ISSN 0965-5425, Computational Mathematics and Mathematical Physics, 2010, Vol. 50, No. 1, pp. 139–156. © Pleiades Publishing, Ltd., 2010.
Original Russian Text © A.A. Danilov, 2010, published in Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki, 2010, Vol. 50, No. 1, pp. 146–163.

Unstructured Tetrahedral Mesh Generation Technology[1]

A. A. Danilov

Institute of Numerical Mathematics of the Russian Academy of Sciences, ul. Gubkina, 8, Moscow, 119333 Russia
e-mail: a.a.danilov@gmail.com

Received November 17, 2008

Abstract—We present a robust unstructured tetrahedral mesh generation technology. This technology is a combination of boundary discretization methods, an advancing front technique and a Delaunay-based mesh generation technique. For boundary mesh generation we propose four different approaches using analytical boundary parameterization, interface with CAD systems, surface mesh refinement, and constructive solid geometry. These methods allow us to build a flexible grid generation technology with a user friendly interface.

DOI: 10.1134/S0965542510010124

Key words: advancing front technique, constrained Delaunay triangulations mesh generation.

INTRODUCTION

Unstructured tetrahedral meshes are widely used in mathematical modelling due to their simplicity and ability to represent complex domains. A tetrahedral mesh is called conformal, if each its two tetrahedra have one common vertex, one entire common edge, or one entire common face, or do not have any common points. The similar definition applies to conformal triangulations. Mesh conformity is an important and desirable property. In this paper we use a notion "triangulation" for both tetrahedral and triangular meshes, if it does not lead to an ambiguity.

Mesh generation process consists of two stages: surface meshing and volume meshing. At the first stage we construct a surface conformal triangular mesh. The surface mesh can be produced in several different ways. At the second stage we mesh the volume inside the domain into tetrahedra while preserving given surface triangular mesh on the boundary.

In 2D case, we can always construct a boundary conforming triangulation without insertion of new points inside a given polygon. In 3D case, for some polyhedra no triangulation is possible without insertion of additional points. An example of such polyhedron is shown in Fig. 1—Schönhardt prism [1]. Insertion of just one point inside the convex polyhedron can solve the problem as shown by Joe [2]. George [3] has shown the existence of a tetrahedral boundary conforming mesh for an arbitrary polyhedron with possible addition of several internal points. However, robust construction of a conformal tetrahedral mesh with fixed boundary triangulation is a challenging problem.

There exist two main approaches for tetrahedral mesh generation: methods based on Delaunay triangulations (DT), and advancing front technique (AFT). The Delaunay triangulation approach does not guarantee the boundary conformity. Several techniques are proposed to match the boundary: local mesh transformations [4], mesh refinement [5], and constrained Delaunay triangulations (CDT) [6]. On the other hand, the advancing front technique does not have issues with boundary recovery since it starts from the given surface mesh [7]. However, AFT methods have difficulties in successfully closing up their fronts. Combining both AFT and DT ideas, one proposed hybrid methods [8]. Our volume mesh generation method is a combination of a conventional AFT method and a DT based fallback method.

In this paper we present several methods for defining and meshing domain boundaries, and a combination of the two methods for volume meshing. In Section 1 we propose four methods for boundary meshing using analytical boundary parameterization, interface with CAD systems, surface mesh refinement, and solid constructive geometry (SCG). In Section 2 we discuss a classical AFT method for volume meshing. This method does not guarantee us a successful meshing. That is why we use a fallback method based on DT and boundary faces recovery. A simplified version of this George's method [9] is discussed in Sec-

[1]The article is published in the original.

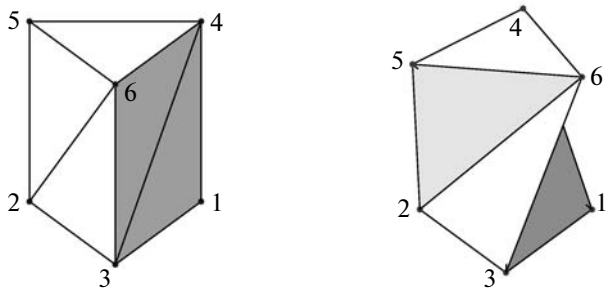

Fig. 1. The Schönhardt prism: convex prism and twisted non-convex prism (the hidden edge is 1–5). No conformal triangulation is possible without insertion of additional points.

tion 3. In Section 4 we briefly discuss robustness issues of volume meshing. In Section 5 we provide several application examples.

The presented technology is implemented as a part of Ani3D package [10], the successor of Ani2D package [11]. Both packages provide tools for unstructured mesh generation, anisotropic metric-based adaptation, finite element discretization, and linear system solving. These packages are publicly available. In each section we pay special attention to our actual Ani3D implementations of discussed methods. Some technical information will be provided for our AFT method as well.

1. SURFACE MESH GENERATION

In this section we will discuss several methods for defining and meshing the surface, which represents a domain boundary. The main idea is to split the surface into several pieces sharing common boundary lines. Though in general these lines may be curves, for simplicity we just call them lines. Then, assuming that we have discretization of the boundary lines, we can use an advancing front method on each piece. Since all pieces of the surface have the same discretization on the boundary lines, the resulting surface mesh will be conformal.

Keeping this idea in mind we propose four different approaches for surface meshing. They are based on:

(1) analytical boundary parameterization;

(2) interface with CAD systems based on a boundary representation (BREP) model;

(3) surface mesh refinement and coarsening;

(4) surface meshing of solid constructive geometry models.

Analytical boundary parameterization method is used for simple domains: spheres, boxes, cylinders, and polyhedra with planar faces. Complex domains like a union or an intersection of several simple primitives can be processed with solid constructive geometry approach. More complex domains designed in CAD systems can be meshed with a help of an interface between a CAD system and a mesh generator. A domain boundary can also be defined by a surface mesh produced by an unattainable CAD system. Since the resulting volume mesh quality strongly depends on surface mesh quality, an additional surface mesh refining may be required. On the other hand, too fine surface mesh can produce unnecessarily fine volume mesh, in this case an additional surface coarsening can be performed. These preprocessing tools allow us to adopt and use many available surface meshes.

1.1. Analytical Boundary Parameterization

This is a straightforward application of the general idea discussed above. We split the boundary surface of the domain into several pieces Σ_i called patches. Each of them is parameterized separately by a smooth function:

$$\Sigma_i = \{(x, y, z) \,|\, (x, y, z) = f_i(u, v), (u, v) \in \Omega_i \subset \mathbb{R}^2\}.$$

A boundary line Γ_k between two adjacent patches is also parameterized:

$$\Gamma_k = \{(x, y, z) \,|\, (x, y, z) = g_k(t), t \in [t_k^0, t_k^1]\}.$$

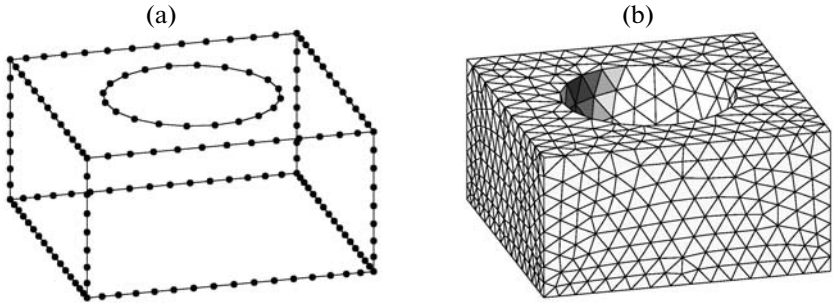

Fig. 2. Analytical boundary parameterization: (a) boundary lines discretization; (b) surface mesh.

We also need the mappings (u_{ik}, v_{ik}) from a boundary line parameterization space to the surface parameterization space for each patch:

$$g_k(t) = f_i(u_{ik}(t), v_{ik}(t)), \quad t \in [t_k^0, t_k^1].$$

In general, for some functions f_i, g_k these mappings u_{ik}, v_{ik} may appear to be analytically non-computable. Thus this method is applicable only for very simple domains.

Alternatively, the user can define a separate boundary line parameterization in (u, v) space for each separate patch. Let us consider a patch i parameterized by function $f_i(u, v)$. We can represent its boundary line Γ_k by a curve in (u, v) space. For instance, if we use the mapping $u \longrightarrow (u, v_{ik}(u))$, then in $\mathbb{R}^3$ the boundary line will be parameterized as follows:

$$\Gamma_k = \{(x, y, z) | (x, y, z) = f_i(u, v_{ik}(u)), u \in [u_k^0, u_k^1]\}.$$

These parameterizations of Γ_k will be different for different patches, but they should represent the same geometry. The latter approach is the one used in Ani3D package for analytical boundary representation.

In order to create a surface mesh for each patch, boundary lines are discretized first. Let us consider a parameterized boundary line:

$$(x, y, z) = g(u) = f(u, v(u)), \quad u \in [u_0, u_1].$$

We create new points on the line:

$$P_i = g(\tau_i), \quad i = 0, 1, \ldots, n.$$

Parameter points $\tau_i \in [u_0, u_1]$ are selected by an edge subdivision method according to desired distance between points P_i.

Once the boundary discretization on the patch is constructed, we use the 2D AFT method to construct a triangular mesh. In this AFT method we work simultaneously in both (u, v) space and (x, y, z) space. The 2D AFT method will be briefly discussed in Section 2.

This method for surface meshing is a general approach. Parameterization functions may be complicated for complex surfaces and especially for their intersection lines. Thus this method is applicable in rather simple cases. An example of initial geometry and generated surface mesh is presented in Fig. 2.

1.2. Interface with CAD System

In many cases a domain can be modeled in a CAD system. Most of CAD kernels use a boundary representation (BREP) model as an internal model representation. In this model four types of topological entities are used: vertices, edges, faces, and volumes. Beside this, the BREP model provides adjacency relationships between the entities. Each face and each edge have their own parameterization stored by the CAD kernel. For instance, NURBS are widely used in CAD kernels for surface parameterizations.

The topology model and the parameterizations of lines and surfaces provide all necessary data to apply analytical boundary parameterization method. Several ways may be used to obtain this data. One option is to implement a special file reader. This reader loads all data from CAD file, and works with NURBS or other types of used parameterizations. Another option is to query all data directly from the CAD kernel, or through a special abstracting interface layer between the CAD kernel and the AFT generator. The sec-

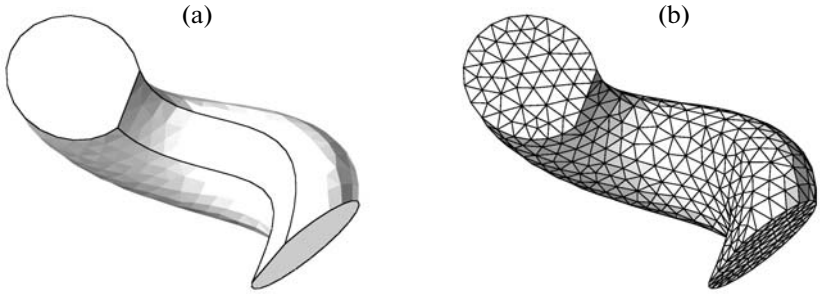

Fig. 3. Interface with CAD system: (a) model in CAD system; (b) surface mesh.

ond option is more appealing, since there is no need to implement different file format readers and different parameterization types.

In Ani3D package we used the interface layer approach on the basis of a common functional interface. The layer can be easily used to switch between different CAD kernels without the need of rewriting the mesh generator itself. The Common Geometry Module (CGM) [12] is chosen as an abstracting layer: it integrates with OpenCASCADE [13], ACIS, and Pro/Engineer geometry kernels. Interfacing a CAD system with a mesh generator is mainly a technical task. An example of a CAD model and a surface mesh are illustrated in Fig. 3.

1.3. Surface Mesh Refinement and Coarsening

In some cases the domain already has a boundary mesh. However, the surface mesh may be ill-shaped, like meshes exported from CAD systems. In this case constructing a volume mesh inside the domain will be hampered or will result in an ill-shaped volume mesh. In Ani3D package we introduced a new technique for surface mesh modification [14]. The basic idea is to split the surface into several nearly flat polygons, and remesh them. In order to construct a nearly flat polygon, we define a criterion, and use it to add triangles to the polygon. We use the following criterion. Triangle T nearly lies in plane P, if an angle between the normals to the triangle T and to the plane P is small enough, and a distance from the triangle T to the plane P is also small. The user can control the value of admissible deviation.

After the surface is split into several nearly flat polygons, we create a new boundary discretization of these polygons. This boundary discretization should be consistent across different polygons. To achieve this, we compute a mesh size function in the vertices and interpolate it on the edges. According to this metric we discretize the edges, thus ensuring consistency across polygons.

For each polygon we construct a new triangular mesh. To this end, we make a projection of the polygon and its new boundary discretization on a plane, construct a new triangular mesh using planar AFT starting from the new boundary discretization, and project the new mesh from the plane back to the surface. Since the original polygon was nearly flat, a surface distortion caused by the projection will be insignificant.

At the end of this process a new conformal surface triangulation is created. The detailed algorithm is presented in Algorithm 1.

Algorithm 1. Surface mesh modification

for all vertices V_i, **do**

Compute a mesh size function $h_i = h(V_i)$ based on the size of surrounding triangles

in the initial surface mesh

end for

while there are unexamined triangles **do**

Pick an arbitrary triangle T_0 from the initial mesh

Construct a plane P containing triangle T_0

Initialize a new polygon $M = \{T_0\}$

for all neighboring triangles T_k **do**

if T_k nearly lies in P **then**

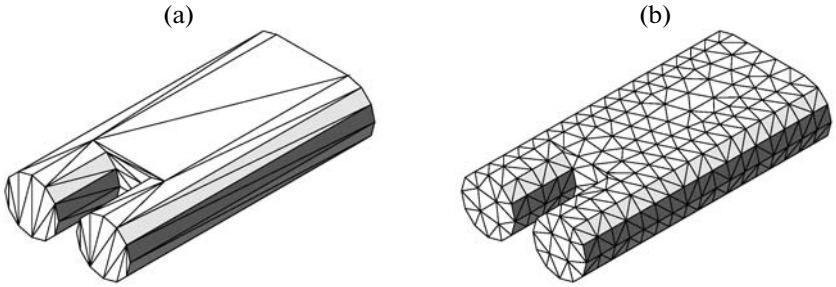

Fig. 4. Surface mesh refinement: (a) initial surface mesh; (b) refined surface mesh.

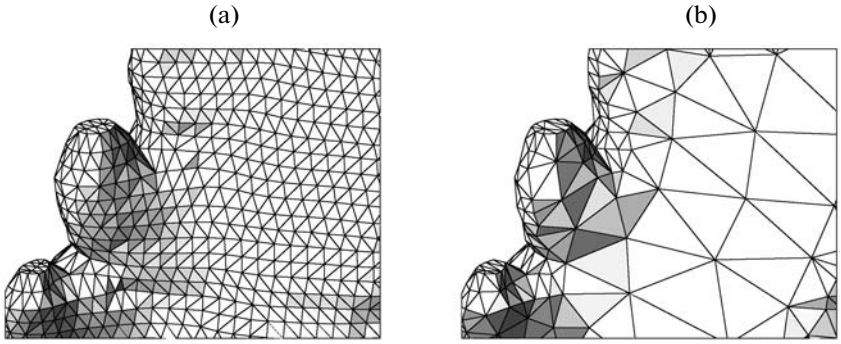

Fig. 5. Surface mesh coarsening: (a) initial surface mesh; (b) coarsened surface mesh.

Add triangle T_k to the set M
end if
end for
for all boundary edges E_j of polygon M **do**
Linearly interpolate the mesh size values h_i on the edge
Create an edge discretization based on the interpolated metric
end for
Project polygon M and its new boundary discretization on the plane P
Construct a new triangular mesh using planar AFT
Project a new mesh from P back to M
Discard M from initial mesh
end while

This method is useful for remeshing an ill-shaped surface mesh exported from CAD system into a more regular surface mesh. This technique may also be used for coarsening an initial fine mesh. To this end, we use a moderate admissible deviation for triangle normals. In this case polyregions will have much more triangles. Then, for each polyregion we can construct a new coarse triangular mesh resulting in a new more coarse surface mesh. Both examples of surface refinement and coarsening are presented in Figs. 4 and 5 respectively.

1.4. Solid Constructive Geometry

Some domains are rather complex for analytical representation, but still simple enough for using huge CAD systems. Important examples are intersections and unions of simple objects (primitives) like boxes, spheres, cylinders. This type of geometry representation is frequently called solid constructive geometry (SCG). The idea of implementing boundary meshing for SCG models is as follows [15]. For a simple primitive a surface mesh is generated using an analytical parameterization. Two surface meshes representing two primitives are intersected to provide a new surface mesh for a union or an intersection of the two primitives. The new object is then treated as another primitive.

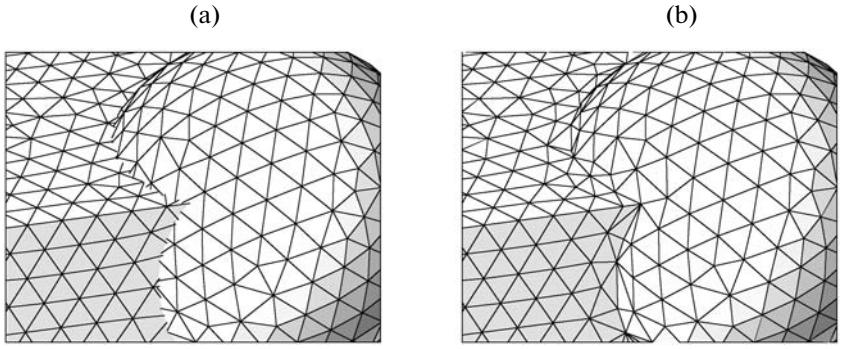

Fig. 6. Solid constructive geometry: (a) two surface meshes; (b) conformal surface mesh.

The mesh intersection is performed in three stages.

At the first stage we intersect two surface meshes, and construct a polyline of the intersection. This line splits each surface mesh into three parts: two surface parts and an intersection band. Depending on a logical operation used (union, intersection or difference), one of two parts will lie either inside or outside the resulting object. The other surface part will lie on the surface of the object.

At the second stage we remove triangles from the intersection band, as well as triangles which do not lie on the new surface.

At the third stage we remesh the band near the intersection polyline ensuring conformity of the final mesh, and perform local mesh optimizations in order to improve the mesh quality near the intersection line.

The detailed algorithm is presented in Algorithm 2.

Algorithm 2. Surface mesh intersection

Intersect two meshes: $\Gamma = M_1 \cap M_2$

for each mesh M_i **do**

for each triangle T **do**

Determine if T intersects Γ, or T lies inside the new surface, of T lies outside it

end for

Split M_i into three parts: triangles M_i^{band} forming intersection band, triangles M_i^{in} lying inside the new surface, and triangles M_i^{out} lying outside it

end for

Replace mesh parts M_i^{band} with a new conformal mesh M^{band}

Unite the new intersection band mesh M^{band} with two other surface parts M_i^{in}

An example of this approach application is shown in Fig. 6. In Ani3D package the user may create and use his own arbitrary primitives by providing their boundary meshes. In particular, user may crop an interesting region from an already obtained surface mesh by intersecting it with a rectangular box.

2. ADVANCING FRONT TECHNIQUE

In this section we discuss an advancing front technique which is used to construct a volume mesh inside a bounded domain. We define the front as a set of oriented triangular faces forming a closed conformal surface mesh. The idea of AFT is to exhaust this front by advancing it inside the domain and constructing new tetrahedra. The front actually divides the domain into two parts: already meshed one and the remaining part. At each step a new tetrahedron is constructed and the front is advanced. If the front is void, then the mesh is fully constructed, and AFT algorithm terminates.

Let us discuss one step of this method. In Fig. 7 we show main substeps in 2D case.

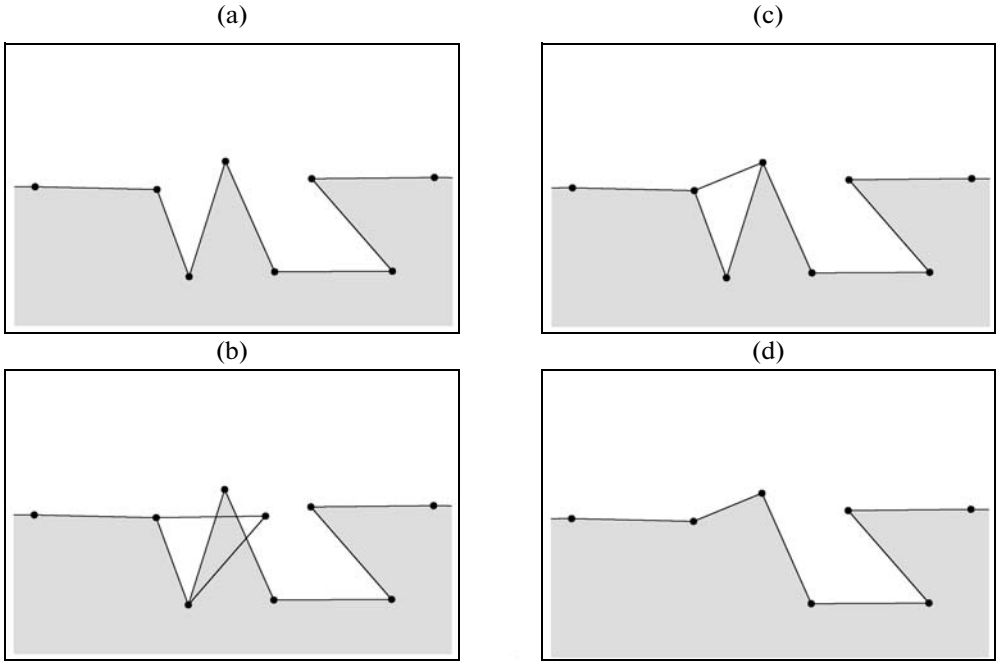

Fig. 7. One step of AFT method in 2D case: (a) current front; (b) constructed triangle; (c) suitable triangle; (d) updated front.

We start by selecting an unvisited face with the smallest area from the current front, we call it the current face. We construct a new tetrahedron based on the current face, i.e. we construct a normal vector from the center of the face, and place the fourth vertex on its end (see Fig. 7b). The length of the vector is based either on a user-defined mesh size, or on a product of a current face size and a user-defined stretching factor k. In the latter case, AFT method will result in a quasi-uniform mesh with $k = 1$, and it will result in an automatic mesh coarsening away from the boundary with $k > 1$. Automatic mesh coarsening will be discussed in the next subsection.

We are going to find a suitable tetrahedron and add it to the mesh. A tetrahedron is assumed to be suitable if it does not intersect the front, does not degenerate, and does meet other user requirements. We start from just constructed tentative tetrahedra and check if it is suitable. If not, we iterate through the front vertices and try to use them as the fourth vertex of the tentative tetrahedron, until a suitable tetrahedron is found (see Fig. 7c).

If no tetrahedron was found, then we skip and mark the current face as visited. These marks will be cleared later and faces will be revisited. If a suitable tetrahedron was found, then we add it to the mesh and update the front: we delete the current face, and either delete from the front, or add to the front each of the other three faces (see Fig. 7d).

We repeat the whole process until the front is void or all faces are marked as visited. In the last case we clear the marks and restart the whole process. If the front can not be advanced further, then another method should be used to mesh the remaining part of the domain. At the end we can perform a simple mesh smoothing in order to improve mesh quality.

The formal algorithm of AFT method is presented in Algorithm 3.

The similar technique is used in 2D case. In planar 2D case the mesh can always be constructed using just AFT method. The 2D case on a parameterized surface differs from

Algorithm 3. Advancing front technique

while front is not void **do**
  Pick the smallest face $F = \triangle ABC$ from the current front Γ
  if all faces are already visited **then**
    Stop AFT
  end if

Determine the desired mesh size near F
Compute a tentative position for the new point P
Locate front triangles Γ_B and front points V_B in a vicinity of P
if there are front points near P **then**
  Set D as the nearest front point
else
  Set $D = P$
end if
while tetrahedron $ABCD$ is tangled, or is degenerated, or intersects Γ_B **do**
  Pick another candidate D from V_B,
  if there are no more candidates **then**
    Mark F as visited, and goto 2
  end if
end while
Add tetrahedron $ABCD$ to the mesh
for each oriented face F_i of $ABCD$ **do**
  if inverted face F_i^* is in the front **then**
    Remove F_i^* from the front
  else
    Add F_i^* to the front
  end if
end for
end while

The similar technique is used in 2D case. In planar 2D case the mesh can always be constructed using just AFT method. The 2D case on a parameterized surface differs from the planar 2D case in that part, where a new point is constructed to form a triangle. A new point is searched in (u, v) space in such a way that the resulting triangle will be isosceles in (x, y, z) space. An intersection between a new tentative triangle and the existing front is checked in (x, y, z) space.

2.1. Finiteness of AFT Algorithm

In this subsection we will go into the details of AFT method used in Ani3D package. Let us first consider the case, when an automatic mesh coarsening is used. In our AFT method we follow the next rule. Addition of a new vertex should not decrease the minimal pairwise distance ρ between vertices in the mesh. On a regular AFT step we consider the front triangle ABC, where a, b, c are its edge lengths. Let M be a barycenter of the triangle. We construct a normal to plane ABC going through point M, and position the point D on its end. Distance $MD = h$ is chosen automatically according to coarsening factor k. We use the following formula for h:

$$h = \sqrt[3]{\frac{a^3 + b^3 + c^3}{3}} k \sqrt{\frac{2}{3}}.$$

According to mean inequality, $h \geqslant \sqrt{\frac{a^2 + b^2 + c^2}{3}} k \sqrt{\frac{2}{3}}$, with equality being achieved only when $a = b = c$. Keeping in mind that $k \geqslant 1$, we get $h^2 \geqslant \frac{2a^2 + 2b^2 + 2c^2}{9}$. Now we consider the length of AD, $AD^2 =$

$AM^2 + MD^2$. Length of AM is computed by median length formula: $AM = \frac{2}{3}\sqrt{\frac{2b^2 + 2c^2 - a^2}{4}}$. We get $AM^2 = \frac{2b^2 + 2c^2 - a^2}{9}$, and finally

$$AD^2 \geqslant \frac{2b^2 + 2c^2 - a^2}{9} + \frac{2a^2 + 2b^2 + 2c^2}{9} = \frac{a^2 + 4b^2 + 4c^2}{9}.$$

Since triangle side lengths a, b, c are not less than ρ, then $AD \geqslant \rho$ as well. In case of $k = 1$ and equilateral triangle ABC we get $AD = a = b = c$. The same applies for other edges BD and CD. Thereby the fourth point D will be distanced enough from the triangle.

Additionally, we will check that there are no other front points in ρ-vicinity of the fourth point D. If such points exist, then the point D is not considered, and the search is started from the nearest front point. Following these rules we can guarantee that the minimal pairwise distance between points is not less than ρ. In this case the total number of vertices will be bounded above, and the total number of tetrahedra will be bounded above as well.

The case, when the value of h is defined according to the user-defined mesh size function $F(x)$, is similar. If $F(x) > F_0 > 0$ in the whole domain, then the minimal pairwise point distance is limited bellow, and the number of tetrahedra is limited above. If $F(x)$ may be arbitrary small, then AFT algorithm may end up in infinite refinement loop.

Summing up, if automatic mesh coarsening is used, or if user-defined mesh size function is isolated from zero, then the total number of points and tetrahedra is bounded above, thus the number of AFT iterations is also bounded above.

2.2. Complexity of AFT Algorithm

In order to analyze the complexity of the proposed algorithm, we focus on the following operations with the front:

(1) add a new face into the front;

(2) delete an existing face from the front;

(3) pick the face from the front with minimal area;

(4) select the faces from the front in a specified search region.

Effective implementation should use effective data structures. In Ani3D we use a heap of faces with the smallest face in the root, and an octree-based search tree for searching faces in 3D space. Computational costs for adding and deleting a face are proportional to $\log N$, where N is the number of faces in the front. Searching the face with minimal area takes a fixed number of operations, and searching the faces in specified search region is proportional to $(\log N + K)$ operations, where K is the number of found faces in the search query. Let us examine the last estimate in detail.

An octree is organized as a conventional binary tree whose vertices have eight rather than two descendants. Each vertex of an octree is associated with a part of a bounding cube $H \times H \times H$. The root of the tree is associated with the entire cube, whereas its descendants correspond to the eights of the original cube. Transition from an ancestor to descendants implies a partition of the ancestor's cell into eight equal subcells associated with eight descendants. The level of the tree vertex is the number of graph edges in the path from the tree root to the vertex. The tree root has level 0, and its direct descendants have level 1. In each vertex we keep a list of front triangles assigned to the vertex. For each triangle we define its center as a barycenter of a triangle, and its radius R as the largest distance from its center to its vertices. In order to choose the octree vertex referring to the triangle with radius R, we

(1) compute the minimal n, such that $R \leqslant 2^{-n} \cdot H$; this number n will be the level of an octree vertex;

(2) search an octree vertex with level n associated with the cubic cell containing the center of the triangle.

Advantages of the octree structure are as follows. One of the main operations in AFT is an intersection check between a tetrahedron and the current front. Let us consider the sphere B covering the tetrahedron: if triangle T_i from the front intersects with tetrahedron, it intersects with B as well. For each front triangle T_i we define the sphere B_i with the center in a center of the triangle, and radius equal to the radius of the triangle. If T_i intersects with B, then B_i intersects with B. Clearly, B and B_i intersect iff the distance ρ_i between their centers is less than the sum of their radii, R and R_i, respectively. Therefore, if a triangle inter-

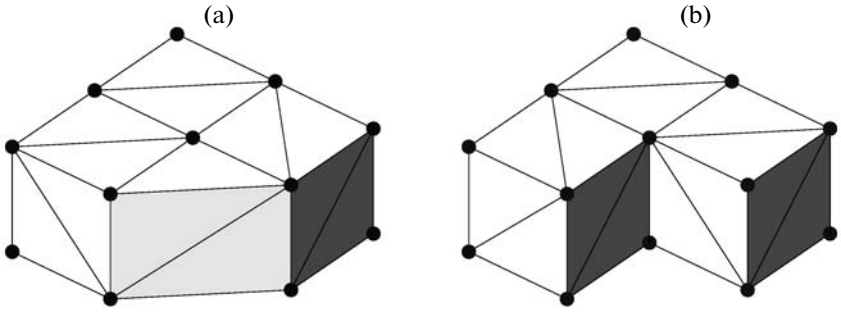

Fig. 8. Two meshes with the same sets of vertices: (a) volume mesh of the convex hull; (b) triangular surface mesh.

sects with the tetrahedron then $\rho_i \leqslant R + R_i$. Recall that $R_i \leqslant 2^{-\mathrm{lev}(i)} \cdot H$, where lev($i$) is the level of the octree vertex associated with the triangle T_i. The octree structure allows us to quickly find all triangles satisfying $\rho_i \leqslant R + 2^{-\mathrm{lev}(i)} \cdot H$. Computational cost is proportional to $(\log N + K)$, where N is the total number of triangles in octree, and K is the number of triangles satisfying the search conditions.

Thus, the computational cost of AFT step k is proportional to $\log N_k$, where N_k is the number of triangles in the current front. Total computational costs will be bounded above by the value proportional to $M \log N_{\max}$, where M is the total number of constructed tetrahedra in the mesh, and $N_{\max} = \max_k N_k$ In practice construction of quasi-uniform mesh with size h gives the following rule: $N_{\max} \sim H^2 h^{-2}$.

3. DELAUNAY-BASED TECHNIQUE

In this section we discuss the DT based method for robust volume meshing preserving boundary faces. We use it as a fallback method in case AFT fails to fully mesh the domain. Though the DT method may be applied to the entire domain, we use it only for the small parts remained after AFT meshing. The DT method is more time consuming than the AFT method, and does not provide convenient way to control the mesh size. The general idea of this method is proposed by George in [9]. We present here a simplified version of this method. We start from the set of boundary points, consider the convex hull of this set, and create the Delaunay triangulation of this convex hull. In order to recover the boundary of the domain, we intersect the surface triangulation with the volume Delaunay mesh, and split the latter. In order to restore the boundary mesh conformity, we shift newly created points inside the domain.

This algorithm is split into three independent stages: construction of the Delaunay triangulation, recovery of the domain geometry, and boundary conformity restoration. Each of these parts will be discussed in detail in the following three sections.

3.1. Delaunay Triangulation

Given a finite point set $S = \{P_1, \ldots, P_n\}$, the 3D Delaunay triangulation is such tetrahedrization $M = \{T_1, \ldots, T_m\}$, that for each tetrahedra T_i no points from S lie strictly inside the circumsphere of T_i. Different methods exist to produce the Delaunay triangulation. In Ani3D package the conventional incremental algorithm [16] is used. We start from an auxiliary initial mesh, and add points P_j into it one at a time. At each step we check and modify the mesh in order to maintain the Delaunay criterion. At the end we remove auxiliary tetrahedra from the mesh.

3.2. Geometry Recovery

At this stage we have two different meshes with the same sets of vertices. The first mesh is a tetrahedral volume mesh of a convex hull, and the second mesh is a triangular surface mesh of a domain boundary. An example of these meshes is shown in Fig. 8. The objective of the stage is to intersect both meshes, add points of intersections to the set of vertices, and split tetrahedra and triangles in such a way, that the resulting mesh will have the given boundary.

This objective is attained in two steps. First we check intersections of the tetrahedral mesh with the edges of the triangular mesh. For each edge of the surface mesh we split intersecting tetrahedra in the volume mesh to ensure that the surface edge is present in the volume mesh either entirely, or as a union of

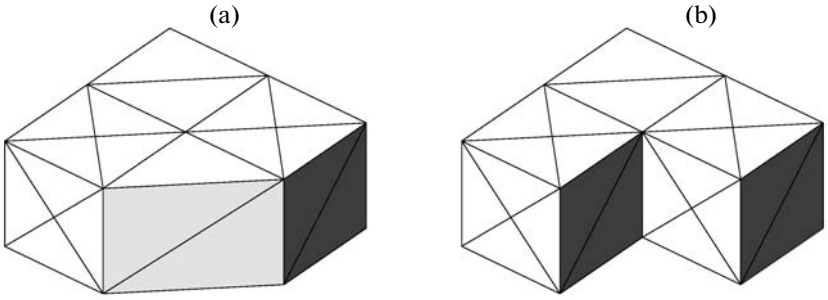

Fig. 9. Geometry recovery: (a) refined volume mesh; (b) mesh representing examined geometry.

several smaller edges. The second step is similar, we check intersections of the tetrahedral mesh with the triangles of the surface mesh. For each triangle we split intersecting tetrahedra to ensure that the surface triangle is present in the volume mesh either entirely, or as a union of smaller faces. Detailed algorithm is presented in Algorithm 4.

Each time an intersecting tetrahedron has to be split, we create an intersection point, and split only elements of the volume mesh. At the end we will get a refined volume mesh, with its faces fully covering the surface mesh.

Keeping the elements of the volume mesh lying inside the examined domain, we remove remaining elements. After that we will get the volume mesh representing the examined domain. In Fig. 9 we show the refined volume mesh, and mesh representing our geometry. We notice that the trace of this mesh on the surface may not coincide with the given surface mesh: it may have several additional points on the surface (compare Fig. 8b and Fig. 9b). These points will be removed from the boundary in the third stage.

Algorithm 4. Geometry recovery

procedure Process_Edges(V_1, V_2)
  Find tetrahedra incident to V_1, which intersect edge V_1V_2
  if new intersection point V is created **then**
    Split corresponding tetrahedra in the volume mesh
    Process_Edges(V, V_2)
  end if
end procedure

procedure Process_Faces(V_1, V_2)
  for each face F in surface mesh **do**
    if face F intersects edge V_1V_2 **then**
      Find the intersection point V
      Split corresponding tetrahedra in the volume mesh
      Process_Faces(V_1, V)
      Process_Faces(V, V_2)
      Exit
    end if
  end for
end procedure
for each edge V_1V_2 in surface mesh **do**
  Process_Edges(V_1, V_2)
end for
for each edge V_1V_2 in volume mesh **do**
  Process_Faces(V_1, V_2)
end for

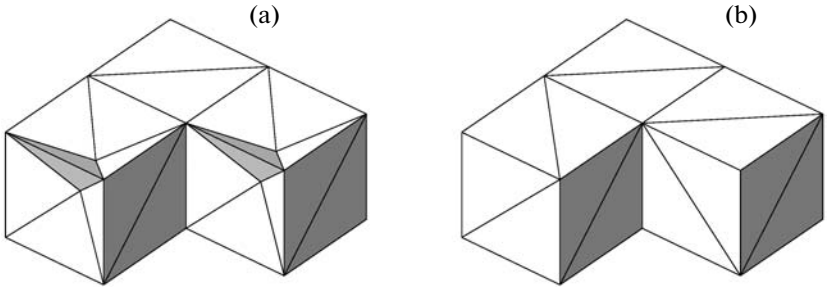

Fig. 10. Boundary faces recovery: (a) "dents" on the surface; (b) final volume mesh.

Locate and delete tetrahedra lying outside the domain.

3.3. Boundary Conformity Recovery

The third stage is the elimination of extra points from the boundary surface. This is done by shifting these points inside the domain.

All extra points are divided into two sets: those lying on the edges of the initial boundary and those lying inside the input triangular faces. The elimination in both cases is performed in a similar way. We consider in detail the first case.

Let P be a point on the surface edge that should be removed from the surface. Let us assume for simplicity that our surface mesh is a 2-manifold. With several adjustments this algorithm may be applied in general case. We shall work with a set of tetrahedra T surrounding the point P. The hull H of this set consists of triangles of two types: those lying on the surface of the domain, and those lying inside the domain. Moving P inside the domain will result in appearance of a "dent" on the surface. The dent is actually a union of two polygonal cones. In general, the bases of those cones will lie in different planes. However, if the initial surface edge and two neighbouring triangles are lying in the same plane, then the cone bases will lie in the same plane as well. An example of the dents with coplanar cone bases is shown in Fig. 10a. Polygonal cones should be additionally meshed into tetrahedra. This can be easily done, if the polygonal base of each cone is meshed into triangles without additional points. The latter is known to be always possible. These two polygonal bases form a surface part of the hull H.

We mesh these polygonal bases into triangles, and construct a new set of tetrahedra by connecting new triangles with the vertex P. These tetrahedra will be degenerated at this moment. Then we move the point P inside the domain maintaining the mesh conformity, and avoiding tangling the mesh. It is shown [9] that the point always may be shifted inside a nonempty part of the domain. The resulting volume mesh is shown in Fig. 10b.

The second case, when the point lies inside the surface triangle, differs by the fact that the dent is a single polygonal cone. The full algorithm is presented in Algorithm 5.

The proposed method always constructs the volume mesh preserving boundary faces. This mesh may contain nearly flat and nearly degenerated elements, therefore a postprocessing optimization is required.

Algorithm 5. Boundary conformity recovery

for each extra vertex V on the surface edge **do**
  Find the set T of neighbouring tetrahedra
  Determine two boundary parts of T corresponding to surface faces
  Remesh these parts into triangles eliminating vertex V
  Create new tetrahedra based on these triangles and vertex V
  Move V inside the domain restoring conformity
end for
for each extra vertex V on the surface face **do**
  Find the set T of neighbouring tetrahedra
  Determinate the boundary part of T corresponding to the surface face
  Remesh this part into triangles eliminating vertex V

Create new tetrahedra based on these triangles and vertex V

Move V inside the domain restoring conformity

end for

4. ROBUSTNESS

In this section we will discuss robustness issues which may appear in the presented volume meshing technology. First, let us assume that all arithmetic operations are exact. In this case AFT method may fail to construct a mesh for entire domain, but it will construct a valid conformal mesh for some part of the domain. The other part will be meshed by DT based method. According to George [9] the method works for arbitrary fronts, but may create nearly flat elements, still not degenerated in case of exact arithmetic.

Now let us consider the case of inexact arithmetic. In AFT method only the intersection tests heavily depend on arithmetic precision. In Ani3D package we use a fail-safe intersection test. It may wrongly report non-intersecting triangles as intersection due to round-off issues. But it will never report actually intersecting triangles as non-intersecting. Thus we slightly narrow the possibility for front advance, but we will always have a valid conformal mesh at the end. In DT based method roundoff errors may lead to wrong edge-to-face intersection, and also may result in degenerated elements after point movements. Intersection problems are mainly due to very bad shaped front left after AFT method. We added several checks into AFT method, preventing it from generating fronts with acute dihedral angles and nearly crossing edges. Still, in some rare pathological cases degenerated elements may appear after DT based method. In this case we only rely on a mesh post-processing which may untangle and improve the mesh. At the present Ani3D package does not provide any untangling post-processing tools. In practice surface remeshing or mesh size change may help to avoid degenerated elements.

A good quality surface mesh, and appropriately chosen mesh size or coarsening factor are the key factors for valid volume mesh generation.

The overall technology combines several algorithms for surface meshing and volume meshing, but it does not guarantee construction of a good quality mesh. A good quality mesh is produced by additional post-processing of the mesh. It commonly consists of smoothing, remeshing, refinement and adaptation. In this section we will make a brief overview of available methods and strategies.

Smoothing is used to improve mesh quality by moving vertices and preserving the topology. Several approaches are available. The simplest one is a Laplacian smoothing: vertices are moved to the average location of their neighbours. Instead of averaging techniques one may use some physically-based techniques, e.g. apply attraction or repulsion forces to the vertices. The above methods are heuristic: they do not guarantee quality improvement. One can use optimization-based methods to improve mesh quality as much as possible [17, 18]. However, this approach is generally more time consuming.

Remeshing is used to change the mesh topology by edge flipping, edge collapsing and other simple operations [16, 19]. It may also be used to refine the mesh in user-defined regions of interest, and other adaptation purposes.

In Ani3D package we do not provide any post-processing inside mesh generation library (AniAFT). The powerful post-processing tool is available in the metric based adaptation library (AniMBA) from the same Ani3D package. This tool may be readily applied to possibly ill-shaped meshes generated by the AFT/DT method.

5. APPLICATION EXAMPLES

In this section we present several examples of application of our technology. All the tests were carried out using Ani3D package on a Pentium D 2.8 GHz machine. The first example is used to estimate the time complexity of AFT method. The second example illustrates both AFT and Delaunay-based methods used together. In the third example we apply surface mesh coarsening on input and mesh optimization on output. The fourth example demonstrates the use of CGM interface with OpenCASCADE.

In each case we calculate the quality of the mesh. We use the following formula for tetrahedral cell quality:

$$q_T = 36\sqrt{2}\frac{V}{\Sigma l_{ij}^3}.$$

In this formula V is the oriented volume of tetrahedron, and l_{ij} is the length of ij edge. The value of q_T normally lies in [0, 1]. For equilateral tetrahedra $q_T = 1$, for degenerated tetrahedra $q_T = 0$, and for tangled

Table 1.

N_F	N_T	Time, s	Time/N_T, s
132	132	0.03	2.273×10^{-4}
508	844	0.22	2.607×10^{-4}
2034	4163	1.05	2.522×10^{-4}
8074	18759	4.77	2.543×10^{-4}
32874	84006	21.92	2.609×10^{-4}
132832	357559	95.67	2.676×10^{-4}
537180	1524614	429.58	2.818×10^{-4}

Table 2.

N_T	$q > 10^{-1}$	$q > 10^{-2}$	$q > 10^{-3}$	$q > 10^{-4}$
1524614	1522691	1836	82	5

tetrahedra $q_T < 0$. With respect to entire mesh, we are interested in minimal cell quality $q_{\min}$ and in quality distribution. We use logarithmic scale, as it gives-better understanding of the distribution.

5.1. Time Complexity of AFT

In this section we estimate the average time complexity of AFT algorithm. We run series of tests and measure the computational time of AFT volume mesh generation. In all tests we use the same unit cube as the geometrical domain. In each subsequent test we decrease the mesh size of the initial front by two. Therefore the number of triangles (N_F) in initial front increases roughly by a factor of four. We use AFT method with automatic mesh coarsening inside the domain with coarsening factor $c = 1.2$. In this case most of the tetrahedra are gathered near the boundary, and the number of tetrahedra (N_T) increases about the same speed as the number of triangles.

For each test run we measure the generation time and compute the average time per tetrahedron. The results are presented in Table 1. As we can see, the average time per one tetrahedron does not depends strongly on the number of front triangles.

In the last test AFT method started from the initial front with 537180 triangles and finished with 2770 triangles in the front. It meshed only 99.87% of the volume. In Table 2 we present the quality distribution of the mesh. The minimal cell quality in the meshed part is $q_{\min} = 1.121 \times 10^{-4}$. Only five tetrahedra have quality smaller than 10^{-3}, $N_F = 537180$.

5.2. Combination of AFT and DT Methods

In this section we illustrate the combination of the two methods. We apply AFT method to construct mesh in the most part of the domain. Then we mesh the remaining parts using DT technique.

We start from an initial surface mesh of a 3D scanned dragon model, which is shown in Fig. 11a. It has 54296 vertices and 108582 triangles. All vertices belong to the set of equally distanced XY-planes. In each plane vertices are connected with edges forming a contour line of the model. Neighbouring contour lines are connected with triangles forming conformal surface mesh.

At the first stage, we apply AFT method with automatic coarsening inside the domain with coarsening factor $c = 1.3$. AFT meshes only 99.67% of the volume constructing 250461 tetrahedra and leaving 1718 triangles in the final front. The final front and the unmeshed part of the domain are shown in Fig. 11b. The resulted mesh quality is presented in the first row of Table 3. The minimal tetrahedra quality is $q_{\min} = 3.698 \times 10^{-7}$.

At the second stage we apply DT based method in order to mesh the remaining part. The unmeshed part consists of several compact regions with bad shape. Therefore the DT method provides the mesh with very poor quality. However, this mesh is less than 1% of the volume. The quality of the whole mesh after

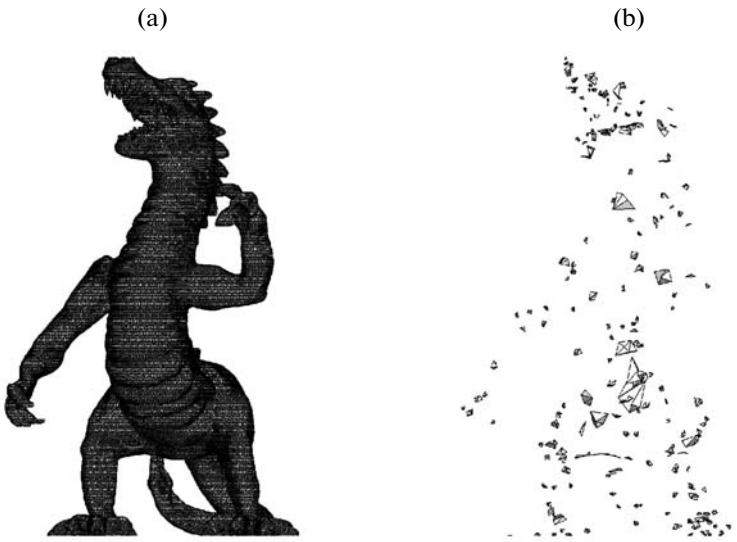

Fig. 11. Dragon model. Application of AFT method: (a) initial front; (b) final front.

both AFT and DT methods is shown in the second row of Table 3. The minimal cell quality is now $q_{\min} = 1.232 \times 10^{-14}$.

At the final stage we apply postprocessing mesh smoothing based on AniMBA library from Ani3D project. Results are shown in the third row of Table 3. The minimal cell quality is $q_{\min} = 1.910 \times 10^{-7}$, which is close to the minimal quality after AFT method, but in this case the whole domain is meshed. The final mesh has 75 665 vertices, 108582 boundary faces, and 267 475 tetrahedra.

5.3. Surface Coarsening

In this section we illustrate the use of surface mesh pre-processing and volume mesh post-processing. We start from the same initial surface mesh used in the previous example (see Fig. 11a). Suppose we are not interested in such fine surface mesh, and we want a more coarse mesh both on the surface and inside the domain. The initial mesh has 108 582 triangles on the surface. We apply surface mesh coarsening method discussed in Section 1.3. The resulting mesh (see Fig. 12a) has only 54 412 triangles, nearly two times smaller.

The combination of AFT and DT methods is used for volume meshing. At the first stage, we apply AFT method with automatic coarsening inside the domain with coarsening factor $c = 1.6$. AFT meshes only 99.36% of the volume constructing 102 094 tetrahedra and leaving 1536 triangles in the front. The mesh quality after AFT method is presented in the first row of Table 4. The minimal cell quality is $q_{\min} = 1.965 \times 10^{-7}$.

At the second stage we apply DT based method in order to mesh the remaining part. The mesh quality after both AFT and DT methods is shown in the second row of Table 4. The minimal tetrahedra quality is now $q_{\min} = 1.129 \times 10^{-9}$.

At the post-processing stage we use mesh smoothing based on AniMBA library. Mesh quality is shown in the third row of Table 4. The minimal cell quality is $q_{\min} = 8.268 \times 10^{-8}$. The final mesh has 39 417 vertices, 54 412 boundary faces, and 125383 tetrahedra. In Fig. 12b we present the cross section of the volume mesh.

Table 3.

Mesh	$q_{\min}$	N_T	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}
AFT	3.699×10^{-7}	250461	248359	1923	166	11	1	0	1
AFT + DT	1.232×10^{-14}	254764	249674	3428	1020	340	138	71	93
Smoothing	1.910×10^{-7}	267475	258449	8269	718	30	6	0	3

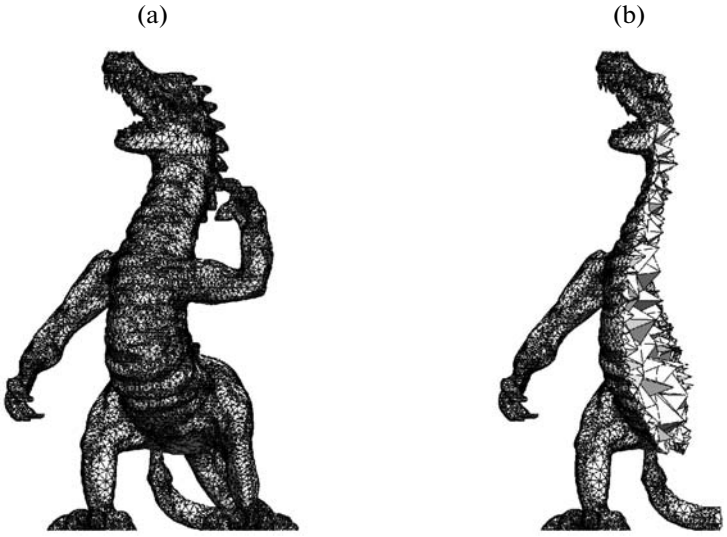

Fig. 12. Coarse dragon mesh: (a) coarsened survade mesh; (b) cross section of a volume.

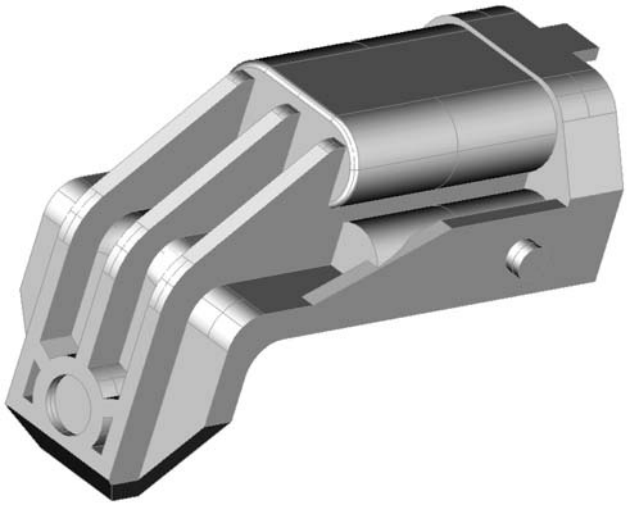

Fig. 13. Original model in Open CASCADE CAD kernel.

5.4. Interface with CAD System

In this section we illustrate the use of an interface with OpenCASCADE CAD kernel. We tested our CGM integration on a sample model 62_shaver1 from OpenCASCADE website [13]. The BREP model has 266 vertices, 403 edges, and 153 faces (see Fig. 13). The surface was meshed into a quasiuniform mesh with 38 335 vertices and 76 666 triangles (see Fig. 14).

The combination of AFT and DT methods is used for volume meshing. At the first stage, we apply AFT method with automatic coarsening inside the domain with coarsening factor $c = 1.0$. AFT meshes only

Table 4.

Mesh	$q_{\min}$	N_T	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}
AFT	1.965×10^{-7}	102094	100733	1298	59	3	0	0	1	–
AFT + DT	1.129×10^{-9}	105852	101942	2752	758	269	68	32	22	9
Smoothing	8.268×10^{-8}	125383	115179	9945	244	13	0	0	0	2

Fig. 14. Surface mesh of a CAD model.

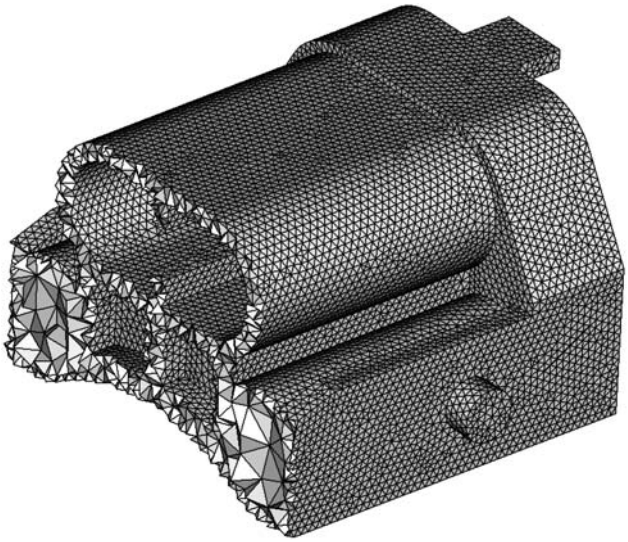

Fig. 15. Cross section of the volume mesh.

99.84% of the volume constructing 206 562 tetrahedra and leaving 830 triangles in the front. At the second stage we apply DT based method in order to mesh the remaining part. The minimal tetrahedra quality after both AFT and DT methods is $q_{\min} = 1.044 \times 10^{-10}$.

At the post-processing stage we use mesh smoothing based on AniMBA library, and mesh quality highly improves. The minimal cell quality is now $q_{\min} = 1.780 \times 10^{-2}$. The final mesh has 60 204 vertices, 76 666 boundary faces, and 229 554 tetrahedra. In Fig. 15 we present a cross section of the volume mesh.

Total time of mesh construction is 12 min 30 s. Time distribution is as follows: surface meshing—8 min 12 s, volume meshing—1 m 53 s, post-processing improvement—10 s. The rest of the time was spent on final mesh topology check and other auxiliary operations.

6. CONCLUSIONS

In this paper we proposed the technology for unstructured tetrahedral mesh generation. We discussed several boundary discretization methods, including analytical definition, interfacing with CAD system,

discrete surface modification, and CSG-based surface meshing. For robust volume meshing we proposed the combination of two techniques: AFT as the primary method and DT as the fallback method. In combination with the post-processing tools, this technology produces tetrahedral meshes with good quality. Several application examples were demonstrated.

ACKNOWLEDGMENTS

This research was partly supported by the Russian Foundation for Basic Research through grant 08-01-00159-a and by RAS program "Optimal methods for problem of mathematical physics."

The author thanks R. Garimella for the assistance with the abstracting interface layer with CAD systems, and K. Nikitin for the CSG technology software. The author is grateful to Yu. Vassilevski for the long-term support, fruitful discussions and valuable comments.

REFERENCES

1. E. Schönhardt, "Über die Zerlegung von Dreieckspolyedern in Tetraeder," Math. Ann. **98**, 309–312 (1928).
2. B. Joe, "Three-Dimensional Boundary-Constrained Triangulations," in *Proc. 13th IMACS World Congress, 1992*, pp. 215–222.
3. P.-L. George and H. Borouchaki, "Maillage Simplicial d'un Polyédre Arbitraire," C. R. Acad. Sci. Paris, Ser. I **338**, 735–740 (2004).
4. H. Borouchaki, F. Hecht, E. Saltel, and P.-L. George, "Reasonably Efficient Delaunay Based Mesh Generator in 3 Dimensions," in *Proc. 4th Inernat. Meshing Roundtable, 1995*, pp. 3–14.
5. Q. Du and D. Wang, "Recent Progress in Robust and Quality Delaunay Mesh Generation," J. Comput. Appl. Math. **195**, 8–23 (2006).
6. J. R. Shewchuk, "Constrained Delaunay Tetrahedralizations and Provably Good Boundary Recovery," in *Proc. 11th Internat. Meshing Roundtable, 2002*, pp. 193–204.
7. Y. Ito, A. Shih, and B. Soni, "Reliable Isotropic Tetrahedral Mesh Generation Based on an Advancing Front Method," in *Proc. 13th Internat. Meshing Roundtable, 2004,* pp. 95–106.
8. Y. Yang, J. Yong, and J. Sun, "An Algorithm for Tetrahedral Mesh Generation Based on Conforming Constrained Delaunay Tetrahedralization," Comput. Graphics **29**, 606–615 (2005).
9. P.-L. George, H. Borouchaki, and E. Saltel, "Ultimate Robustness in Meshing an Arbitrary Polyhedron," Internat. J. Numer. Meth. Engng. **58**, 1061–1089 (2003).
10. 3D Generator of Anisotropic Meshes, http://sourceforge.net/projects/ani3d/
11. 2D Generator of Anisotropic Meshes, http://sourceforge.net/projects/ani2d/
12. The Common Geometry Module, http://cubit.sandia.gov/cgm. html
13. Open CASCADE Technology, http://www.opencascade.org/
14. Yu. Vassilevski, A. Vershinin, A. Danilov, and A. Plenkin, *Tetrahedral Mesh Generation in Domains Defined in CAD Systems* (Inst. Num. Math., Moscow, 2005), pp. 21–32 [in Russian].
15. K. Nikitin, "Surface Meshing Technology for Domains Composed of Primitives," *Diploma Thesis* (Dept. Mech. Math., MSU, Moscow, 2007) [in Russian].
16. P.-L. George and H. Borouchaki, "Delaunay Triangulation and Meshing," in *Application to Finite Elements* (Hermes, 1998).
17. S. A. Ivanenko, "Variational Methods for Adaptive Grid Generation," Comput. Math. Math. Phys. **42** (6), 830–844 (2003).
18. V. Garanzha, "Max-Norm Optimization of Spatial Mappings with Application to Grid Generation, Construction of Surfaces and Shape Design," in *Proc. Minisymp. in Int. Conf. OFEA, 2001*, pp. 61–74.
19. A. Agouzal, K. Lipnikov, and Yu. Vassilevski, "Adaptive Generation of Quasi-Optimal Tetrahe-dral Meshes," East-West J. **7**, 223–244 (1999).