A Scheme for Edge-based Adaptive Tetrahedron Subdivision

Detlef Ruprecht[1] and Heinrich Müller[2]

[1] Andersen Consulting, Otto-Volger-Straße 15, D-65843 Frankfurt, Germany
[2] Informatik VII, University of Dortmund, D-44221 Dortmund, Germany

Abstract. A new scheme for adaptive refinement of tetrahedra based on an edge criterion is presented. This scheme guarantees consistent subdivision without cracks without the necessity for checks between neighboring tetrahedra of a mesh, allowing a recursive algorithm with low space requirements and parallel implementation.

1 Introduction

Tetrahedra are a basic structural element in many applications working with solid models, e.g. finite element computations [2,4] or solid modeling [14]. They are also useful for implicit surface modeling [8] and visualization, for the deformation of solid models [15], and have even been used for flow visualization [11].

In many of these applications, there is a need to adapt the structure of tetrahedral meshes to the level of detail necessary. In FEM computations, a finer mesh is often required in the vicinity of strong force gradients [4]. For the deformation of solid models, an adaptive subdivision of tetrahedra is desired where deformation varies strongly [15]. For explicit calculation and visualization of implicit surface models, tetrahedra near the surface need to be refined [8]. For visualization of mesh data by iso-surfaces or volume rendering [12,17,6,18–20], adaptive and multilevel representations and mesh reduction techniques have been used for accelerating processing and rendering of large data sets. An example of particular interest for the application of this work to the field of volume visualization is the FEM-based approach of [7].

A number of methods to obtain surfaces from space subdivisions have been developed in recent years. Most of them divide the space into cubes. From these cubic meshes, surfaces can be derived with the marching cubes algorithm [10], or adaptively by tree-based methods [13,16]. Unfortunately, these methods tend to produce a large number of surface polygons and, as has been pointed out [8], cubic meshes are unsuitable for a number of applications, particularly when arbitrary given points have to be included in the division of the space. For these cases, a tetrahedral subdivision appears to be more adequate.

Methods for the construction of a tetrahedral mesh dividing a given polyhedral object are mainly known from the FEM literature [1,5]. However, little is known about adaptive subdivisions of such tetrahedralizations to fit particular precision requirements.

One approach presented by Hall and Warren [8] uses criteria for the tetrahedra. If a tetrahedron is split, e.g. because the implicit surface to be described has a zero contour within it, adjacent tetrahedra must be partially split as well to avoid cracks, i.e. discontinuities in the tetrahedral approximation of the surface. This is checked in a second pass over the tetrahedra. Finally, the respective tetrahedra are split by projecting from their centroid to the surface.

Bornemann et al. [4] and Bey [3] present a modification of a method by Zhang [21], which also uses a tetrahedron based refinement criterion causing a regular subdivision of affected tetrahedra. Adjacent tetrahedra again have to be treated in a separate step. Here, only a few simple configurations are allowed for the subdivision of adjacent tetrahedra. Tetrahedra not fitting these requirements are regularly subdivided.

In the following, we present a new algorithm for adaptive tetrahedron subdivision which has two advantages compared to the methods outlined above:

- There is no need to check neighboring tetrahedra. This allows a depth-first recursive approach with small memory requirements.
- The subdivision does not require the addition of a centroid. This reduces the number of new tetrahedra created in a recursive step and possibly also leads to an improved aspect ratio of the created tetrahedra.

The algorithm described below divides tetrahedra based on the subdivision of their edges. This guarantees consistent subdivision of neighboring tetrahedra without interdependency checks, as the edge criterion can be independently evaluated for each tetrahedron containing the edge.

The requirement that the subdivision be controlled by the subdivision of edges appears to be a useful method, as it is easy to define criteria for the splitting of edges, e.g.

- for implicit surface modeling, edges whose endpoints have opposite sign should be split.
- for deformations, if the deformation of the midpoint of an edge differs from the midpoint of the deformed edge more than some threshold, this edge should be split.
- for visualization, if the interpolation along the edge differs from the correct data values by a given threshold.

The path to the solution presented in sect. 3 lies in a subdivision of the triangles on a tetrahedron's surface which depends only on which edges are split. As neighboring tetrahedra share surface triangles, cracks and T-junctions are effectively avoided as the subdivision of the shared triangles is independent of the tetrahedra. We will present a subdivision scheme for surface triangles, and show that for each configuration of the four surface triangles of a tetrahedron, the tetrahedron can be subdivided without the use of additional auxiliary interior points, like a centroid. The subdivision scheme for triangles is outlined in the following sect. 2.

2 Triangle subdivision

We use the following scheme for the subdivision of triangles, which has already proven to be useful for the adaptive subdivision of triangle meshes in geometric modeling by free form deformation [15] (Fig. 1):

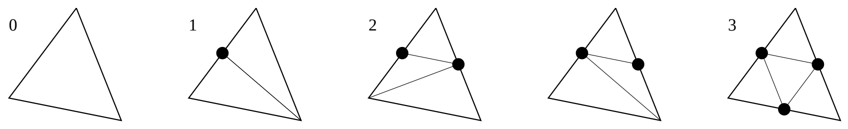

Fig. 1. Configurations for triangle subdivision.

No edges split: There is nothing to do.
One edge split: The triangle is split along the line from the splitpoint of the split edge to the opposite corner of the triangle, giving two new triangles.
Two edges split: The two splitpoints are connected. The remaining quadrilateral is divided starting from the splitpoint of the longer edge. If both diagonals have equal length, the diagonal starting from the splitpoint of the edge with the lower index is used (note that this requires that edges are ordered). This results in three new triangles.
Three edges split: All splitpoints are connected. This results in four new triangles.

For the case of two split edges, if edges are split at their midpoint, starting from the longer edge is equivalent to splitting the quadrilateral along the shorter diagonal. This helps to improve the aspect ratio of the created triangles.

As the subdivision of the triangles is solely driven by the subdivision of its edges, which are shared by neighboring triangles, this scheme ensures that the triangles fit together without cracks or T-junctions, and without having to maintain information about neighboring triangles. This allows storing the information in a rather simple data structure consisting of a list of edges which are referenced by the list of triangles. The subdivision can be performed in a *breadth-first approach* by testing each edge whether it needs to be split and then reforming all triangles.

Alternatively, a *depth-first approach* is possible where each triangle is recursively subdivided until none of the edges needs splitting. In this case, the space requirements are only in the order of the refinement depth, independent of the number of tetrahedra. However, this requires recomputing the edge criterion for each triangle an edge is part of, i.e. it doubles the number of times the edge criterion has to be evaluated.

For both methods, the processing time is linear in the number of created triangles, which is obviously optimal [15].

3 Tetrahedron Subdivision

We will now show that the triangle subdivision scheme presented above allows a subdivision of the tetrahedron for all configurations of subdivisions of the four surface triangles. This is non-obvious, and we have to take into account all possible configurations.

For a tetrahedron with its six edges, we have $2^6 = 64$ possible configurations of split edges. We will first show that after elimination of symmetrical cases, only 11 configurations remain (Fig. 2).

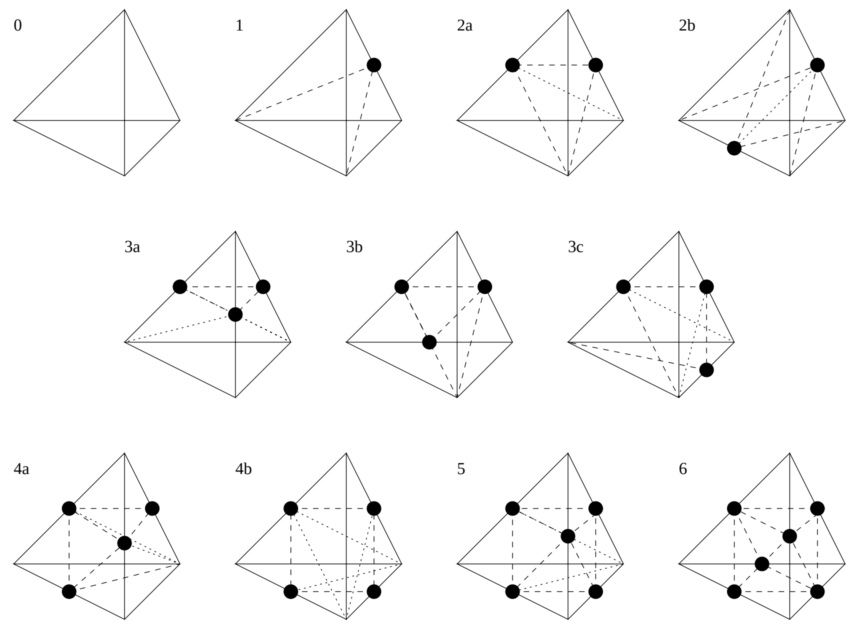

Fig. 2. Configurations for tetrahedron subdivision.

Theorem 1. *There are 11 possible configurations of split edges for a tetrahedron.*

Proof. We will determine the number of possible configurations for each number of split edges separately.

No split edges: Obviously, this is a single configuration.
One split edge: Because of the rotational symmetry of the problem, this is also a single configuration.
Two split edges: If we keep the first split edge fixed, all cases where the second split edge belongs to the same triangle as the first are symmetric. The only case left is the second split edge being diagonally opposite to the first. Thus, we get two configurations.
Three split edges: Starting with any of the two configurations for two split edges, we cannot add a third split edge without at least two of them belonging to the same triangle. So we can start considering one surface triangle with two split edges. Now, the third split edge can either belong to the same triangle, share a corner with the other two split edges, or share a corner with either one of the split edges. For the last case, it is irrelevant which one of the other edges, so we have three configurations.
Four split edges: With four split edges, we have two unsplit edges, which is symmetrical to the situation with two split edges. Thus, we get two configurations.
Five split edges: Five split edges give one unsplit edge, which leads to one configuration.
Six split edges: There is of course only one configuration with all edges split.

If we add up the seven cases above, we get $1 + 1 + 2 + 3 + 2 + 1 + 1 = 11$ configurations. □

Now, we have to show that for each of these 11 configurations, a tetrahedron subdivision is possible. This is slightly complicated by the fact that triangles with two split edges can be divided in two ways depending on the triangle geometry.

Theorem 2. *For all possible configurations, a subdivision of the tetrahedron using the triangle subdivision scheme above is possible without adding an inner point.*

Proof. We will again consider all 11 configurations in succession.

Configuration 0: There is nothing to do.
Configuration 1: The two triangles adjacent to the split edge are split in half. The two new edges, together with the tetrahedron edge diagonally opposite to the split edge, form an inner triangle of the tetrahedron which splits the tetrahedron in half.
Configuration 2a: The triangle containing the split edges is divided into three triangles. Each of these forms a new tetrahedron with the opposite corner of the original tetrahedron.

Configuration 2b: All surface triangles are split in half. By adding a new internal edge connecting the two splitpoints, we get four new tetrahedra.

Configuration 3a: This configuration is a bit more complex, as we have three triangles with two split edges each, where the geometry must be taken into account. The connecting lines of the splitpoints cut off one corner of the tetrahedron, leaving a prism. This can be split into three tetrahedra if the new edges dividing the quadrilateral form a chord, i.e. a single polyline. This is always the case, as the split point of the longest of the three split edges is common to two of the added diagonals. Regardless of the orientation of the third diagonal, it will always be connected to one of the two others.

Configuration 3b: Here, one of the triangles is divided in four by the edges connecting three splitpoints. Each of the new triangles forms a new tetrahedron with the opposite corner.

Configuration 3c: This case is again complicated by the fact that two of the triangles contain two split edges. We have to add an internal edge connecting the diagonally opposite splitpoints. Now, this edge, together with the edges created on the two triangles with one split edge each, cuts off a new tetrahedron, leaving two deformed pyramids which share one triangular surface. These two pyramids can be split into two tetrahedra each independently of each other.

Configuration 4a: This case is similar to configuration 3c in that the inner edge connecting the diagonally opposite splitpoints together with the non-ambiguous edges cut off two triangles. This again leaves two pyramids with a common triangle which can be split into tetrahedrons arbitrarily.

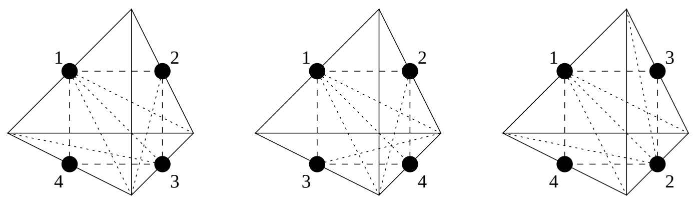

Fig. 3. Subconfigurations for configuration 4b.

Configuration 4b: This is the most complicated configuration as all four surface triangles contain two split edges. We note that the new edges connecting the four splitpoints form an inner quadrilateral which splits the tetrahedron into two prisms. These two prisms can be split into three tetrahedrons each, if their surface quadrilaterals are split by diagonals which form a chord. We have to show that regardless of the outer diagonals, we can always choose the diagonal splitting the inner quadrilateral consistently for both prisms. As the geometric criterion for the outer diagonals is based on the length of the neighboring edges, and the length is a transitive measure, we have a strict ordering of the four split edges. After removing the symmetric cases, we are left with the three subconfigurations shown in Fig. 3, where split point 1 belongs to the longest edge and split point 4 to the shortest. As we can see, for the case of Fig. 3a and Fig. 3b, at least one of the two prisms has outer diagonals meeting in one point so that the choice of the inner diagonal is arbitrary. For case 3c, both prisms constrain the orientation of the inner diagonal, but – fortunately – both produce the same constraint. Thus, we can always split the tetrahedron into six new tetrahedra.

Configuration 5: In this case, we can trivially cut off two tetrahedrons and are left with a pyramid and a prism sharing a quadrilateral. As the pyramid can be split either way, the orientation of the inner diagonal can be determined by the outer diagonals of the prism.

Configuration 6: Cutting off tetrahedra on all four corners leaves an octahedron, which can be split into four tetrahedra by adding an inner edge connecting two diagonally opposite corners of the octahedron. To minimize distortions of the created tetrahedra, the shortest of the three possible inner diagonals should be chosen.

As we have been able to give a construction for the subdivision into tetrahedra for all configurations, the proof is complete. □

With this construction of the subdivision of tetrahedra it is guaranteed that neighboring tetrahedra will fit together without cracks after the subdivision is finished, as the subdivision is driven by the surface triangles which are shared by neighboring tetrahedra. This is similar to the case of the subdivision of triangles described in sect. 2. Again, we do not have to maintain information about neighboring triangles, so we can use a data structure and algorithms similar to the ones described in sect. 2 which give a processing time linear in the number of created tetrahedra, which is again optimal.

4 Discussion

The scheme for adaptive refinement of tetrahedra presented above appears promising for a number of applications in solid and implicit modeling, visualization, and finite element calculations. It allows the use of a simple refinement criterion defined on the edges of the tetrahedra. The required processing time is linear in the number of produced tetrahedra, which is optimal. With a depth-first algorithm similar to the one described for the adaptive subdivision of triangles elsewhere [15], the space requirement is proportional to the depth of the subdivision only. Therefore, the method does not require holding the tetrahedral model in whole in memory, making it suitable for large applications.

The properties of the created tetrahedra should be studied, e.g. if we define the aspect ratio of a tetrahedron as the ratio between the circumscribed and inscribed sphere, what distribution of aspect ratios evolves after several steps of refinement with various edges splitting criteria? It is known that if a tetrahedron is repeatedly split completely as described in configuration 6, the resulting tetrahedra belong to one of at most three congruence classes [3]. This means that the tetrahedra preserve a reasonable shape.

From our experience with the adaptive subdivision scheme for triangles, we are confident that the method will generally lead to well-behaved tetrahedra [15]. A rationale for this is the fact that for common edge splitting criteria, longer edges are more likely to be split than shorter ones, which tends to even out the edge lengths of tetrahedra, thereby improving the aspect ratio. Independently, schemes for improving the aspect ratio, like the method of Laplacian smoothing [9] can be used as a post-processing step. For some of the applications mentioned here, these methods might have to be modified so that the initial points of the tetrahedralization remain unaffected.

For applications where a surface has to be extracted from the tetrahedra, e.g. for rendering implicitly defined surfaces, an additional complication arises from the fact that even for a tetrahedralization with good aspect ratios, the derived triangulated surface might contain thin triangles. In this case, it is better to defer the smoothing step to the created triangles. A practical approach to solve this problem has been discussed by Hall and Warren [8].

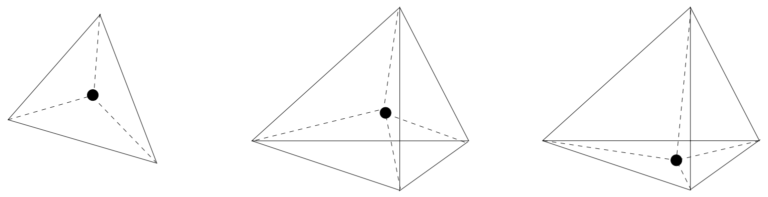

Fig. 4. Configurations for volume-driven subdivision of triangles (left) and tetrahedra. If more than one face of the tetrahedron satisfies the subdivision criterion, the configuration shown on the right is iteratively applied to the subtetrahedra incident with these faces.

Adaptive refinement by our algorithm is solely driven by the subdivision of edges which are shared by neighboring triangles. However, it might happen that there are surface of volume criteria which may enforce a subdivision although the edges do not satisfy the criterion of subdivision. For the visualization of implicit algebraic surfaces, for example, a tetrahedron may contain a zero contour of the defining function which does not intersect its edges. A solution for triangle subdivision may be a volumetric subdivision by placing a new point, for instance its centroid, inside the triangle and connect it to the vertices of the triangle (Fig. 4, left). For tetrahedra, the analogous subdivision is to insert a point inside the tetrahedron, and connect it to the four vertices (Fig. 4, middle). If in the tetrahedral case an additional criterion of subdivision for faces is available, faces which satisfy the criterion are subdivided like the triangle before. If only one face has to be subdivided, the tetrahedral subdivision is obtained by connecting the subdivision point on the face with the vertex of the tetrahedron opposite to it (Fig. 4, right). If more than one face has to be subdivided, one of the faces is selected in order to subdivide the tetrahedron according to this rule. In the resulting subdivision at most one of the subdivided faces is incident to the new tetrahedra. If one is incident, the rule is applied again to the respective tetrahedron. The volume subdivision criterion is only applied if none of the faces has to be subdivided, which in turn are only subdivided if none of the edges of the tetrahedron has to be split.

The problem with the approach just sketched is that the triangles and tetrahedra tend to become thin. If the breadth-first algorithm mentioned in sect. 2 is used rather than the depth-first approach, the unmodified edge-driven scheme may also be applied by deriving additional edge criteria from surface or volume criteria. Such criteria have also been discussed for visualization of implicit surfaces [8].

References

1. Baker, T.J.: Developments and trends in three dimensional mesh generation, Applied Numerical Mathematics **5** (1989) 275–304.
2. Bank, R.E., Sherman, A.H. , and Weiser, H.: Refinement algorithms and data structures for regular local mesh refinement, in Scientific Computing, eds. R. Stepleman et al., (IMACS North-Holland, Amsterdam, 1983) pp. 3–17.
3. Bey, J.: Tetrahedral Grid Refinement, Computing **55** (1995) 355–378.
4. Bornemann, F., Erdmann, B., and Kornhuber, R.: Adaptive multilevel-methods in three space dimensions, Preprint SC 92–14, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Heilbronner Straße 10, 10711 Berlin, July 1992.
5. George, P.L.: Automatic Mesh Generation (John Wiley & Sons Ltd., 1991).
6. Giertsen, C: Volume Visualization of sparse irregular meshes, IEEE Computer Graphics and Applications **12**(2) (1992) 40–48.
7. Grosso, R., Lürig, Ch., Ertl, Th.: Adaptive Multilevel Finite Elements for Mesh Optimization and Visualization of Volume Data, to appear in: Proc. Visualization '97, IEEE Computer Science Press, 1997
8. Hall, M., Warren, J.: Adaptive polygonalization of implicitly defined surfaces, IEEE Computer Graphics and Applications, Nov. 1990, pp. 33–42.
9. Hermann, L.R.: Laplacian-isoparametric grid generation scheme, Journal of the Engineering Mechanics Division of the American Society of Civil Engineers **102/EM5** (1976) 749–756.
10. Lorensen, W.E., Cline, H.E.: Marching cubes: A high resolution 3D surface construction algorithm, Computer Graphics **21(4)** (1987) 163–169.
11. Max, N., Becker, B., Crawfis, R.: Flow volumes for interactive vector field visualization, In Proceedings Visualization '93, eds. G.M. Nielson and D. Bergeron (IEEE Computer Science Press, 1993) pp. 19–24.
12. Max, N., Hanrahan, P., Crawfis, R.: Area and Volume Coherence for Efficient Visualization of 3D Scalar Functions, Computer Graphics **24(5)** (1990) 27–33.

13. Müller, H, Stark, M.: Adaptive generation of surfaces in volume data, The Visual Computer **9** (1993) 182–199.
14. Paoluzzi, A., Bernardini, F., Cattani, C., Ferrucci, V.: Dimension-Independent Modeling with Simplicial Complexes, ACM Transactions on Graphics **12** (1993) 56–102.
15. Ruprecht, D., Nagel, R., Müller, H.: Spatial free form deformation with scattered data interpolation methods, Computers & Graphics **19** (1995) 63–71.
16. Schmidt, M.F.W: Cutting cubes – visualizing implicit surfaces by adaptive polygonalization, The Visual Computer **10** (1993) 101–115.
17. Shirley, P., Tuchman, A.: A Polygonal Approximation to Direct Scalar Volume Rendering, Computer Graphics **24(5)** (1990) 63–70.
18. Silva, C., Mitchell, J., Kaufman, A.: Fast rendering of irregular grids. In Proceedings ACM Symposium on Volume Visualization 1996 pp. 15–23
19. Wilhelms, J., Gelder, A., Tarantino, P., Gibbs, J.: Hierarchical and parallelizable direct volume rendering of irregular and multiple grids. In Proceedings IEEE Visualization 1996 pp. 57–65
20. Yagel, R., Reed, D., Law, A., Shih, P.-W., Shareef, N.: Hardware assisted volume rendering of unstructured grids by incremental slicing. In *Proceedings ACM Symposium on Volume Visualization 1996* pp. 55–63
21. Zhang, S.: Multilevel Iterative Techniques, PhD thesis, Pennsylvania State University, 1988.