

## Chapter 17 Calibration & more: The final stages in setting up a visual stimulus

Here are the main things to think about before assuming that the visual stimulus you programmed is going to appear on the screen as you wish it to do.

These issues include calibration (measuring the relationship between gun values and the light emitted by the display) and various other issues.

The following table is a summary of the issues we are going to address in this chapter.

	<b>time</b>	<b>space</b>	<b>luminance</b>	<b>color</b>
<b>time</b>	Missing frames  Phosphor persistence	Protect your screen resolution from lab assistants  Record your display settings and your viewing distance	Warming up  Slow temporal changes  Quantization errors	Warming up  Slow temporal changes
<b>space</b>		Pixels not square	Stationarity  Power drain	Stationarity  Power drain
<b>luminance</b>			Variation between monitors  Gamma correction	Chromatic variance for low/high gun values
<b>color</b>				Variation between monitors  Gun drain

Always calibrate with the conditions as much as possible like the experiment as you can manage, same computer, same monitor, same screen resolution etc. etc.

### **Timing issues**

#### **1. Missing frames.**

If you are putting up a series of images in quick succession (for example 1 flip for every monitor refresh) it's important to make sure that drawing the offscreen images isn't taking too long. If the offscreen drawing takes too long then you may be missing **refreshes** or **frames**. This means that the program will be taking longer (and the stimulus will be moving more slowly, or present for longer) than you specified in the code. The way to check this is to record the times that the flips happen and make sure they match the frame rate of the monitor (as specified in the monitor control panel). If you are missing refreshes there are two main solutions to the problem (1) Try lowering the refresh rate of the monitor in the control panels to give you a little more time for each flip. (2) Try to do as much computational work as possible before you have to start quickly

throwing up flips (e.g. if you are putting up a series of face images, load them all into memory beforehand rather than loading them from disk which is slow).

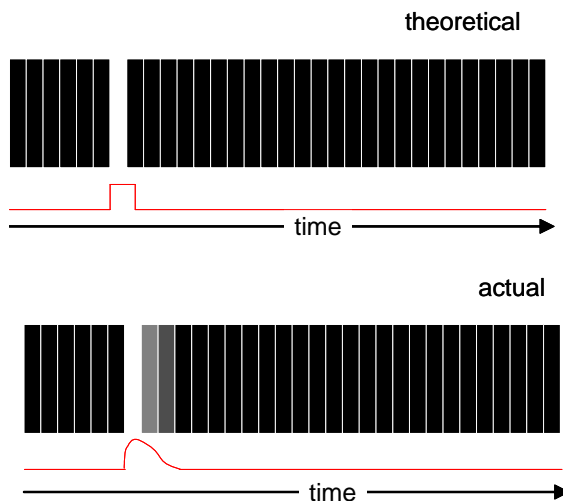
## 2. Phosphor persistence.

This is the phenomenon whereby even if you ask Psychtoolbox to only flash a gun for a single frame, that gun will still admit some energy after the frame is over. This is a particular issue for LCD and videoprojectors since these devices use relatively slow phosphors in comparison to good old fashioned CRT monitors. The issue of phosphor persistence is often an important issue for experiments where you are flashing up a stimulus briefly – imagine you have a 60Hz monitor and you are measuring how long a stimulus needs to be presented to be visible. When the stimulus is theoretically only up for a single frame (16.67ms) it is in reality presented for longer than that if it isn't immediately masked by some other stimulus. Of course this effect is most significant when a stimulus is only up for a very small number of frames. Another case where this is important is when you want to create a moving or flickering stimulus. The slow decline of the phosphors can result in a moving dot seeming to “streak” across the screen. For moving dot stimuli it is much better to use a CRT monitor if you can find one.

## Space

### 1. Pixels are not square (isotropic)

The pixels in monitors are not perfectly square. So if you want a perfectly square stimulus you need to calculate the length and width of each pixel. The best way to do this is simply to measure the extent of the screen's width and height (remember there's often a black rim around the edge of the display which shouldn't be counted) and divide that by the number of pixels. You can then try to adjust the monitor to make the pixels more square, or modify the code for your stimulus to compensate for the pixels not being perfectly square.



### 2. Calculating visual angle

Generally visual stimuli are expressed in terms of visual angle – i.e. the size of the visual stimulus as subtended on the retina. This is because for most visual tasks it is the size of the image on the retina that is important rather than the size of the image on the screen.

To calculate the visual angle subtended by a stimulus involves two stages. (Make sure you use the same units, e.g. cm for both stages).

#### 1) Calculate the width of a single pixel in cm.

(i) Measure the width and height of the screen (there is usually a black rim, don't measure that) in cm

(ii) Find out the number of pixels for that monitor horizontally and vertically

width of single pix in cm=width screen in cm/number of horizontal pixels (call this  $\text{pixH}$ )

height single pixel in cm=height screen in cm/number of vertical pixels (call this  $\text{pixV}$ )

It's possible that your pixels aren't square. It may be that you don't care. If you do, then you might want to adjust the settings on your monitor to make the pixels square. If your pixels still aren't square then you may have to have different values for the number of pixels per degree in the horizontal and vertical directions. If the pixels are reasonably square you can average vertical and horizontal values together to give you a single value (pix)

(iii) Measure the viewing distance of your subject (from the middle of their two eyes) to the center of the monitor in cm. (call this viewD)

2) Calculate the visual angle subtended by a single pixel

$\text{degperpix} = \text{atan}(\text{pix}/\text{viewD})$

It's often also useful to calculate the number of pixels within 1 degree of visual angle.

$\text{pixperdeg} = 1/\text{degperpix}$

Here's a program you can use to automatically calculate pixperdeg and degperpix

```
>params.res=[1024 768]
```

```
>params.sz=[30 24];
```

```
>params.vdist=57;
```

```
>[pixperdeg, degperpix]=VisAng(params)
```

Note that there are two values for pixperdeg and degperpix – the horizontal and the vertical values. If they are reasonably similar and your experiment doesn't depend critically on the stimulus being perfectly scaled along horizontal and vertical dimensions then you can simply average the two values of pixperdeg and the two values of degperpix to get an approximation of the visual angle subtended by each pixel

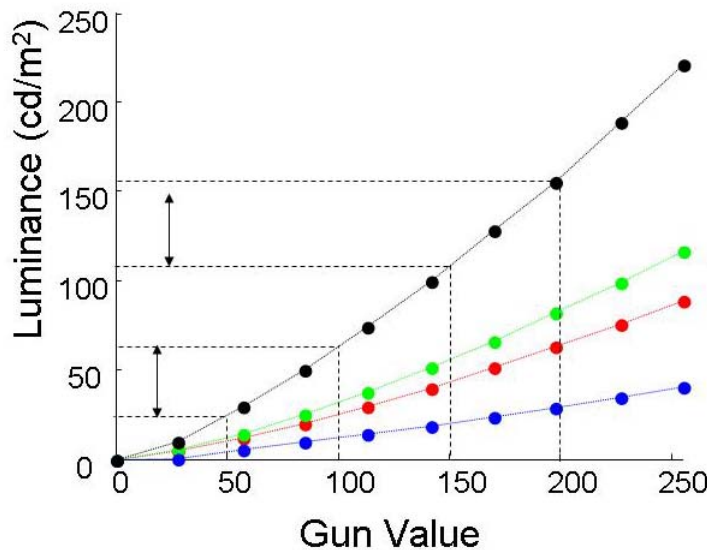
```
1 function [pixperdeg, degperpix]=VisAng(params)
2 % function [pixperdeg, degperpix]=VisAng(params)
3 %
4 %     Takes as input a structure containing:
5 % struct.res - the resolution of the monitor e.g. struct.res=[1024 768]
6 % struct.sz - the size of the monitor width then height
7 %     (needs to match the order you listed struct.res)
8 %     e.g. struct.sz=[30 24]
9 % struct.vdist - the viewing distance
10 %     e.g. struct.vdist=57;
11 %     Note that struct.vdist and struct.sz should be in the same
12 %     units (cm or inches)
13 %
14 %     Calculates the visual angle subtended by a single pixel
15 % Returns the pixels per degree
16 % and its reciprocal - the degrees per pixel (in degrees, not radians)
17 %
18 % written by IF 7/2000
19
20
21 degperpix=2*((atan(params.sz./(2*params.vdist))).*(180/pi))./params.res;
22 pixperdeg=1./degperpix;
23
```

## Luminance

### 1. Different monitors vary a lot in their relationship between gun value and luminance

This is true even of two monitors of the same brand. The monitor output for a given gun value may also vary depending on what computer you are using to drive it. If your experiment depends in any way on the luminance of the stimuli then you need to think carefully about your choice of monitors. Either you need to calibrate your monitors, so each monitor is displaying the same luminance value, or for experiments where this is less of a concern you don't want to run everyone from condition A on monitor 1 and everyone from condition B on monitor 2. You also don't want to swap monitors half way through the experiment if you can possibly help it.

### 2. Gamma Correction: The relationship between gun value and luminance is nonlinear and different for each gun

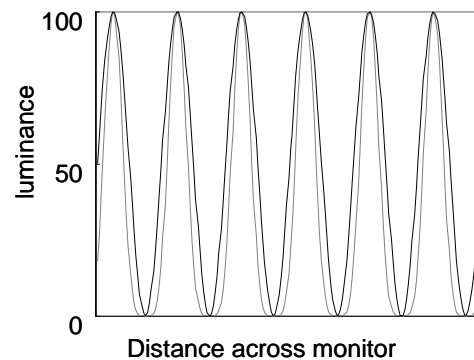


Up until now you have expressed image intensity in terms of gun values. But you should be aware that monitors vary a surprising amount in how these gun values translate into actual luminance or chromaticity values, and the relationship is not linear. Here I show how luminance is related to the gun value for the red, green and blue guns, and for all three guns on with an equal value (black line).

One way of thinking about this is that doubling the gun value does **not** result in a doubling of the luminance. In

the example shown here the gun value of 75 results in an image that is about 42  $\text{cd/m}^2$ , while a gun value of 150 results in an image that is about 109  $\text{cd/m}^2$ . Another way of thinking about this is that gun value 100 results in an image that is about 39  $\text{cd/m}^2$  higher in luminance than gun value 50. But gun value 200 results in an image that is about 51  $\text{cd/m}^2$  higher in luminance than gun value 150.

If you are using any stimuli that vary continuously in luminance, like Gabors or gratings, you will need to calibrate your monitor. Otherwise your stimulus will not actually be a grating (or whatever it is that it is meant to be). In the figure here, the black curves represent a true sinusoid. The gray curves represent how that sinusoid would be represented on the screen if you simply sent gun values to a typical monitor without calibrating. Notice how the bright parts of the sinusoid have gotten much narrower, and the dark parts of the sinusoid have



gotten much broader. This is because the gamma curve relating gun value to luminance is much steeper when the luminance is high.

A typical set of gamma correction measurement will look something like this:

GunValue	red Y	green Y	blue Y
0	0	0	0
28	4.4	5.3	.2
57	11.7	14.3	5.7
85	20.1	25.1	9.6
113	29.5	37.4	13.9
142	40.2	51.5	18.8
170	51.2	66.3	23.8
198	63.0	82.0	29.0
227	75.7	99.4	34.6
255	88.6	116.9	40.3

Rather than measuring the luminance output for every gun value we tend to fit these data using a gamma function of the form:

$$I = \alpha g^\gamma + \beta$$

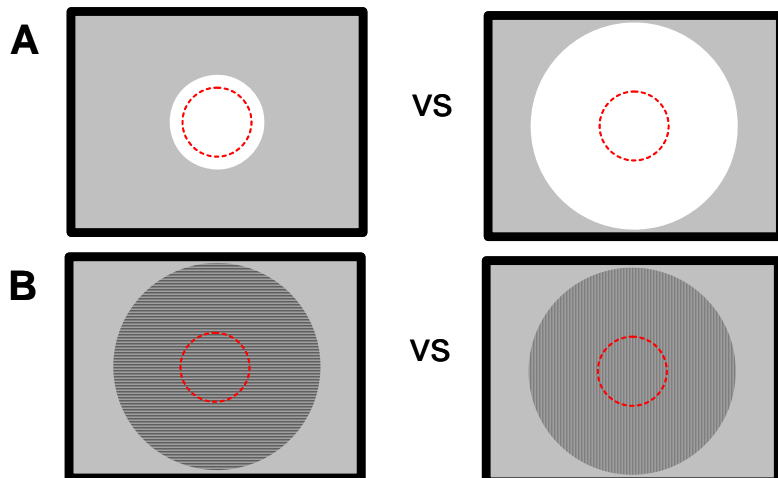
Where I is the intensity of the gun (or guns if you are calibrating for grayscale with all three guns on at an equal gun value), and g is the gun value.

**3. Stationarity: The relationship between gun value and luminance/color actually changes over the spatial extent of the monitor**

The luminance and color of a stimulus will depend on where it is on the monitor so you should always calibrate the region of the monitor that will actually be used to present the stimuli. If you are running an experiment where contrast is critical (e.g. comparing a contrast discrimination task where the stimulus is presented in different locations of the monitor) then it is worth measuring the relationship between gun value and luminance for the different locations, and making sure the monitor is reasonably homogenous over space. If not you may need different calibration tables for different locations.

**4. Power drain – the intensity of a pixel may depend on the extent to which neighboring pixels are also being driven**

Note that the amount of power drain tends to be worse between pixels on the same horizontal dimension than for pixels that are aligned vertically (this is certainly true for a CRT monitor, I don't know if LCD monitors have a



similar issue). This can be an important issue if you are comparing (for example) contrast discrimination thresholds for vertical and horizontal gratings. More mutual power drain along the horizontal dimension might mean that your horizontal gratings are actually slightly lower contrast. Good monitors shouldn't suffer too badly from this problem, but it's still worth checking. The following would be sensible ways of checking for power drain. The dotted red line represents the part of the screen that the photometer is measuring. Make sure the test patch is big enough to cover the whole region measured by your photometer.

In (A) you check whether the luminance output of a bright patch depends on how much more of the screen is also being driven. You want the measured luminance of the central bright patch to remain constant. If not, then you have a power drain problem. This will be a real nightmare if your stimuli vary in a way that will affect power drain. If they don't then the best work around is to calibrate using calibration patches that are as similar as possible to the stimuli you will use in your experiment. (B) is specifically designed to check whether power drain differs depending on whether stimuli are oriented horizontally or spatially. If you did not see any evidence for power drain with test A you should be OK, but if you are comparing vertical and horizontal gratings it might still be worth double checking using B. Note that the lines have to be small compared to the aperture of the photometer or you will get slightly different values depending on how many lines are fitting into the apertures. You really want the lines to be 1 pixel in width/height.

### *5. Monitors take time to warm up*

When calibrating a monitor or running an experiment you should turn on the monitor at least 5 minutes before the actual experiment begins (often experiments take 5 minutes to get started so this isn't something you have to worry a lot about). If the experiment depends heavily on luminance then you might want to measure the luminance of your monitor every minute after it's been started in the morning and measure how long it is before your monitor is stable.

### *6. Temporal stability: The relationship between gun value and luminance/chromaticity will change slowly over time*

When you are running an experiment you should, before the experiment begins

- 1) ALWAYS tape over the buttons that allow you to change monitor settings and add a warning to people not to change these without consulting you.
- 2) Write down the exact settings in the monitor control panel (resolution, bit setting etc. And put a note on the computer warning people not to change those settings without consulting you.

It is amazing how often a new lab assistant will change these settings without telling anyone. This quickly leads to a very bad lab atmosphere especially when the monitor concerned belongs to a graduate student who is trying to finish up their thesis.

Even without someone fiddling with the controls, monitors still change gradually over time, so you should recalibrate periodically. If your experiment depends critically on contrast/color then recalibrating once a months should be fine. Otherwise you should probably recalibrate every couple of months, and certainly when you start a new experiment.

### *7. Quantization errors.*

As you have learned, on most monitors the guns only take a certain number of discrete levels, generally 8 bits (256 different levels). When you display an image, you will generally provide a matrix that similarly contains integers between 0 and 255. If you send in non-integers you need to bear in mind that the gun value will simply be rounded to the nearest integer – sending in the gun value 145.3 will simply give you a gun value of 145. So be wary of this when calculating (for example) the rms contrast of a stimulus, or when setting the background to be the mean contrast of the stimulus.

For fine contrast discrimination tasks you may sometimes need to produce more discrete levels than 256. There are a variety of ways to do it, which won't be discussed here.

## Color

A color space is an abstract mathematical model describing how colors can be represented. Typically, because we have three cones, color spaces are represented in terms of three axes (since more are unnecessary).

Examples of color spaces include:

**1. Cone space** – color and intensity is represented in terms of the relative and absolute firing rate of human LM and S cones. This color space represents cone outputs, and tends to be mainly used by vision scientists.

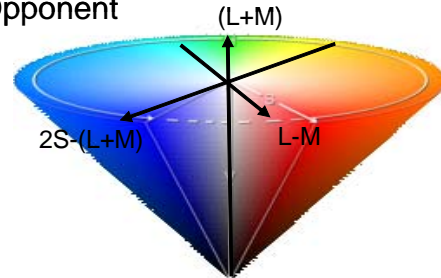
**2. Opponent color space** – This contains three axes – a luminance axis, a red-green axis and a blue-yellow axis. This is again a color space that tends to be used by vision scientists. In this case the color space is thought to represent color processing in the later parts of the retina and the cortex.

**3. RGB** – color can be represented in terms of red, green and blue intensity

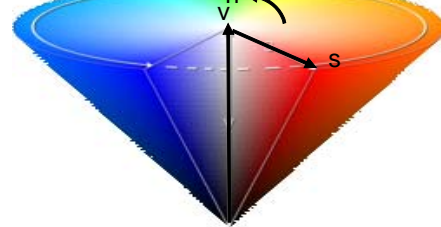
**4. HSV** – this represents color in terms of hue, saturation and brightness

**5. CIE color** – This is one of the earliest color spaces. It has the axis Y which represents luminance, X which represents redness, y which represents greenness and z which represents blueness ( $z=1-(x+y)$ ). CIE also uses the vectors X ( $X=Yx/y$ ) and Z ( $Z=Yz/y$ ).

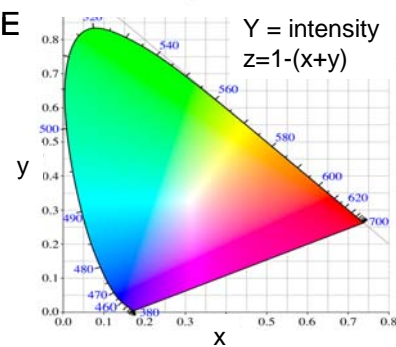
### Opponent



### HSV



### CIE

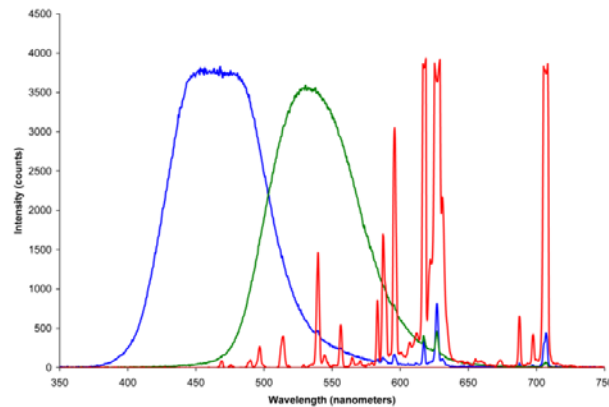


### 1. Different monitors vary in their relationship between gun value and chromaticity

Similar issues apply to color as to luminance – while equal gun values will tend to look gray, in fact that exact shade of gray will differ considerably across monitors. If your experiment depends in any way on the color of stimuli then you need to think carefully about your choice of monitors.

Either you need to calibrate your monitors, so each monitor is displaying the same color values, or for experiments where this is less of a concern you don't want to run everyone from condition A on monitor 1 and everyone from condition B on monitor 2.

When you look at the wavelengths emitted by a typical CRT monitor you can see that (as would be expected) the blue gun mostly emits short wavelengths, the green gun emits longer wavelengths and the red gun emits mostly longer wavelengths.



With a high quality spectrometer you can measure the energy as a function of wavelength, usually in 10-30nm bins. These are becoming surprisingly cheap – the GregtagMacbeth Eye One photometer will run you about \$250 (they used to run about \$7-12k). Cheaper or older colormeters will give you the chromaticity of each phosphor in some sort of 3d color space.

## 2. Gun drain/power stealing

For some monitors there is power stealing across the three guns. I.e. The luminance output of the red gun is smaller when the green gun is also on at full intensity. The best way to test for this is to measure the output of each of the three guns independently at maximum intensity (gun value of 255) and then measure the output of the three guns on together for a stimulus of the same size. The overall luminance should be the sum of the maximum luminance of the three individual guns. In my experience CRT monitors show a lot of power stealing (up to 10% drop in luminance), so this is something you do need to compensate for when calibrating. If you are using a monochromatic stimulus (e.g. all guns on at equal gun value) then just make sure you use gamma correction tables based on all three guns being on at equal intensities (don't just sum the gamma tables for the individual guns, since you will underestimate the luminance of your stimuli). If your stimuli are varying in color (and aren't just red, green and blue) then compensating for power stealing gets a little more complicated. We'll go into that in more detail later since full chromatic calibration is mostly for the brave.

## 3. The chromaticity of a monitor is only likely to be stable for intermediate luminance values.

Over a wide range of luminances if you have a fixed ratio of red, green and blue guns, then the stimulus will stay roughly the same in chromaticity – e.g. when red, green and blue guns have the same value the stimulus will be close to gray. This tends to fall apart for very low gun values (of less than 50), and to a lesser extent for very high gun values of 225 or more. So if your stimuli are falling into that range, then you want to be wary of slight shifts in chromaticity for high and low luminance stimuli. This is probably slightly due to slight inaccuracies in the fit of the gamma function for very low gamma values, but is mostly due to the chromaticity of the guns actually changing slightly when they are on at very low or very high intensities.

A typical set of chromaticity values for a monitor might look like this in CIE space:

For red gun

GunValue	x	y	z
0	0.5695	0.1406	0.2899
28	0.5544	0.0899	0.3557
57	0.5182	0.2493	0.2325
85	0.5220	0.1741	0.3039
113	0.5722	0.2164	0.2114
142	0.5025	0.2117	0.2858
170	0.5891	0.2011	0.2099
198	0.5784	0.1498	0.2717
227	0.5089	0.1526	0.3384
255	0.5367	0.1813	0.2820

For a green gun

GunValue	x	y	z
0	0.2407	0.4564	0.3029
28	0.2472	0.4828	0.2700
57	0.3736	0.3916	0.2348
85	0.3028	0.4270	0.2702
113	0.2391	0.4369	0.3240
142	0.2979	0.3893	0.3127
170	0.2436	0.3840	0.3724
198	0.2325	0.4966	0.2709
227	0.2869	0.4506	0.2625

255                    0.3477                    0.4177                    0.2346

## Calibration

The goal is of course to be able to measure the luminance and chromaticity of a monitor, and then use those measured values to specify the exact color and luminance you want the monitor to produce. Here's code that will let you do that.

### Grayscale calibration

If you don't care about the chromaticity of your stimulus then calibration is very simple. You simply find the best fitting gamma function that describes the transformation from luminance to intensity, when all the guns are on at equal value:

$$I = \alpha g^\gamma + \beta$$

Then you simply need to calculate the inverse of that gamma function, which is of course:

$$g = ((I - \beta) / \alpha)^{1/\gamma}$$

so if we want to find the gun values for a linear set of intensity values we simply plug in a linear set of gun values from the minimum possible, to the maximum possible and calculate the corresponding gun value.

Suppose

```
>calib,gv=[0 28 57 85 113 142 170 198 227 255];
>calib.measuredgray= [0.3000 0.5000 4 10 20 35 70 110 140 180];
>calib.graygamma.alpha=0.0002141;
>calib.graygamma.beta=0.0023;
>calib.graygamma.gamma=2.4666;
```

We can calculate a linear desired set of luminance values that spans the full range encompassed by the monitors:

```
>desiredlum=linspace(min(calib.measuredgray), ...
    max(calib.measuredgray), 256);
```

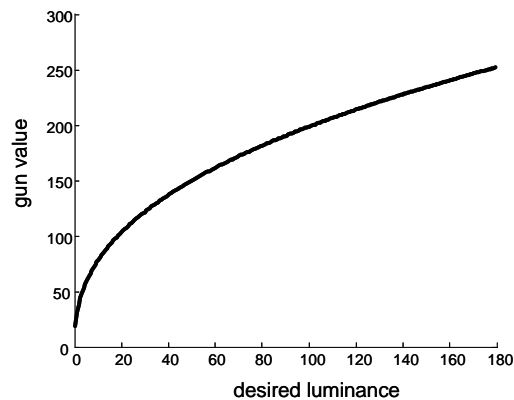
We can then calculate the inverse gamma table – i.e. the gun values you need to step linearly from the minimum luminance to the maximum luminance.

```
calib.inversegray=((desiredlum-calib.graygamma.beta) ...
    ./calib.graygamma.alpha) ...
    .^(1./calib.graygamma.gamma);
```

We can then associate our inverse gamma function with a psychtoolbox window so as to create a linearized window. To do this we first need to create a 256 rows x 3 columns matrix

```
>gtable=[calib.inversegray; calib.inversegray;
calib.inversegray]';
```

Note that because you can't actually get a luminance of 0 the inverse gamma table starts just above 0. You then need to scale the gtable so it goes between 0-1 instead of from 0-255.



```
>calib.graytable=scaleif(gtable, 0, 1)
```

Once you have done this, then there is a nice linear relationship between gun value and luminance. If you put in normalized gun value of 1 it will give you maxlum (180), a normalized gun value of 0 will give you minlum, and a normalized gun value of 0.5 it will give you a luminance value that is  $90.15 = (\text{maxlum} - \text{minlum})/2 + \text{minlum}$ .

### *Color calibration using spectral data and calculating LMS values*

Let's say you want to produce a particular response in L, M and S cones. The spectral output of the monitor will be a weighted sum of the three phosphors, so the overall light at each wavelength will be:

$T = SI$  – where T is the overall spectral output of the monitor

$I = [I_r ; I_g ; I_b] / (\text{max}I_r + \text{max}I_g + \text{max}I_b)$  – this is a 1 x 3 matrix containing the relative luminance of each gun

$S = [S_r S_g S_b]$  - where S is a n x 3 matrix where n is the number of samples of wavelength

We can then go on to calculate cone outputs as follows:

$C = [L M S]^T$  – this describes the spectral absorption spectrum of L, M and S cones.

$O = CSI$  - describes the output of the cones for a given set of luminances for each of the cone types

$M = CS$  is called the calibration matrix of the monitor and should be a three by three matrix

[0.2732 0.9922 0.1466

0.1034 0.9971 0.2123

0.0117 0.1047 1.0]

To calculate the luminance needed on each gun to achieve a desired output for each of the three cones we just work our way backwards

$$I = M^{-1}O$$

Where O is the desired output for each of the three cone types.

This technique is pretty standard and doesn't take account of gun drain (the luminance when all three guns are on full being less than would be predicted by each gun being on separately at full)