

GLOBAL
EDITION



Fluency With Information Technology

Skills, Concepts, & Capabilities

SIXTH EDITION

Lawrence Snyder

ALWAYS LEARNING

PEARSON



Global Edition

LAWRENCE SNYDER

UNIVERSITY OF WASHINGTON

Global Edition contributions by

CHETHAN VENKATESH

M S RAMAIAH INSTITUTE OF TECHNOLOGY

Pearson

Boston Columbus Indianapolis New York San Francisco Upper Saddle River
Amsterdam Cape Town Dubai London Madrid Milan Munich Paris Montreal Toronto
Delhi Mexico City Sao Paulo Sydney Hong Kong Seoul Singapore Taipei Tokyo

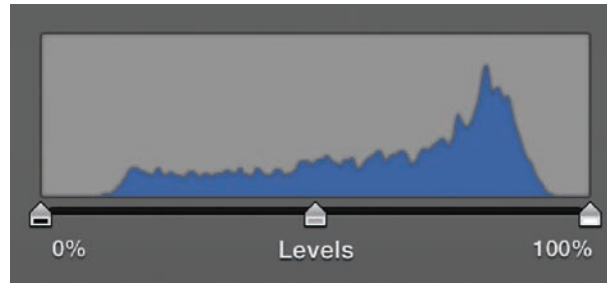


Figure 8.6 The Levels graph for the GGGM photo; the horizontal axis is the 256 pixel values, and the vertical axis is the number of pixels in the image with that value.

Increasing Brightness and Contrast

Photo manipulation software usually gives the ability to increase the brightness and contrast of photos, and this one looks like it could use those enhancements. *Brightness* refers to how close to white the pixels are; *contrast* is the size of the difference between the darkest and lightest portions of the image. What? Isn't black 0000 0000 0000 0000 0000, the darkest, and white, 1111 1111 1111 1111 1111 1111, the lightest? It's true that they are the most extreme encodings, but an image may not have any true black or true white pixels in it. In fact, photo manipulation software often gives the values of the pixels in a clever Levels graph (see Figure 8.6).

In the pixel distribution of the GGGM photo, 0 percent is called the *black point*, or all zeroes in an unmanipulated picture, and 100 percent is the *white point*, or all ones. The midpoint is called the *gamma point*—let's not think about why—and it is the midpoint in the pixel range. From the graph we see that this photo's pixels are clumped in the middle gray range, with no extremes of white or black; it doesn't have much contrast and it's not very bright.

Let's consider brightening it first. What does that mean? We want all the pixels to be nearer intense white, but to keep their relative relationships. So, if we add, say 16, to each pixel, then a pixel in her cheek, which is 197, 197, 197 ■, would become 213, 213, 213 ■. It doesn't look like a huge change, but when applied everywhere, the image brightens noticeably.



Binary Addition

Okay, back to binary. We can apply (and review) the binary given in the last section. Let's implement the addition of 16 to 197. First of all, what is 197 in binary? Our table tells us

Number being converted		197	69	5	5	5	5	1	1
Place value		128	64	32	16	8	4	2	1
<i>Subtract</i>		69	5				1		0
Binary Number		1	1	0	0	0	1	0	1

that it is 1100 0101, ■; of course, 16 is a power of 2, so it is simply 1 0000. Writing the addition in the usual form

1100 0101	binary representing decimal number	197
+ 1 0000	binary representing decimal number	16
1101 0101	binary representing decimal number	213

the addition works as always. We can check the result using our binary to decimal algorithm to see that the result is 213. Changing all three bytes makes that pixel lighter, . To increase brightness for the rest of the picture, we add 16 to all pixels. A pixel in her wrist has a gray value of 157 , which is 1001 1101 in binary. It presents a carry situation:

1	carry digit	
1001 1101	binary representing decimal number	157
+ 1 0000	binary representing decimal number	16
1010 1101	binary representing decimal number	173

↑

The resulting gray value is 173, .

After adding 16 to all pixels, we have the brighter image shown in Figure 8.7(b). In terms of the Levels graph in Figure 8.6, we have shifted the entire graph 16 positions to the right, as shown in Figure 8.8(b).



Figure 8.7 Three versions of the GGGM photo: (a) original, (b) brightened by “adding 16,” and (c) contrast improved by increasing highlights more than shadows.

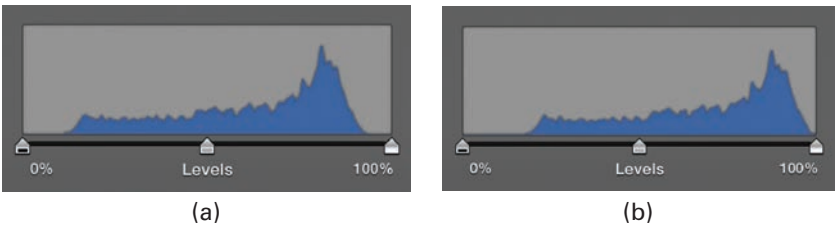


Figure 8.8 Levels graph for GGGM photo: (a) original and (b) after brightening by adding 16. Notice that the graph is simply shifted right by 16.

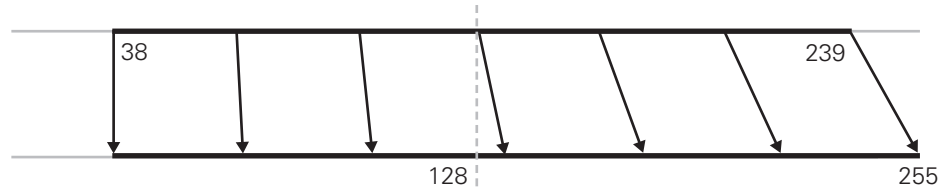


Figure 8.9 “Stretching” the pixel values right from the range 38–239 to 38–255.

Contrast

The photo got lighter when we added brightness, but it hasn’t improved enough. We don’t need the whole thing to be lighter as much as we need to increase the difference between the light parts and the dark parts, that is, increase the contrast. We can do that by using similar ideas.

Our goal is not to shift the Levels diagram right, but rather to “stretch it out” toward the right. That is, we want to add an amount to each pixel as we did before, but this time we need to add a smaller amount for dark pixels and a larger amount for light pixels. By leaving the dark pixels pretty much the same, and increasing the light pixels, we increase the contrast between the dark and light areas of the image (see Figure 8.7(c)).

To understand adding different amounts to each pixel, see the diagram in Figure 8.9. The (solid) upper line shows the pixel values as they are in the original Levels graph, as shown in Figure 8.6. The smallest pixel value is 38 and the largest is 239. The arrows show how we want to increase the amount we lighten each pixel gradually so that they span the range 38 to 255.

It’s easiest to think about the operation as converting the original value to a new value, and to use Figure 8.9 as a guide. For every original pixel P_o , we subtract the amount of the lower end of the range; in our case that’s 38,

$$P_o - 38$$

giving us its position along the dark part of the top line. That tells how much to increase each pixel position; smaller (darker) numbers get lightened less than larger (lighter) numbers. Then we multiply by the size of the new interval divided by the size of the old interval. This is how much to stretch the old range (38–239) so that it just matches the new range (38–255):

$$\frac{(255 - 38)}{(239 - 38)} = \frac{217}{210} = 1.08$$

Finally, we add the low end of the original range back in again, to return each pixel to its new position along the second line. This gives us the equation for the value in each pixel position of the new image:

$$P_n = (P_o - 38) * 1.08 + 38$$

rounding to a whole number. Checking, we verify that pixel value 38 isn’t changed at all, and pixel value 239 is changed to 255. Additionally, pixel 197 is lightened by 13 because

$$\begin{aligned} P_n &= (197 - 38) * 1.08 + 38 \\ &= 159 * 1.08 + 38 \\ &= 171.72 + 38 \\ &= 209.72 \\ &= 210 \end{aligned}$$

and pixel 157 is lightened by 10 because

$$\begin{aligned}
 P_n &= (157 - 38) * 1.08 + 38 \\
 &= 119 * 1.08 + 38 \\
 &= 128.52 + 38 \\
 &= 166.52 \\
 &= 167
 \end{aligned}$$

The result, shown in Figure 8.7(c), is quite nice. Your great-great-grandmother's hair is dark, and the detail in her sleeve is accentuated. It's possible to increase by more or less than 1.08, of course. (Image processing software usually gives us a slider.) Here, we chose 1.12 and made sure that any value that exceeded 255 was set to 255.

Of course, the computer is performing all of these multiplications in binary, which works like decimal multiplication, as you probably guessed. We will leave the details of binary multiplication to computer engineers.

Adding Color

So far, we have not used the RGB characteristics of our image encoding. But now we will add color by changing the subpixels by different amounts. Whenever the 3 bytes differ in value, we can perceive color. We will continue our idea of changing the pixels by different amounts on different parts of the photograph. As is standard in image manipulation, recognize three parts of the photograph:

Pixel Type	R Change	G Change	B Change
Highlights	+8	0	-4
Midrange	+9	+6	-4
Shadows	+15	0	-6



The amount shown is the amount by which I propose to change the subpixels. These settings produce an effect similar to “antique” in image processing software, a slightly pink-brown tint known as sepia.

But, how do we define the three different regions of the image? We can define “highlights” as the lightest 25 percent of the pixels and “shadows” as the darkest 25 percent of the pixels. We do this by counting the pixels.

A useful fact is that there are $600 \times 800 = 480,000$ pixels total in this photo. If we start at pixel value 38 and find the number of 38 pixels and add that number to the number of 39's and the number of 40's, . . . until we get a total of about 120,000, we know that that's the boundary between the shadows and the midrange portions of the image. In the GGGM photo, the largest shadow subpixel value is 134. Starting at the other end, at 255, and doing the opposite gets us the boundary between the highlights and the midrange. The largest midrange pixel is 233 (see Figure 8.10).

To summarize, the highlights, midrange, and shadows are defined as follows:

Highlights 25%	255–234	121,339
Midrange 50%	233–135	239,540
Shadows 25%	134–38	119,121

The approximate nature of these ranges will not affect the quality of the image.

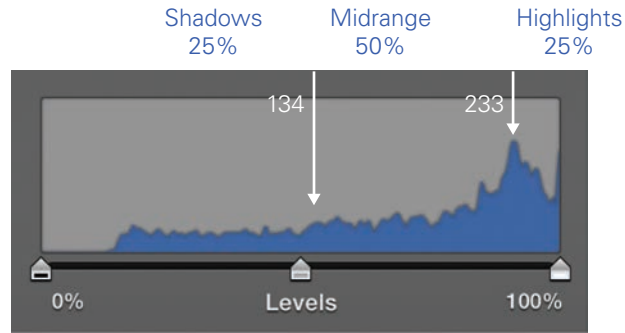


Figure 8.10 The Levels graph for the image in Figure 8.7(c), and boundaries between the highlights, midrange, and shadows.

The algorithm to colorize the image (see Figure 8.11) is simple. For each pixel, get the red subpixel (any one will work) and check its range. Then, using the color modifications given above for that portion of the image, adjust the color of each subpixel.



8.8 Getting the Gray Out. Using the coloring algorithm, give the new value for a pixel with a gray value of 1010 1100.

Summary of Digital Color

Color is represented by three quantities: red intensity, green intensity, and blue intensity, or RGB. Together, these three values form a pixel; the constituent parts are called subpixels. Standard computer equipment assigns 1 byte to each intensity, giving a range from 0 to 255. The intensities are given as binary numbers, which allows software to “compute on the image.”

Using a scanned black-and-white photo, we manipulated it in several ways. We brightened it by adding 16 to each subpixel. This preserved the gray color—all three subpixels have the same value—and pushed each pixel closer to white. The picture was brighter, but not

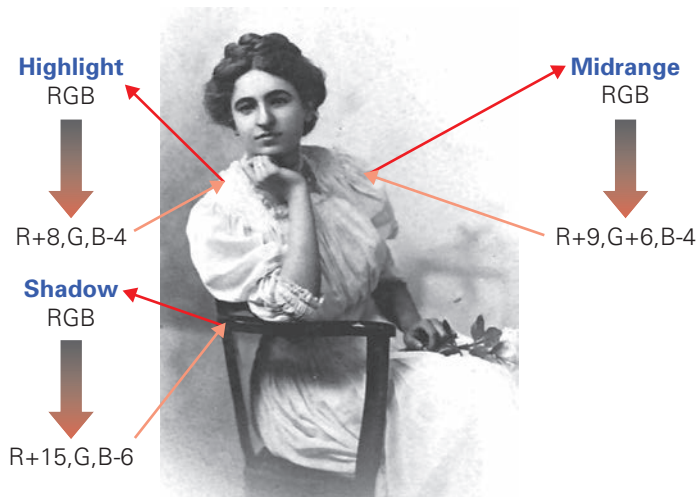


Figure 8.11 The gray is changed to sepia by checking each pixel, deciding if it's a highlight, midrange or shadow, and replacing the pixel with a corresponding revised RGB assignment.

much better. So, we increased the contrast by redefining each pixel so they would be brightened by a different amount depending on how much larger it was than the darkest subpixel value of 38. The darks changed little, but the highlights were lightened as before. Finally, we “split the RGB values” so that they were no longer the same, causing the image to be colored. We applied an algorithm to adjust the color differently over different parts of the image. Commercial image processing software is more sophisticated when it performs these transformations, generally to produce a more artistic effect, but the process is exactly the same.

And as we transformed the image, we learned the basics of binary arithmetic, which—it turned out—we already knew, because we know decimal arithmetic. It’s the same, just limited to 0 and 1.

Digitizing Sound

In this section we discuss digitizing again, but this time we focus on digitizing sound rather than images because it is equally interesting and slightly easier. The principles are the same when digitizing any “continuous” information.

An object—think of a cymbal—creates sound by vibrating in a medium such as air. The vibrations push the air, causing pressure waves to emanate from the object, which in turn vibrate our eardrums. The vibrations are transmitted by three tiny bones to the fine hairs of our cochlea, stimulating nerves that allow us to sense the waves and “hear” them as sound. The force, or intensity of the push, determines the volume, and the **frequency** (the number of waves per second) of the pushes is the pitch. Figure 8.12 shows a graph of a pure tone sound wave. The horizontal axis shows time and the vertical axis shows the amount of positive or negative sound pressure.

From a digitization point of view, the key is that the object vibrates continuously, producing a continuously changing wave, which is called **analog** information. As the wave moves past, say, a microphone, the measured pressure changes smoothly. When this pressure variation is recorded directly, as it was originally by Thomas Edison with a scratch on a wax cylinder, and then later with vinyl records, we have a continuous (analog) representation of the wave. In principle, all of the continuous variation of the wave is recorded. Digital representations work differently.

Analog to Digital

To digitize continuous information, we must convert the data to bits. For a sound wave, we use a binary number to record the amount that the wave is above or below the 0 line at a given point on our graph; that is, the amount of positive or negative sound pressure.

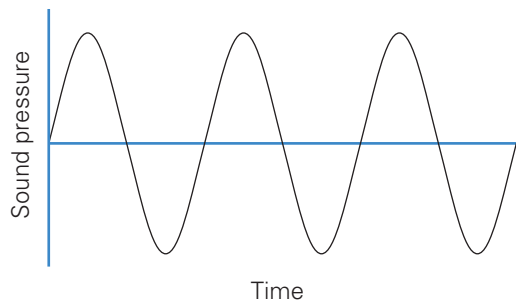


Figure 8.12 Sound wave. The horizontal axis is time; the vertical axis is sound pressure.