Gamma Correction

Technical Memo 9

Alvy Ray Smith
September 1, 1995

Abstract

Gamma is one of the most misunderstood concepts in computer imaging applications. Part of the reason is that the term is used to mean several different things, completely confusing the user. The purpose of this memo is to draw distinctions between the several uses and to propose a fixed meaning for the term: **Gamma is the term used to describe the nonlinearity of a display monitor**. This is the historical first meaning of the term. A corollary of this definition is that all other uses of the term gamma should be dropped, or the imaging industry will continue to stumble. In particular, this means that gamma should not be used to describe nonlinearity introduced into the image data itself. Image data, especially images intended for reuse, should always contain linear data, as commonly assumed by all computer graphics algorithms.

Historical Gamma Definition

All computer display monitors (and all TV sets) are nonlinear. This means that if the voltage on their electronics is doubled, their brightness does *not* double as you might expect it to. In fact, it varies approximately as the square of the voltage. If it varied by the square exactly, we would say it had a *gamma* of 2. In other (mathematical) words, gamma is an exponent. But all monitors are slightly different, so the actual gamma of a monitor might be anywhere from 1.4 to 2.6, instead of 2. Very typical gamma numbers are 1.8 for the PC and Mac worlds and 2.2 for the broadcast TV world (and for PCs using TV graphics boards, such as Truevision Targa+), but these should not be taken as gospel. They vary from display to display, even on displays from the same manufacturer. Any gamma other than 1 is *nonlinear*.

The term gamma comes from the electronics industry. The equation derived by electronics engineers to describe the intensity *I*, or brightness, out of a cathode-ray tube (CRT) for a given voltage *V* applied to it its inputs is

$$I = kV^{\gamma}$$

where the exponent of *V* is the Greek letter gamma (and *k* is some proportionality constant)[1]. Most computer displays are still based on CRTs. So the historical

[1] Equivalently, the intensity is related by the same exponent γ to the number of electrons in the CRT that strike the phosphors on the face of the tube and cause them to emit light. This is because the number *N* of electrons is directly proportional to the voltage *V*.

meaning of gamma is that it is a number indicating the degree of nonlinearity of a given CRT.

The proposal here is that this—gamma is a measure of the nonlinearity of a display device—be the *only* meaning that gamma have in imaging. Another use of this term is discussed in the next section, then the difficulties resulting from multiple, inconsistent uses of the term are discussed after that.

The important point to take from this section is that gamma is a property *local* to each individual display device. Unfortunately, it is not a property that can be obtained by software query of the display, at least with today's hardware. The best that software can do, without input from the user, is **guess** a particular display's gamma.

Newer, Corrupted Gamma Definition

Many imaging applications today use the term gamma to mean a nonlinearity introduced into the image data itself, independent of display of that data. This is typically presented to the user by a curve that defines the nonlinearity to be introduced—a "compensation" curve, or "tonal response" curve, or "correction" curve, etc. A common implementation lets the user fix the two endpoints of the curve with one control per endpoint, then provides a third, central control called "gamma" that controls the curvature of the central part of the curve. The result of applying this curve to an image is that the original pixels in it are replaced with new pixels containing the nonlinearity described by the curve. The details of the implementation need not concern us here. The important point is that the term gamma in this context is used to mean a nonlinearity introduced into the pixel data itself. Notice that this means the nonlinearity is *global* to all display devices (on which that image might be displayed). Equivalently, the nonlinearity of a display device is compounded by the nonlinearity introduced into the data itself when it is presented to the user.

It is clear why this corrupted definition arose. In both cases, gamma is a single number or exponent defining the curvature (nonlinearity) of a curve. In both cases, the apparent brightness of the display image is altered by changing this exponent. So what is the problem?

The Assumption of Linearity

The problem is that all computer graphics computations **assume** linear images. This means simply that, for instance, half red plus half red gives full red. This is fundamental to the industry. For example, the composition of images uses linear interpolation of images as the algorithm, which assumes the images being composited together are linear. Another example is antialiasing of edges to rid them of the jaggies. The classic computer graphics solution assumes linear data. All nonlinearity must be removed before entering the computer graphics world and then reintroduced, if necessary, when exiting it.

There are two ways to take care of this mismatch between the reality of nonlinear displays and the assumption of linear computations:

1. Take care of the nonlinearity in the display, or
2. Take care of it in the data.

Only choice 1 preserves image data for later use—for *reuse*. This is why gamma in the display context is proposed as the only use of the term. As you can probably foresee, I am going to argue an even stronger point: Nonlinearity should **never** be stored in an image. Or, if it is, then this nonlinearity must be noted in the storage format in such a way that it is known how to remove it to retrieve linear data.

Consequences

Unfortunately, many applications force you to take care of display nonlinearity by making image data nonlinear. They do this by assuming the default monitor gamma is 1, or linear (which, of course, it is not). The user then introduces the nonlinearity necessary to make the image “look right” on the monitor by varying the application’s brightness compensation control, often unfortunately called “gamma”. This replaces the images of the pixel—which hopefully are linear by default—with nonlinear pixels. This works, so long as you never use the resulting image for another image computation, and so long as the next display you show the image on (including ink on paper) has the **same** nonlinearity as your original display. This has "worked" often enough or closely enough in the past for the mistake to have been tolerated, but it is only an accident when it works.

To state the problem another way: The mistake works so long as monitors of identical gamma are used for any single image and so long as nothing further is done with the image, computationally—ie, it is only displayed thereafter on the monitors of the given gamma. You can see why the problem is rampant: This set of conditions is often met in the case of a single user generating a single image for no future reuse (computationally).

But the whole idea of an image composition program (Altamira Composer, for example) is to take images from many sources and composite them together to form new images (which may then be used by someone else as a component in yet another image, and so on). Images or sprites are reused; reuse is a fundamental of modern multimedia content creation. Such an application **must** assume linear data in its images for these computations, and hence nonlinearity must be handled by the display.

Consider this: Kodak PhotoCD images have a gamma of 2.2 embedded in the image data[2]. This assumes that all monitors they are to be displayed on have a gamma of 2.2. But we know that many popular monitors for PCs have gammas of 1.8. And for applications, like Altamira Composer, that have to assume linear image data, the PhotoCD images are nonlinear and must be de-compensated before they can be used in the application. Unfortunately, Kodak does not supply a

[2] To be careful with my terminology, I should say here that Kodak has embedded a nonlinearity in PhotoCD data equivalent to displaying linear data on a monitor with gamma 2.2.

programmer's interface (API) that allows the linear data to be extracted from its format. So computer graphics applications have to sacrifice 1-2 bits per channel per pixel to get linear data. Furthermore, an applications programmer must **know** that Kodak has assumed 2.2. This fact is not stored in the image file. I learned it by reading the documentation that showed the encoding formulas used by Kodak. This documentation shows exponents of value .45 in several places. I happen to know that 1/.45 is about 2.2 so could easily recognize the embedded gamma correction for what it was. I advocate, of course, that Kodak not correct (delinearize) its image data, but since it is too late presumably for them to change their format, then they should provide an API that allows developers to decorrect (linearize) their data with as much accuracy as possible.

Consider this: A user exports an image from Photoshop on a Mac (gamma 1.8 nominally) to Photoshop on a PC with a video display (gamma 2.2 nominally). In general the PC image should look too bright, so the user will either debrighten the display, putting it into who knows what state, or will recompensate it in the imaging app, thus introducing a new nonlinearity into the data and possibly throwing away some bits. This happens very often. This assumes Photoshop is used in its default way: Monitor gamma is assumed to be 1 and nonlinearity is wired into the pixel data[3]. I am not just picking on Photoshop. Microsoft Imager does the same thing. It has, in fact, a slider labeled γ on its main display that affects image data, not the display.

The Altamira Composer solution to the gamma problem assumes the default monitor gamma is 1.8 for PCs. This can be changed to a particular display's gamma in the View Options menu. The application corrects for the nonlinearity of the display during the display process, not the computation process. The data in images inside Altamira Composer can always be assumed to be linear, and linear data is always stored to files.

If images are imported to Altamira Composer from applications that have forced the display nonlinearity into the data, a user will—not surprisingly—get unexpected results. For this reason, Altamira Composer provides for the removal of the unwanted nonlinearity with a so-called "degamma" process. Of course, there is no way for Altamira Composer to know exactly what nonlinearity has been stored in the imported data. (That is the problem with storing the nonlinearity in the data.) It just guesses by using the monitor gamma setting in View Options to degamma the data. So for nonlinear images imported from a broadcast video app, a user might want to set this option to 2.2.

Another common source of nonlinear image data are applications supporting the Targa (*.tga*) image format and Windows (*.bmp, .dib*) image format[4]. Altamira Composer provides an option called TGA, BMP Degamma Correct in File Op-

[3] There are ways around this in most imaging applications, but unfortunately they are not the default ways so ordinary users never discover them.

[4] The Windows image (bitmap) format is apparently derived from the Targa format, so it is not surprising that they share this problem.

tions to automatically take out the nonlinearity in these images. It defaults to No (since Altamira Composer itself stores linear data in these two formats). If images are imported from applications storing nonlinear data in these formats, this option is set to Yes. Caution: Targa apps are usually for broadcast TV with typical gamma of 2.2 (or even higher). Windows apps usually assume 1.8.

Gamma can be confusing, as the above probably illustrates. Here are the simple rules Altamira Composer uses and what I am advocating that imaging applications do as a matter of course: Images are always assumed to be linear. Gamma is applied only to the display of images and not to the data of the images. The display[5] is assumed to be nonlinear (because it is). Applications separate computation from display cleanly, and gamma correct for the local display only in the display process.

To get compatible results between imaging applications written under the (I trust you believe sensible) "new" guidelines offered here and those written the "old" way: Set the monitor gamma assumption in all the "old" imaging applications to the same (greater than 1) value—presumably to that matching one's usual display monitor. Most applications provide a way to do this. This transfers the nonlinearity correction in those apps from the computation process to the display process, as it should be, leaving linear data in the images themselves.

A desirable consequence of all this is that it would be very convenient for imaging software if display devices provided gamma correction tables settable by software. That way, each imaging app could work completely in linear space, knowing that the display step would be correctly compensated by the local monitor for its local nonlinearities. Believe it or not, this was the way it was done 20 years ago, but the idea got lost along the way, leading to the mess described in this memo. Unfortunately, it is probably too late to change. The technique offered here is the best that can be done short of changing all the hardware.

[5] As is probably clear, I include as "display" ink on paper, video monitors, flat-panel displays, and photographs. They are all nonlinear in different ways.