7

2D Scalar Visualization

In this chapter, we will discuss techniques for visualization of two-dimensional scalar fields. This simpler scenario will allow us to introduce several techniques that are useful in more complicated situations. We will start with color mapping, where many of the issues discussed in Chapter 6 will be important to an effective choice of color map. We then proceed to the use of height-fields (or 2.5D visualization as it is sometimes called). Finally, we look at the problem of visualizing multiple scalar fields in a single picture, and the problems that are involved. In this chapter, we will denote the scalar fields we are trying to visualize by $f(x,y), g(x,y)$, *etc.*.

7.1 Color Mapping

By far, the simplest way of visualizing a single 2D scalar field is by giving each different scalar a different color. We create a function $m : \mathbf{R} \rightarrow \text{RGB}$ which maps every value in the image of the scalar field to a color. We call m a *color map*. The critical issue is the choice of m, and we will see that different visualization tasks call for different color maps.

The first visualization task we will discuss is *quantitative measurement*. Specifically, we want to look at a certain point in the picture and be able to tell, as precisely as possible, what is the value of the field there. This entails looking at the point in the scalar field and then comparing the color with a labeled axis that portrays the entire range of scalars in the figure. In chapter 6, we learned that we are quite sensitive to changes in brightness, but in addition, we are quite prone to adaptation: different shades of gray look different compared to their surroundings. This means that an absolute reading of a grayscale image is bound to be inaccurate. At the same time, we know that hue perception is a lot more stable, especially when other psychovisual components such as brightness and saturation are held constant. Because of this, *hue-variant* color maps are more suited for quantitative measurement tasks. Figure 7.1 illustrates the difference in the context of this visualization task: try to determine the exact thickness of the ozone layer above your home town in the two visualizations.

The next visualization task we can imagine with 2D scalar fields is *qualitative measurements*. In this case, we have a set of locations in the map, and we simply

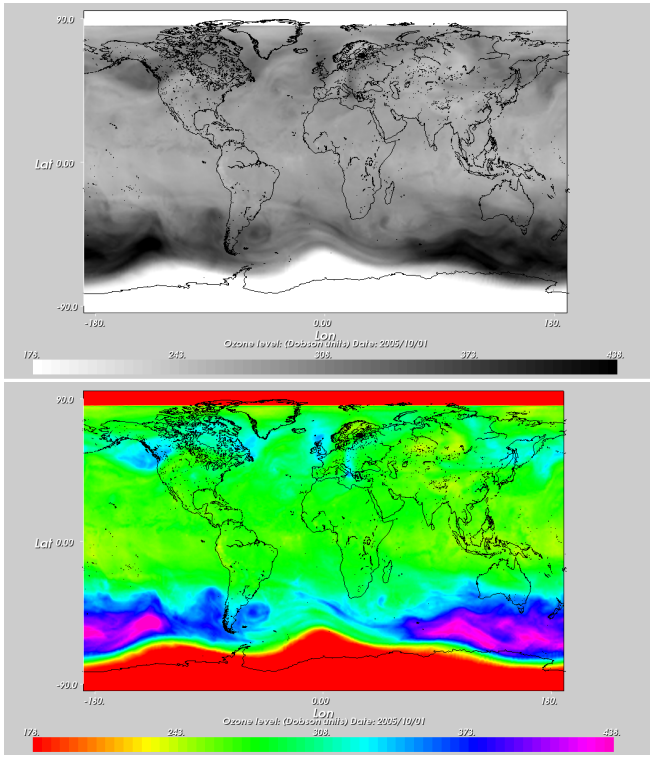


Figure 7.1: For quantitative measurements, a colormap that changes hue (right) is much more effective than one that changes brightness. For qualitative measurements, the opposite is true.

want to order them by the scalar depicted in the visualization. The ease with which we can perform this task depends critically on two things: there being a consistent total order between the colors in the colormap, and this order being preserved by m. The human visual system does not ascribe any consistent ordering among changes in hue — in fact, the wrapping around of hue in the HSV color system is one of its defining features. Brightness, however, has an intuitive ordering that is trivial to preserve in the ordering, so *brightness-variant* colormaps are ideal candidates for this visualization task. Figure 7.1 again provides a good example. Try now to place your hometown's ozone density in relation to (say) New York, US, London, UK and Quito, Ecuador. Notice that with the brightness colormap, you don't need to look at the colormap more than once. This makes it a much more effective visualization.

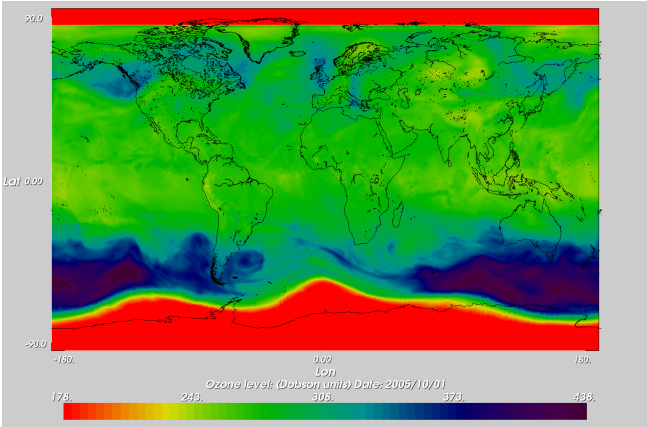


Figure 7.2: A colormap with redundant cues such as the one displayed here allows a wider variety of visualization tasks to be performed efficiently.

7.1.1 Redundant Cues

Our visual system perceives colors along three axes, and scalar fields store a single value at each point in the field. By using only hue or brightness, we are in effect underutilizing our visual ability. It is usually possible to create better visualizations by applying the principle of *redundant cues*.

The principle states that we should generally use as much information channels as possible to convey information. In our case, we have a "hue" channel, a "saturation" channel and a "brightness" channel. Both of the visualizations in Figure 7.1 use a single information channel to convey the ozone layer density at each point. As we have seen, each of these channels is better suited for a particular task. However, if they don't interfere in one another, we can use *both of them* in a single colormap. Figure 7.2 illustrates the result. Notice now that both visualization tasks previously mentioned are easy to perform in this single figure. This principle is applicable in many other situations besides color mapping, and is one very important practical aspect involved in creating effective visualizations.

Some standard colormaps When designing colormaps, visual phenomena that we encountered before need to be taken into account, like the Bezold-Brücke phenomenon. This is especially important in the case of colormaps that change both color and hue, since changes in hue sometimes cause a change in perceived brightness and vice-versa. Some standard colormaps have been designed to account for these issues, and they are presented in Figure 7.3. These are a reasonable first choice in colormaps you should consider before trying anything else.

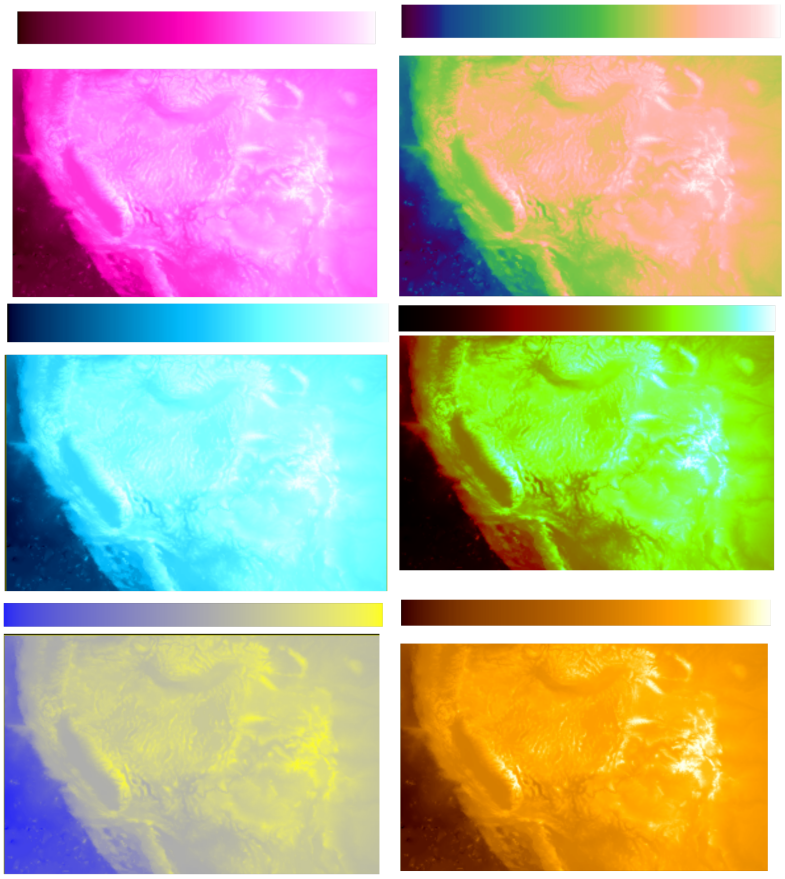
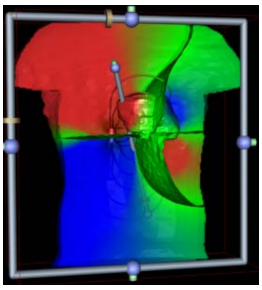


Figure 7.3: Some standard colormaps. (FIXME: figure out how to do this sans subfigures later). Optimal is Levkowitz's optimal color scale [?]; Rainbow is Pizer's rainbow color scale [?] .



Cultural Issues Color can be used to great effect in scientific visualization. However, colors tend to have meaning attached to them that is entirely cultural in nature. Because these connotations change in people with different backgrounds, sometimes a colormap design can "backfire". For example, the almost universally accepted "green good, red bad" colormap is *not* universal. This is particularly true in situations where experts are highly trained, and used to a particular color scheme. The side figure shows an example of a colormap where green is neutral, red is positive, and blue is negative. It is very important, then, to be aware of such cultural issues when designing a visualization.

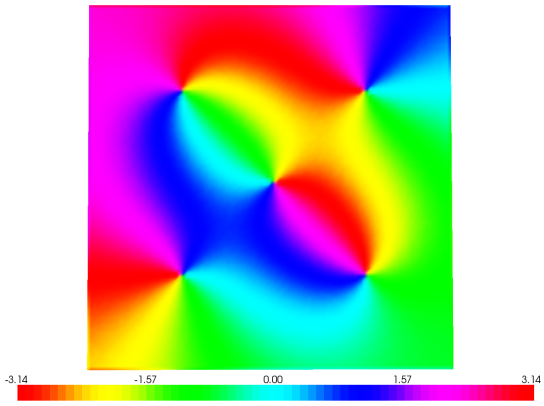
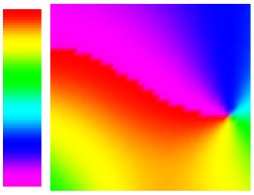


Figure 7.4: A colormap should respect the topology of the elements in the field. When visualizing phases, for example, it should always be the case that $m(-\pi) = m(\pi)$.

7.1.2 Continuity, bijectivity

Some datasets have fields whose values are not strictly scalars, even if the value is encoded by a real number. For example, we could measure the phase of a periodic phenomenon in every point in space. In this case, 0 and 2π should be assigned the same value, since the two phases are equivalent. Direction is another example of this type of data. The basic issue here is one of *topology*: m must preserve the topology of f (technically, m should be a *homeomorphism*). Intuitively, values that are close to one another in the set of original values should still be close to one another in the colormap.

One attribute of color that has a natural loop in its topology is hue, which makes it attractive for portraying phase information. Figure 7.4 shows an example of a synthetic dataset that stores phase information. Notice how the colors for $-\pi$ and π are the same, which ensures the topology is preserved. Compare this to the small picture on the side, which shows an inset of the same dataset, now visualized with a slightly different colormap. Even though the colormap is still one-to-one with the set of scalar, the topologies do not match. As a result, a purple-red boundary shows up as an artificial seam.



Sometimes we find figures created using colormaps specifically designed not to be bijective. Typically, these colormaps change the color very rapidly as m traverses the set of possible values. The rationale behind this is that it becomes possible to assess how quickly the scalar field is changing by comparing the speed in color variation. This is illustrated in Figure 7.5. For this visualization task of comparing local gradient magnitudes, you should consider using contours instead. We discuss them in the following section.

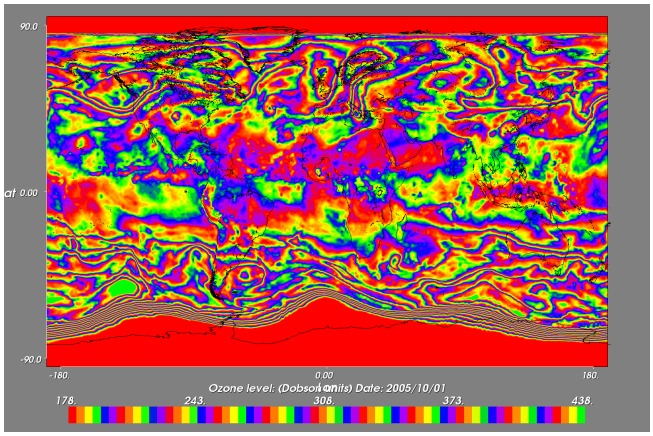


Figure 7.5: Sometimes a colormap is designed specifically not to be bijective. Although this allows us to perceive changes in gradient magnitude (faster color changes indicate higher gradient magnitude), this is a distracting visualization. Section 7.2 describes a more appealing alternative.

7.2 Isocontours

The colormap in Figure 7.5 is distracting, due to its rapidly changing colors. However, on closer inspection, there are features that provide extra information about the data being depicted. For example, in places with high gradient, our eyes naturally follow the lines that form with the same color (for example, the lines that run east-west north of Antarctica, and to a lesser extent the lines that run east-west on the North American Pacific Northwest). Instead of using such a cluttered visualization, we can formalize the notion and use the techniques directly. The lines we are following are a set of points in the visualization where the scalar value remains constant. They are called *isolines* or *isocontours*. Formally, they are the *preimage* of a certain value in the range of the scalar field: $c(k) = x|f^{-1}(x) = k$.

Figure **??** shows how contour lines help enrich a visualization with much of the content that was introduced by the colormap in Figure 7.5. Notice that we can clearly see differences in gradient magnitude by inspecting how close the isocontours are: the denser the contours are, the higher the gradient magnitude. The lines themselves show regions where the ozone density is equal, and are perpendicular to the direction in which the scalar field changes the fastest. In addition, contour lines always enclose regions of similar scalar values, which is hard to see in the colormapped figure alone.

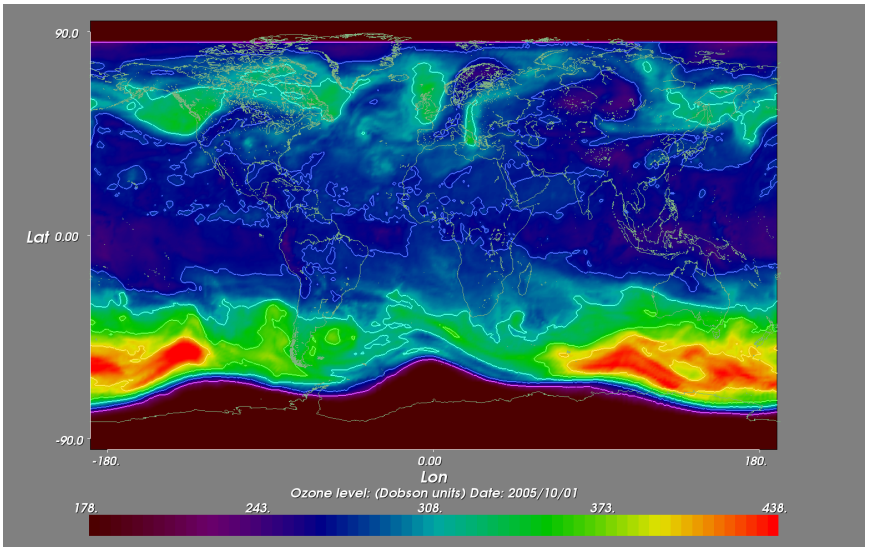
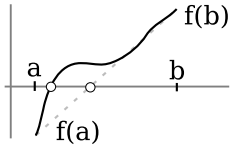


Figure 7.6: Figure **??** enriched with isocontours. Contour lines add extra information to a visualization with relatively little clutter.

7.2.1 Computing Isocontours



This section describes how isocontours are computed for the two most common 2D geometric datatypes: triangles and squares. In this section, we will make extensive use of the intermediate value theorem. It states that given a continuous function of a single variable $f(x)$ and two values a and b such that $f(a) < 0$ and $f(b) > 0$, there exists a root of f somewhere between a and b. We cannot know exactly what is the value of the root or whether it is the only one, so we will settle for an approximate root value — usually given by assuming the function is linear between a and b.

Most algorithms for computing contours are known as *marching methods*. These methods work by inspecting each cell independently of any others (it *marches* through the cells) and applying the intermediate value theorem *on the edges of each cell, by looking at the values at the nodes.*. We will start with triangular grids, which are actually simpler to understand and compute, and then move on to regular quadrilateral grids.

Marching Triangles The method works by inspecting one triangle at a time. The algorithm examines the scalar values at the mesh nodes, and deciding which triangle edges straddle a piece of the desired contour. Figure 7.7 illustrates some of the cases. The algorithm determines that there exists a piece of the contour

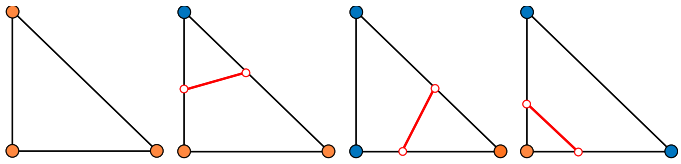


Figure 7.7: Some of the possible cases for marching triangles. Different colors denote different signs.

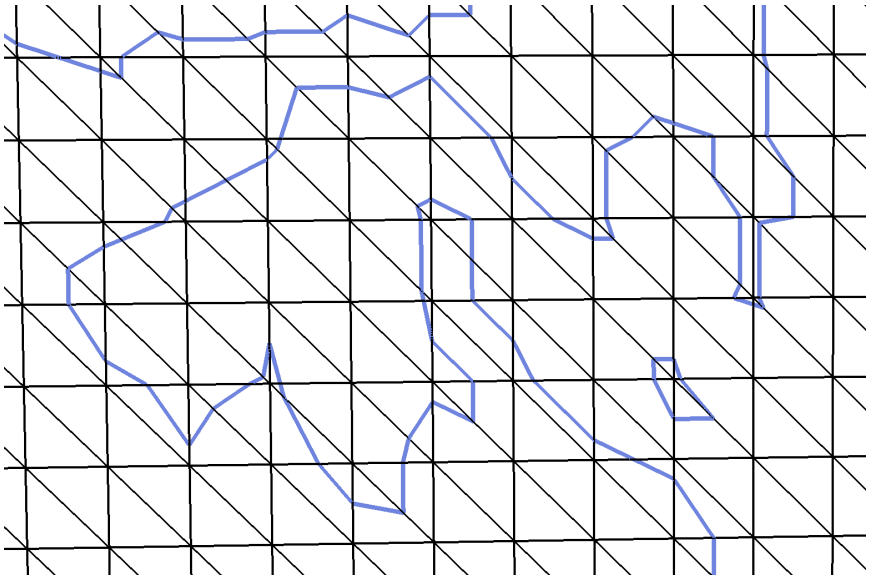


Figure 7.8: Marching triangles applied to the ozone dataset.

crossing the cell at the edge points, and then outputs a line that connects these points.

Even though there are 8 possible configurations for marching triangles, a simple observation reduces the cases by half. Because contours are invariant with flipping the signs of the vertices at all the cells, we only need to look at 4 of them. In fact, every configuration that outputs a line is equivalent, by rotational symmetry. Figure 7.8 shows an example of the output generated by the algorithm.

Marching Squares The marching algorithm for squares is directly analogous to the triangle case. Here, there are more vertices per cell, so it is natural to expect more cases. However, as we will see, there are additional complications. There

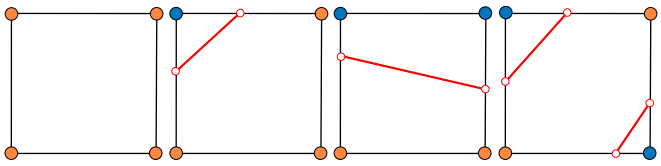


Figure 7.9: The possible cases for marching squares.

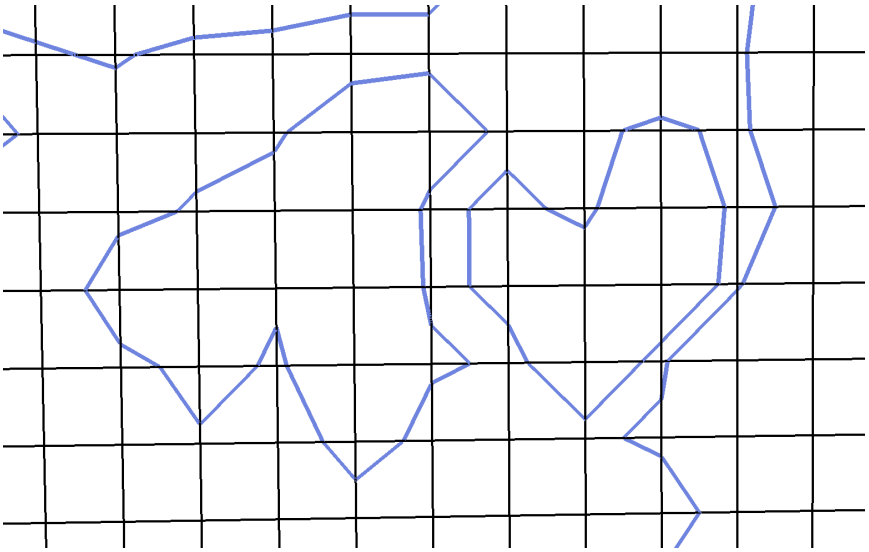
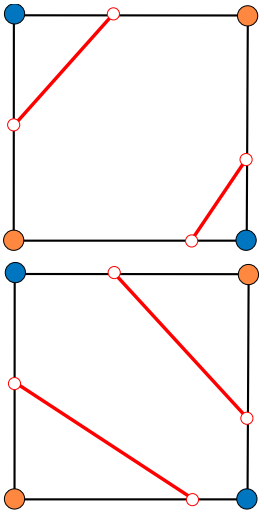


Figure 7.10: Marching squares applied to the ozone dataset.

are 16 possible configurations, but just like with triangles, we can significantly reduce the number of cases. After exploiting symmetry, we end up with four distinct cases, shown in Figure 7.9. Figure 7.10 shows an example output generated by the algorithm.



7.2.2 Ambiguities in marching methods

The attentive reader will have noticed an issue with Figure 7.10. Even though each case is internally consistent, and the set of alternatives produces a plausible contour, the rightmost case is ambiguous. There are two possible alternatives, and we selected one of them arbitrarily. How to choose the correct possibility is what we now discuss. First, however, we consider one alternative that is not, in fact, used in practice.

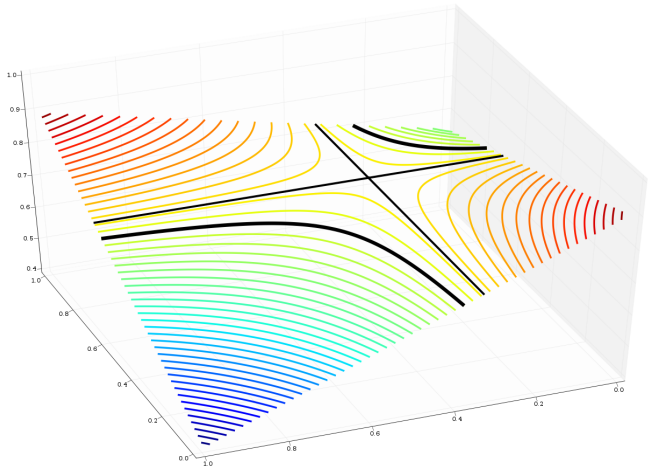


Figure 7.11: Bilinear interpolation generates hyperbolic surfaces, as can be seen by this plot. An example contour is highlighted, and the asymptotes for the hyperbolas are shown as straight black lines. Every contour is a hyperbola, and they all share the asymptotes.

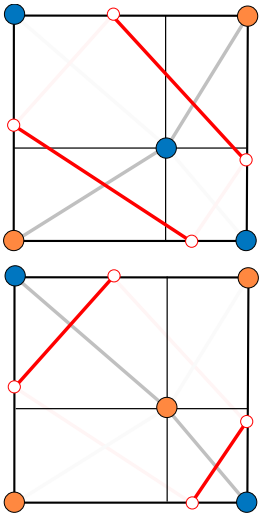
A simple way of fixing the problem would be to split every square into a triangle and then running Marching Triangles on the result. This is not a very good idea for two reasons. First, this generates many more line segments than would be necessary (up to twice as many), so it is an inefficient solution. Second, and most importantly, this does not actually solve the ambiguities: the choice involved in splitting the square is exactly the same as the choice in the two cases. Both of them are consistent, but if we pick one arbitrarily, we are back to the problem we started with. In fact, Figure 7.8 was created by splitting the squares of Figure 7.10. Notice the contours are markedly different.

The algorithm we present here is called the *asymptotic decider*, for reasons we shall see shortly. It was introduced by Hamann and Nielsen [**?**], and is only one way of solving this ambiguity. It solves the problem by assuming the values are interpolated linearly inside the cell, and then *choosing the case that respects the interpolation*. This is an important point: if the values inside the cell need to respect a different interpolation criterion, then the asymptotic decider cannot give the correct results. For bilinear interpolation, however, the techniques gives a surprisingly simple criterion that decides which of the two cases should be chosen.

The first insight contained in the algorithm is that *Bilinear interpolation generates hyperbolic surfaces*. Also, every contour line inside a bilinearly interpolated cell will be a section from a hyperbola. Finally, all these hyperbola share the exact same asymptote. These can be readily seen in Figure 7.11.

We use this information to determine which of the cases should be used.-

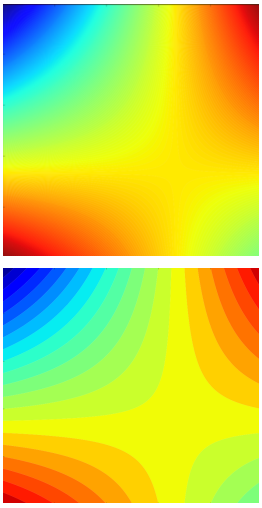
Specifically, we compare the value of the scalar field *at the asymptotes* with the scalar value of the contour we are extracting. Notice from Figure 7.11 that the asymptotes divide the cell into four quadrants: two of those hold the contours above the asymptotes, and the remaining two hold the contours below. This means that the scalar values on straight lines irradiating out of the asymptotic will change monotonically (can you see why?). From this, it follows that if the sign of the scalar value at the asymptote is different than the scalar value at a vertex, the contour must cross a line that connects both. This is enough to decide which case is appropriate.



7.3 Height Fields

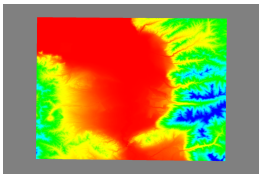
In this section, we will present a different 2D scalar visualization technique that is very simple to implement, and can be effective for certain types of visualization tasks and data. *Height fields* represent data values by how raised (or lowered) they are compared to a base plane. They are sometimes known as 2.5D plots, because they appeal, to some extent, to our three-dimensional intuition.

Height fields are direct analogs of the 2D plots we have investigated in Chapter 5, so many of the same principles will apply. In addition, 2D scalar field contains much more data than a 1D function, so visual clutter plays a central role. Clutter, in fact, will usually determine whether height fields will be effective.



The main reason for using height fields is for giving better contextual gradient information. Height fields portray changes in scalar value as changes in height, so the higher the gradient magnitude is, the steeper the height field "slope" will be at any point. One important information this gives is the constancy of gradients along lines. Colormaps are not very effective at portraying constancy of gradients, because there is no clear-cut way of assigning a constant rate of change in color. In fact, Figure 7.11 is an example of a height field. Notice how it is easy to realize the scalar value is changing linearly across the borders — this is certainly not the case in a purely colormapped image, such as the top one on the side. Sometimes, the same effect can be achieved by quantizing the colormap (bottom one on the side), but we lose resolution and uniqueness, which as we discussed previously, are desirable properties of a colormap.

Another (perhaps obvious) application of height fields is for the visualization of topographical maps. We use the height information to portray exactly elevation changes in regions in a map. This is very effective because of its evocative effect. Our visual system is conditioned to noticing ridgelines and valleys, and using height fields adds much to the effectiveness. Figure **??** shows an example of using height fields for this purpose. Notice that the figure has the same "amount" of information as the visualization on the side, but is much more effective because of the additional cues given by the height field.



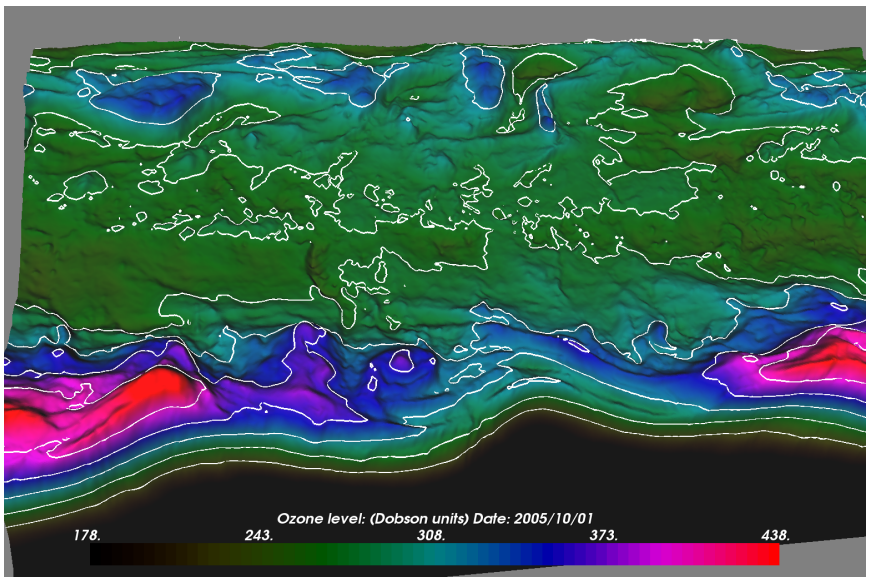


Figure 7.12: Height field visualization of the ozone dataset. Coupled with contour lines and colormaps, height fields can help adding gradient information to a visualization. Because it is partially a three-dimensional visualization, clutter and occlusion become more pronounced, limiting its utility.

Height fields and banking We have already mentioned that height fields are a direct analog of 1D plots. Because of this, the principle of banking applies in the same way as the plotting case. For height fields, this sometimes means that the height is distorted in what might seem an unnatural way. For example, in Figure 7.12, the elevation is actually stretched twice as much as the physical units suggest. Even though this means that the absolute angles are not correctly portrayed, the choice is justified in this case becuase we are using the height field to enhance qualitative measurements (specifically, gradient differences). Absolute angle measurements should not be made directly from heightfield pictures in any case, because the perspective transformation involved will induce errors. The top figure on the side shows a physically correct stretching: the mountain slopes are accurate. The bottom figure is a reproduction of Figure 7.13. Notice that it is much easier to see gradient differences in the bottom figure.

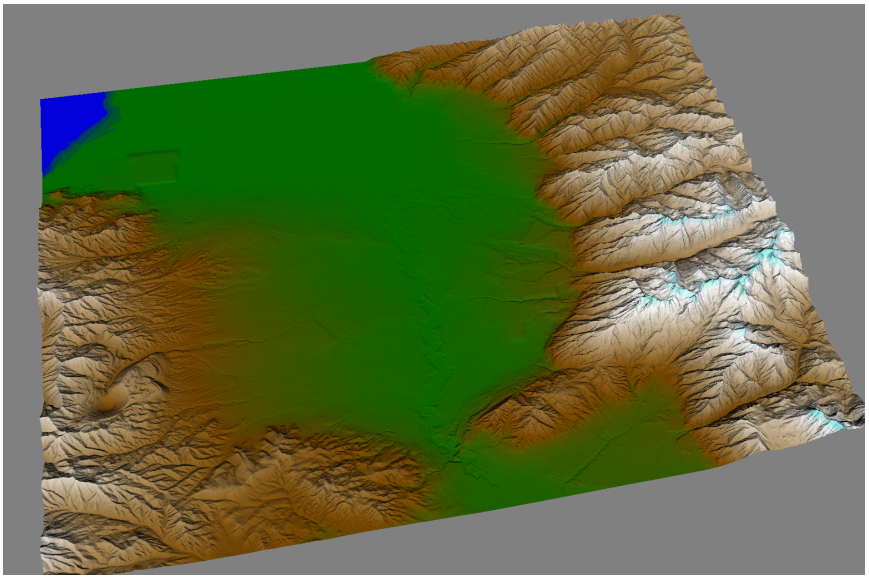


Figure 7.13: A visualization of the Wasatch front in Utah, US. A default colormap is not particularly effective, but by combining a culturally aware colormap with height fields and contours, we get a good visualization.

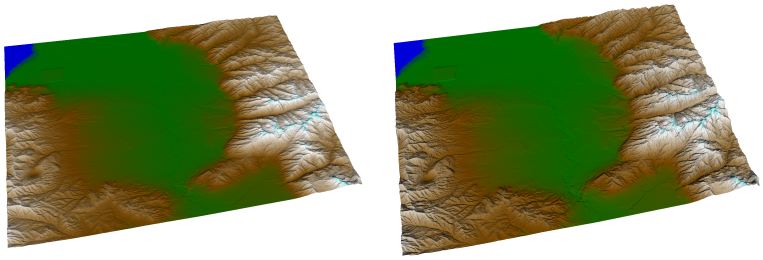


Figure 7.14: Banking vs. unbanking heightfields. Even though the visualization on the left has physically correct stretching and accurate slope angles, the visualization on the right is more effective for telling different slopes apart.

7.4 Multivariate visualization

Sometimes it is necessary to create a single visualization that conveys information about more than one scalar field. In this section, we will investigate mostly the problem of visualizing two scalar fields in a single picture. The reason for this is

that bivariate visualization is already sufficiently challenging on itself, and serves to present most of the issues that will be present in the general multivariable case.

7.4.1 Bivariate colormaps

One of the most straightforward solutions is to design what is known as a *bivariate colormap*. A bivariate colormap assigns a color to each value in the field as a function of *both* values. Since we perceive color information along three (relatively) independent axes, we can choose a colormap that is a surface in a color space, instead of a curve. Our function m then will be $\mathbf{R}^2 \rightarrow \text{RGB}$. There are many possible choices for m, but many of them are inadequate because they fail at least one of the criteria for colormaps we discussed in section 7.1.2. In addition, it becomes difficult to make sense out of a local change in the colors. Using regular colormaps, as a color changes on a line through the field, the values either increase or decrease. Now, a change in colors translates to a 2D vector corresponding to a potential change in any of the two fields. Because the human visual system is inherently bad at telling these local changes in color apart, understanding the figure becomes much more challenging.

FIXME Figures.

Effectiveness One of the biggest drawbacks in multivariate visualization is that it becomes progressively harder to tell what exactly is being portrayed in a figure. Because there is inherently more information being displayed per unit area, there is less room for redundant cues, which, as we have seen before, are an essential component of a clear and effective visualization. Bivariate colormaps should only be used, then, when either the audience is well-trained in their meaning, or when there is no risk of confusion.

7.4.2 Multivariate descriptors

We haven't discussed this in class, so don't worry about it too much. (It's still an important idea, but it won't be in your midterm.)

One way of overcoming the inherent issues in the visualization of multiple scalar fields $f_0, f_1, \ldots$ is to create a new field $f\star = F(f_0, f_1, f_2, \ldots)$ that represents some important relationship between the functions at a certain spot, and visualize those instead. As we mentioned before, one of the hardest visualization tasks using bivariate colormaps is to distinguish differences between the gradients at a point. It turns out that there exists a simple derived scalar field we can compute that gives a good indication of this, proposed by Edelsbrunner *et al.* [**?**]. We restrict the discussion to two scalar fields in 2D, and refer the reader to the paper for the more general case.

The idea behind the descriptors is that they describe how different the gradients are at any point. Given two scalar fields f_0 and f_1, we can compute their gradients ∇f_0 and ∇f_1. To determine how they are related, we compute the *norm*

of the cross product between the two gradients. This will be highest whenever the gradients are exactly perpendicular to one another, and zero when the gradients exactly align.

These descriptors are useful to show the local similarities and differences between two functions, but they require additional training on the receiving end of the visualization, since they have not yet seen widely adoption. We believe, however, that they provide a more effective means of comparison than a bivariate colormap for most situations.

FIXME Figures

7.5 Problems

1. Show the assertions made in Section 7.2.2: namely, that contours of bilinear cells are hyperbola, and that they all share the same asymptotes.

2. Show the value of the scalar field at the asymptotes of a general bilinear cell.