Extracting surfaces from volume data

Lecture #10: November 18, 2002
Lecturer: Denis Zorin
Scribe: Chien-Yu Hung

After scanning an object by CAT or MRI devices, we can get data defined in a volume. In this lecture we discuss several techniques for extracting surfaces from such data.

1 Linear Interpolation

In the simplest case the data is a set of scalar values defined at nodes of a regular 3D grid. Let $f(i,j,k)$ be the value defined at the grid node (i,j,k). Using trilinear interpolation, we construct a continuous function over the big cube: $f(u,v,w) = \sum i,j,k f(i,j,k)B(u,v,w)$ of the object. The surface that we want to extract is an isosurface (level set) of this function $f(u,v,w) = C$.

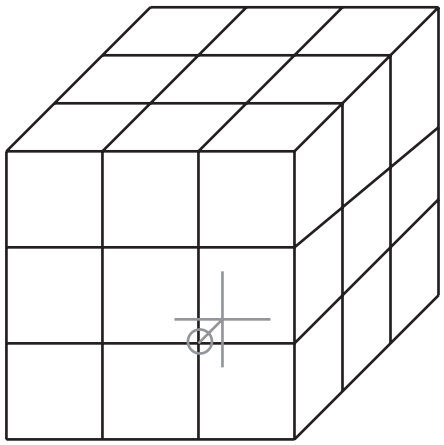


Figure 1: Values $f(i,j,k)$ are defined at grid points.

1.1 1D Case

If a function is defined at integer parameter values, one can define it for intermediate parameter values using linear interpolation, i.e. just connecting the values at integers with straight line segments.

The line segments of the line between i^{th} and $(i+1)^{th}$ point can be expressed by linear interpolation as $f(t) = (1 - (t - i)) + f(i + 1)(t - i)$, for $i \leq t < i + 1$. By summing these equations for all i we can get the equation of the complete interpolating p.w. linear function.

$$L(t) = \sum_i [f(i)(1 - (t - i)) + f(i + 1)(t - i)] \tag{1}$$

Note that $f(i)$ only influences the intervals $[i - 1, i]$ and $[i, i + 1]$. We can write the the previous equation as

$$L(t) = \sum_i f(i)B(t - i) \tag{2}$$

where $B_i(t) = 1 - |t|$, for $-1 < t < 1$, and zero otherwise. The functions $B_i(t) = B(t - i)$ can be regarded as the basis functions; the complete interpolant is the sum of basis functions weighted by function values at integers.

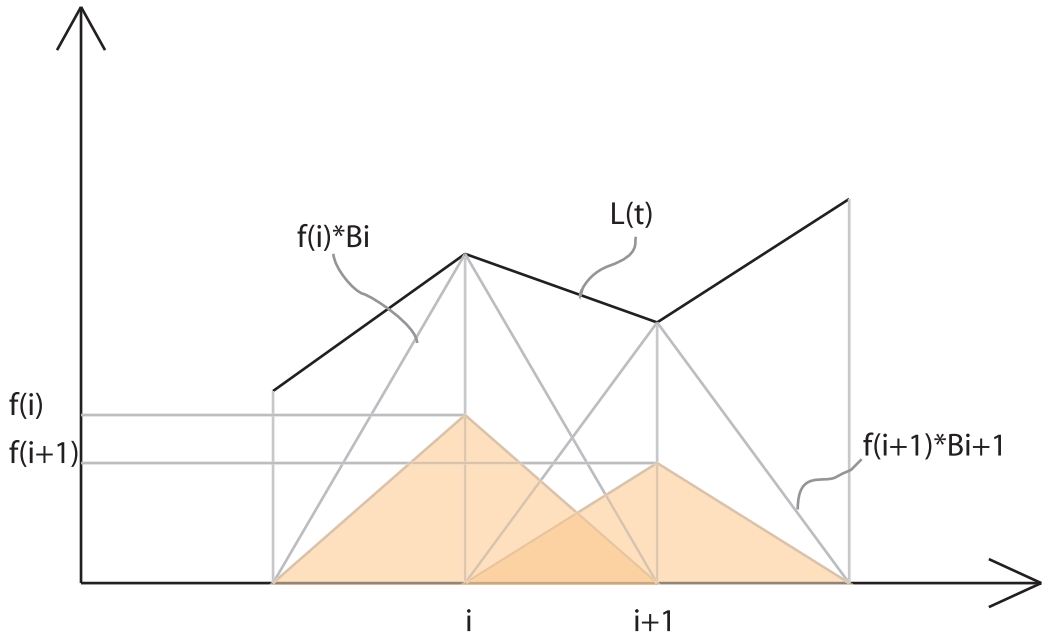


Figure 2: The line equation between i^{th} and $(i + 1)^{th}$ points

1.2 2D and 3D Cases

In 2D case, the values are defined at points (i, j) in the plane with integer coordinates. The simplest way to extend linear interpolation to 2D is to use B(u-i)B(u-j) as our bases functions:

$$L(u, v) = \sum_{i,j} f(i, j)B(u - i)B(v - j) \tag{3}$$

This is also known as bilinear interpolation.

Similarly in 3D we can just take the product of three 1D basis functions:

$$L(u,v,w) = \sum_{i,j,k} f(i,j,k)B(u-i)B(v-j)B(w-k) \tag{4}$$

Usually we use $B_{i,j,k}(u,v,w)$ to replace $B(u-i)B(v-j)B(w-k)$. This is also known as trilinear interpolation.

It is important to note that the level set of $L(u,v)$ is *not* a piecewise linear surface: the equation $L(u,v,w) = 0$ is not linear (it includes terms vw, uv, uw and uvw. A method called *marching cubes* computes piecewise linear approximation to this set.

A different approach is to choose bases functions that are linear; to do this we need to split each cube into tetrahedra. There is a unique linear function interpolating values at four vertices, unless vertices are coplanar. This can be easily seen by counting the degrees of freedom. A linear function in 3D is defined by four coefficients: $au + bv + cw + d$. Requiring it to have given values for four vertices with fixed $u_i, v_i, w_i)$ $i = 1..4$, yields four linear equations in a, b, c, d. The complete interpolant can still be written in the same form as $L(u,v,w)$ but with different basis functions

$$L'(u,v,w) = \sum_{i,j,k} f(i,j,k)B_{i,j,k}(u,v,w) \tag{5}$$

which are all piecewise linear. For a linear function, the level sets are planes, and can be extracted using a simple algorithm called *marching tetrahedra.*

2 Marching Tetrahedra

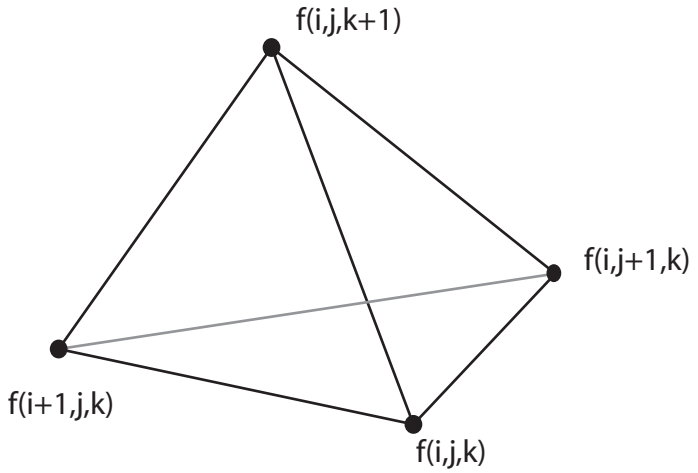


Figure 3: A tetrahedron used in marching tetrahedra.

Each cube of the grid in 3D can be split into 5 or 6 tetrahedra (depending on the method used for splitting the cube). The surface patch contained in this tetrahedron is where $L'(u, v, w) = 0$. We assume that the function values are nonzero at the vertices. In this case there are three possibilities shown in Figure 4: the black circles indicate positive values. Note that the first and last case are identical, because one can be transformed to the other by changing signs. For each tetrahedron where there are at least two vertices with different signs, we construct either triangular or quadrilateral face. The vertices of each face are exactly the points on the edges of the tetrahedron satisfying $L'(u, v, w) = 0$. Because the function L' is linear in space, it is also linear on each edge. At the endpoints of the edge, it coincides with the known function values. So we can find the points on the edges satisfying $L'(u, v, w) = 0$ as an affine combination edge endpoints p_0 and p_1:

$$\frac{f_1 p_0}{f_0 + f_1} + \frac{f_0 p_1}{f_0 + f_1}$$

where f_0 and f_1 are function values at the endpoints.

The algorithm proceeds by going over all tetrahedra, and generating a triangle or a quad (usually replaced with two triangles) for each tetrahedron with a sign change.

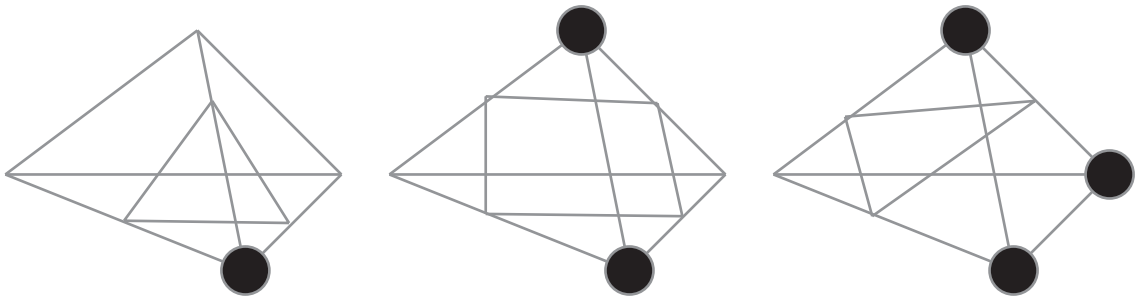


Figure 4: 3 cases for evaluating surface patch in a tetrahedra

3 Marching Cubes

In 2D space, it is easy to convert a square grid into triangle grid; this gives us only twice as many faces. For cubes, we get at least a factor of 5; further more the quality of mesh generated by marching tetrahedra is usually not very good.

The *Marching cubes* method does not require splitting cubes into tetrahedra, but allows one to generate an approximation to the level set of the trilinar interpolant. The resulting meshes are usually of somewhat better quality and have lower triangle count.

At the same time, there are many more cases we have to consider and some additional ambiguity problems which do not occur for tetrahedra. There are 256 possible ways of assigning positive and negative values to the vertices of the cube. Taking symmetries into consideration, these 256 cases can be reduced to 15.

However, in some cases there is more than one topologically distinct way to construct triangles from the points on edges where $L(u, v, w) = 0$. Because we consider each cube independently, it is possible that inconsistent decisions are taken for adjacent cubes.

3.1 Ambiguity

To understand the ambiguity better, we start with the 2D case. The only ambiguity in this case is shown in Figure 6: the signs are different at any two adjacent vertices of the square. Both (a) and (b) are valid ways to extract edges of the isocontour. In 2D case the problem is less severe, as no topological artifacts can occur.

In 3D case, if inconsistent choices are made for two adjacent cubes, the resulting mesh may have holes, which means that it is topologically different from the isosurface that it approximates (isosurfaces never have boundaries).

An example is shown in Figure 7. One way to solve this problem is to make correlated decisions at adjacent cubes. Another approach is to decide between different cases using evaluation at an interior point, which is particularly suitable in 2D. For bilinear interpolation, the isocontours are hyperbolas, if we exclude degenerate cases when the points of isocontours are exactly between corners.

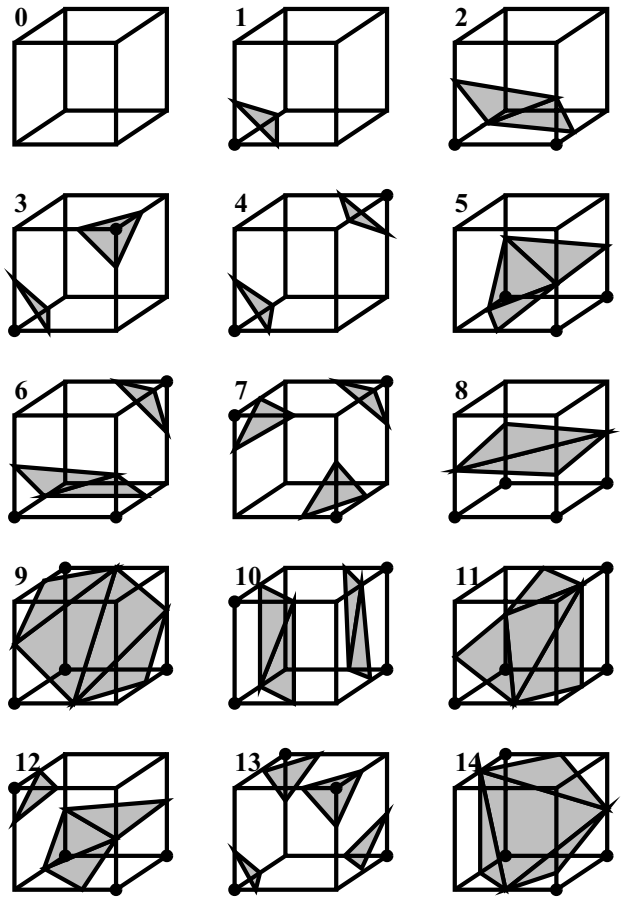


Figure 5: 15 basic cases of the marching cubes algorithm.

Then the sign of the interpolated function at the intersection of the axes of the hyperbolas determines which pair has to be used.

More specifically, it can be easily shown that the isocontour are given by an equation of the form $(u-a)(v-b)-c=0$, with a, b, and c depending on the values of the function at vertices; then the point (a,b) is the point of intersection of the axes of the hyperbola. It should be on the same side w.r.t. the approximation of the isocontour, as the two vertices of the square at which the sign of the function is the same as at (a,b).

In 3D there are a few more cases to consider; specifically there may be several face ambiguities of the same type as the 2D ambiguity described above, and there are so-called interior ambiguities. After all cases are enumerated the total number is 33 (Figure 9).

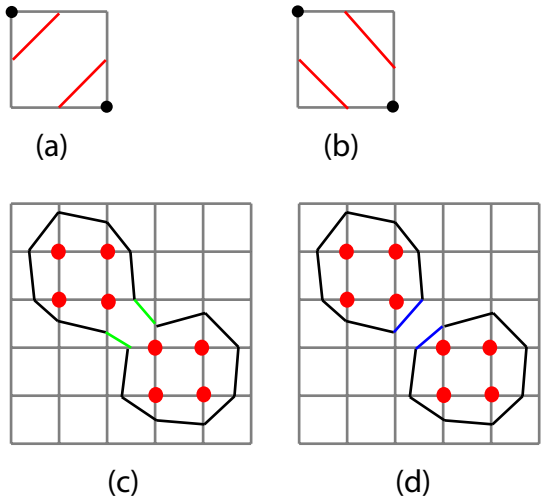


Figure 6: ambiguous situation in 2D case

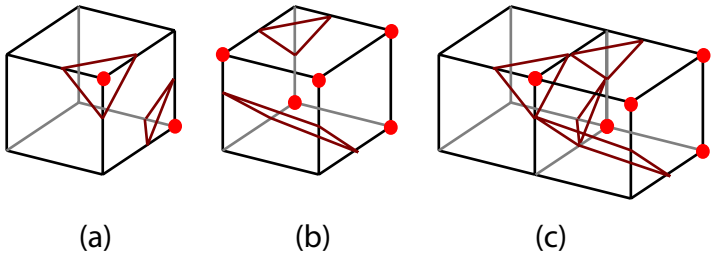


Figure 7: An example of an ambiguous situation in 3D case.

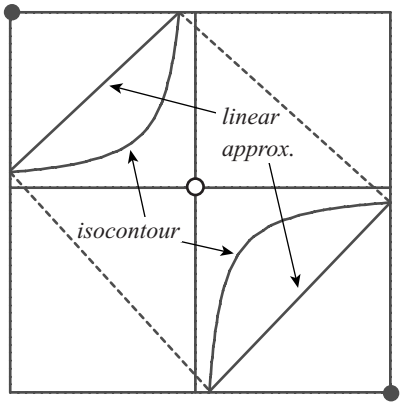


Figure 8: Evaluating an additional point, 2D case. If the value of the interpolated function at the point marked with empty circle is positive, then the correct linear approximation to the isocontour is given by solid lines. If it is positive, the correct approximation would be given by dotted lines.

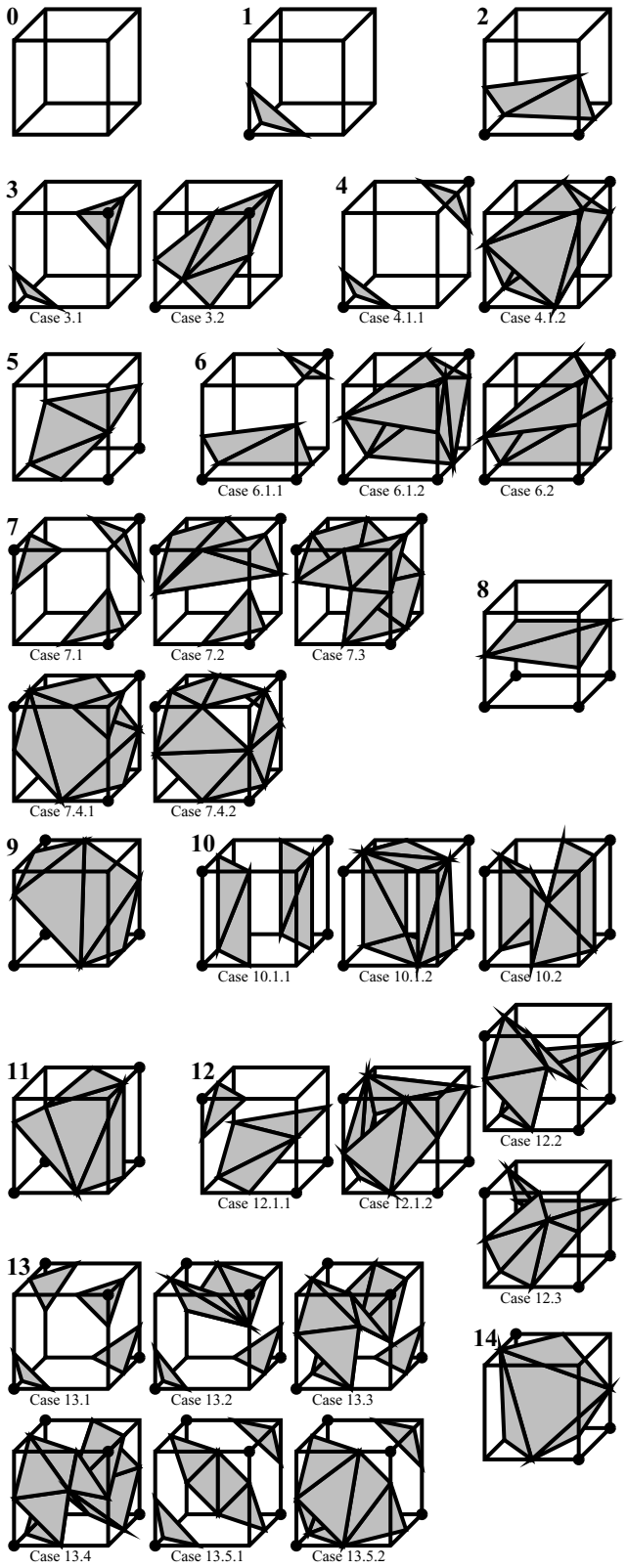


Figure 9: Complete enumeration of all ambiguous cases, from E.V. Chernyaev, "Marching Cubes 33 : Construction of topologically correct isosurfaces", CERN Report,CN/95-17, 1995.