Real-Time Gamma Correction Procedure

Faisel G. M. Al-Hamadi & Mahfoth M. B. Al-Ani
Department of Computer Science, College of Science, University of Baghdad, Baghdad-Iraq.
Received: 19/1/2004 Accepted: 23/4/2005

Abstract

In the current research work real time gamma correction (contrast balancing) procedure were performed. The motivation behind this research work is to reproduce an accurate image on the monitor for an uncorrected image on a particular computer system. But, in the real world, it isn't quite this simple, especially when an image needs to look good on different systems, or platforms because most computers, or more specifically, most computer systems, do not work in exactly the same way. However, the current work exploit the advantage that the most monitors work in about the same way with respect to gamma correction to reproduce correction image. Different images from real world were tested using current procedure to illustrate efficiency of it which give good results.

المستخلص

في البحث الحالي خوارزمية لتصحيح كاما (موازنة التمايز) بوقت معالجة حقيقي وتم أختبار هذة الخوارزمية على صور واقعية مختلفة وكانت النتائج جيدة. إن المحفز الرئيسي خلف هذا العمل البحثي هو لاعادة إنتاج صورة واضحة (مصححة) على شاشة الحاسبة في بيئة تشغيلية معينه (نظام تشغيل معين). ولكن يصبح العمل أكثر صعوبة عند استخدام أنظمة تشغيل مختلفة وذلك بسبب أختلاف ألية عمل هذة الانظمة. على كل حال فأن هذا البحث تم فيه استغلال ميزة أن معظم أنظمة العرض تعمل بنفس الطريقة بالنسبة لعملية تصحيح كاما لاعادة إنتاج صورة مصححة.

Introduction

The term *gamma correction* means doing graphics color math accounting for the distortion that the color will eventually go through when displayed on a monitor or in other term (Gamma correction controls the overall brightness of an image. Images which are not properly corrected can look either bleached out, or too dark. Trying to reproduce colors accurately also requires some knowledge of gamma. Varying the amount of gamma correction changes not only the brightness, but also the ratios of red to green to blue) [1].

In Windows color data is usually represented as a set of 3 BYTE values for red, green and blue color components, where (0,0,0) represents black and (255,255,255) represents white. Depending on the output device medium values like (128,128,128) are displayed differently, thus an area of (128,128,128) pixels might look brighter or darker than an area with alternating black and white pixels.

When using a cathode ray tube (CRT) display the cause of this behavior is the non-linear brightness curve of the tube, while in the case of printing usually a black dot being printed is bigger than it ought to be, so the resulting picture looks ”darker” even though white is still white and black is correctly black. This *non-linear distortion* is not correctable by the linear brightness and contrast parameters. The most-known standard used to correct the non-linear brightness distortion is known as "gamma correction" according to the formula [2]:

$$\boldsymbol{color}' = 255 \times (\frac{\boldsymbol{color}}{255})^{\frac{1}{gamma}} \qquad \textbf{(1)}$$

color = not corrected color component value, standardized to values between 0.0..1.0

color'= corrected color component value, standardized to values between 0.0..1.0

gamma = correction value, for screens typically. 1.5-1.9

The phosphors on the face of the monitor tube luminance when struck by the beam of the electron gun sweeping out the scan lines. By increasing the intensity of the beam, the phosphor dot luminance's more brightly, and by reducing the intensity of the beam, the phosphor glows less brightly. Unfortunately, the output of the phosphor is not directly proportional to the impinging beam strength. The response of the phosphors usually looks something like this

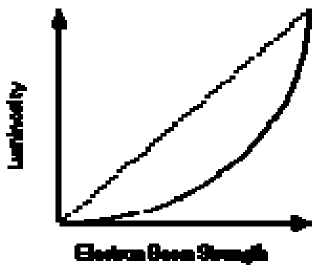


Figure (1)

Where the dotted line shows ideal linear response and the solid line approximates the observed response of a normal phosphor. This response characteristic is due to physical phenomena and can be described via a function of the form (eq.(1)). Typically monitors have a gamma from anything between 1.5 and 2.8

Subjectively, this causes the colors on the screen to appear darker than expected. Based on this behavior, an inverse gamma correction function can be applied to compensate for this non-linear response:

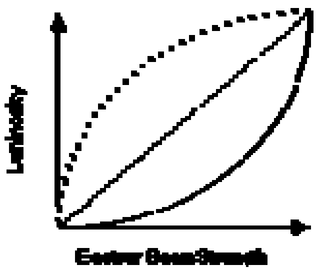
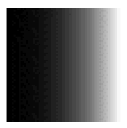
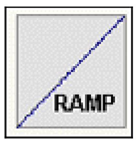


Figure (2)

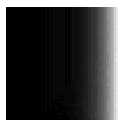
Here, the solid line again represents the uncorrected phosphor response, the large dotted line is the inverse gamma function, and the fine dotted line is the resultant, linearized color response.



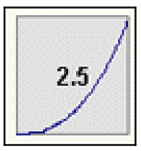
Sample Input to monitor



Graph of input



Output from Monitor



Graph of Output L = V ^ 2.5

Figure (3)

Setting the gamma value

To adjust the gamma value "roughly", set the contrast and brightness of your monitor to your preferred/comfortable level and use whatever gamma software comes with your hardware.

In the absence of other information it is usual to set the effective gamma to 2.2, this is the standard employed for PhotoCD's. On SGI machines this normally means setting a "gamma" value of around 1.2, this is because SGI's are normally configured with a natural gamma of 2.6, the combined gamma value is 2.6 divided by this number [3].

All well written image viewers should provide options for controlling the gamma or they should get the value from the system being used. For example on UNIX machine, the **xv** viewer can be called with the option **-gamma 2.2**. Unfortunately at the time of writing none of the two main WEB browsers support gamma correction for images in WWW pages, as a result the images tend to look dark on uncorrected monitors. This is a major problem when it comes to distributing images on a wide variety of systems. Fortunately the PNG format does support the specification of a colourmetric model and will go a long way to providing consistent image appearance across monitor hardware [4].

Applying Gamma Correction filter to an image

A Gamma Correction filter is primarily used for color matching purposes on CRT based displays, but it can also be used for creative image processing. This article shows how the gamma filter is calculated and provides an implementation .

1. About the Gamma Filter

Gamma correction was developed to make it easier to adjust color displayed on Cathode Ray Tube (CRT) displays. CRT displays produce a light intensity (luminance) proportional to the input voltage raised to a power. Since no two CRTs are exactly alike in their luminance characteristic, a way of adjusting the input image so the displayed colors match a reference is needed. This is done by adjusting the colors to be displayed by a power which is termed gamma. Since gamma lightens or darkens the colors in an image it can be used for image processing effects as well as the normal color profile adjustment. Gamma is normally restricted to the range 1/5 to 5, where a value less than 1 lightens the image; a value of 1 leaves the image unaffected and a value greater than 1 darkens the image.

2. Calculating Gamma

To apply a gamma filter, each color is factored using the inverse power of the gamma value (equation (1)). Since the gamma calculation involves powers, and for a given value of gamma and the output color value is constant, it makes sense to pre-calculate the new color values and store them in an array so the calculation doesn't need to be repeated each time. Then the new value can be looked up simply as the index of the entry in a one-dimensional array (the following source codes were written with visual basic 6):-

```
Private m_fGamma As Double
Private m_red(0 To 255) As Byte
Private m_green(0 To 255) As Byte
Private m_blue(0 To 255) As Byte

Private Sub createGammaTable()
Dim i As Long
Dim lValue As Long
  For i = 0 To 255
    lValue = (255#*((i / 255#) ^ (1# / m_fGamma)))
    If (lValue > 255) Then lValue = 255
    m_red(i) = lValue
    m_green(i) = lValue
    m_blue(i) = lValue
  Next i
End Sub
```

Note that the red, green and blue gamma values are all set to the same amount in this example. A different gamma could be used for each to achieve colorization effects .

Applying the Filter and Experimental Results

Once the gamma arrays have been calculated it is relatively simple to apply them to an image. For each pixel in the array, read the red, green and blue values and then replace them with their lookup within the gamma tables :

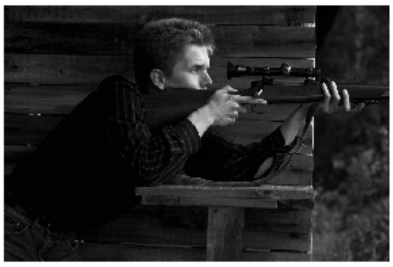
```
    Dim bDib() As Byte
    Dim bDibDst() As Byte
    Dim tSA As SAFEARRAY2D
    Dim tSADst As SAFEARRAY2D
```

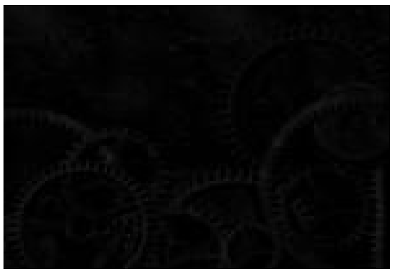
Get the bits in the from DIB (Device-Independent Bitmap) section:

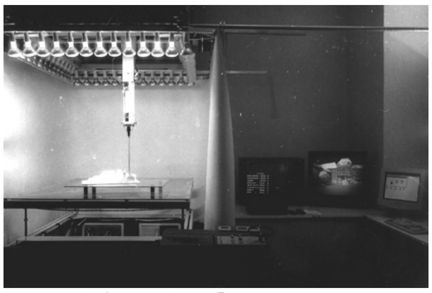
```
    CopyMemory ByVal VarPtrArray(bDib()), VarPtr(tSA), 4

    Dim x As Long
    Dim y As Long
    Dim xEnd As Long
    Dim yEnd As Long

    xEnd = cSrc.BytesPerScanLine() - 3
    yEnd = cSrc.Height - 1

    For x = 0 To xEnd Step 3
      For y = 0 To yEnd
        bDibDst(x + 2, y) = m_red(bDib(x + 2, y))
        bDibDst(x +1, y) = m_green(bDib(x + 1, y))
        bDibDst(x, y) = m_blue(bDib(x, y))
      Next y
    Next x
  CopyMemory ByVal VarPtrArray(bDibDst), 0&, 4
  CopyMemory ByVal VarPtrArray(bDib), 0&, 4
```

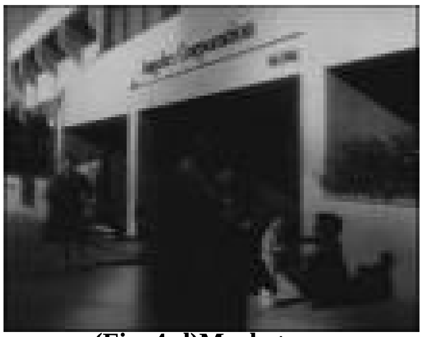
Experimental results

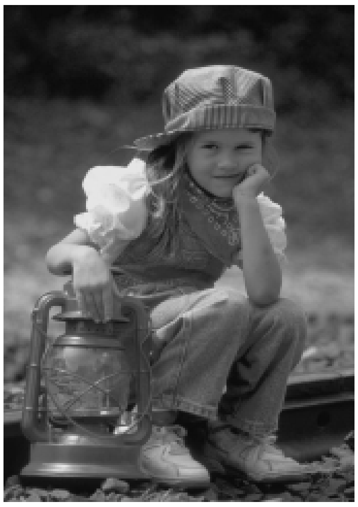
Natural images as shown in figure (4) are subjected to gamma correction with. The amount of gamma is estimated by using visual perception (objective fidelity criteria). A range of possible gamma values from 0.2 to 3.8 are used to give the better results (figure (5)). However this is the off-line case (i.e. to show the program implementation and its results),thus this program tested using digital camera with real-time processing (frame rate=15 FPS) with image size up to 350X400).

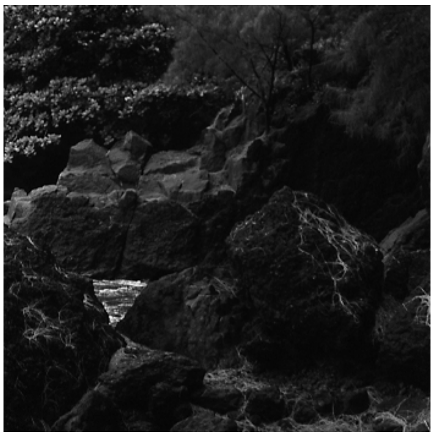

(Fig. 4-a) Hunter


(Fig. 4-b) Deshler


(Fig. 4-c) Laboratory


(Fig. 4-d)Market


(Fig. 4-e) Girl


(Fig. 4-f) Scene

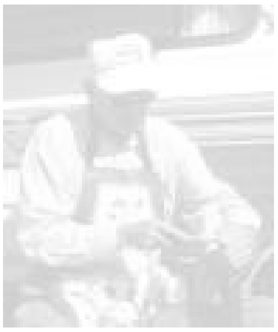
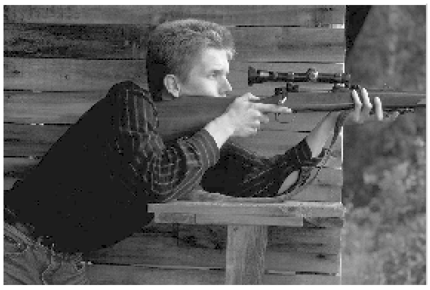
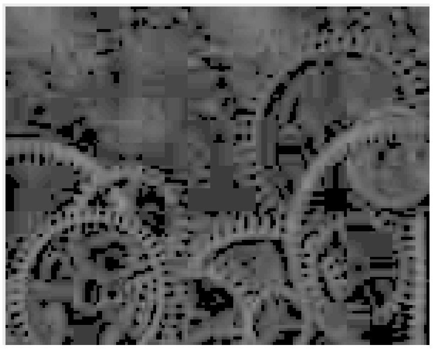
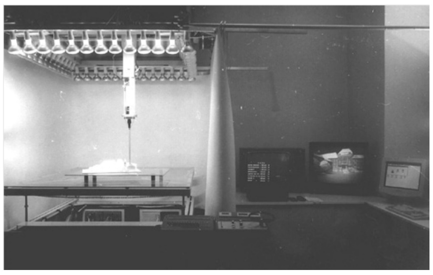

(Fig. 4-g) Man

Figure (4): Original images (theses images obtained by using www.googleImage.com search engine with keyword "Low Intensity")


(Fig. 5-a) Hunter


(Fig. 5-b) Deshler


(Fig. 5-c) Laboratory

(Fig. 5-d)Market

(Fig. 5-e) Girl

(Fig. 5-f) Scene

Fig. 5-g) Man
Figure (5): Gamma corrected images where (a) gamma=2.2, (b) gamma=3.8, (c) gamma=1.8, (d) gamma=1.4, (e) gamma=0.2, (f) gamma=2.2, (g) gamma=2.6

Conclusion and future work

The sample code demonstrates applying gamma to a True Color DIB Section. This type of code can be used as a simple way to highlight, brighten or darken images .

A 24 bit color system can display 2^24 or 16.7 million colors simultaneously. However, the same hardware may be capable of displaying many more colors, just not more than 16.7 million simultaneously. Anyone who has worked on an 8-bit color system (256 colors) knows that they can display 256 colors at any given time and can change their palette of 256 colors as their needs change. Similarly, people with 16-bit colors systems (65536 colors) can display more than 65,536 colors by chaining their palettes. With 24 bit color systems, many people overlook this option of rearranging their palette. Most software doesn't explicitly tell a user how to change the palette and low end graphics cards may not allow the capability to extend the palette to a different set of 16.7 million colors. But if you work with 16 bit color, or 8 bit color have a really nice graphics card with a 10-bit DAC or 12-bit DAC (Digital Analog Converter) it would behoove you to take note of this palette switching capability.

Bit-Division One very effective way of changing the palette is by dividing up the 24 bits differently among the red, green and blue channels [5]. Typically the red, green, and blue channels are each assigned 8 bits of color. Should a user desire more control over the colors in one channel, however, these bits could be divided differently. Had Monet been a computer artist while doing his studies of blue, he could have assigned 10 bits to blue, and 7 each to red and green, thereby offering a more blue palette. Since most graphics applications and file formats do not support this method, however, it is not very practical to implement, unless you are a computer programmer writing your own graphics package.

References

1. Y.-C Chang and J.F. Reid. *RGB calibration for analysis in machine vision*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 5(10):1414Œ1422, **1996**.
2. R.P. Kleihorst, R.L. Lagendiik, and J. Biemond. *An adaptive order-statistic noise _lter for gamma-corrected image sequences*. IEEE Transactions on Image Processing, 6(10):1442Œ1446, **1997**.
3. X. You and G. Crebbin. *Image blur identi_cation by using higher order statistic techniques*. In Proceedings of the 3rd IEEE International Conference on Image Processing, pages 77Œ80, Lausanne, Switzerland, **1996**.
4. M.A. Ibrahim, A.W.F Hussein, S.A. Mashali, and A.H. Mohamed. *A blind image restoration systemusing higher-order statistics and Radon transform*. In Proceedings of SPIE - The International Society for Optical Engineering, volume 3388, pages 173Œ181, Orlando, FL, USA, **1998**.
5. D.T. Kuan, A.A. Sawchuk, T.C. Strand, and P. Chavel. *Adaptive restoration of images with speckle*. In Proceedings of SPIE - The International Society for Optical Engineering, volume 359, pages 29Œ38, San Diego, CA, USA, **1983**