Gray-centered RGB color space

Stephen J. Sangwine and Todd A. Ell
University of Essex, Department of Electronic Systems Engineering,
Wivenhoe Park, Colchester, CO4 3SQ, UK.
and 5620 Oak View Court, Savage, Minnesota, USA.

Abstract

RGB color spaces are widely used in image processing. In this paper the authors discuss the use of RGB color space in which the origin is centered in the RGB cube at middle gray. This may appear a trivial difference from the traditional RGB color space in which the origin is at black, yet it is shown that significant advantages accrue from offsetting the origin, as is traditionally done in signal processing with bipolar samples. The geometry of vectors (pixel values) in gray-centered RGB space is described, and related to familiar properties from the hue-saturation-intensity (HSI) representation, which is explained as a different coordinate system, based on RGB. Finally, to illustrate the geometric advantage of centering the origin, the paper presents a simple but effective clipping algorithm for dealing with out of range vectors which can arise from filtering operations applied in RGB space. The effectiveness of the clipping algorithm is demonstrated with some example images.

1. Introduction

RGB color spaces are widely used in image capture, image processing and image storage [1]. Most image capture devices operate in an RGB color space, and so do many display devices such as CRT and LCD monitors. Many image file formats also use RGB color space (not all do, as some lossy formats like JPEG use luminance/chrominance color spaces).

RGB pixels stored in image files, and passed to display devices, are conventionally represented so that (0, 0, 0) represents black and (M, M, M) represents white, where M is the maximum sample value that can be represented (e.g. 255 for an 8-bit sample). The three primary colors, red, green and blue are represented by (M, 0, 0), (0, M, 0) and (0, 0, M) respectively in this representation. The reason for this representation is that these values are those required to drive a digital-to-analog converter in a CRT display, [2], black being obtained by the absence of beam current.

An RGB color space is characterized by its primary chromaticities, and various standard RGB spaces exist. Perhaps the most significant is the sRGB space[3], which is widely supported for Internet applications and which uses the same chromaticities as the ITU-R BT.709 high definition television standard [4].

Table 1: *Unit RGB coordinates and the colors represented.*

R	G	B	Color name
0	0	0	Black
0	0	1	Blue
0	1	0	Green
0	1	1	Cyan
1	0	0	Red
1	0	1	Magenta
1	1	0	Yellow
1	1	1	White

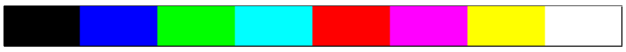
Figure 1: *RGB color bars*

In all of these RGB color spaces, the range of allowable pixel values falls within a cube in the RGB coordinate space. It is normal in image processing to normalise the coordinates of this cube so the coordinate components fall in the range 0 to 1 inclusive. We call this in what follows, the *unit RGB cube*. In this normalised representation the vertices of the cube represent the three primary colors, the three secondary colors (combinations of the primaries in pairs), plus black and white. Table 1 lists the coordinate values and the color names, and Figure 1 shows what the colors look like, in the same sequence as listed in the table.

Perhaps because of the widespread use of the above representation, many attempts to process images in RGB space have used the unit RGB cube representation. In the next section we argue for the use of a different coordinate system that is trivially derived from the unit cube, but which offers significant advantages. We have used this color space in previous work on vector filtering, [5], but until now we have not presented a clear and complete discussion of its properties. The main motivation for using this color space is that the geometrical manipulation of pixel vectors in this space can be simply related to perceptual properties of color, and we seek to design and discover linear vector filters that perform useful manipulations of images, based on geometrical operations on the pixel vectors [6].

Before we explain the gray-centered RGB color space, we remark on another, not quite so trivial, change of representation from the unit RGB cube, but which is nevertheless a simple change of coordinates: the hue-saturation-intensity (HSI) representation. The HSI 'color space' is simply RGB space mapped to a cylindrical polar coordinate system, and it is well described in the literature [1, 7]. The black–white axis of RGB space becomes in the HSI coordinate system an intensity axis, and perpendicular to this are planes of constant intensity. On each of these planes one imagines a reference direction which is conventionally the projection of the red axis of RGB space. A given pixel value has a *hue* which is the angle subtended at the intensity axis between vectors directed from the intensity axis to the pixel value, and from the intensity axis along the reference direction. The hue values of the primary colors are: red 0°, green 120°, blue 240°; and for the secondary colors: yellow 60°, cyan 180°, magenta 300°. Achromatic or gray colors have no defined hue. The third coordinate, saturation, varies somewhat in its definition, but essentially measures distance from the achromatic or intensity axis. The primary and secondary colors are fully saturated, and have saturation values of 1 conventionally. HSI space has been used for many image processing purposes, but there are problems, particularly with the fact that hue is undefined for achromatic pixels.

2. Gray-centered RGB color space

It is a small change from the above representation (the unit RGB cube) to gray-centered RGB color space. In gray-centered RGB color space, the unit RGB cube is translated so that the coordinate (0, 0, 0) represents mid-gray (halfway between black and white). The translation is achieved by subtracting (½, ½, ½) from each pixel value in unit RGB space. In terms of unnormalized RGB values, we subtract half the full-scale range, that is 128 for an 8-bit sample, or 32768 for 16 bits. This translation is obviously equally trivially reversed by adding (½, ½, ½). We may note that exactly this translation of origin is done in standard signal processing, where unipolar samples from an ADC are converted to bipolar samples for processing by subtracting half the full-scale range. The conversion can also be regarded as from unsigned natural binary to signed two's complement.

To explain what this translation of origin achieves, think of all pixel values as vectors directed away from the origin of the image color space. Since the origin in the gray-centered RGB color space represents mid-gray, the pixel vector represents by its length and direction how much the pixel color differs from mid-gray. It may not be obvious, but all pixels with a common direction away from the origin have the same *hue*. What is more, vectors directed in opposite directions from the origin have opponent colors (they have hues that differ by 180°). That is, any two pixel values (r_1, g_1, b_1) and (r_2, g_2, b_2) in the gray-centered RGB color space have the same color if there exists a positive constant k such that:

$$(r_1, g_1, b_1) = (kr_2, kg_2, kb_2)$$

and they have opponent colors if there exists a positive constant k such that:

$$(r_1, g_1, b_1) = (-kr_2, -kg_2, -kb_2)$$

These two pixel values have the same hue in the first case and hues differing by 180° in the second case, except when $r_1 = g_1 = b_1$, when the pixels are achromatic. In this case, where vectors are directed along the black–white axis, black and white may be regarded as in some sense 'opponent colors'. Pixel values with greater magnitude represent more saturated colors than pixels in the same direction with lesser magnitude, but of course in some directions in gray-centered RGB color space, increase of pixel magnitude increases intensity much more than saturation, and for directions at about 45° to the black–white axis, increasing magnitude increases saturation and intensity together.

To illustrate these points, Figures 2 and 3 show some color ramp images. (These color ramps are, of course, expressed in unit RGB color space in order to be displayed and included in the electronic form of this paper, but their representation in gray-centered RGB color space is what is of interest in this paper.) All the pixels in a given color ramp image are aligned in one direction in gray-centered color space. The magnitude of the pixel vectors varies uniformly from left to right along the image, passing through zero at the center of the image (where the color is always mid-gray). Thus the colors to the left of the centre of the image are opponent colors to those on the right. In terms of the perceptual quantities *hue*, *saturation* and *intensity*, these images have constant hue from centre to left edge and from centre to right edge (and the two hues are opponent hues, with a difference of 180°); saturation varies from a maximum at the left edge to zero at the centre and back to a maximum at the right edge; intensity varies monotonically from left to right edge. The first set of color ramps (in Figure 2) shows colors along a line from one vertex (a primary color) to the opposite vertex (a secondary color) of the RGB cube. (The first of these is from white to black, which is along the luminance axis rather than from a primary color.) The second set of color ramps (in Figure 3) shows colors along a line from the center of one face of the RGB cube to the center of the opposite face. These colors do not have standard names, and it is noticeable that

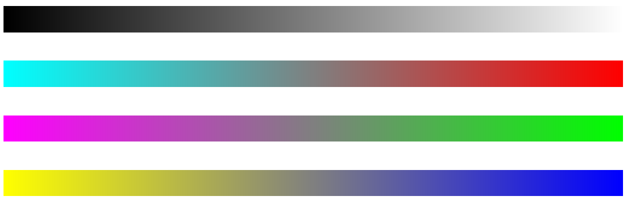
Figure 2: *Color ramp images from vertex to vertex of the RGB cube. Top to bottom: grayscale, red/cyan, green/magenta, blue/yellow. Vector directions: (1,1,1), (1,-1,-1), (-1,1,-1), (-1,-1,1), respectively*

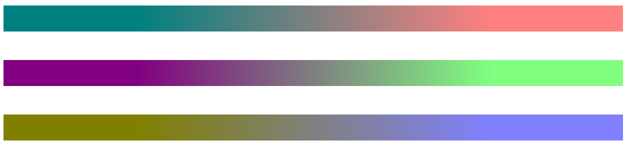
Figure 3: *Color ramp images from face to face of the RGB cube. Top to bottom, vector directions: (1,0,0), (0,1,0), (0,0,1).*

in each case one extreme of the color ramp appears more saturated than the other. It is striking that in all these color ramp images the colors in each half of the ramp are perceived as the same color. (This is not surprising to those familiar with the HSI space, as it is well-known that hue is a strongly perceptual quantity.) This has obvious implications for filtering and it shows that an approach based on the geometrical manipulation of vectors in gray-centered RGB color space is worth pursuing. Finally, before leaving this discussion of the gray-centered RGB color space it is worth briefly discussing the effect of the translation of origin on the frequency content of the image. Since a constant value has been subtracted from the image, the only effect on the Fourier spectrum of the image is to change the value of the DC or zero frequency component. All other coefficients in the Fourier spectrum are unaffected by the translation of origin.

3. A clipping algorithm that preserves 'color'

In this final section we present a simple, but effective, clipping algorithm that converts an out-of-range gray-centered RGB value into an in-range unit RGB value ready for output to a display device or image file. The significance of this is not that it is a particularly good clipping algorithm, as there are much more sophisticated algorithms known. Rather, it illustrates that the gray-centered RGB space permits a particularly simple manipulation of the pixel values based on vector concepts (the same concepts that can be used to devise vector filters).

Display devices and image file formats handle finite range samples. Often the range of samples is characterised as 8-bit or 16-bit unsigned natural binary (numeric ranges of 0 to 255 or 0 to 65535 respectively). Any pixels with RGB components outside the unit range must be truncated in some way to produce an in-range value. Out-of-range values can be generated in processing algorithms (such as linear filters).

A desirable characteristic of any clipping algorithm is that it should not distort hue. The reason for this is that hue changes are perceptually highly noticeable, and if a clipping algorithm changes the hue of a pixel, the result may be noticeable when the clipped image is viewed. The gray-centered RGB color space provides a simple way to ensure that hue is preserved: clip the pixel vector without changing its direction relative to the mid-gray origin of the color space. Note that this cannot be achieved by clipping the pixel RGB components independently – the clipping must be done in a vector sense. A suitable algorithm is as follows: compute $M = \max(|r|, |g|, |b|)$ where (r, g, b) is the pixel value expressed in gray-centered RGB space, and $|x|$ denotes the modulus of x. The clipped value is then given by:

$$(r', g', b') = \begin{cases} (r, g, b) & \text{if} \quad M \leq 1.0 \\ (r/M, g/M, b/M) & \text{if} \quad M > 1.0 \end{cases}$$

A simple way to test this algorithm is to convert an image into gray-centered RGB color space, and then scale the pixel values by a constant such as 1.5 (this must obviously be done using a numeric range for the pixel samples which is larger than the normal 8-bit or 16-bit range, for example by using floating-point). This causes some pixel values to fall outside the unit gray-centered RGB cube. In fact, it expands the unit cube by the scale factor so that all pixel values are further from mid-gray. The algorithm above may then be applied, and the clipped values converted back into a standard 8-bit or 16-bit integer format for output to a file and/or display. Some results are shown in Figures 4 and 5. In the first case, the clipping algorithm was independent clipping of the RGB samples. In the second case, the algorithm above was applied in gray-centered RGB color space. Note that the scaling applied to these images is much more than would be expected from most processing algorithms – in order to demonstrate the effects clearly.

4. Conclusion

In this paper we have presented a reasoned argument for the use of a gray-centered RGB color space and demonstrated its advantages with a simple but effective way to handle out-of-range RGB values by a vector clipping algorithm that preserves hue. Pixel vectors in the gray-centered RGB color space can be related in a simple geometrical way to the familiar color properties of hue, saturation and intensity, yet are amenable to geometrical manipulation in order to implement vector filtering operations.

References

[1] H. Palus. Representations of colour images in different colour spaces. In Sangwine and Horne [8], chapter 4, pages 67–90.

[2] R. E. N. Horne. Colour video systems and signals. In Sangwine and Horne [8], chapter 5, pages 93–114.

[3] International Electrotechnical Commission, Geneva. *IEC 61966-2-1 (1999-10) Multimedia Systems and Equipment – Colour Measurement and Management – Part 2-1: Colour Management – Default RGB Colour Space – sRGB*, 1999.

[4] ITU. *ITU-R Recommendation BT.709-2: Parameter Values for the HDTV Standards for production and International Programme Exchange*. Geneva, (`http://www.itu.ch/`), 1990.

[5] S. J. Sangwine, B. N. Gatsheni, and T. A. Ell. Linear colour-dependent image filtering based on vector decomposition. In *Proceedings of EUSIPCO 2002, XI European Signal Processing Conference*, volume II, pages 274–277, Toulouse, France, 3–6 September 2002. European Association for Signal Processing.

[6] S. J. Sangwine and T. A. Ell. Mathematical approaches to linear vector filtering of color images. In *First European Conference on Color in Graphics, Imaging and Vision (CGIV 2002)*, pages 348–351, University of Poitiers, France, 2–5 April 2002. The Society for Imaging Science and Technology.

[7] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Addison-Wesley, Reading, MA, third edition, 1992. Reprinted with corrections 1993.

[8] S. J. Sangwine and R. E. N. Horne, editors. *The Colour Image Processing Handbook*. Chapman and Hall, London, 1998.

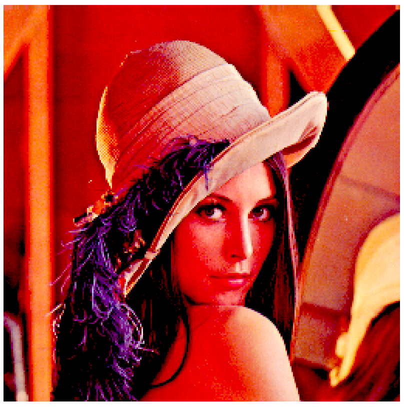

Figure 4: *Lena image scaled by 1.5 in gray-centered RGB color space and clipped using independent clipping of the RGB samples.*

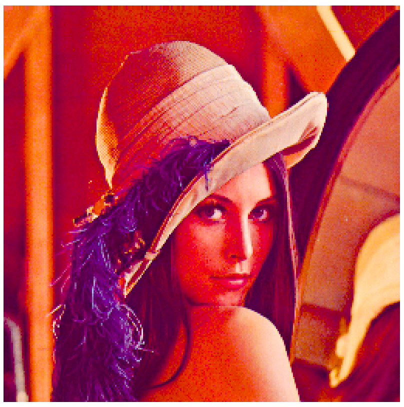

Figure 5: *Lena image scaled by 1.5 in gray-centered RGB color space and clipped using the algorithm in section 3.*

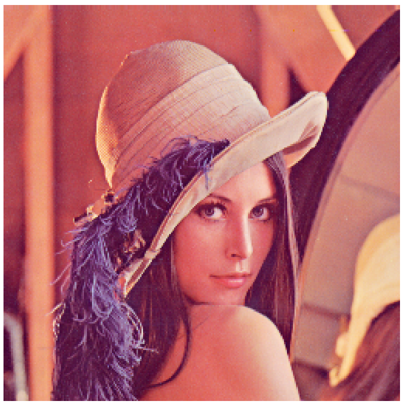

Figure 6: *Original lena image.*