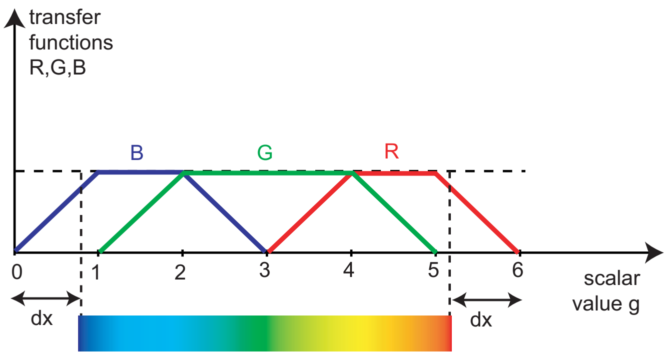

Figure 5.1. Construction of rainbow colormap.

Rainbow colormap. Colormap design is also influenced by application or domain-specific conventions and traditions. For example, many engineering and weather forecast applications use a blue-to-red colormap, often called the *rainbow* colormap (see Figure 5.1). This colormap is based on the intuition that blue, a "cold" color, suggests low values, whereas red, a "hot" color, suggests high values. We can construct a rainbow colormap using three transfer functions R, G, B, shown in Figure 5.1. The rainbow colormap is constructed by the code in Listing 5.1. This constructs three trapezium-shaped transfer functions R, G, and B, ranging from zero to one. The functions are centered at different locations on the scalar value (horizontal) axis, which determines the blue-to-red colormap structure. The functions overlap almost everywhere, which makes the hues in the colormap vary smoothly. The parameter `dx` $\in [0, 1]$ controls the amount of pure blue and red used at the beginning and the end of the colormap, respectively, which gives more aesthetically pleasing results than a colormap containing only mixed hues. In terms of the interpolation theory discussed in Chapter 3, these transfer functions can be seen as piecewise linear basis functions used to interpolate between their corresponding primary colors.

Figure 5.2(a) shows the rainbow colormap applied to a 2D slice from a computer tomography (CT) volumetric dataset.[1] Data values indicate tissue density, with high values corresponding to hard structures, such as bones, and low values corresponding to soft tissues, such as the brain, skin, fat, and muscles. The lowest data value corresponds to air. Using the rainbow colormap, we can relatively easily distinguish the harder tissues (color mapped to yellow, orange, and red), and the air (color mapped to dark blue).

[1]More on CT and medical visualization will be presented in Chapter 10. Slicing is explained further in Section 8.1.2.

```
void c(float f, float& R, float& G, float& B)
{
   const float dx = 0.8;
   f = (f<0)? 0 : (f>1)? 1 : f;            //clamp f in [0,1]
   g = (6-2*dx)*f + dx;                    //scale f to [dx,6-dx]
   R = max(0,(3-fabs(g-4)-fabs(g-5))/2);
   G = max(0,(4-fabs(g-2)-fabs(g-4))/2);
   B = max(0,(3-fabs(g-1)-fabs(g-2))/2);
}
```

Listing 5.1. Rainbow colormap construction.

Rainbow colormaps are one of the widest used colormap types in data visualization, and they are included, and often provided as default, in numerous visualization applications [Borland and Taylor 07]. However, the rainbow colormap has also several important limitations:

- Focus: Perceptually, warm colors arguably attract attention more than cold colors. In our example in Figure 5.2(a), this means our attention is attracted more towards the high data values. This can be desirable if these are our values of interest (goal 4). However, depending on the application, these may not be the values where we want to focus on.

- Luminance: The construction of the rainbow colormap proposed in Figure 5.1 and Listing 5.1 states that all generated colors have the same HSV luminance, i.e., the same maximum of the R, G, and B color components (see Section 3.6.3, Listing 3.2), except for the "tails" of the colormap, where the luminance is slightly increasing, respectively slightly decreasing. However, as explained by Section 3.6.3, Equation 3.23, perceived luminance and HSV luminance are far from being identical. As such, luminances of the rainbow colormap entries vary non-monotonically. This leads to users being potentially attracted more to certain colors than to others, and/or perceiving certain color contrasts as being more important than others. The latter issue can create problems for goal 5. This issue can be corrected by adjusting the rainbow colormap entries so that they use the same hues, but have maximal luminance as being given by Equation 3.23.

- Context: Hues can have application-dependent semantics. The rainbow colormap is based on the assumption that "warm" colors, such as yellow and red, are perceived as being associated with higher data values, whereas "cold" colors such as blue suggest low values. This assumption is based on the interpretation of the rainbow colormap as a heat, or temperature, map. While this might be a good encoding for visualizing a temperature dataset, this association is much harder and more unnatural to do in case of, for example, a medical dataset.

- Ordering: The rainbow colormap assumes that users can easily order hues from blue to green to yellow to red, such as used by this colormap. While this ordering can be learned by a sustained use of the rainbow colormap, we cannot assume that any user will order hues in this particular manner. When this assumption does not hold, goal 2 is challenged.

- Linearity: Besides the colormap invertibility requirement, visualization applications often also require a *linearity* constraint to be satisfied. To explain this, assume, for example, that we visualize a linear function $f(x) = x$ with the rainbow colormap. The result of the visualization is actually identical to the color bar displayed at the bottom of Figure 5.2(a). It can be argued that this colormap, which essentially maps value to hue, is not linear. Some users perceive colors to change "faster" per spatial unit in the higher yellow-to-red range than in the lower blue-to-cyan range. Hence, a linear signal would be perceived by the user as nonlinear, with possibly undesirable consequences for goals 3 and 5.

Other colormap designs. Many other colormap designs are possible. Figure 5.2 shows five such colormaps applied to our 2D CT slice, apart from the already discussed rainbow colormap.

Grayscale: Figure 5.2(b) shows a *grayscale* colormap. Here, we map data values f linearly to luminance, or gray value, with f_{min} corresponding to black and f_{max} corresponding to white. Most medical specialists, but also nonspecialists, would agree that the grayscale produces a much easier-to-follow, less-confusing visualization on which details are easier to spot than when using the rainbow colormap in Figure 5.2(a). The grayscale colormap has several advantages. First, it directly encodes data into luminance, and thus is has no issues regarding the discrimination of different hues. Second, color ordering is natural (from dark to bright), which helps goal 2. For medical visualizations, for example, the black-to-white colormap can be more effective than the rainbow one, as these grayscales map intuitively to the ones visible in X-ray photographs. Finally, rendering grayscale images is less sensitive to color reproduction variability issues when targeting a range of display or print devices. However, on the negative side, telling differences between two gray values, or addressing goal 3, is harder than when using hue-based colormaps. A more subtle problem occurs if our dataset

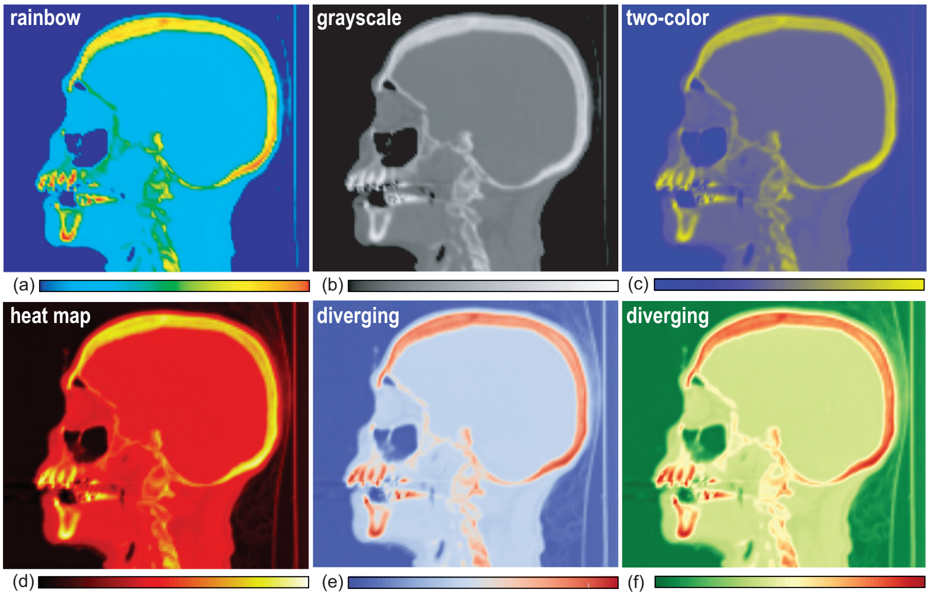

Figure 5.2. Scalar visualization with various colormaps.

to display does not form a *compact* domain (such as the 2D slice in Figure 5.2). Examples of such datasets are scattered 2D point clouds, parallel coordinate plots, or node-link graph drawings, such as the ones shown in Figures 11.32, 11.29, and 11.23 respectively for information visualization applications (Chapter 11). In such cases, we cannot use the grayscale colormap, or for that matter any colormap which includes very bright colors, since such colormaps will make data points appear too similar to the background color, and thus make them potentially invisible.

Two-hue: Figure 5.2(c) shows a *two-hue* colormap. The colormap entries are obtained by linearly interpolating between two user-selected colors, blue and yellow in our case. The two-hue colormap can be seen as a generalization of the grayscale colormap, where we interpolate between two colors, rather than between black and white. If the two colors used for interpolation are perceptually quite different (such as in our case, where they differ both in hue but also in perceived luminance), the resulting colormap allows an easy color ordering, and also produces a perceptually more linear result than the rainbow colormap. Also, in case none of the two colors that define this colormap is too bright, the result is suitable to be used for non-compact data displayed on a white background.

However, a disadvantage of this type of colormap is that it arguably offers less dynamic range: In general, we can distinguish between more hues (such as in the case of the rainbow colormap) than between an equal number of mixes of the same two hues.

The two-hue colormap in Figure 5.2(c) uses both hue and luminance differences of its two defining colors to create a higher dynamic range. However, imagine using this colormap on a 3D shaded surface. In such cases, the luminance perceived in the final visualization is ambiguous, as it can be due to the luminance of the colors present in the colormap, the luminance due to the shading used for the 3D surface, or a combination of both. A simple way to correct this issue is to use a corrected isoluminant two-hue colormap, where all entries have the same luminance, similar to the luminance-corrected rainbow colormap discussed earlier, such as a two-hue colormap generated using red and green as end colors. The disadvantage of this design is that less colors can be individually perceived, leading to challenges for goals 1, 4, and 5.

Heat map: Figure 5.2(d) shows a colormap typically known as a *heat map*, or *heated body colormap*. The intuition behind its colors is that they represent the color of an object heated at increasing temperature values, with black corresponding to low data values, red-orange hues for intermediate data ranges, and yellow-white hues for the high data values respectively [Borland and Taylor 07, Moreland 09]. Compared to the rainbow colormap, the heat map uses a smaller set of hues, but adds luminance as a way to order colors in an intuitive manner. Compared to the two-hue colormap, the heat map uses more hues, thus allowing one to discriminate between more data values. Eliminating the right end of the colormap (thus, using yellow rather than white for the highest data value) allows using this colormap also for non-compact domains displayed on a white background. However, its strong dependence on luminance makes the heat map less suitable for color coding data on 3D shaded surfaces, just as the two-hue nonisoluminant colormaps. Together with the grayscale map, the heat map is a popular choice for medical data visualization.

Diverging: Figures 5.2(e,f) show two final colormap examples, known under the name of *diverging*, or *double-ended scale*, colormaps. Diverging colormaps are constructed starting from two typically isoluminant hues, just as the isoluminant two-hue colormaps. However, rather than interpolating between the end colors c_{min} and c_{max}, we now add a third color c_{mid} for the data value $f_{mid} = (f_{min} + f_{max})/2$ located in the middle of the considered data range $[f_{min}, f_{max}]$, and use two piecewise-linear interpolations between c_{min} and c_{mid}

and between c_{mid} and c_{max}, respectively. Additionally, c_{mid} is chosen so that it has a considerably higher luminance than c_{min} and c_{max}. In Figure 5.2(e), we used a diverging colormap with $c_{min} = blue$, $c_{max} = red$, and $c_{mid} = white$. This can be interpreted as a temperature colormap, where the average value (white) is considered neutral, and we want to emphasize points which are "colder," respectively '"warmer," than this value. In Figure 5.2(f), we used a diverging colormap with $c_{min} = green$, $c_{max} = red$, and $c_{mid} = brightyellow$. Since the maximum luminance of this colormap is lower than that of pure white, we can use this colormap also on non-compact data domains. Diverging colormaps effectively consist of two two-hue colormaps for the left, respectively right, halves of the considered data range. Diverging colormaps are good for situations where we want to emphasize the deviation of data values from the average value f_{mid}, and also effectively support the task of assessing value differences (goal 3).

Zebra colormap: In some applications, we want to emphasize the variations of the data rather than absolute data values (goal 5 of our initial task taxonomy). This is useful when we are interested in detecting the dataset regions where data changes the most quickly or, alternatively, stays constant. For this goal, we could simply use one of the continuous colormaps presented in Figure 5.2 on the magnitude of the gradient $\|\nabla f\|$ of our scalar dataset f. However, this solution implies all the challenges of choosing a "good" continuous colormap which we previously outlined. Additionally, we can only tell the absolute value of our data's rate of change, not the *direction* in which that rate of change is maximal.

A different solution is to use a colormap on the scalar dataset f containing two or more alternating colors that are perceptually very different. When the data values change, the colors change abruptly, yielding easily detectable band-like patterns in the visualization. Figure 5.3 illustrates this design. In the left image, we visualize a scalar function $f(x, y) = e^{-10(x^4+y^4)}$, whose shape is quite similar to the Gaussian shown in Figure 2.1. Instead of a height plot, we now use a grayscale colormap that maps the function value range $[0, 1]$ to 30 distinct shades of gray. The result is a smooth image (see Figure 5.3(a)). Although we can clearly distinguish low values (shown as black) from high values (shown as light gray or white), telling how the data changes within the intermediate data-range region is hard, since we cannot easily tell apart gradients of similar-grayvalue shades. If we use instead a colormap that maps the same value range to an alternating pattern of black and white colors, we obtain the zebra-like result shown in Figure 5.3(b). Thin, dense stripes indicate regions of high variation speed of the function, whereas the flat areas in the center and at the periphery of the image indicate regions of slower variation. We can now also see better