Scalar Visualization

Motivation

Visualizing scalar data is frequently encountered in science, engineering, and medicine, but also in daily life. Recalling from earlier, scalar datasets, or scalar fields, represent functions $f:D \rightarrow R$, where D is usually a subset of R^2 or R^3. There exist many scalar visualization techniques, both for 2D and 3D datasets. In this chapter, we present a number of the most popular scalar visualization techniques: color mapping, contouring, and height plots.

Scalar Visualization

Color Mapping

Color mapping is a common scalar visualization technique that maps scalar data to colors, and displays the colors on the computer system. The scalar mapping is implemented by indexing into a *color lookup table*. Scalar values then serve as indices into this lookup table.

The lookup table holds an array of colors. Associated with the table is a minimum and maximum scalar range into which the scalars are mapped. Scalar values greater than the maximum are clamped to the maximum color, scalar values less than the minimum are clamped to the minimum value.

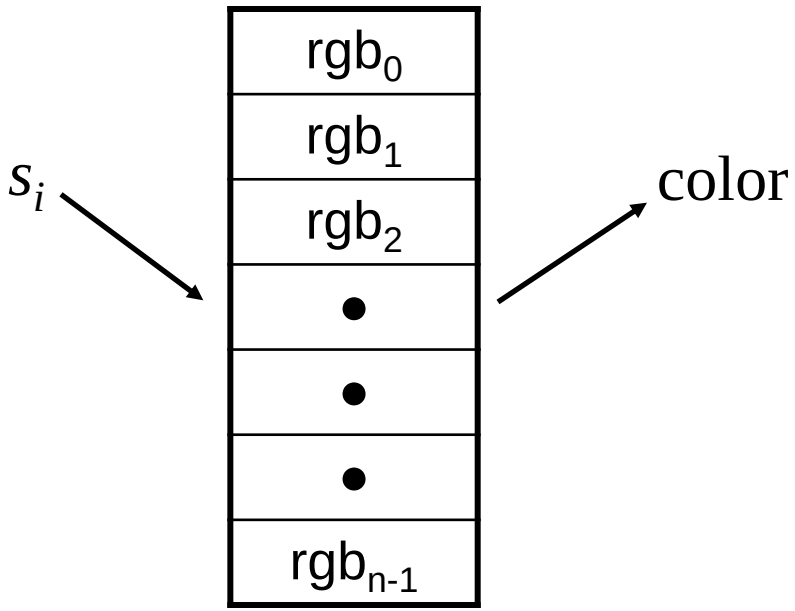
Scalar Visualization

Then, for each scalar value s_i, the index i into the color able with n entries is given as:

$$s_i < \min \quad : \quad i = 0$$

$$s_i > \max \quad : \quad i = n - 1$$

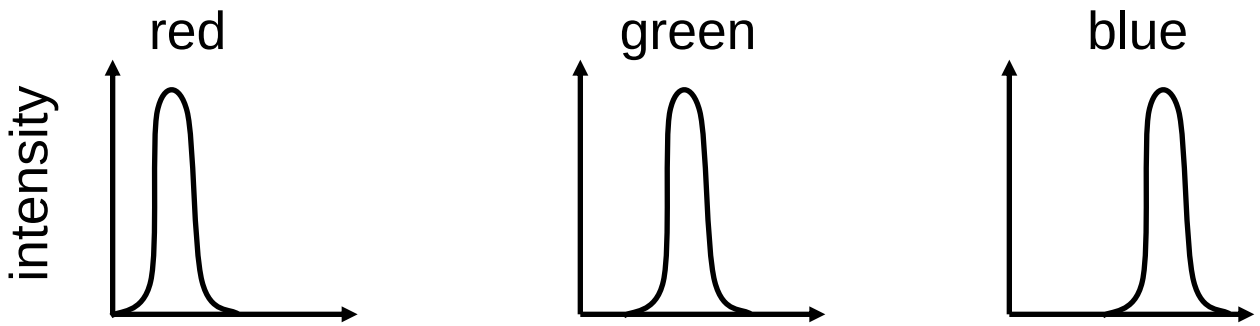
$$\text{otherwise: } i = n \cdot \left(\frac{s_i - \min}{\max - \min} \right)$$



Scalar Visualization

Transfer Functions

A more general form of the lookup table is called *transfer function*. A transfer function is any expression that maps scalar values into a color specification. For example, a function can be used to map scalar values into separate intensity values for the red, green, and blue components.

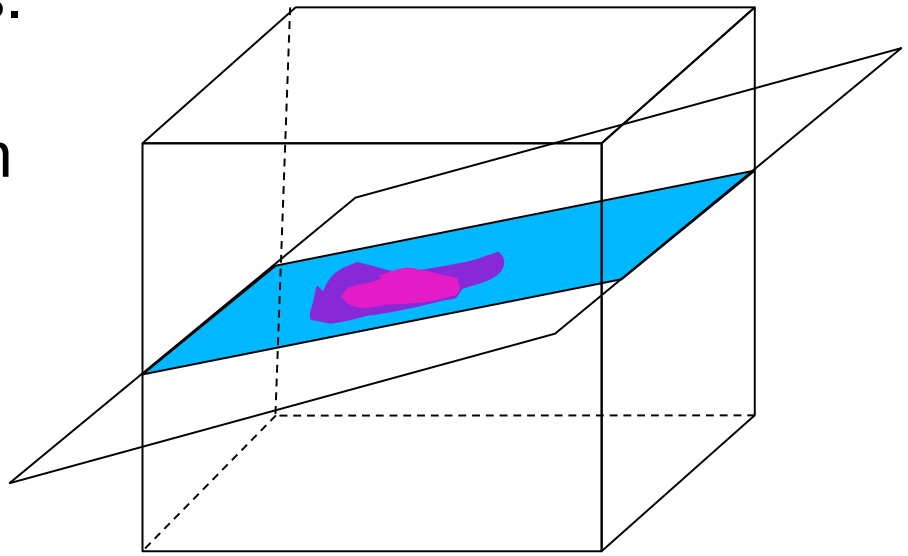


Scalar Visualization

We can also use transfer functions to map scalar data into other information such as local transparency. This will be discussed later when we talk about volume rendering. A lookup table is a discrete sampling of a transfer function. We can create a lookup table from any transfer function by sampling the transfer function at a set of discrete points.

Scalar Visualization

Color mapping is a one-dimensional visualization technique. It maps one piece of information (i.e. a scalar value) into a color specification. However, the display of color information is not limited to one-dimensional displays. Often we use color information mapped onto 1-D, 2-D, or 3-D objects. This is a simple way to increase the information content of our visualization. In 3-D, cutting planes can be used to visualize the data inside.

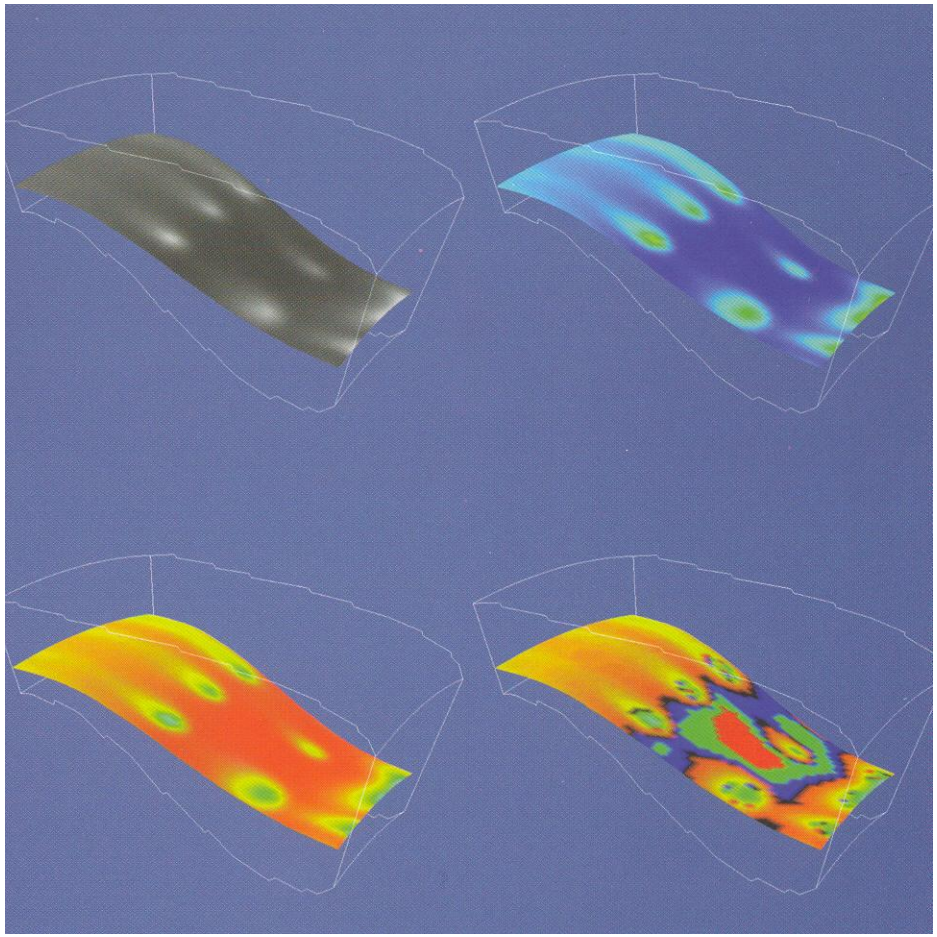


Scalar Visualization

The key to color mapping for visualization is to choose the lookup table entries carefully. Designing lookup tables is as much art as it is science. From a practical point of view, tables should accentuate important features, while minimizing less important or extraneous details. It is also desirable to use palettes that inherently contain scaling information. For example, a color rainbow scale from blue to red is often used to represent temperature scale, since many people associate blue with cold temperatures and red with hot temperatures.

Scalar Visualization

Examples:

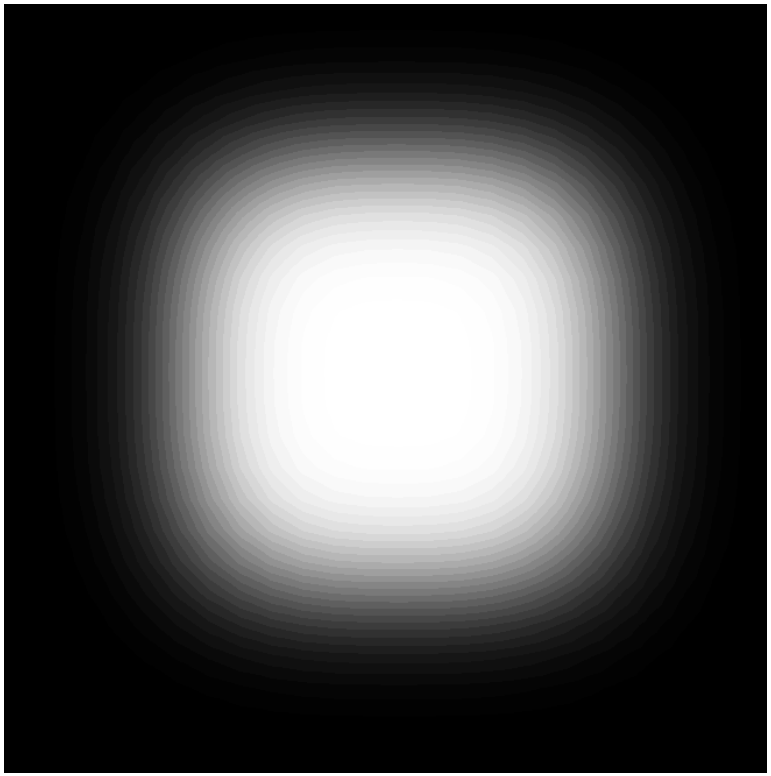


Scalar Visualization

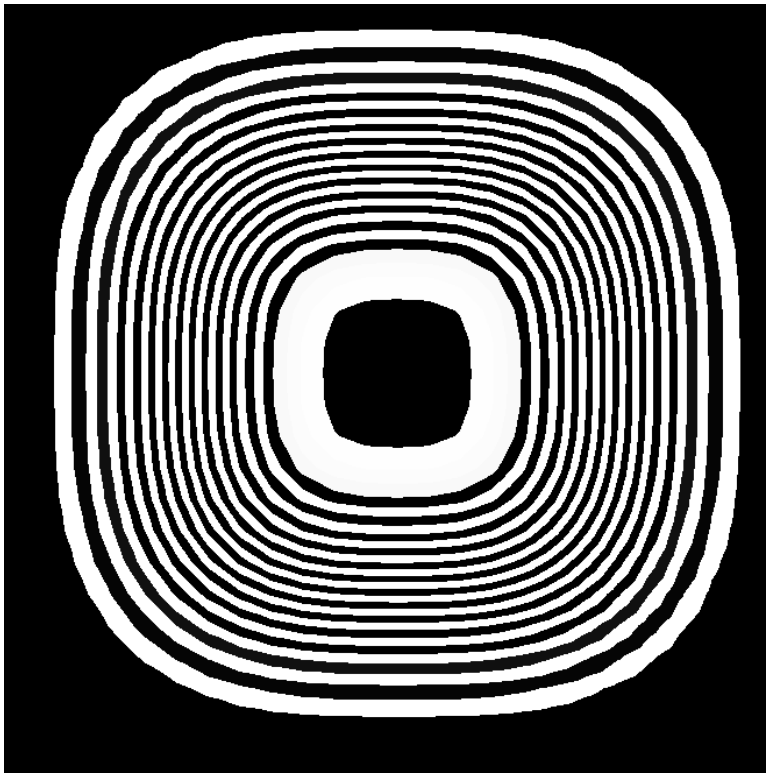
In some applications, we want to emphasize the variations of the data rather than absolute data values. This is useful when we are interested in detecting the dataset regions where data changes most quickly or stays constant. For this goal, we can use a color map containing two or more alternating colors that are perceptually very different. When the data values change, the colors change abruptly, yielding the easily detectable band-like patterns in the visualization.

Scalar Visualization

Example:



luminance color map



zebra color map

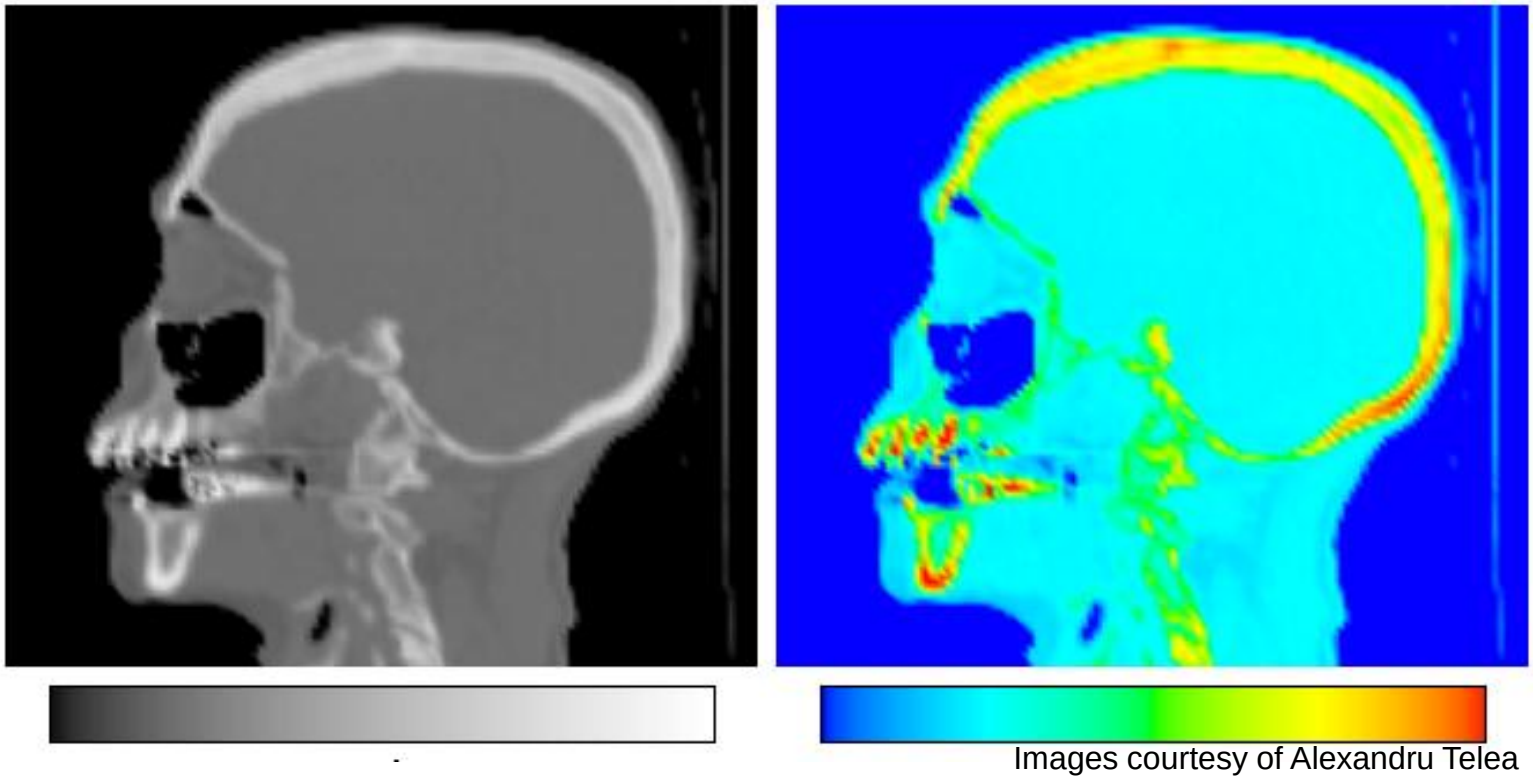
Images courtesy of Alexandru Telea

Scalar Visualization

Many other color map designs are possible. For example, geographical applications often encode landscape height using a particular color map that contains colors, which suggest typical relief forms at different heights, including blue (sea level), green (fields), beige (hills), brown (mountains), and white (mountain peaks). In other applications, such as medical imaging, the simple luminance color map works best. Rainbow coloring would result in loss of linearity due to color values being mapped to hue so that some users perceive the colors to change “faster” per spatial unit in the higher yellow-to-red range than in the lower blue-to-cyan range..

Scalar Visualization

Example:



Images courtesy of Alexandru Telea

luminance color map

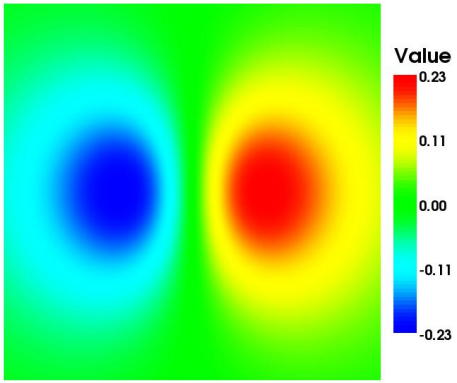
rainbow color map

Scalar Visualization

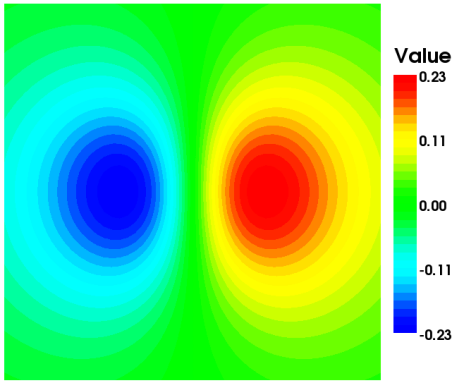
Another important aspect in color map design is the choice of the number of colors N. Choosing a small N would inevitably lead to the **color banding** effect. This is well-known to any computer user who has tried to reduce the number of colors in a color images using image processing programs. Mathematically, color banding produces artifacts identical to undersampling the scalar signal range. Indeed, when we use, for example, just 32 colors to visualize the dataset on the next slide, the effect is practically the same as if we first undersampled the scalar signal to 5 bits and then visualized it via a color map with a high number of colors.

Scalar Visualization

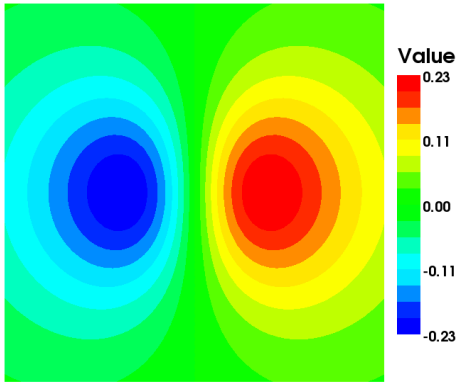
Example:



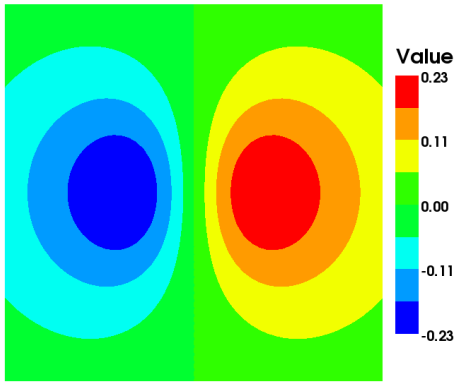
256 colors



32 colors



16 colors



8 colors

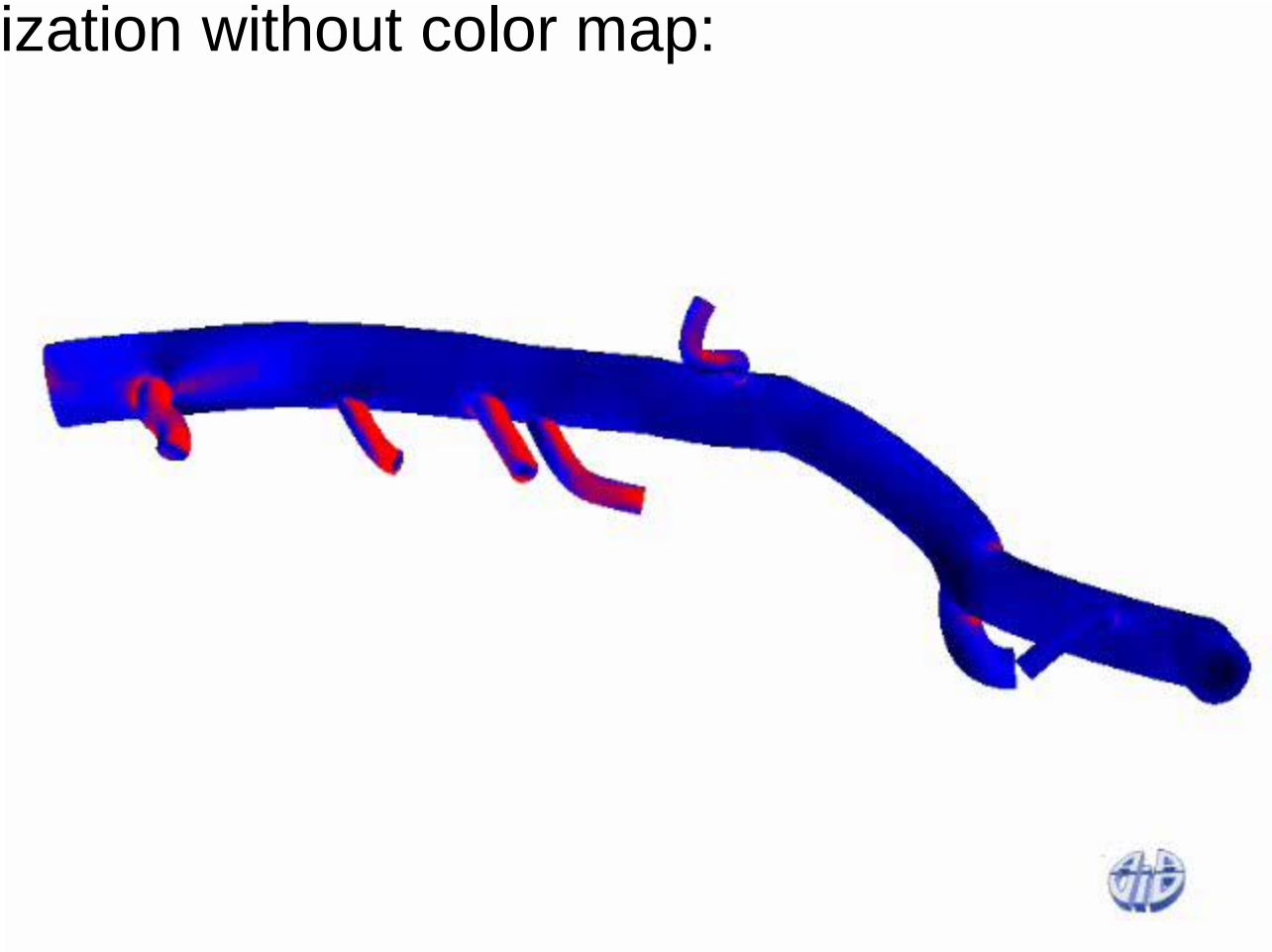
Images courtesy of Alexandru Telea

Scalar Visualization

The previous images also show an important feature that should be included with color maps: a legend explaining how the colors are mapped to the original values. Without this information, the color information can be almost useless as the frame of reference is missing. Providing this context information can help better interpret the image visualizing the data.

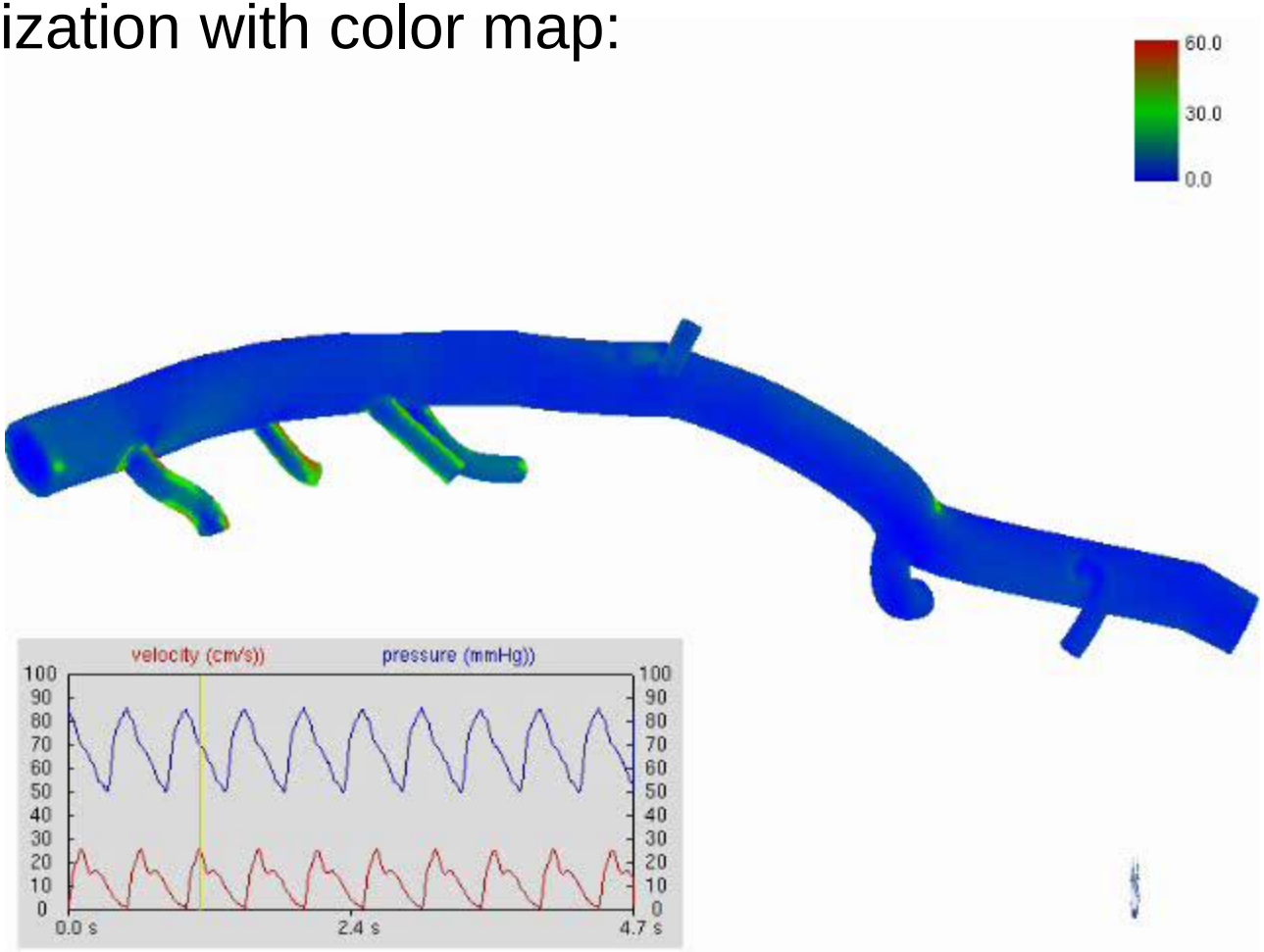
Scalar Visualization

Visualization without color map:



Scalar Visualization

Visualization with color map:



Scalar Visualization

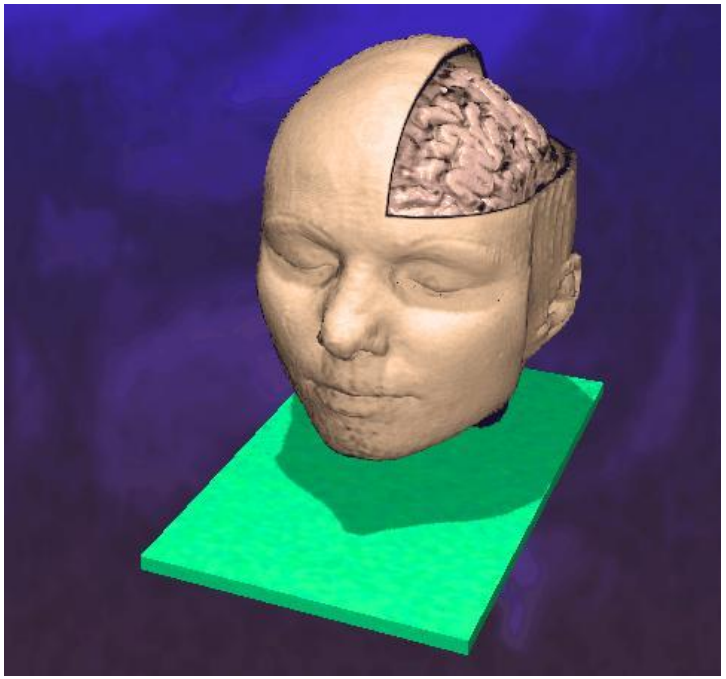
Contouring

A natural extension to color mapping is contouring. When we see a surface colored with data values, the eye often separates similarly colored areas into distinct regions. When we contour data, we are effectively constructing the boundary between these regions. These boundaries correspond to contour lines (2-D) or surfaces (3-D) of constant scalar value.

Examples of 2-D contour displays include weather maps annotated with lines of constant temperature (isotherms), or topological maps drawn with lines of constant elevation.

Scalar Visualization

Three-dimensional contours are called *isosurfaces*, and can be approximated by many polygonal primitives. Examples of isosurfaces include constant medical image intensity corresponding to body tissues such as skin, bone, or other organs. (The corresponding isovalue for the same tissue, however, is not necessarily constant among several different scans.) Other abstract isosurfaces such as surfaces of constant pressure or temperature in fluid flow also may be created.

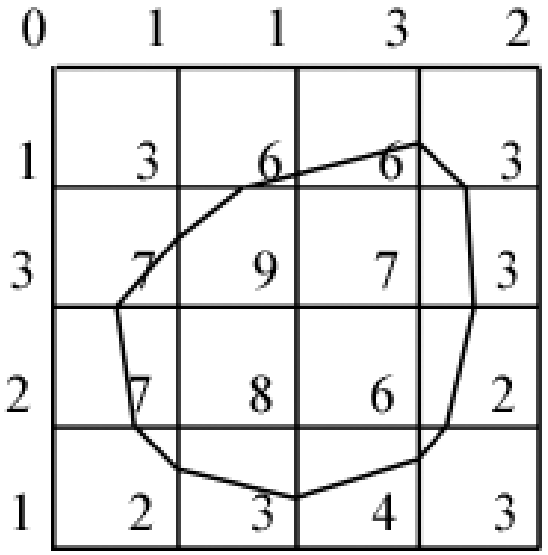


Scalar Visualization

First, we will focus on 2-D contours and how to generate such an isocontour for a given isovalue. Consider a regular grid with scalar values assigned to the grid nodes. Contouring always begins by selecting a scalar value (the isovalue or contourvalue) that corresponds to the contour lines or surface generated. Assuming linear interpolation on the regular grid, we can identify those locations on the edges of the regular grid where the data assumes the isovalue. For example, if an edge has scalar values 10 and 0 at its two end points, and if we are trying to generate a contour line of value 5, then the contour passes through the midpoint of that edge.

Scalar Visualization

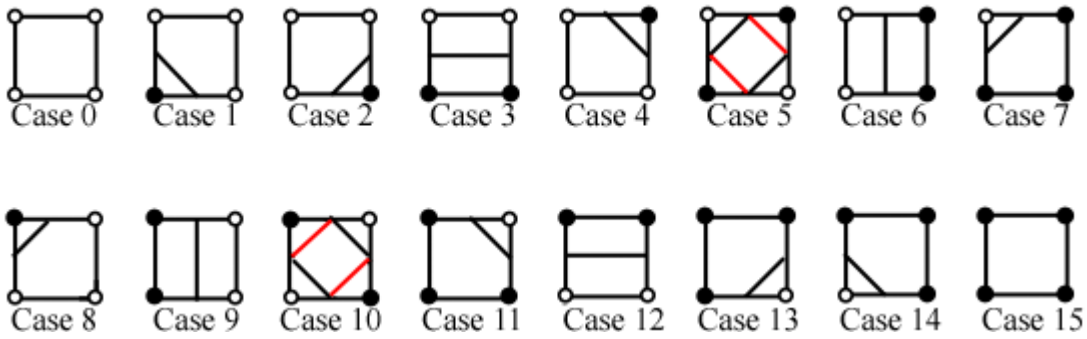
Once the points on all edges are generated, we can connect these points into contours using a few different approaches. One approach detects an edge intersection, i.e. the contour passes through an edge, and then tracks this contour as it moves across cell boundaries. We know that if a contour edge enters a cell, it must exit a cell as well. The contour is tracked until it closes back on itself, or exits a data set boundary.



Scalar Visualization

Marching Squares

Another approach uses a divide and conquer technique, treating cells independently. This *marching squares* algorithm assumes that a contour can only pass through a cell in a finite number of ways due to the linear interpolation used. A case table is constructed that enumerates all possible topological states of a cell, given combinations of scalar values at the cell points.

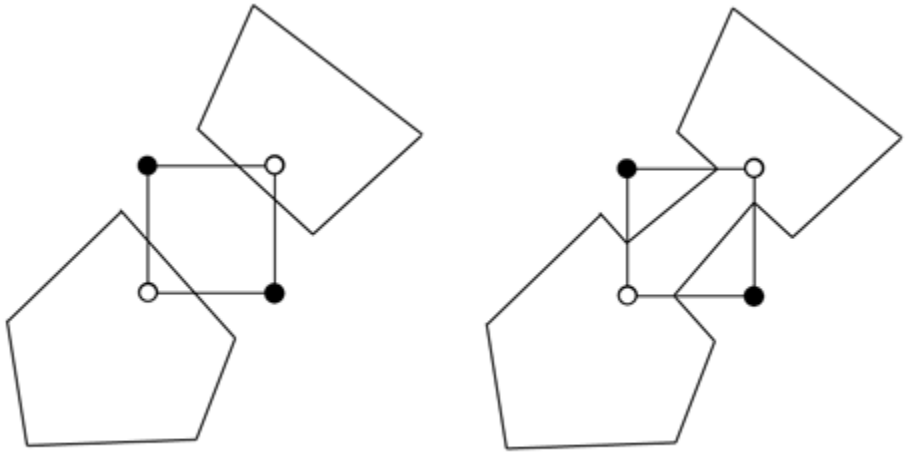


(dark vertices indicate scalar value is above isovalue)

Scalar Visualization

Ambiguities in Marching Squares

While trying this algorithm on different configurations we realize that some cases may be ambiguous. That is the situation for the squares 5 and 10.



As you can see on the previous picture we are not able to take a decision on the interpretation of this kind of situation. However, these exceptions do not imply any real error because the edges keep closed.

Scalar Visualization

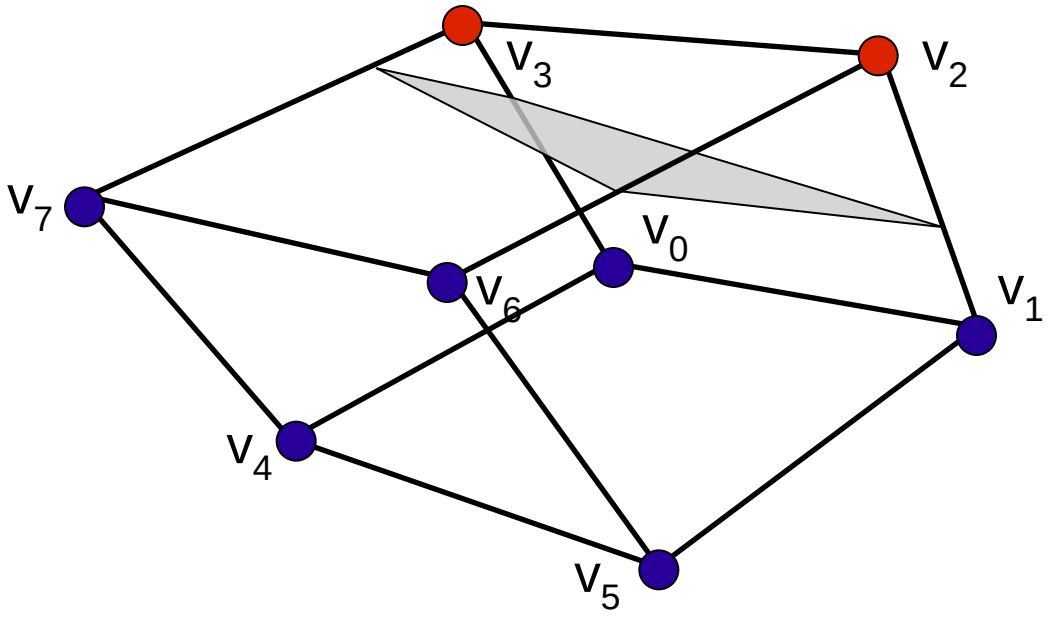
Marching Cubes

Lorensen and Cline introduced Marching Cubes in 1987.

[William E. Lorensen, Harvey E. Cline, „Marching Cubes: A High Resolution 3D Surface Construction Algorithm“, ACM Computer Graphics Vol. 21 No. 4 (SIGGRAPH 1987 Proceedings)]

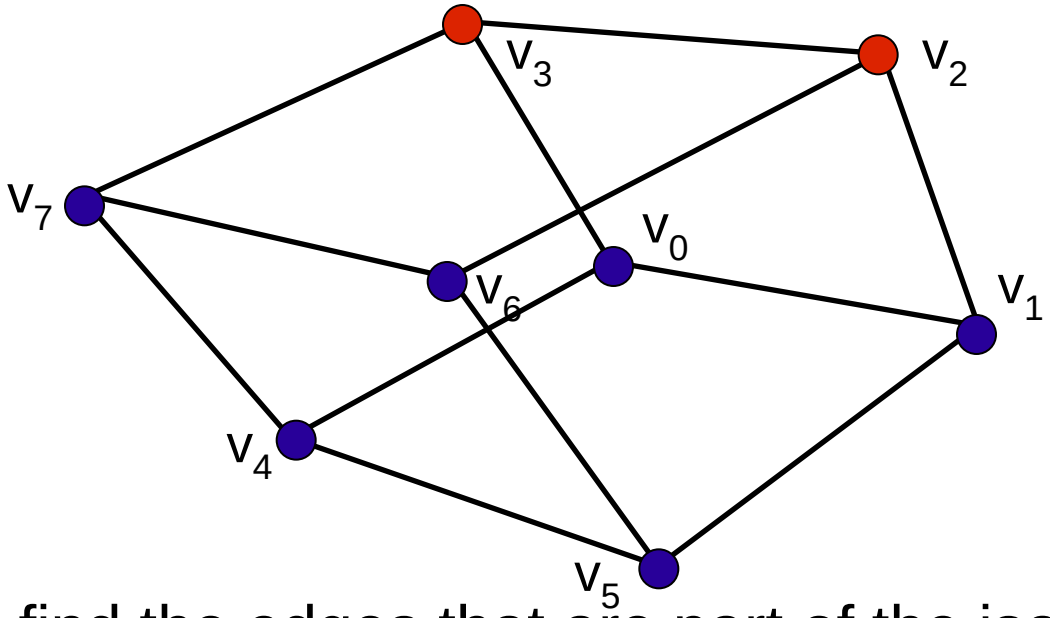
Marching Cubes (MC) is an efficient method for extracting isosurfaces from scalar data set defined on a regular grid. Similar to marching squares, surface segment is computed for each cell of the grid that approximates the isosurface.

Scalar Visualization



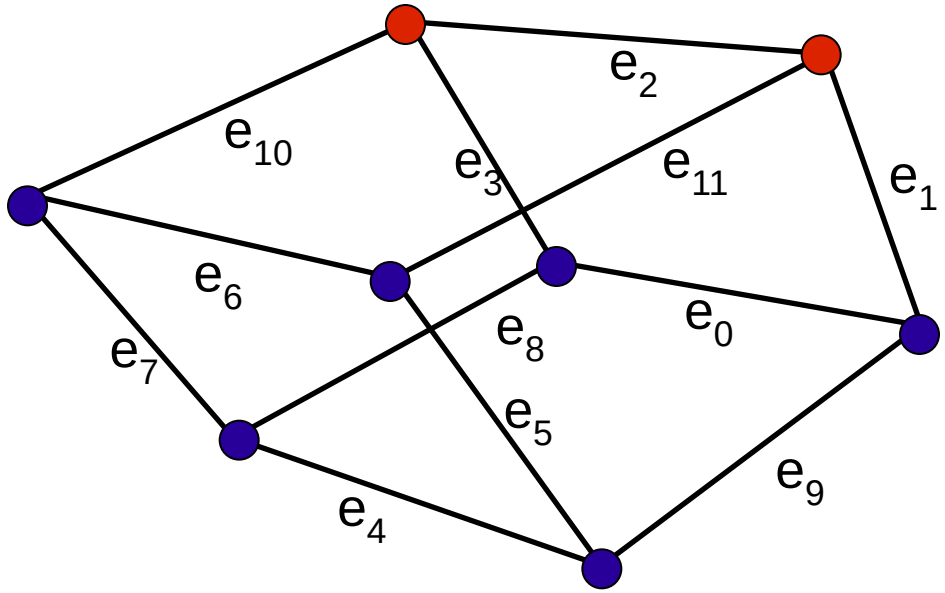
Since the triangulation inside the cell only depends on whether edges exist that intersect the isosurface, we again focus on checking if an edge has values at its vertices in such a way, that one is smaller and one is larger than the isovalue.

Scalar Visualization



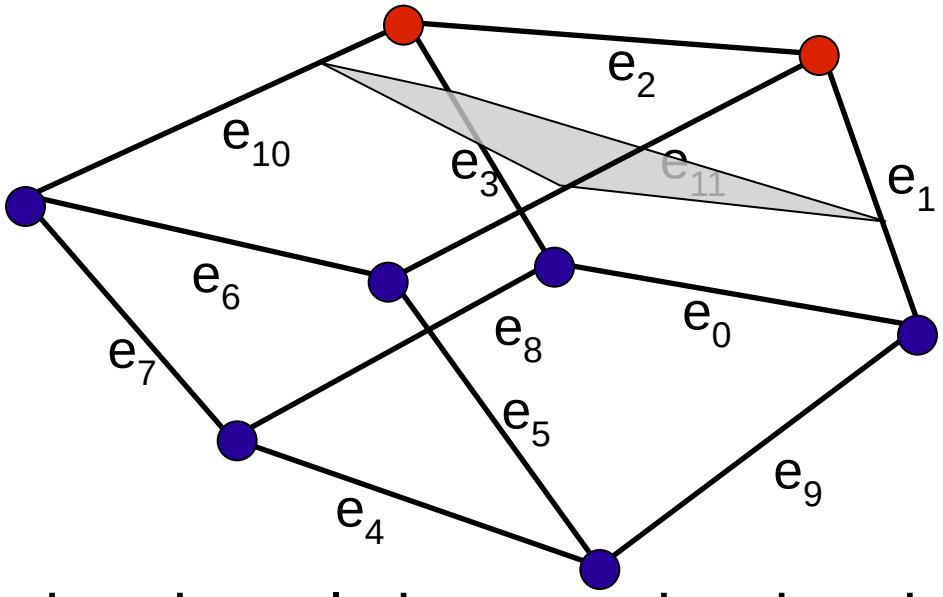
In order to find the edges that are part of the isosurface a lookup table can be used. In order to find the correct entry in this table the vertices are numbered. By setting the corresponding bit for each value larger than the isovalue we get the index referring to the lookup table. There are 256 different combinations possible.

Scalar Visualization



Similarly, the edges are numbered. By generating a bit mask like before using the marked edges we can use the resulting value to point to another lookup table for the triangulation. In the above case the marked edges are 1, 3, 10, and 11.

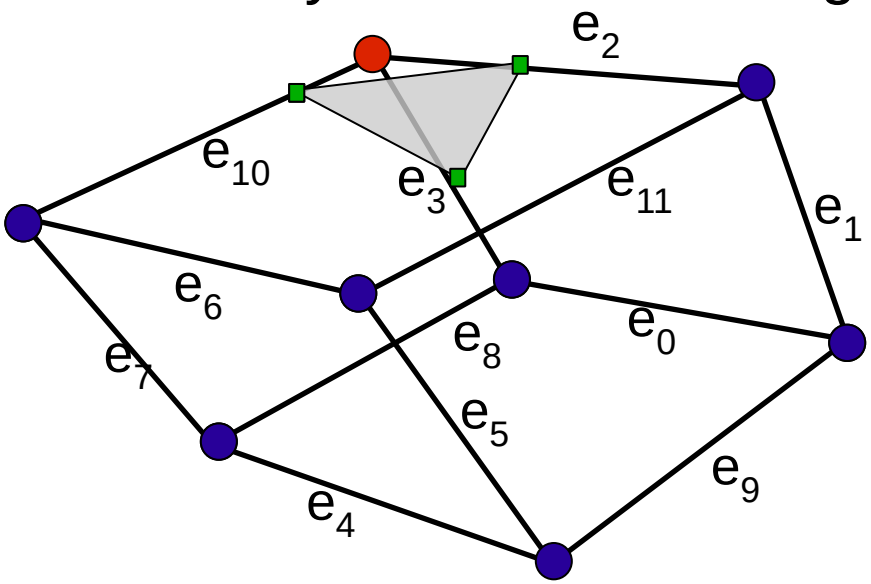
Scalar Visualization



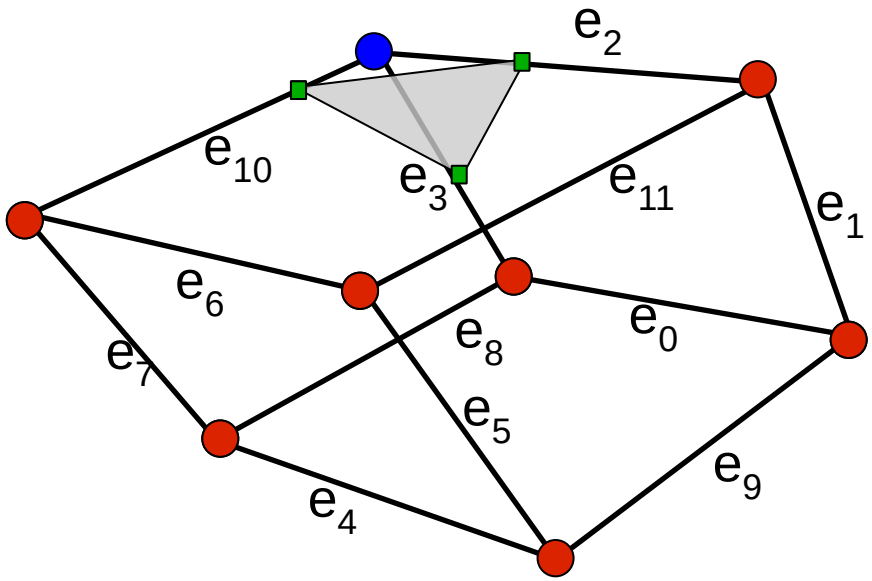
The table for the triangulation contains the triangulations for all 256 cases. Each entry has a list of triangles; the vertices refer to the interpolated points of intersection with the isosurface. The triangles should be oriented mathematically positively for correct backface culling. In our example, the triangulation is $e_1e_3e_{11};e_3e_{10}e_{11}$.

Scalar Visualization

Even though there are 256 possible configurations which have to be triangulated, only 15 of them need to be stored. The remaining ones can be derived from these 16 cases by rotation, mirroring, or inversion.

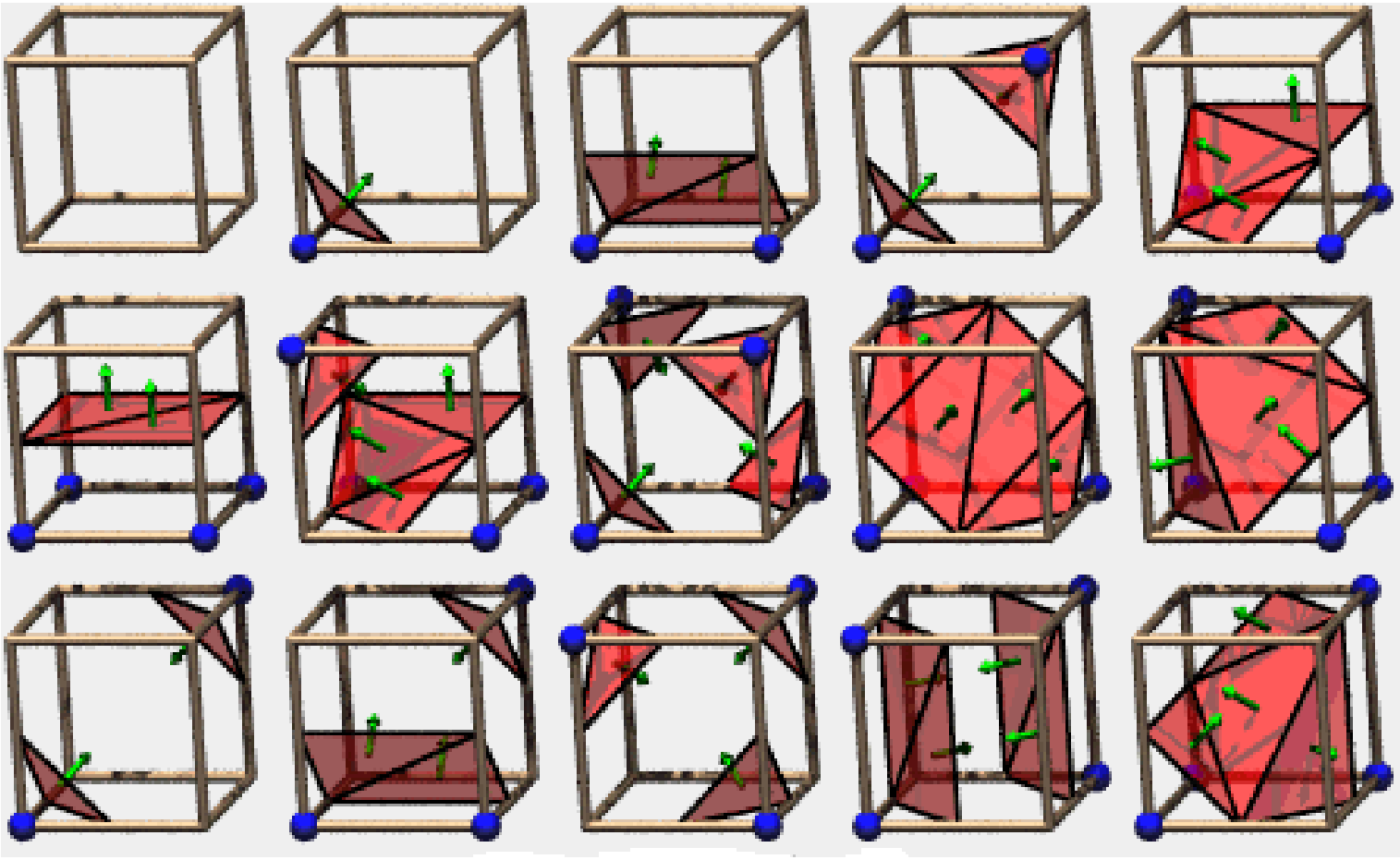


➔$e_2e_3e_{10}$



➔$e_2e_3e_{10}$

Scalar Visualization



All 15 basic cases needed for Marching Cubes

Scalar Visualization

Computation of normal vectors

The quality of the resulting representation of the extracted isosurface can be improved by computing the normal vectors of all vertices. We can exploit the fact, that the gradient of the scalar function

$$\nabla f(x,y,z) = \begin{pmatrix} \frac{\partial f}{\partial x}(x,y,z) \\ \frac{\partial f}{\partial y}(x,y,z) \\ \frac{\partial f}{\partial z}(x,y,z) \end{pmatrix}$$

is always orthogonal to the isosurface. Marching Cubes approximates the gradient at the vertices of the grid as

$$\nabla f(x,y,z) = \begin{pmatrix} \frac{D(i+1,j,j) - D(i-1,j,k)}{\Delta x} \\ \frac{D(i,j+1,j) - D(i,j-1,k)}{\Delta y} \\ \frac{D(i,j,j+1) - D(i,j,k-1)}{\Delta z} \end{pmatrix}$$

and interpolates linearly to determine the gradient at the intersection.

Scalar Visualization

Problems with Marching Cubes

After the article about Marching Cubes was published it turned out that the isosurfaces extracted using Marching Cubes can contain holes under certain circumstances due to ambiguities in the case table. Several follow-up papers exist to fix several issues with Marching Cubes.

Variants

There are variants of Marching Cubes for triangles and tetrahedra as well.

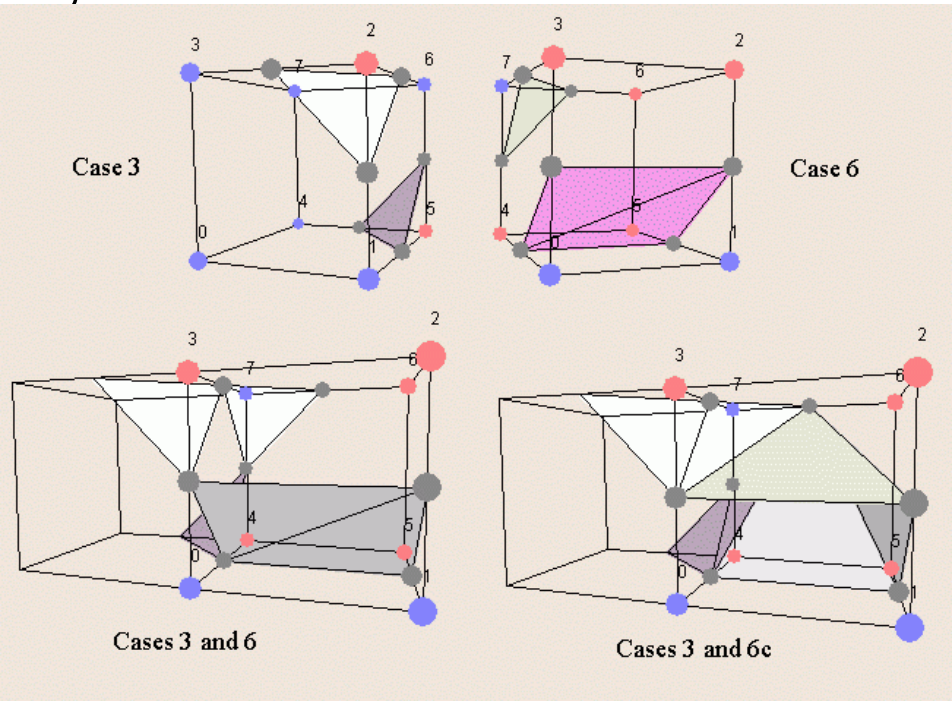
Scalar Visualization

Coping with the Ambiguities

The original set of cases for creating triangles within the cells to generate an isosurface can create holes in some cases. This is basically due to ambiguities, i.e. there are more than one way to generate triangles for some cases. Hence, by introducing additional cases to our case table we can cope with the ambiguities. Obviously, in order to decide which configuration to use, we also have to look at the neighboring cells.

Scalar Visualization

Consider the following two cells. The original marching cubes solution (left) would change the topology of the resulting isosurface, i.e. create holes. By introducing an additional case, this can be fixed.

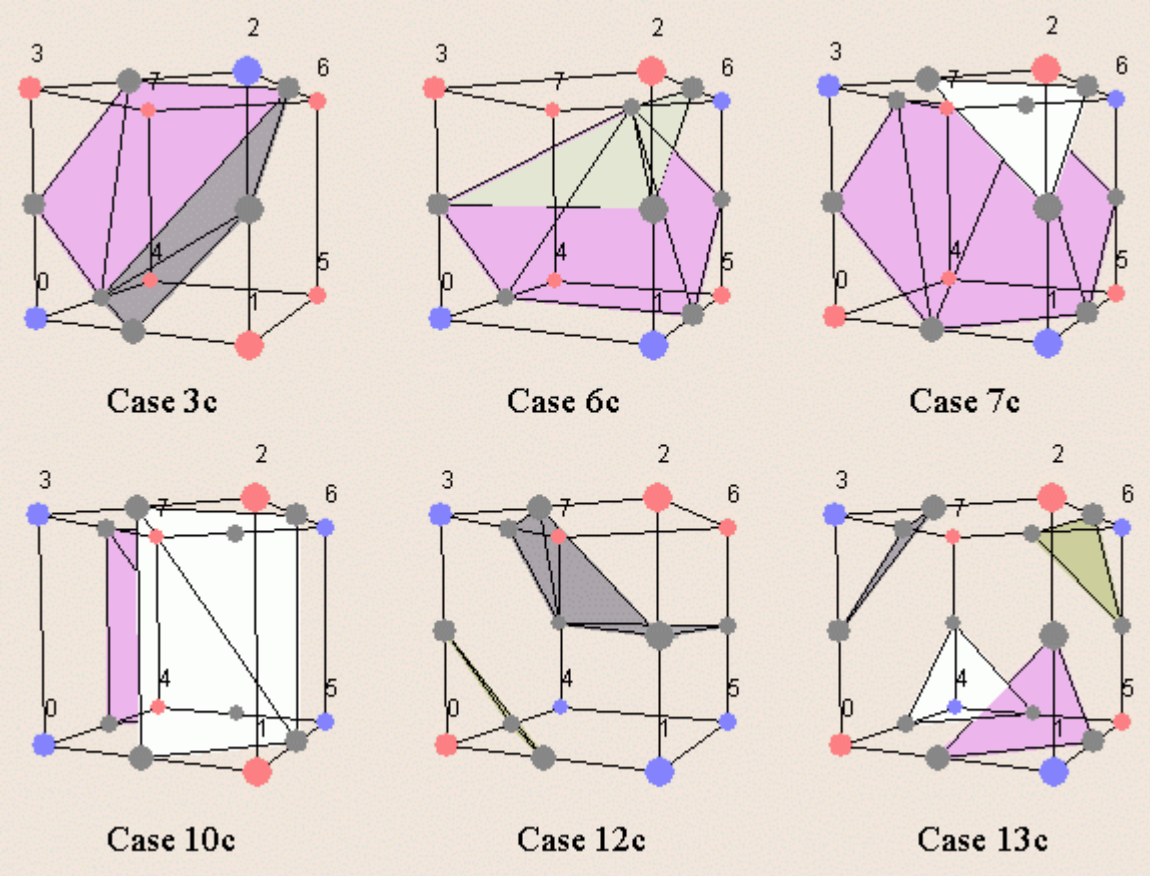


Scalar Visualization

To cope with these topology errors (as holes in the 3D model), 6 cases have been added to the marching cubes cases. These cases have to be used as complementary cases. For instance, in the previous picture, you have to use the case 6c instead of the standard complementary of the case 6. The list of new cases is shown on the next slide.

Scalar Visualization

Additional cases:



Scalar Visualization

We can draw more than a single isosurface of the same dataset in one visualization. Often times this is combined with the use of transparency to allow for several isosurfaces being visible at the same time. The process of rendering several nested semitransparent isosurfaces that correspond to a discrete sequence of isovalues can be generealized to the continuous case, as we will see when we discuss volume rendering.

Scalar Visualization

Example:

The blue opaque isosurface corresponds to a high isovalue, which denotes a hard material such as the enamel present on the tooth upper surface. The beige isosurface corresponds to a lower isovalue, which represents the dentine material inside the tooth. Inside, we can see the tooth nerve chamber.

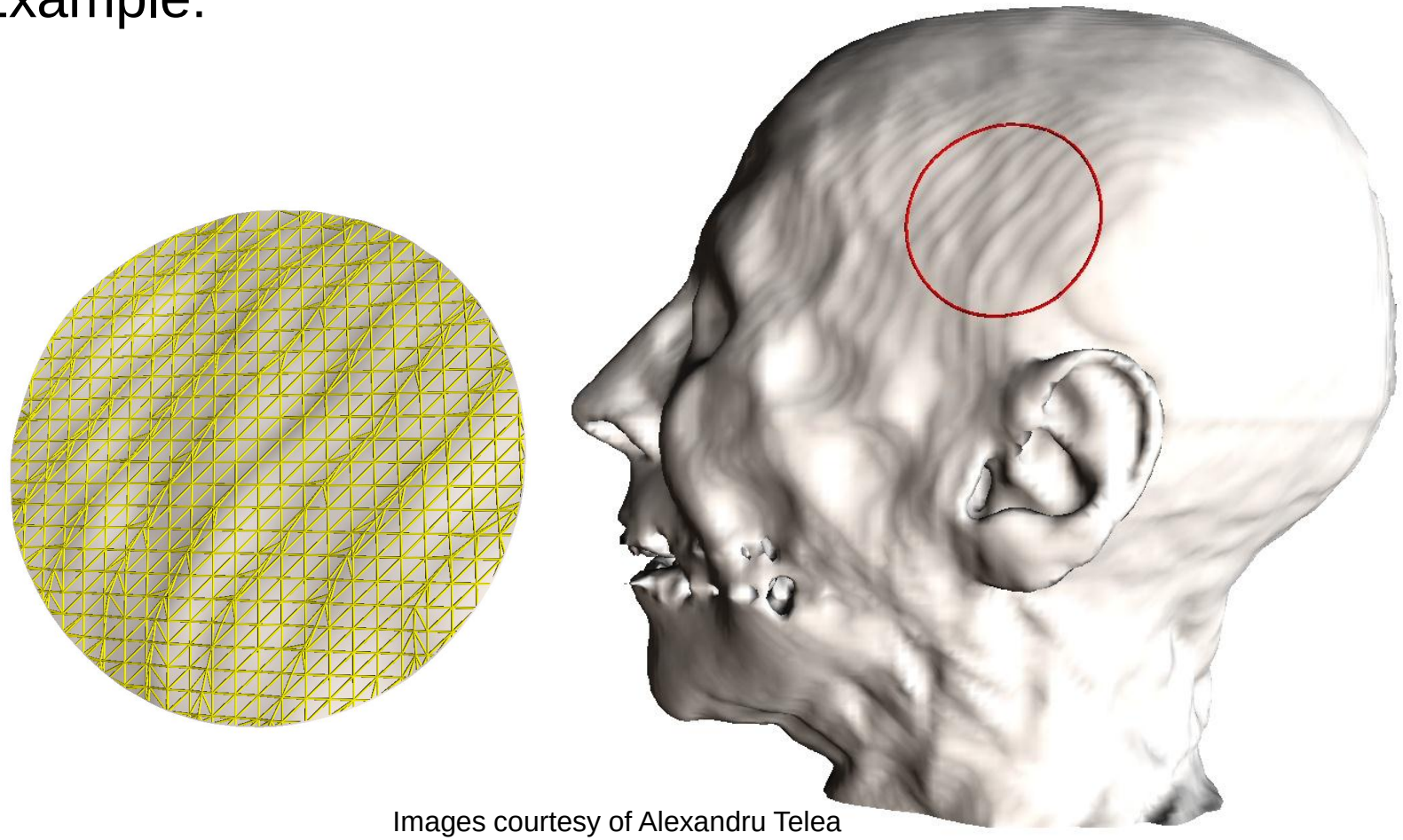
Image courtesy of Alexandru Telea

Scalar Visualization

Interpreting the results of marching cubes must be done with care. For example, holes of size less than the dataset resolution may just be a result from the ambiguity. Also, an isosurface my appear wavy, which is not necessarily an actual feature of the data. Instead, this pattern is caused by subsampling of the original signal on a low-resolution uniform grid.

Scalar Visualization

Example:



Images courtesy of Alexandru Telea

Scalar Visualization

Scalar generation

The two techniques – color mapping and contouring – are simple, effective methods to display scalar information. It is natural to turn to these techniques first when visualizing data. However, often our data is not in a form convenient to these techniques. The data may not be single-valued, i.e. a scalar, or it may be a mathematical or other complex relationship. That is part of the fun and creative challenge of visualization: we must tap our creative resources to convert data into a form we can visualize.

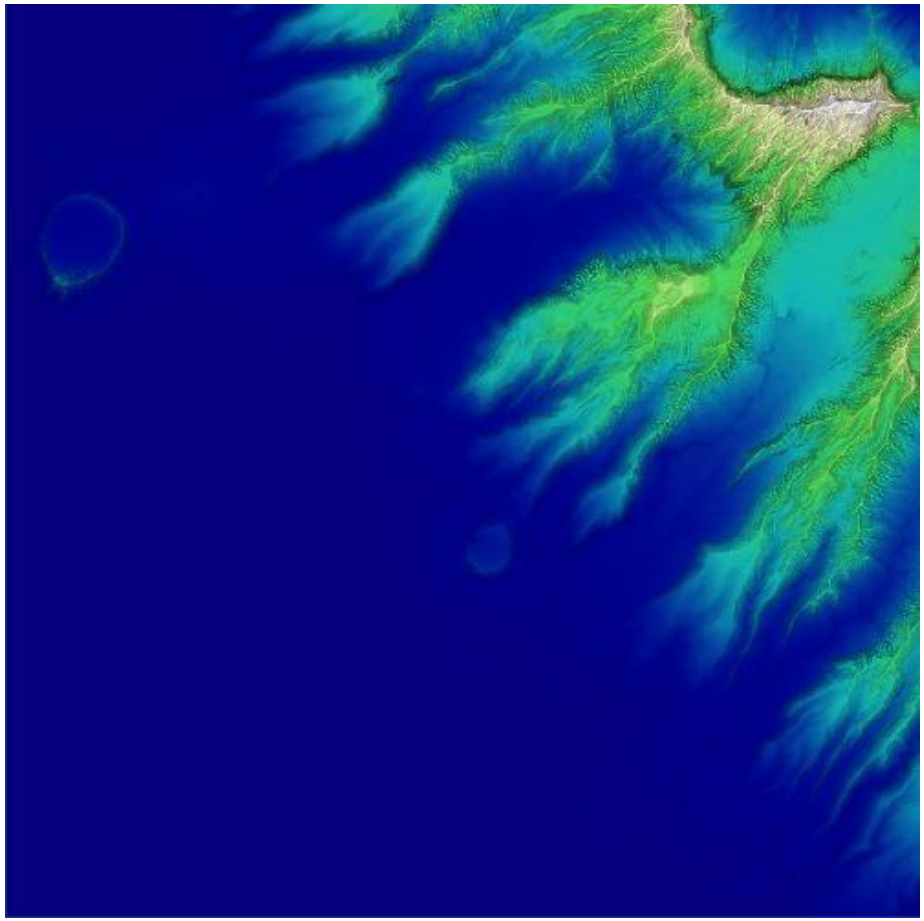
Scalar Visualization

Example

Consider a terrain data set. We assume that the data is given as x-y-z coordinates, where x and y represents the coordinates in the plane, and z represents the elevation above sea level. Our desired visualization is to color the terrain according to elevation. This requires creating a colormap – possibly using white for high altitude, blue for sea level and below, and various shades of green and brown corresponding to elevation between sea level and high altitude. We also need scalars to index into the colormap. The obvious choice here is to extract the z-coordinate. That is, scalars are simply the z-coordinate.

Scalar Visualization

The resulting visualization may look like this:



Scalar Visualization

Height Plots

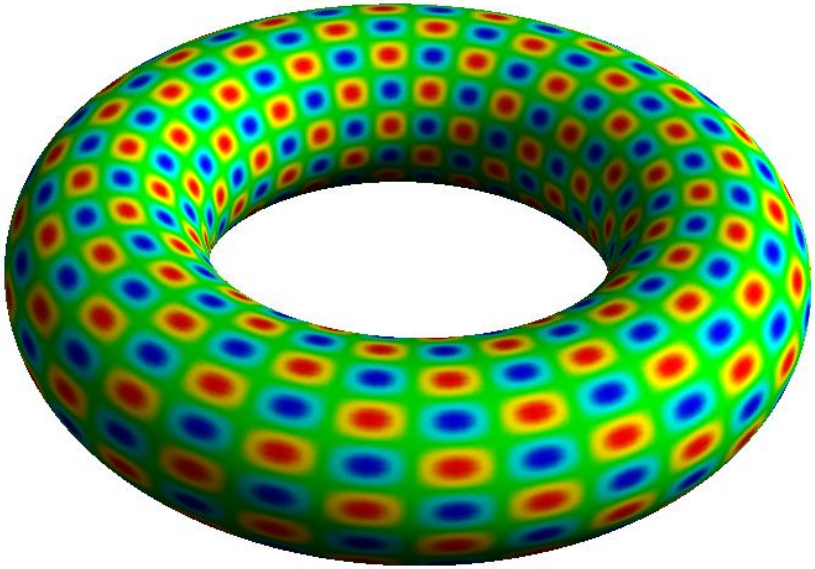
Height plots, also called elevation or carpet plots, were introduced by our first example during the introduction. Given a two-dimensional surface $D_s \in D$, part of a scalar dataset D, height plots cam be described by the mapping operation

$$m: D_s \rightarrow D,\ m(x) = x + s(x)n(x),\ \forall x \in D_s$$

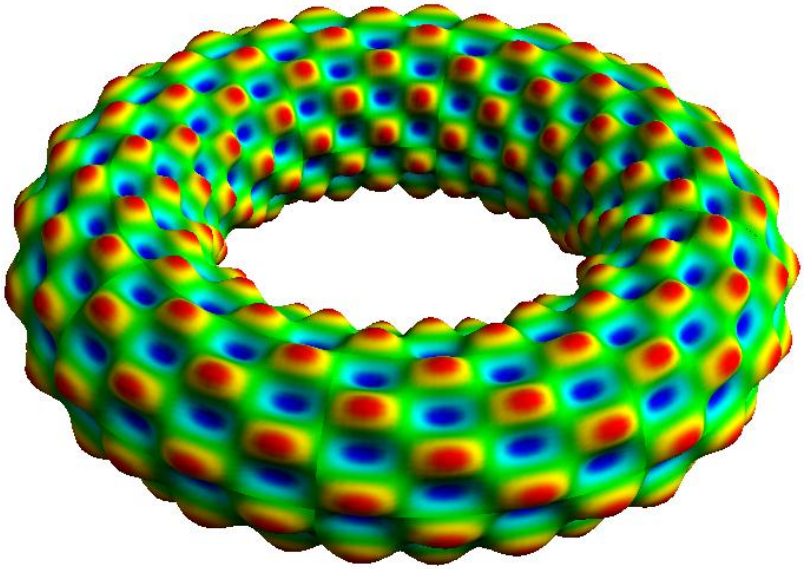
Where $s(x)$ is the scalar value of D at the point x and $n(x)$ is the normal to the surface D_s at x. In other words, the height plot mapping operation "warps" a given surface D_s included in the data set along the surface normal with a factor proportional to the scalar values.

Scalar Visualization

Example: a torus surface (a) and its “warped” variant with the height corresponding to the scalar value.



(a)



(b)

Images courtesy of Alexandru Telea