Adaptive mesh generation using shrink wrapping approach

A. Malyshev[1], A. Zhidkov[2], V. Turlapov[1]
al.s.malyshev@gmail.com|artem.zhidkov@opencascade.com|vadim.turlapov@itmm.unn.ru
[1] Lobachevsky State University of Nizhny Novgorod, Nizhny Novgorod, Russia
[2] Open Cascade, Guyancourt, France

We consider the problem of mesh preparation for subsequent simulations. The Cartesian shrink-wrapping technique is used to construct a triangular mesh on dirty input geometry. Supported imperfections are gaps, overlaps, inconsistent normal orientations and self-intersections. Wrapping is stored using octree. An octree is refined adaptively to follow a form of the original mesh. Initial watertight mesh which is extracted from octree is improved using nodes projection to original triangulation and by local operations such as swap edge, collapse edge, and Laplacian smoothing.

Keywords: *shrink wrapping, octree, bounding volumes hierarchy.*

1. Introduction

Significant performance boost of the computers attracted close attention to automatic mesh generation. Mesh processing covers such engineering practices as mesh healing, mesh construction from triangles soup, adapting design models for simulation etc. The necessity of healing often emerges after data exchange different CAD application. Diverse tolerance handling approaches may ruin model for further use obtained via neutral formats like STEP or IGES.

Usually, healing is performed manually, and this process is tedious and time-consuming. The proposed approach allows reducing the number of manual operations related to detecting and fixing of self-intersections, gaps, overlaps and inconsistent normal orientations.

Generally, there are two main ways to deal with imperfections. Methods of the first group modify existing mesh or CAD. Another group is intended to build a new object, which is close to the original one. The second approach gives more freedom, so it is used in most cases.

On the CAD level, some clean-up simplification techniques were introduced using a virtual topology concept [14]. For meshes, various re-meshing articles were published ([3] and [12]). The common point between them is that they cannot deal with all kinds of topological and geometric problems simultaneously. It is necessary to produce a pipeline of methods where each algorithm fixes only some defects.

The robust industrial healing algorithms are often based on the volumetric representation. Initially, Kobbelt suggested shrink-wrapping concept in [8]. His idea was to wrap a thin deformable surface around the initial mesh. Wang [18] carried out research based Cartesian grid approach to deal with some imperfections. In that study, all the Cartesian cells intersecting the entity are refined recursively to satisfy cell size criterion. Bischoff [1] presented a method to extract manifold meshes from architectural models using dual contouring [5].

Lee [10] overcame the most of these issues using a Cartesian grid approach. Grid size functions ([19] and [20]) were utilized to refine cell adaptively. They took outer (or inner) wrapping constructed on the Cartesian grid of the input mesh as the original watertight deformable surface. After that, the watertight mesh is projected to the surface. At the end, the mesh is refined to obtain better skewness and aspect ratio. The method is based on the Wang's [18] interior-to-boundary approach.

Recent research of Juretić [6] was connected to holes detection based on the intensity of heat fluxes in various places of the input surface mesh. Another way of holes handling was proposed by Martineau [11] where the wrapping and volumetric mesh generator are combined.

Octree usage ([13], [15]) is the well-established technique to represent arbitrary free-form objects in the three-dimensional space. Octree fills the space using hierarchically structured axis-aligned boxes. Lee [10] states that octree construction requires more CPU time compared to the Cartesian grid but the lower amount of memory. In our study, we use octrees to build the adaptive representation of input mesh due to the following reasons:

- Construction simplicity.
- Effective parallelization. Nodes of octree do not depend on each other so that they can be processed in a parallel mode.
- Low memory consumption.

Many computational problems such as initial watertight nodes projection, self-intersection state checking or neighbors detection for a particular node can be effectively solved using accelerating data structures like bounding volumes hierarchy ([2], [9] and [16]). The idea of BVH is to subdivide a set of primitives built on input object recursively until some target criterion is met. In case of mesh input, as we have, no mesh conversion is needed since it is possible to reflect the mesh triangle as a primitive of BVH.

Usually, the result of wrapping is not directly suitable for analysis because of insufficient quality of mesh triangles. Local operators like edge swap, edge collapse [17] are used to improve the quality of the resulting mesh. Also, it is possible to use some decimation techniques [7], but they are not investigated in this study. If the quality of the resulting mesh is not an aim, this algorithm can be applied as automatic feature suppressor [11].

In our study, we present a "boundary-to-interior" method according to Wang terminology [18]. The idea is to convert the initial model to the volumetric form presented as octree and extract initial outer watertight mesh. After that, initial mesh is projected to the original mesh, and possible self-intersections are eliminated. At the end of the algorithm, mesh quality is improved.

2. Algorithm

The input for our algorithm is a mesh presented in stereolithography (STL) format. The majority of CAD systems provide the possibility to save faceted representation in that format due to its simplicity and portability.
The algorithm has one parameter to tune – minimal allowed voxel size. This parameter determines the maximal gap which algorithm can cover.
The algorithm has three main stages:

- Construction of original watertight mesh.
- Projection + Self-intersections elimination.

Работа опубликована при финансовой поддержке РФФИ, грант 18-07-20045\18

- Mesh quality improvements.

2.1 Watertight mesh construction

The octree is a tree-like data structure where each non-leaf node has 8 children. Figure 1 illustrates the octree structure.

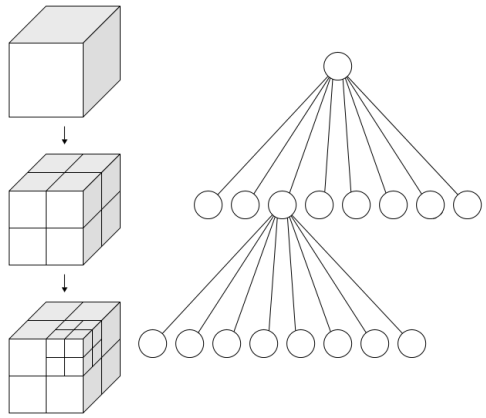

Figure 1: Octree structure.

The root node of voxel octree corresponds to the axis-aligned bounding box of dirty input mesh. The tree is constructed layer by layer and nodes of fixed depth are checked for subdividing in a parallel way. The number of nodes grows exponentially (up to 8^k nodes on the k layer). Adaptivity may significantly decrease this value. Our experiments show linear speed up on trees with more than one hundred thousand nodes. Subdividing is made taking into account the following principles:

- It is not necessary to subdivide nodes which do not intersect input triangles. Subdivision requires effective box-mesh intersection checks. In our research, it is done using BVH where triangles of the input dirty mesh are used as primitives. Separation axis theorem is used to check intersection state between triangles and boxes. Let N be the number triangles in input mesh and M – the number of nodes in the octree. BVH allows checking intersection state in $O(log(N))$ time for each box in case of proper balancing. The tree itself can be built on linear-logarithmic time. So, the full complexity of this operation in case of BVH usage is $O(N * log(N) + M * \log(N))$ while direct naive implementation requires $O(N * M)$ operations.
- Subdivision of intersecting nodes is not obligatory in case of quasi-planar intersecting triangles. MCAD models deal with canonical geometry, and it is a usual case to deal with (Figure 2 and Figure 3). Subdivision check is implemented by calculating the maximal angle between normals of triangles intersecting the current node. Besides, it is possible to construct gauss maps to speed up this check.

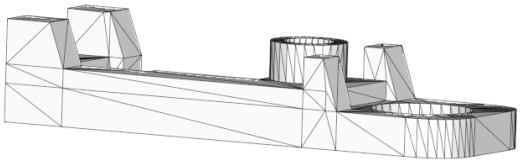

Figure 2: Faceted representation of MBB Gehause rohteil model with planar faces.

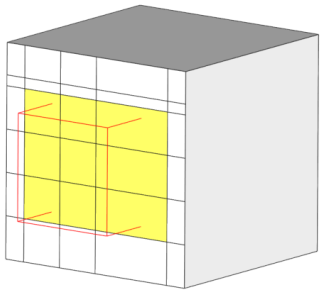

Figure 3: Demonstration of octree node which should not be subdivided (red lines) and intersecting facets of input model (yellow color).

Outer mesh extraction is hardly possible for arbitrary octree. The nature of this problem is connected to possible size jumps of adjacent nodes. In general, their sizes can have 2^n: 1 gradation (Figure 4).

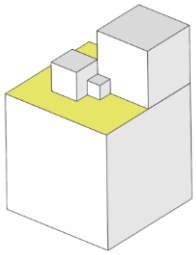

Figure 4: Octree with 8:1 relation. Outer contour extraction will add many points to the yellow face.

Again, effective neighbor detection is based on BVH data structure. For that, leaf nodes are converted to BVH primitives, and simple box intersections are used to check intersection. All neighbors for a particular node can be detected as all nodes intersecting enlarged bounding box of the current node. Since we know the minimal allowed cell size, it is easy to avoid numerical stability problems increasing bounding box by some value smaller than minimal size ("0.1 * minimal size" for example). The result of that process is demonstrated by Figure 5.

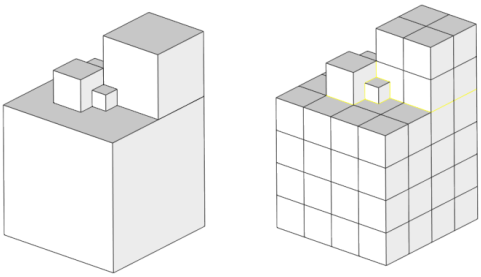

Figure 5: Model conversion to 2:1 form. Original node connections are marked by yellow color, from left to right.

One additional problem should be solved before outer contour extraction. Usually, adjacent nodes have non-manifold connections. In our paper conformity restoration is performed by adding new cells. The resulting tree might break 2:1 gradation rule and its restoration is needed. In general, initial polygons can have up to 8 points. Polygons are not tessellated during initial wrapping construction; it is performed after projection. It allows getting better quality mesh since in projection optimal tessellation may change. To conclude, watertight mesh construction involves the following steps:

1. Construct initial octree.
2. Convert octree to 2:1 form.
3. Resolve non-manifold joints.

The publication of the conference proceedings was supported by the RFBR, grant No. 18-07-20045\18

4. Convert octree to 2:1 form again.
5. Extract outer or inner wrapping.

2.2 Projection and self-intersection elimination

The first and the most critical step in watertight mesh improvement is projection. Nowadays, typical watertight wrapped mesh consists of million triangles. The efficiency of the projection procedure is crucial for the construction of a real industrial application. Direct exhaustive implementation of the projection algorithm requires $O(M * N)$ operations. As an alternative to it, BVH-driven projection gives about $M * \log(N)$ time complexity.

As mentioned above, the projection procedure comprises several healing steps like sticking polygons treatment. Usually, sticking polygons appear when input geometry has a substantial amount of planar places. The list of problems and their origins is presented below:

- Two polygons with equal nodes – move polygon nodes back in projection directions. It happens when the whole polygon is positioned at shell part of input mesh.
- A polygon with wrong normal – remove it. Non-manifold joints resolving adds new nodes some of them are placed inside the input geometry. Another source of that problem is sharp and complicated input geometry.
- A polygon with sticking nodes – collapse degeneracy. It arises when an edge of a polygon is orthogonal to input mesh.
- A polygon without area – remove it. It comes when the original non-projected polygon is entirely orthogonal to input mesh.

It turns out that nearest projection leads to a situation when neighbor nodes are projected in opposite directions. Sometimes, it indicates insufficient amount of octree refinement steps but, typically, the best cell size is unknown. That is why this issue is addressed. First, the accumulated length of node's outgoing edges is calculated, and it is compared with a perimeter of faces containing these nodes. If these two values are too far from each other new position will be assigned to the node using smoothing. Node is projected to original mesh once again after smoothing to sure that node is lies on the input triangulation. Wrong projection incident is shown in Figure 6.

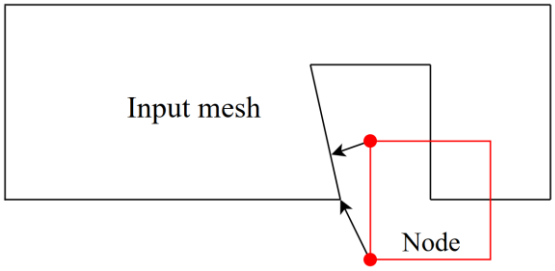

Figure 6: Wrong projection.

The last step of projection routine is self-intersection correction. Unlike the previous problems before, this issue is more geometric than topological. It means that implementation should be robust enough to fight with numerical issues raised from floating point operations performed using IEEE 754 standard. This idea should be kept in mind while developing algorithms for detection and fixing of self-intersections. Self-intersections are obtained using BVH tree to reduce complexity from quadratic to linear-logarithmic. The intersecting pairs are grouped into clusters according to their adjacency. Each cluster is processed independently in a parallel mode that decreases execution time. Intersections are amended using smoothing. The idea is to keep cluster boundary untouched and modify only internal nodes.

2.3 Mesh quality improvements

Usually, outer wrapping is not applicable for analysis after projection due to near degenerated polygons. The ordinary way to improve mesh quality is to apply local topological operators like edge swap, face collapse etc. The following metrics are utilized to check mesh quality:

- Skewness. $skewness = 1 - \frac{2r}{R}$, where r is the radius of the inscribed circle and R is the radius of the circumscribed circle.
- Aspect ratio. $aspect\ ratio = \frac{L}{l}$, where L is the length of the longest side, l is the length of the shortest side.

Thanks to polygons usage on the previous steps of the algorithm, it is possible to build different triangulations and get better mesh quality. The triangulation algorithm is based on the ear-slicing approach [4]. It tessellates arbitrary convex or concave polygon to triangles but in a non-optimal manner. Greedy-algorithm is used to determine the best ear to slice since brute-force check of all possible tessellations is computationally ineffective. Despite the execution time, some workflows like volumetric mesh generation may benefit from optimal tessellation because the mesh is constructed by increasing dimensions from lower to higher and excellent quality in the previous dimension is mandatory to build fine mesh in upper dimensions.

The local mesh operators like edge collapse are used to improve the quality of tessellated triangular mesh. In our algorithm the following operators are implemented:

- Collapse edge. The lengths of all neighbors are calculated for the target edge. If an edge is several times shorter than all its neighbors, the target edge will be collapsed.
- Split edge. Neighbor lengths are calculated like in case before. If the target edge's length is significantly bigger than neighbor lengths a new node will be introduced in edge, while two adjacent faces containing the target edge will be split into four triangles.
- Swap edge. Sometimes two faces containing target edge may be improved by connecting non-shared nodes of triangles and removing existing target edge.

All mentioned operations are safe from self-intersection perspective. It is checked that the mesh is not corrupted during modification. In general, this leads to a situation when arbitrary mesh cannot be converted to an optimal state.

3. Mesh generation examples

Our algorithm is capable of generating watertight meshes for many types of imperfections like non-manifold topology or overlaps. The examples presented in that section are intended to demonstrate obtained performance and applicability of our method for industrial use.

3.1 Geometric primitives: sphere, cylinder, and cube

These cases are aimed to demonstrate basic wrapping abilities and robustness against common defects. The initial mesh, shown in Figure 7, has a missing triangle, that situation emulates erroneous data translation from one format to another. The sphere has a radius of 0.1 m, and the longest side of the missing triangle has a 0.0018 m length. This model is processed with 0.005 m cell size, and the result is shown in Figure 8.

Работа опубликована при финансовой поддержке РФФИ, грант 18-07-20045\18

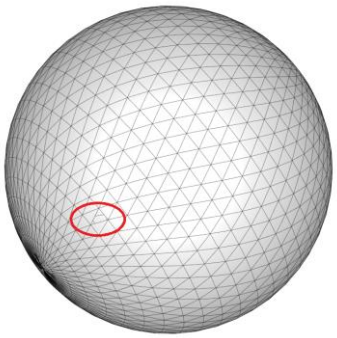

Figure 7: Sphere with missing triangle (marked by the red ellipse).

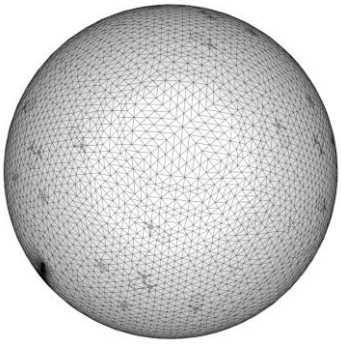

Figure 8: Wrapped mesh.

The mesh presented in the Figure 9 demonstrates the result of inaccurate modeling when two objects are mutually intersecting. This artificial model is constructed using two orthogonal cylinders with a radius of 0.01 m and 0.1 m in length. Wrapped mesh generated using 0.001 m cell size fixes the problem. It should be noted, that this case also shows the conversion possibility from CAD-based mesh to its computer graphics equivalent. The result of wrapping is shown in Figure 10.

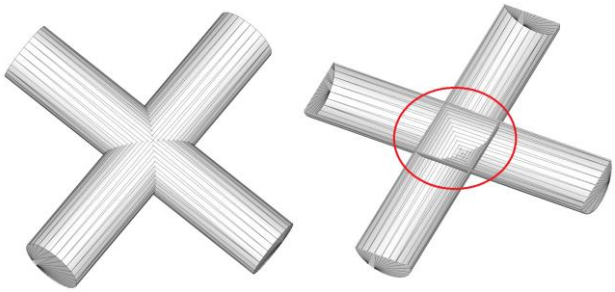

Figure 9: Cylinder models and cylinders with clipping planes (intersection is marked by red color), from left to right.

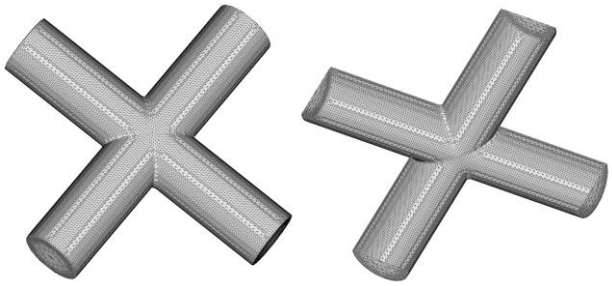

Figure 10: Wrapped cylinders and clipped cylinders, from left to right.

The cube model displayed on the left side of Figure 11 has 0.1 m size length. This model is used to compare execution time, the number of triangles and peak memory consumption of uniform and adaptive versions of the wrapping. Table 1 shows almost linear increasing of all tracked parameters; however, Table 2 demonstrates square complexity for uniform octree construction.

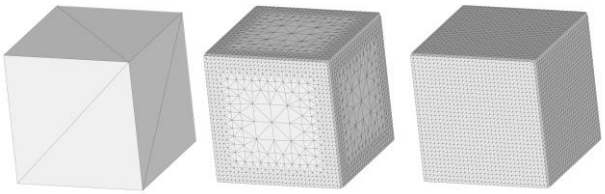

Figure 11: Original box, adaptive wrapping and uniform wrapping from left to right.

Cell size, mm:	Execution time, sec:	Max used memory, MiB:	Number of triangles:
0.5	0.91	44	6912
0.25	2.08	54	16224
0.125	4.50	75	35520
0.0625	9.76	125	74784

Table 1: Adaptive wrapping of the cube model.

Cell size, mm:	Execution time, sec:	Max used memory, MiB:	Number of triangles:
0.5	1.35	51	12288
0.25	5.86	94	49152
0.125	23.60	249	196608
0.0625	96.37	959	786432

Table 2: Uniform wrapping of the cube model.

3.2 Industrial model: engine

The following example is designed to reveal envelope construction capabilities on the real industrial model. The model shown in Figure 12 is the wright whirlwind radial engine. The model sizes are equal to 250, 300 and 550 mm for X, Y and Z axis correspondingly. The CAD model containing 1002 parts is meshed, and the input mesh has 1635339 triangles. This mesh is wrapped with cell size equal to 1.0 mm. The resulting mesh presented in Figure 13 has 1120166 triangles and 559839 nodes. Multiple problems with non-valid topology, self-intersecting mesh are detected and healed during envelope construction.

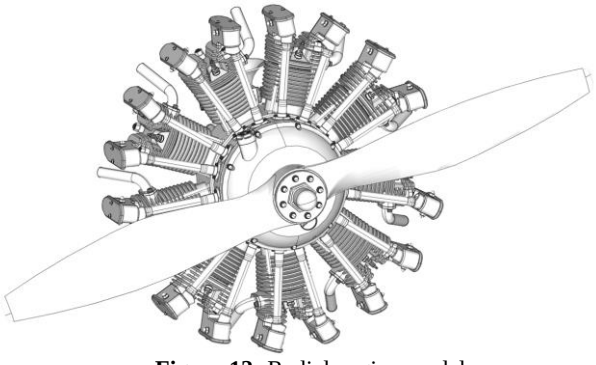

Figure 12: Radial engine model.

The publication of the conference proceedings was supported by the RFBR, grant No. 18-07-20045\18

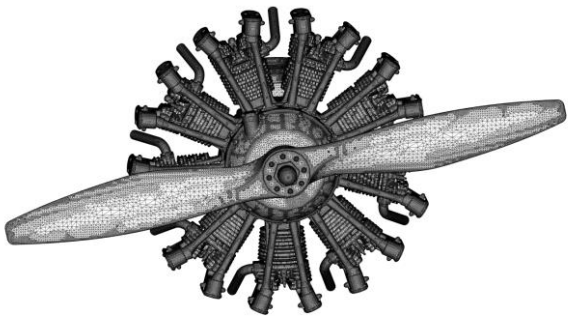

Figure 13: Engine model wrapping, constructed in 48 minutes and 44 seconds on core i5-3450 CPU.

4. Conclusion and further work

This article presents a mesh generation method for surface wrapping on dirty models. No assumptions about quality of input mesh are made. The proposed approach fixes both geometrical and topological problems such as overlaps, open boundary edges, and non-manifold configurations. The octree is used to construct adaptive outer wrapping. The quality of the resulting mesh is controlled by the minimal allowed cell size, and this value defines the gap filling capability of the algorithm. Initial outer wrapping is refined to reach computational mesh quality using local operators, such as edge collapse, edge split, and edge swap. Laplacian smoothing is used for eliminating self-intersections and deeper refinement of the mesh quality.

The method can be further improved in several ways. Some sub-routines should be investigated to be used in parallel mode. Memory usage is another room for improvement since existing implementation is not optimized from a memory point of view.

In many practical cases, model features like sharp edges preserving should be kept, but no delicate algorithms are investigated in that study. The present approach smooths input geometry which is considered as an automatic feature suppression mechanism.

The topological state of the result of the wrapping and meshing can be enhanced by the usage of proximity size functions proposed by Zhu [20]. Specialized algorithms for efficient proximity computation should be developed.

Finally, the proposed algorithm can be adapted for various case studies like protection of intellectual properties, model simplification, and preparation for thermal, fluid or aero analysis.

5. References

[1] Aichholzer, O. and Aurenhammer, F. 1996. Straight skeletons for general polygonal figures in the plane. Computing and Combinatorics, Lecture Notes in Computer Science Volume 1090. (1996), pp 117-126.

[2] Chen, X.D., Yong, J.H., Wang, G., Paul, J.C. and Xu, G. 2008. Computing the minimum distance between a point and a NURBS curve. CAD Computer Aided Design. 40, 10–11 (2008), 1051–1054.

[3] Coupez, T., Digonnet, H. and Ducloux, R. 2000. Parallel meshing and remeshing. Applied Mathematical Modelling. 25, 2 (2000), 153–175.

[4] ElGindy, H., Everett, H. and Toussaint, G. 1993. Slicing an ear using prune-and-search. Pattern Recognition Letters. 14, 9 (1993), 719–722.

[5] Ju, T., Losasso, F., Schaefer, S. and Warren, J. 2002. Dual contouring of hermite data. ACM Transactions on Graphics. 21, 3 (2002).

[6] Juretić, F. and Putz, N. 2014. Applications of heat-diffusion equation for surface wrapping: Hole detection and normal orientation. Engineering with Computers. 30, 3 (2014), 363–374.

[7] Kobbelt, L., Campagna, S. and Seidel, H.-P. 1998. A General Framework for Mesh Decimation. Graphics Interface. (1998), 43–50.

[8] Kobbelt, L.P., Vorsatz, J., Labsik, U. and Seidel, H.-P. 1999. A Shrink Wrapping Approach to Remeshing Polygonal Surfaces. Computer Graphics Forum. 18, 3 (1999), 119–130.

[9] Lauterbach, C., Garland, M., Sengupta, S., Luebke, D. and Manocha, D. 2009. Fast BVH construction on GPUs. Computer Graphics Forum. 28, 2 (2009), 375–384.

[10] Lee, Y.K., Lim, C.K., Ghazialam, H., Vardhan, H. and Eklund, E. 2010. Surface mesh generation for dirty geometries by the Cartesian shrink-wrapping technique. Engineering with Computers. 26, 4 (2010), 377–390.

[11] Malyshev, A., Slyadnev, S. and Turlapov, V. 2017. Graph-based feature recognition and suppression on the solid models. GraphiCon 2017. (2017), 319–322.

[12] Peyré, G. and Cohen, L.D. 2006. Geodesic remeshing using front propagation. International Journal of Computer Vision. 69, 1 (2006), 145–156.

[13] Schneiders, R., Schindler, R. and Weiler, F. 1996. Octree-based Generation of Hexahedral Element Meshes. Meshing Roundtable. December 1999 (1996), 205–216.

[14] Sheffer, A., Bercovier, M., Blacker, T. and Clements, J. 2000. Virtual topology operators for meshing. International Journal of Computational Geometry & Applications. 10, 3 (Jun. 2000), 309–331.

[15] Shephard, M.S. and Georges, M.K. 1991. Automatic three-dimensional mesh generation by the finite octree technique. International Journal for Numerical Methods in Engineering. 32, 4 (1991), 709–749.

[16] Teschner, M., Kimmerle, S., Heidelberger, B., Zachmann, G., Raghupathi, L., Fuhrmann, A., Cani, M.P., Faure, F., Magnenat-Thalmann, N., Strasser, W. and Volino, P. 2005. Collision detection for deformable objects. Computer Graphics Forum. 24, 1 (2005), 61–81.

[17] Wang, D., Hassan, O., Morgan, K. and Weatherill, N. 2006. Enhanced remeshing from STL files with applications to surface grid generation. Communications in Numerical Methods in Engineering. 23, 3 (Sep. 2006), 227–239.

[18] Wang, Z.J. and Srinivasan, K. 2002. An adaptive Cartesian grid generation method for “dirty” geometry. International Journal for Numerical Methods in Fluids. 39, 8 (2002), 703–717.

[19] Zhu, J. 2003. A new type of size function respecting premeshed entities. Proceedings of the 11th international meshing roundtable. (2003), 403–413.

[20] Zhu, J., Blacker, T. and Smith, R. 2002. Background overlay grid size functions. Proceedings of the 11th International Meshing Roundtable. (2002), 65–73.

About authors

Vadim Turlapov (vadim.turlapov@itmm.unn.ru): Professor of Computer Science, Lobachevsky Nizhny Novgorod University.

Alexander Malyshev (al.s.malyshev@gmail.com): Ph.D. student, Lobachevsky Nizhny Novgorod University.

Artem Zhidkov (artem.zhidkov@opencascade.com): Ph.D., OpenCascade, France.

Работа опубликована при финансовой поддержке РФФИ, грант 18-07-20045\18