $(207,62,86)$, then we are left with a difference of $A[i,j]-B[i,j]=(48,-10,-6)$. This difference is called the *color error* of the pixel.

$$\boxed{\begin{array}{l}\text{R=255}\\ \text{G=52}\\ \text{B=80}\end{array}} - \boxed{\begin{array}{l}\text{R=207}\\ \text{G=62}\\ \text{B=86}\end{array}} = \boxed{\begin{array}{l}\text{R=48}\\ \text{G=}-10\\ \text{B=}-6\end{array}}$$

Large color errors degrade the quality of the displayed image, so an algorithm is needed to minimize the total color error of the image. Diffusion dither does this by distributing the color errors among all the pixels such that the total color error for the entire image is zero (or very close to zero).

The algorithm is very simple. Pixels $A[i,j]$ are scanned line by line from top to bottom. In each line, they are scanned from left to right. For each pixel, the algorithm performs the following:

1. Pick up the palette color that is nearest the original pixel's color. This palette color is stored in the destination bitmap $B[i,j]$.
2. Calculate the color error $A[i,j]-B[i,j]$ for the pixel.
3. Distribute this error to four of $A[i,j]$'s nearest neighbors that haven't been scanned yet (the one on the right and the three ones centered below) according to the *Floyd-Steinberg filter* [Floyd and Steinberg 75] (where the X represents the current pixel):

	X	7/16
3/16	5/16	1/16

Consider the example of Figure H.21a. The current pixel is $(255,52,80)$ and we (arbitrarily) assume that the nearest palette color is $(207,62,86)$. The color error is $(48,-10,-6)$ and is distributed as shown in Figure H.21c. The five nearest

neighbors are assigned new colors as shown in Figure H.21b. The algorithm is shown in Figure H.22a where the weights $p1$, $p2$, $p3$, and $p4$ can be assigned either the Floyd-Steinberg values 7/16, 3/16, 5/16, and 1/16 or any other values.

The total color error may not be exactly zero because the method does not work well for the leftmost column and for the bottom row of pixels. However, the results can be quite good if the palette colors are carefully selected.

This method can easily be applied to the case of a monochromatic display (or any *bi-level* output device), as shown by the pseudo-code of Figure H.22b, where $p1$, $p2$, $p3$, and $p4$ are the four error diffusion parameters. They can be the ones already given (i.e., 7/16, 3/16, 5/16, and 1/16) or different ones, but their sum should be 1.

◇ **Exercise H.11:** Consider an all-gray image, where $A[i, j]$ is a real number in the range $[0, 1]$ and it equals 0.5 for all pixels. What image B would be generated by diffusion dither in this case?

◇ **Exercise H.12:** Imagine a grayscale image consisting of a single row of pixels where pixels have real values in the range $[0, 1]$. The value of each pixel p is compared to the threshold value of 0.5 and the error is propagated to the neighbor on the right. Show the result of dithering a row of pixels all with values 0.5.

Error diffusion can also be used for color printing. A typical low-end ink-jet color printer has four ink cartridges for cyan, magenta, yellow, and black ink. The printer places dots of ink on the page such that each dot has one of the four colors. If a certain area on the page should have color L, where L isn't any of CMYK, then L can be simulated by dithering. This is done by printing adjacent dots in the area with CMYK colors such that the eye (which integrates colors in a small area) will perceive color L. This can be done with error diffusion where the palette consists of the four colors cyan $(255, 0, 0)$, magenta $(0, 255, 0)$, yellow $(0, 0, 255)$, and black $(255, 255, 255)$ and the error for a pixel is the difference between the pixel color and the nearest palette color.

A slightly simpler version of error diffusion is the *minimized average error* method. The errors are not propagated but rather calculated and saved in a separate table. When a pixel $A[x, y]$ is examined, the error table is used to look up the errors $E[x + i, y + j]$ already computed for some previously seen neighbors $A[i, j]$ of the pixel. Pixel $B[x, y]$ is assigned a value of 0 or 1 depending on the corrected intensity:

$$A[x, y] + \frac{1}{\sum_{ij} \alpha_{ij}} \sum_{ij} \alpha_{ij} \mathrm{E}[x + i, y + j].$$

The new error, $E[x, y] = A[x, y] - B[x, y]$, is then added to the error table to be used for future pixels. The quantities α_{ij} are weights assigned to the near neighbors of $A[x, y]$. They can be assigned in many different ways, but they should assign more weight to nearby neighbors, so the following is a typical example:

$$\alpha = \begin{pmatrix} 1 & 3 & 5 & 3 & 1 \\ 3 & 5 & 7 & 5 & 3 \\ 5 & 7 & x & - & - \end{pmatrix},$$

Before	R=255 G=52 B=80	R=178 G=20 B=60
R=192 G=45 B=75	R=250 G=49 B=83	R=191 G=31 B=72

(a)

After	R=207 G=62 B=86	R=199 G=16 B=57
R=201 G=43 B=74	R=265 G=46 B=81	R=194 G=30 B=72

(b)

$\frac{7}{16} \times 48 = 21, \quad \frac{1}{16} \times 48 = 3, \quad \frac{5}{16} \times 48 = 15, \quad \frac{3}{16} \times 48 = 9,$
$\frac{7}{16} \times (-10) = -4, \quad \frac{1}{16} \times (-10) = -1, \quad \frac{5}{16} \times (-10) = -3, \quad \frac{3}{16} \times (-10) = -2,$
$\frac{7}{16} \times (-6) = -3, \quad \frac{1}{16} \times (-6) = 0, \quad \frac{5}{16} \times (-6) = -2, \quad \frac{3}{16} \times (-6) = -1.$

(c)

Figure H.21: Diffusion Dither.

```
for i := 1 to m do
 for j := 1 to n do
  begin
  B[i,j]:=SearchPalette(A[i,j]);

  err := A[i,j] - B[i,j];
  A[i,j+1] := A[i,j+1] + err * p1;
  A[i+1,j-1] := A[i+1,j-1] + err * p2;
  A[i+1,j] := A[i+1,j] + err * p3;
  A[i+1,j+1] := A[i+1,j+1] + err * p4;
  end.
```

(a)

```
for i := 1 to m do
 for j := 1 to n do
  begin
  if A[i,j] < 0.5 then B[i,j] := 0
    else B[i,j] := 1;
  err := A[i,j] - B[i,j];
  A[i,j+1] := A[i,j+1] + err * p1;
  A[i+1,j-1] := A[i+1,j-1] + err * p2;
  A[i+1,j] := A[i+1,j] + err * p3;
  A[i+1,j+1] := A[i+1,j+1] + err * p4;
  end.
```

(b)

Figure H.22: Diffusion Dither Algorithm. (a) For Color. (b) For Bi-level.

where x is the current pixel $A[x, y]$ and the weights are defined for some previously seen neighbors above and to the left of $A[x, y]$. If the weights add up to 1, then the corrected intensity above is simplified and becomes

$$A[x, y] + \sum_{ij} \alpha_{ij} E[x + i, y + j].$$

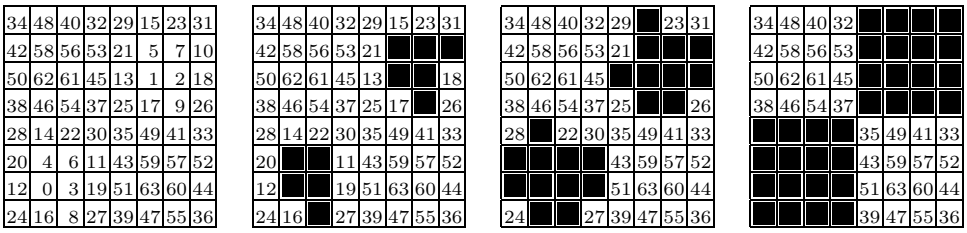
Floyd-Steinberg error diffusion generally produces better results than ordered dither, but has two drawbacks, namely it is serial in nature and it sometimes produces annoying "ghosts." Diffusion dither is serial, since the near neighbors of $B[i, j]$ cannot be calculated until the calculation of $B[i, j]$ is complete and the error $A[i, j] - B[i, j]$ has been distributed to the four near neighbors of $A[i, j]$. To understand why ghosts are created, imagine a dark area positioned above a bright area (for example, a dark sky above a bright sea). When the algorithm works on the last dark A pixels, a lot of error is distributed below, to the first bright A pixels (and also to the right). When the algorithm gets to the first bright pixels, they have collected so much error from above that they may no longer be bright, creating perhaps several rows of dark B pixels. It has been found experimentally that ghosts can be "exorcised" by scaling the A pixels before the algorithm starts. For example, each $A[i, j]$ pixel can be replaced by $0.1 + 0.8A[i, j]$, which "softens" the differences in brightness (the contrast) between the dark and bright pixels, thereby reducing the ghosts. This solution also changes all the pixel intensities, but the eye is less sensitive to absolute intensities than to changes in contrast, so changing intensities may be acceptable in many practical situations.

H.12.4 Dot Diffusion

This section is based on [Knuth 87], a very detailed article containing thorough analysis and actual images dithered using several different methods. The dot diffusion algorithm is somewhat similar to diffusion dither, it also produces good quality, sharp bi-level images, but it is not serial in nature and may be easier to implement on a parallel computer.

We start with the 8×8 *class matrix* of Figure H.23a. The way this matrix was constructed will be discussed later. For now, we simply consider it a permutation of the integers $(0, 1, \ldots, 63)$, which we call *classes*. The class number k of a pixel $A[i, j]$ is found at position (i, j) of the class matrix. The main algorithm is shown in Figure H.24.

The algorithm computes all the pixels of class 0 first, then those of class 1, and so on. Procedure `Distribute` is called for every class k and diffuses the error err to those near neighbors of $A[i, j]$ whose class numbers exceed k. The algorithm distinguishes between the four orthogonal neighbors and the four diagonal neighbors of $A[i, j]$. If a neighbor is $A[u, v]$, then the former type satisfies $(u-i)^2+(v-j)^2 = 1$, while the latter type is identified by $(u - i)^2 + (v - j)^2 = 2$. It is reasonable to distribute more of the error to the orthogonal neighbors than to the diagonal ones, so a possible weight function is $\text{weight}(x, y) = 3 - x^2 - y^2$. For an orthogonal neighbor, either $(u - i)$ or $(v - j)$ equals 1, so $\text{weight}(u - i, v - j) = 2$, while for a diagonal neighbor, both $(u - i)$ and $(v - j)$ equal 1, so $\text{weight}(u - i, v - j) = 1$. Procedure `Distribute` is listed in pseudo-code in Figure H.24.



(a) (b) (c) (d)

Figure H.23: 8×8 Matrices for Dot Diffusion.

```
for k := 0 to 63 do
 for all (i,j) of class k do
  begin
  if A[i,j] < .5 then B[i,j] := 0 else B[i,j] := 1;
  err := A[i,j] − B[i,j];
  Distribute(err,i,j,k);
  end.

procedure Distribute(err,i,j,k);
 w := 0;
 for all neighbors A[u,v] of A[i,j] do
  if class(u,v) > k then w := w+weight(u − i,v − j);
 if w > 0 then for all neighbors A[u,v] of A[i,j] do
  if class(u,v) > k then A[u,v] := A[u,v] + err×weight(u − i,v − j)/w;
end;
```

Figure H.24: The Dot Diffusion Algorithm.

Once the coordinates (i,j) of a pixel $A[i,j]$ are known, the class matrix gives the pixel's class number k that is independent of the color (or brightness) of the pixel. The class matrix also gives the classes of the eight near neighbors of $A[i,j]$, so those neighbors whose classes exceed k can be selected and linked in a list. It is a good idea to construct those lists once and for all, since this speeds up the algorithm considerably.

It remains to show how the class matrix, Figure H.23a, was constructed. The main consideration is the relative positioning of small and large classes. Imagine a large class surrounded, in the class matrix, by smaller classes. An example is class 63, which is surrounded by the "lower classes" 43, 59, 57, 51, 60, 39, 47, and 55. A little thinking shows that as the algorithm iterates toward 63, more and more error is absorbed into pixels that belong to this class, regardless of their brightness. A large class surrounded by lower classes is therefore undesirable and may be called a "baron." The class matrix of Figure H.23a has just two barons. Similarly, "near-baron" positions, which have only one higher-class neighbor, are undesirable and should be avoided. Our class matrix has just two of them.

◇ **Exercise H.13:** What are the barons and near-barons of our class matrix?

◇ **Exercise H.14:** Consider an all-gray image where $A[i, j] = 0.5$ for all pixels. What image B would be generated by dot diffusion in this case?

Another important consideration is the positions of consecutive classes in the class matrix. Figure H.23b,c,d show the class matrix after 10, 21, and 32 of its lowest classes have been blackened. It is easy to see how the black areas form 45° grids that grow and eventually form a 2×2 checkerboard. This helps create diagonal, rather than rectilinear dot patterns in the bi-level array B, and we know from experience that such patterns are less noticeable to the eye. Figure H.25a shows a class matrix with just one baron and one near-baron, but it is easy to see how the lower classes are mostly concentrated at the bottom-left corner of the matrix.

25	21	13	39	47	57	53	45
48	32	29	43	55	63	61	56
40	30	35	51	59	62	60	52
36	14	22	26	46	54	58	44
16	6	10	18	38	42	50	24
8	0	2	7	15	31	34	20
4	1	3	11	23	33	28	12
17	9	5	19	27	49	41	37

(a)

14	13	1	2
4	6	11	9
0	3	15	12
10	8	5	7

(b)

Figure H.25: Two Class Matrices for Dot Diffusion.

A close examination of the class matrix shows that the class numbers in positions (i, j) and $(i, j + 4)$ always add up to 63. This means that the grid pattern of $63 - k$ white pixels after k steps is identical to the grid pattern of $63 - k$ black pixels after $63 - k$ steps, shifted right four positions. This relation between the dot pattern and the diffusion pattern is the reason for the name *dot diffusion*.

◇ **Exercise H.15:** Figure H.25b shows a 4×4 class matrix. Identify the barons, near-barons, and grid patterns.

Experiments with the four methods described in this section seem to indicate that the dot diffusion method produces best results for printing because it tends to generate contiguous areas of black pixels, rather than "checkerboard" areas of alternating black and white. Modern laser and ink-jet printers have resolutions of 600 dpi or more, but they generally cannot produce a high-quality checkerboard of 300 black and 300 white alternating pixels per inch.

Bibliography

Floyd, R., and L. Steinberg (1975) "An Adaptive Algorithm for Spatial Gray Scale," in *Society for Information Display 1975 Symposium Digest of Technical Papers*, p. 36.

Jarvis, J. F., C. N. Judice, and W. H. Ninke (1976) "A Survey of Techniques for the Image Display of Continuous Tone Pictures on Bilevel Displays" *Computer Graphics and Image Processing* **5**(1):13–40.

Jarvis, J. F. and C. S. Roberts (1976) "A New Technique for Displaying Continuous Tone Images on a Bilevel Display" *IEEE Transactions on Communications* **24**(8):891–898, August.

Knuth, Donald E., (1987) "Digital Halftones by Dot Diffusion," *ACM Transactions on Graphics* **6**(4):245–273.

Roetling, P. G. (1976) "Halftone Method with Edge Enhancement and Moiré Suppression," *Journal of the Optical Society of America*, **66**:985–989.

Roetling, P. G. (1977) "Binary Approximation of Continuous Tone Images," *Photography Science and Engineering*, **21**:60–65.

> While wondering what he should do in this emergency he came upon a girl sitting by the roadside. She wore a costume that struck the boy as being remarkably brilliant: her silken waist being of emerald green and her skirt of four distinct colors – blue in front, yellow at the left side, red at the back and purple at the right side. Fastening the waist in front were four buttons—the top one blue, the next yellow, a third red and the last purple.
>
> L. Frank Baum, *The Marvelous Land of Oz*