1d2d3d: ONE, TWO, AND THREE DIMENSIONAL COLOR IMAGES
August 29, 2011

Carl Heiles

Contents

1. INTRODUCTION

This tutorial discusses the use of color in representing arrays of data. Normally we think of image display as just that—a way to display an image. However, you can combine more information into a single image by the combined use of color and intensity. Using color for these purposes is great, but using it intelligently is not trivial. There are three basic uses for color, and below we discuss each in the correspondingly-numbered section:

(1) Using a 256-entry color table to represent a single quantity. Consider this as a *one-dimensional* mapping of the quantity into color. The most common example is the ordinary grayscale image, in which the degree of whiteness or blackness represents the value of the quantity represented in the image (such as light intensity). Many astronomers like to replace the simple grayscale with pseudo color (often called "false color") because it's easier to discern changes of color than small changes of brightness. However, this has its problems: firstly, almost 20% of the human male population is at least partly colorblind; secondly, color reproductions in printed journals are very expensive.

(2) Using 256 *colors* to represent the value of a single quantity, such as the mean velocity of a spectral line, together with 256 values of *lightness* (brightness of the color: black is zero, full brightness is maximum lightness of a particular color) to represent a different quantity, such as intensity of the spectral line. You are using two visually independent qualities—color and lightness—to represent two quantities, so this is a *two-dimensional* mapping onto the image plane.

(3) Using red, green, and blue (r,g,b) to represent the values of three independent quantities, with the 256 lightnesses of each color proportional to its quantity. For example, using red for IR, green for optical, blue for X-ray. This is a *three-dimensional* mapping onto the image plane. You can't represent more than three dimensions in a color image because the eye is independently responsive to only three colors.

2. ONE-DIMENSIONAL COLOR: THE 8-BIT COLOR TABLE

2.1. The Basic Idea

In your computer's memory, an image consists of a two-dimensional array of data value d in each pixel of the image in which d can take on 256 different values ranging from 0 to 255. On your screen, the image consists of projected light in each pixel, which we call the intensity I.

There is a one-to-one relationship between d and I, so there are also 256 different possible values for I. Generally speaking, this relationship is specified by the "color table". The color table is often nonlinear so as to emphasize weak or strong features. Sometimes it's an equal mixture of red, green, and blue (r,g,b), which produces a grayscale image like a black-and-white photo; even though there isn't any color in a grayscale image, the relationship is *still* called the color table.

And often the mixture is engineered to produce color; this is often called a “false-color” image.

Figure 1 shows four identical images of the X-ray sky (data from ROSAT), all with the same stretch but different color tables. The top two are grayscale and reversed grayscale; the reversed grayscale enhances the visibility of dim features and is generally better for printed images. The bottom two show two false color images, one (“STD GAMMA-II”) having striking color differences so that it emphasizes particular intensities and one with a much smoother transition among successive colors.

2.2. Linear Mapping, both Direct and Reversed

In this discussion, the word “Intensity” (I) means the 256 different possible combinations of light intensity and color, one for each value of d. So I means not only the lightness of the image, as it would be in grayscale image, but also the color if it’s a false-color image.

The simplest mapping between data value d and intensity I is a linear one with

$$I = d \tag{1}$$

In this case, a data value $d = 255$ gives white and $d = 0$ gives black. This *direct mapping* is the default manner in which images are displayed on the computer screen: there is a black background on which the image is painted with increasing data values being increasingly white. However, on a piece of paper the relationship is usually reversed, because paper is white and provides a naturally white background. Thus, in this *reversed mapping*, we want to paint the image with increasing data values being increasingly black. This is also a linear mapping, but reversed:

$$I = 255 - d \tag{2}$$

Printed images usually look *much* better with the reversed mapping, because printers have a hard time giving a uniformly black area with no streaks. This is the *first* reason why printed images should be made with a reversed mapping. The *second* reason is that in scientific journals, images with the reversed mapping are reproduced much better. The *third* reason is that making the paper black uses lots of printer toner, which is expensive.

2.3. Nonlinear Mapping

The linear mapping is often not very useful because you usually want to highlight weak features or bright features; we’ll see an example below. The most commonly used nonlinear mapping uses a

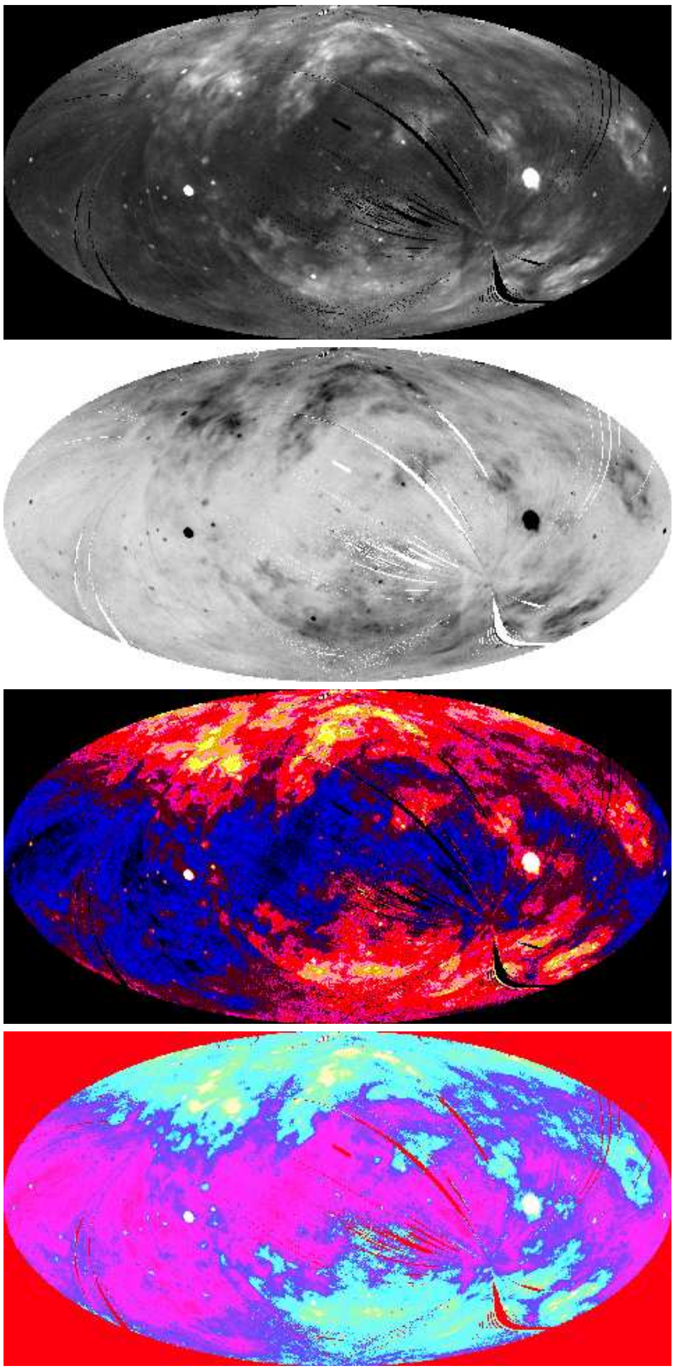

Fig. 1.— Examples of a one-dimensional color image: the XM ROSAT intensity map of the sky. From top to bottom we have: grayscale, reversed grayscale, false-color using “STD GAMMA-II” (IDL’s colortable number 5), false-color image using “Hue Sat Value 1” (IDL’s colortable number 21).

power law (this is the photographer's "characteristic curve") together with a "stretch", which cuts off the image at dim and bright intensity levels:

$$I = 255 \left(\frac{d - d_{bot}}{d_{top} - d_{bot}} \right)^{\gamma}, \; d = d_{bot} \rightarrow d_{top} \tag{3a}$$

$$I = 0, \; d \leq d_{bot} \tag{3b}$$

$$I = 255, \; d \geq d_{top} \tag{3c}$$

In a reversed mapping, you'd substitute $(255 - d)$ for d in the above equations.

There is one other commonly used nonlinear mapping, the so-called "histogram equalization" technique. In this technique, the mapping is modified so that all of the 255 colors are used in an equal number of pixels. Read about it in IDL's documentation on **hist_equal**.

2.4. Why We Need Nonlinear Mapping

Never forget that the idea is to turn the data array into an image that conveys information to the brain. The idea is *not* to be so strictly quantitative that details of interest are obscured.

You want to bring out details of interest. For example, for many images of the interstellar gas you want to emphasize weak structures at the expense of the fidelity gained by a strict proportionality between image brightness and data value. To this end, choose a color table and experiment with the image transfer function, using the techniques discussed in our tutorial "IDIDL—Image Display and Manipulation in IDL". At minimum, this involves changing the span of the data values represented in the image and raising the data values within that span to a power: a power less than unity to emphasize weak features, larger than unity for strong ones.

2.5. Considerations in Choosing a Color Table

For images to be printed on paper, a grayscale color table is usually best because color reproductions are often pretty bad unless you use an inkjet printer (although color laser printers have gotten a lot better in recent years). However, for other purposes, such as lecture slides, a color table is often preferred because you can see more details. And how do you choose a color table?

There is no all-purpose ideal color table. We have, among others, the following considerations:

- Sometimes you want to emphasize features of a certain intensity; in this case, a color table with strong contrasts between adjacent I values is good, as we did for the middle panel of Figure 1, for which we used IDL's "Standard Gamma II" (color table number 5).

- Sometimes you want to represent a wide range of d values without emphasizing one over another; in this case, you want a monotonically and smoothly changing color with data value. IDL's color tables satisfying this criterion include: Hue Sat Lightness 1 and 2 (numbers 19 and 20); and Hue Sat Value 1 and 2 (numbers 21 and 22) (For both pairs, number 2 might need to be reversed because white is at the low-intensity end of the range). Our example is the bottom panel of Figure 1, for which we used IDL's "Hue Sat Value 1" (color table number 21).

- Ideally, you want no visual ambiguity: a non-colorblind person would never suffer any ambiguity about the relationship between color and intensity. For most colorblind people it's red and green that cause the problem; this means don't use red and green in the same color table.

- Ideally, you would have the most intense portions of the image be represented by the physiologically brightest color in the color table. But, violating this sensible approach, it's often *disadvantageous* to have the least intense portions of the image be represented by the physiologically dimmest color (which is blue), because it will not clearly show weak features against the black background of zero intensity.

The very popular color table used by the VLA's AIPS uses red for the most intense parts of the image; the physiologically brighter white represents a lower intensity level. Despite this departure from a purist philosophy, the AIPS color table is quite good. Many other color tables that people use are simply awful when considered from a visual, not to mention logical, perspective.

2.6. The Colorbar

OK, you've carefully chosen your colortable and nonlinear transfer function and made a beautiful one-dimensional image that makes the mental impact you desire—just like we did in Figure 1. You're not done! There's no *quantitative* information in Figure 1!

There's nothing more annoying than seeing a nice one-dimensional image without an indication of how the grayscale or color relates to data value. You do this with a colorbar. Shame on you for even *thinking* of generating a 1-d image without a colorbar! See our 2-d and 3-d figures below for examples of colorbars.

3. TWO-DIMENSIONAL COLOR: AN 8-BIT COLOR TABLE WITH 24-BIT COLOR

3.1. The Basic Idea

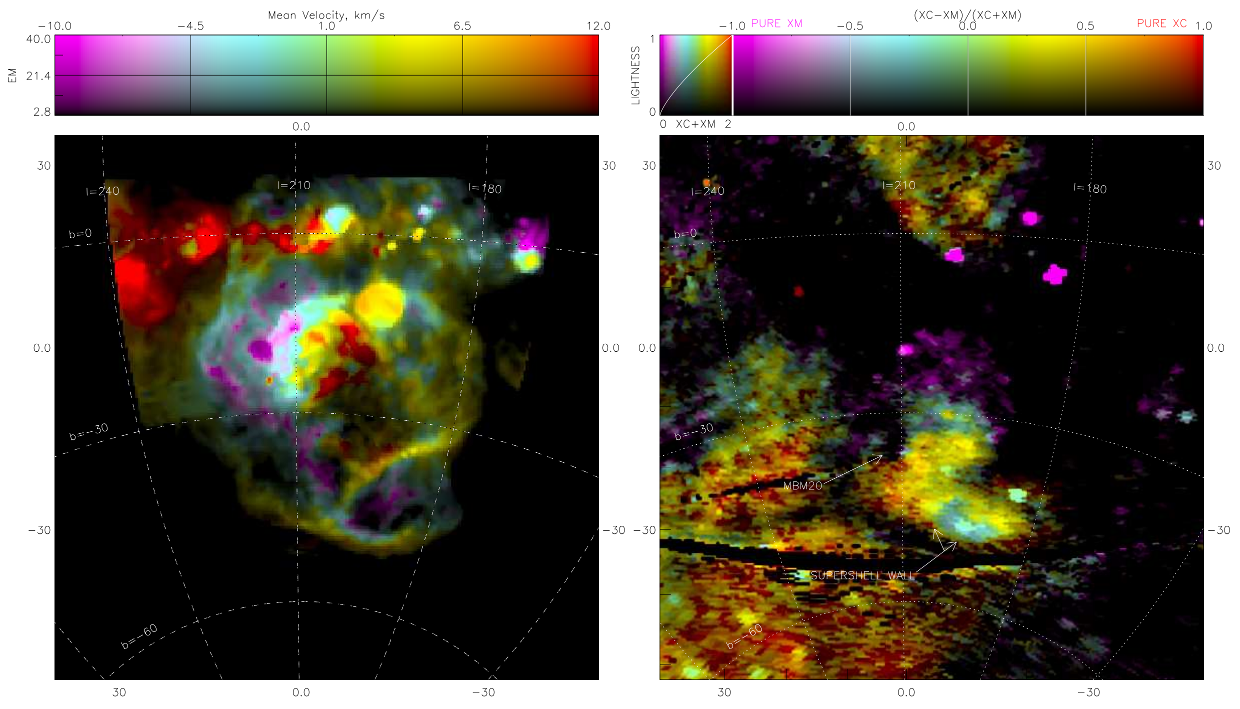

Fig. 2.— Examples of two-dimensional color images. On the left panel, color represents the mean velocity of the Hα line in the Orion-Eridanus superbubble. On the right panel, color represents the X-ray energy, and thus the hot gas temperature, in the superbubble. These images use the physiologically-desirable *pseudo* colortable. Obviously (we hope!), these images make little sense with a black/white printer!

Here, the idea is to use color to represent the value of a single quantity, such as the mean velocity of a spectral line, and use the lightness (apparent brightness of the color: black is zero, full brightness is maximum lightness) to represent a different quantity, such as the integrated intensity of the spectral line. This is a *two-dimensional* mapping onto the image plane. Most astronomical applications use color for velocity and lightness for intensity. The left-hand panel of Figure 2 shows an example, a map of the Hα WHAM survey data in the Orion-Eridanus superbubble; color represents mean velocity, lightness represents the line brightness raised to the 0.56 power. It's easy to see from this that the Hα emission from the left side of the superbubble is approaching, so it's on the near side of the expanding bubble, and the right side is receding.

The color dimension doesn't have to be velocity! The right-hand panel of Figure 2 shows an alternative. Here, the lightness represents the sum of the intensities of the XC and XM X-ray bands (from ROSAT) and color represents the fractional excess of the XC over the XM band, again for the Orion/Eridanus superbubble. XC is 0.25 kEv and XM is 0.75 kEv, so on the colorbar at the top, X-ray energy—i.e., temperature of the X-ray emitting hot gas—decreases from left to right. There is a clear temperature gradient from just above middle to bottom—i.e, as the distance from the Orion cluster (near center top, just below the top coordinate grid) increases, the temperature decreases. And as the temperature decreases, the brightness (and, probably, the gas density) increases, which probably indicates rough pressure equality from top to bottom—as is expected from superbubble dynamics.

To create a 2-d color image you begin by making two images. One is a 1-d gray scale image of the intensity, and you can fiddle with its colortable (which, for gray scale, means the conversion from d to I discussed in §2). The other is a 1-d non-gray scale image of the other quantity, which for now we will call velocity. Let's call these two images $I(x, y)$ and $V(x, y)$, where (x, y) are the image coordinates.

Suppose you look at the V image, which is represented by a 256-element colortable. It has lots of colors and it is uniformly bright because it has no information about I. To insert this intensity information, you make an intensity-modulated V image, which we'll call V_I:

$$V_I(x, y) = \frac{I(x, y)}{I_{max}} \times V(x, y) \tag{4}$$

This combines the two images into a single image in which color represents velocity and brightness the intensity. Note that V_I cannot be represented by a 256-element colortable, so you need decomposed color for a 2-d image.

3.2. Choosing a Color Table

You might think that a natural choice would be the spectrum, from red to blue, representing positive to negative velocities. However, this is an exceedingly poor choice because it ignores the fundamental fact that the human visual system has a very low sensitivity to blue light. Therefore, the brain will perceive all negative velocities, represented by blue, as less intense than the corresponding positive velocities, even though the line intensities are the same or larger.

In other words, you need to account for human physiology in the choice of the color table. If you do this properly, then the sensations of color and lightness will satisfy the desired condition of being nearly orthogonal.

The two images of Figure 2 used this physiologically-desirable colortable. In contrast, the two images of Figure 3 use the rainbow-type colortable. For the left-hand image the blues are easy

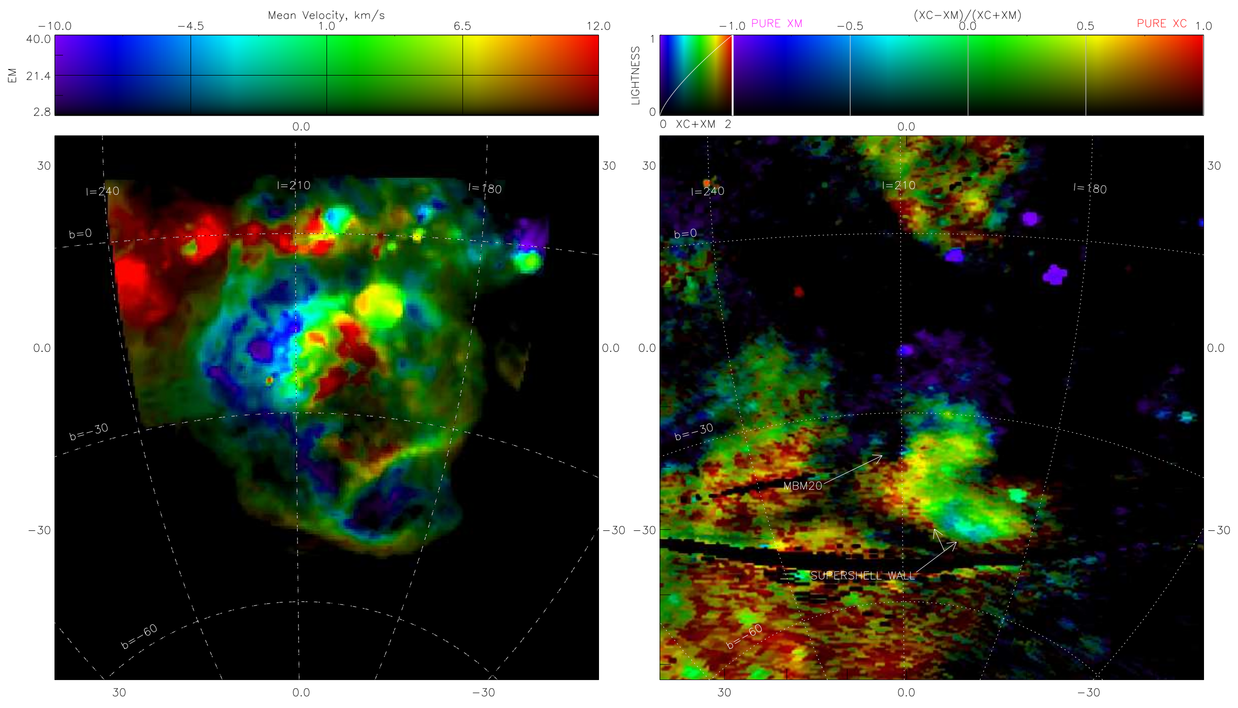

Fig. 3.— The same images as in Figure 2, but with using a physiologically-incorrect rainbow-like colortable.

to see, at least in some areas; this is because they are adjacent not to the black background, but to brighter in other colors. However, in the right-hand image the highest-energy emission, which is located near the center of the image, is dim and lies against the black background; it is barely visible. Using a physiologically-desirable colortable is particularly important for the right-hand panel.

There is a purist version that maximizes orthogonality. However, Tim Robishaw and I have found that a slight revision to this purist concept (namely, removing a bit of red to make more green) gives a color table that has a little more variation in distinguishable color. Figure 4, left panel, shows the physiologically undesirable “rainbow” color table, which contains the dark-looking saturated blue. In contrast, the right panel shows the physiologically orthogonal version: the saturated blue is brightened by adding other colors to make it a brighter pastel color.

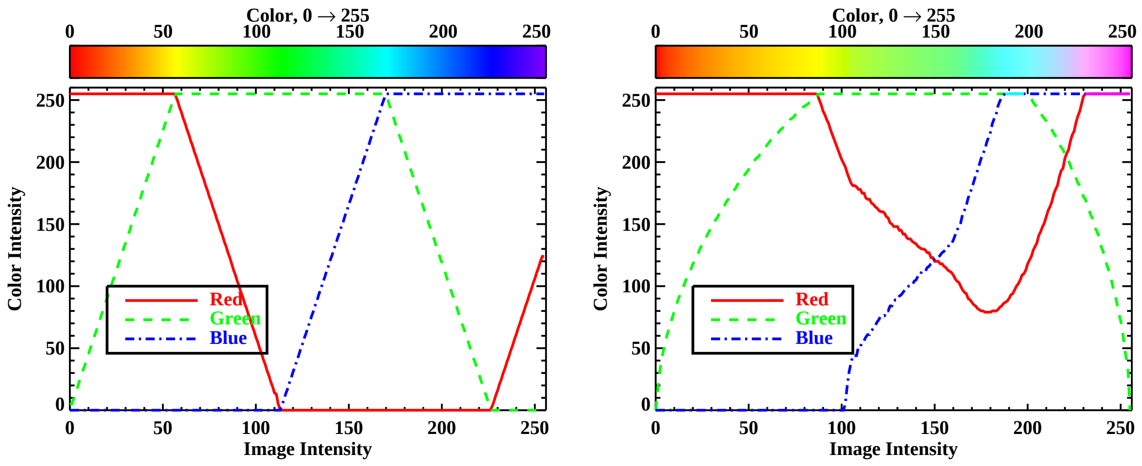

Fig. 4.— Left, a "rainbow" color table; Right, a psychologically correct color table in which all colors are perceived to be of approximately equal brightness. This plot is best viewed in color!

3.3. Generating the color image

Having defined the color table, generating the color image is easy if you have the image of velocities; the velocities V range between two extremes. The velocities of interest have a smaller extent, and you choose this range to span the full color range. Call the limits of this range V_{min} and V_{max}. A common mistake is to make V_{min} and V_{max} too far apart, so you're not gaining the full range of colors on the image. This is almost *never* desirable. On the other hand, going the other way, by making V_{min} and V_{max} "too close" together, can be useful for showing gross features. For example, Figure 5 shows the left panel of Figure 2 with the velocity range compressed—from (–10 to +12) km/s in Figure 2 to (–5 to +7) km/s in Figure 5. We see the global structure—left hand on the near side with negative velocities, right side on the far side—but no details.

Be sure to experiment with these values!

3.4. Colorbars for 2-d Images

Without a colorbar, you're not done! The colorbar has to indicate both intensity and velocity. The colorbar is a rectangular area with two dimensions, so you use one for intensity and one for color as in our illustrative figures.

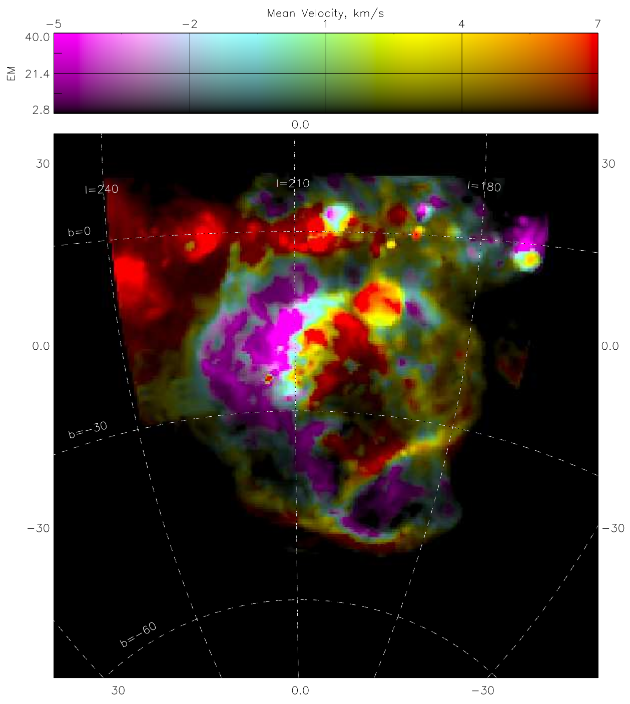

Fig. 5.— The same image as the left-hand panel of Figure 2, but with the colors spanning the smaller velocity range (–5 to +7) km/s.

4. THREE-DIMENSIONAL COLOR: WITH 24-BIT COLOR

Here you want to represent three independent quantities by three independent, orthogonal-to-the-eye colors. Figure 6 shows two examples. On the left panel, red represents the Hα line intensity, green the 21-cm line intensity, and blue the XM band X-ray intensity; on the right panel. green and blue are reversed.

These two images strike one's brain differently. On the right panel, details of weak HI—represented in blue—are hard to see because these weak features are projected against the black background. Not only is blue on black hard to distinguish, but it's especially hard to see detailed structure with blue on black. And the XM image, which is in green, dominates the conception, which is probably not what you want (unless you really want to emphasize the XM angular distribution!). In contrast, the left hand image shows the XM emission well because this emission is mainly concentrated in one blob; and it shows the HI emission well because green shows up well against the black background. It really matters for which image you use blue! Experiment!

You can use two color schemes. One is the obvious red, green, blue; these correspond directly to

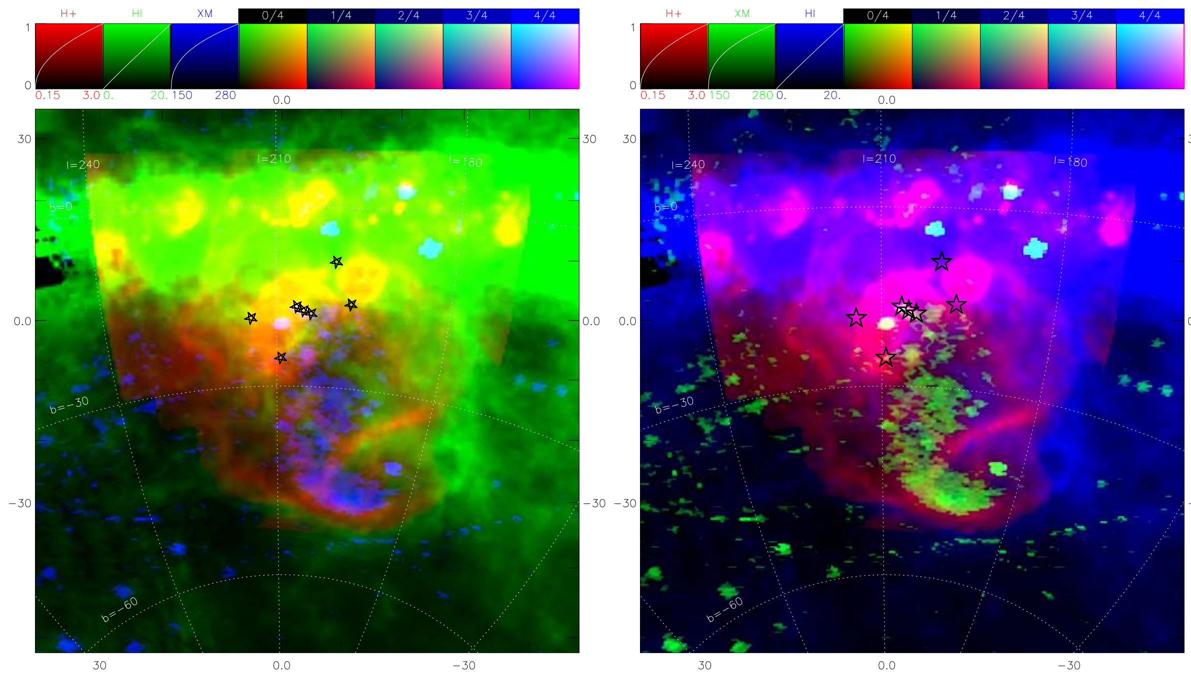

Fig. 6.— Examples of a three-dimensional color image in which red, green, and blue represent different datasets. The representations of HI and XM by green and blue are reversed in the two panels.

the three cones of sensitivity in the eye. Of course, as Figure 6 illustrates, his scheme is susceptible to the poor blue-sensitivity of the human visual system. In principle, instead of using [(r,g,b)] you can use the complements of (r,g,b), i.e. use [white - (r,g,b)] . These are the colors cyan, magenta, and yellow, which are $cyan = (white - red) = (green + blue)$, $magenta = (white - green) = (red + blue)$, $yellow = (white - blue) = (red + green)$. This color scheme like a reversed mapping, for which the image gets darker when the astronomical data values are higher. This endows this color scheme with a highly unusual property: regions with low signal are bright and have well-defined colors; regions with high signal are dark with poorly-defined colors. So this is the scheme of choice when displaying the velocity of weak features is important, and the velocities of strong features don't matter.

4.1. Colorbars for 3-d Images

Without a colorbar, you're not done! Colorbars for 3-d images are hard because paper and images are 2-d by nature, while the colorbar has to indicate intensities for 3 colors and how these colors appear to the eye with all of their different combinations. We have attacked this with the scheme shown in our example Figure 6. Here, a bar across the top of the image is divided into eight squares. The three leftmost squares give the characteristic curves for red, green, and blue, and are labeled with the quantity depicted. The five rightmost squares exhibit how the colors mix when combined. In each square, blue has a different intensity, constant within the whole square, running from 0, 0.25 . . . 1.0 of full intensity. Within each square, red is zero at the left and maximum at the right; green is zero at the bottom and maximum at the top. So the top right corner of each of these five squares is full red and green: the first square, with no blue, shows yellow and the last, with full blue, white.

5. APPENDIX: MAKING 1D COLOR IMAGES IN IDL

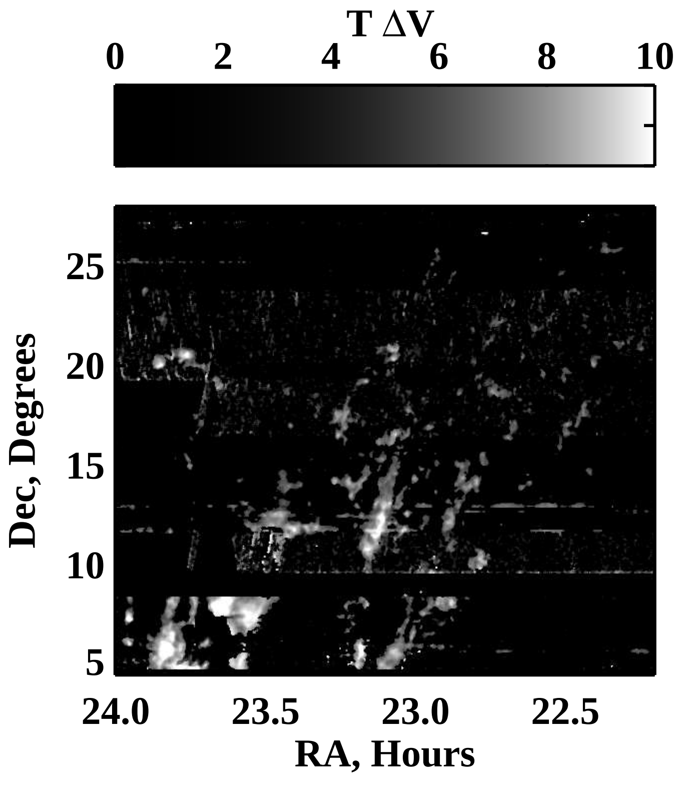

Fig. 7.— The 1-d image produced by the sample IDL code. It's high-velocity clouds.

Here is an example: a close cousin of the IDL code that we used to make Figure 7, which is the intensity part of the 2d example of §6. The input image is named `brtimg`, which get's clipped and contrasted with gamma. We set ps=0 for X window and ps=1 for PostScript.

```
;this is extracted from ~heiles/courses/handouts/images/1dcolor_eg.idl

;CLIP WITH brtmin, brtmax and ADJUST IMAGE CONTRAST WITH gamma...
gamma = 2.5
brtmin = 0.00
brtmax = 10.
brtimg= (brtmin > (brtimg < brtmax))^gamma

;IF YOU WANT PS, OPEN THE FILE...
if ps then ps_ch, '1dcolor_eg.ps', /defaults, /color, xs=6, ys=8

;POSITION THE WINDOWS YOURSELF IF YOU WANT; see text below

;DEFINE THE ASPECT RATIO OF THE IMAGE, WHICH IS 541 BY 470 PIXELS...
aspect=541./470.

;LOAD THE COLORTABLE YOU WANT; NUMBER 0 IS GRAYSCALE...
loadct,0

;DISPLAY THE IMAGE
display, bytscl(brtimg), ra_axis, dec_axis,  aspect=aspect, $
        /xsty, xtit='RA, Hours', xra= reverse( minmax(ra_{axis)), $
        /ysty, ytit='Dec, Degrees'

;DEFINE THE POSITION FOR THE COLORBAR AND DISPLAY IT...
cbarposn = [!x.window[0],!y.window[1]+0.05,!x.window[1],!y.window[1]+0.05+0.10]
colorbar, crange=[brtmin, brtmax], cgamma=gamma, $
     xtit=textoidl('T \DeltaV'), yticks=2,yminor=1, pos=cbarposn

if ps then ps_ch, /close
```

Comments:

1. If your X-window or PS-window is too small, you might get an error message something like

   ```
   % DISPLAY: Normalized POSITION[2:3] must be less than 1.
   % Execution halted at: DISPLAY           731
   ```

This means that there isn't enough space on the window for the colorbar. In this case, increase the size of the X-window (with IDL's `window` command) or the PS-window (using the `xsize, ysize` keywords in the call to `ps_ch` or `psopen`).

2. We used grayscale for this image; you can load any colortable you want, or generate your own.

3. `display` will clip the image, but not adjust for contrast (i.e., with gamma as in §2.3). When gamma is not unity, you have to apply it before entering `display`, and in this case it's easiest to clip the image beforehand as well, as we've done here. And you need to tell `colorbar` what you've done.

4. The `aspect` keyword specifies the aspect ratio. If the x and y axis keyword are specified, the default is to use their min,max values to determine the aspect ratio. This is fine when the units of x and y are the same. Here, however, x is in hours and y in degrees, so we must specify the aspect ratio.

5. We specified the colorbar `position` in terms of where `display` chose to place the image. If you want to specify these positions within the plotting window yourself, take a look at `img_char_posns.pro`. You'd use its output `imgposn` for the `position` keyword in `display`, and you'd use its output `cbarposn` for the `position` keyword in `colorbar`.

6. APPENDIX: MAKING 2D COLOR IMAGES IN IDL

Here is a close cousin of the IDL code that we used to make Figure 8. The input images are `brtimg` and `colimg`, which are clipped and contrasted before being displayed in `display_2d.pro` . We set `ps=0` for X window and `ps=1` for PostScript. The two position axes are `ra_axis` and `dec_axis`.

```
;this is extracted from ~heiles/courses/handouts/images/2dcolor_eg.idl

;DEFINE THE CLIPPING RANGE and GAMMA for brtimg...
;CLIP brtmin WITH brtmin, brtmax and ADJUST IMAGE CONTRAST WITH gamma...
gamma = 1.25
brtmin = 0.00
brtmax = 10.

aspect= 541./470.

;DEFINE THE CLIPPING RANGE FOR colimg...
colmin = -400
```

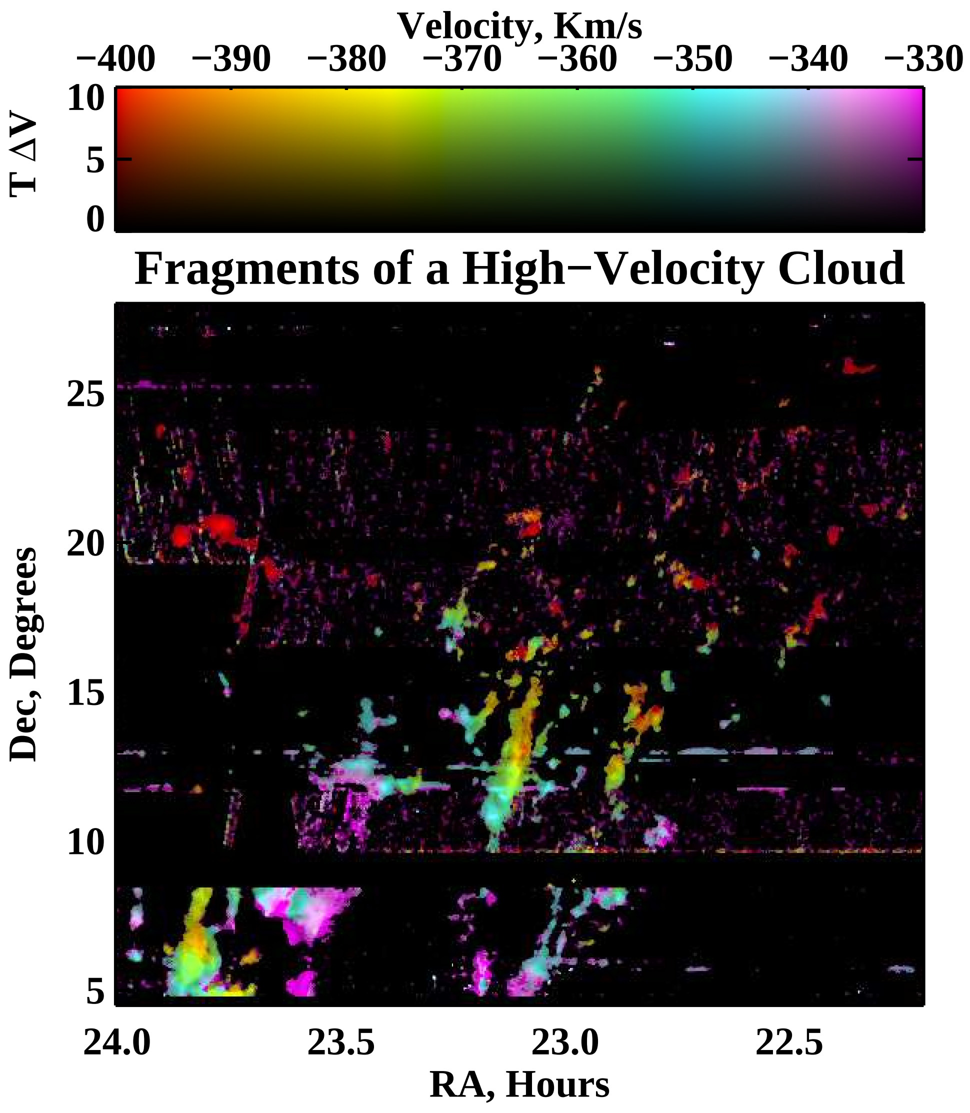

Fig. 8.— The 2-d image produced by the sample IDL code. It's high-velocity clouds.

```
colmax = -330

xtitle='RA, Hours'
ytitle='Dec, Degrees'
cbar_xtitle= 'Velocity, Km/s'
title='Fragments of a High-Velocity Cloud'

if ps then ps_ch, '2dcolor_eg.ps', /defaults, /color, xs=8, ys=10

display_2d, ra_axis, dec_axis, brtimg, colimg, $
```

```
    brtmin, brtmax, gamma, colmin, colmax, $
    xtitle=xtitle, title=title, ytitle=ytitle, $
   cbar_xtitle=cbar_xtitle, aspect=aspect

if ps then ps_ch, /close
```

Comments:

1. In contrast to `display`, `display_2d` will clip the image and also adjust for contrast with gamma as in §2.3). Everything is self-contained, so you don't modify the images beforehand.
2. As above, the `aspect` keyword specifies the aspect ratio.

7. APPENDIX: MAKING 3D COLOR IMAGES IN IDL

We have a clumsily-written but reliable procedure to produce an image like those of Figure 6. A severe restriction is that the image must have dimensions 541×541; you can use `congrid` to convert your image to this size. The procedure is called `rgbimg.pro` and it is reasonably well-documented.

8. APPENDIX: THE IDL CODE FOR FIGURE 4

This is the batch file (invoked with '@') used to create the ps file for Figure 4. Note that `loadct,0` is called after calling `pseud0_ch`!!

```
;USE DESIRED PLACEMENT OF IMAGE IN WINDOW TO CALCULATE POSITION KEYWORDS
FOR THE IMAGE AND THE COLORBAR...
w_left=0.16
w_rght=0.05
w_bot=0.16
w_top=0.14
space=0.03
width=0.1
img_cbar_posns, w_left, w_rght, w_bot, w_top, space, width, $
        imgposn, cbarposn, f_hor, f_ver

;CREATE AN XWINDOW IF PS IS 0...
if ps eq 0 then window,7,xs=f_hor*500, ys= f_ver*400
```

```
;CREATE THE COLORTABLE...
pseudo_ch,colr

;IMPORTANT NOTE: WHEN USING NONDECOMPOSED COLOR (PS OR X WITH
;PSEUDOCOLOR), YOU ***MUST*** INCLUDE THE FOLLOWING STATEMENT; AND
;INCLUDING IT WITH DECOMPOSED COLOR IS OK:
loadct,0

if ps then psopen, 'pseudoplot.ps', xsize=5, ysize=4, /inch, /color, $
  /times, /bold, /isolatin1

;MAKE NICE AXES AND PS FONTS USING SYSTEM VARIABLES...
!p.font=ps-1
!p.thick=6
!x.thick=4
!y.thick=4

;WRITE THE COLORBAR BEFORE CALLING SETCOLORS...
colorbar, pos=cbarposn, crange=[0,255], rgb=colr, color=!black, $
  xtit=textoidl('Color, 0 \rightarrow 255');;;;, ps=ps

;CALL SETCOLORS FOR THE VECTORIZED PORTION OF PS...
setcolors, /sys

;DO THE PLOT...
plot, colr[*,0], /nodata, position=imgposn, /noerase, $
        xra=[0,256], /xsty, xtit='Image Intensity', $
        yra=[0,260], /ysty, ytit='Color Intensity' , color=!black
oplot, colr[*,0], color=!red
oplot, colr[*,1], color=!green, lines=2
oplot, colr[*,2], color=!blue, lines=3

indxcyan=where( colr[*,1] eq 255 and colr[*,2] eq 255, countcyan)
if countcyan ne 0 then oplot, minmax(indxcyan), [255,255], color=!cyan

indxcyan=where( colr[*,0] eq 255 and colr[*,2] eq 255, countcyan)
if countcyan ne 0 then oplot, minmax(indxcyan), [255,255],
color=!magenta

legend, ['Red', 'Green', 'Blue'], lines=[0,2,3],
colors=[!red,!green,!blue], $
        position=[20,100], thick=[6,6,6], textcolors=[!red,!green,!blue]

if ps then psclose
setcolors, /sys

;RETURN EVERYTHING BACK TO NORMAL...
!p.font=-1
!p.thick=0
!x.thick=0
!y.thick=0
```

It is a pleasure to thank Tim Robishaw for teaching me a lot, providing software, and constructive comments.