Barbara Głut [ID]
Tomasz Jurczyk [ID]

TREE-BASED CONTROL SPACE STRUCTURES FOR DISCRETE METRIC SOURCES IN 3D MESHING

Abstract   *This article compares the different variations of the octree and kd-tree structures used to create a control space based on a set of discrete metric point-sources. The created control space supervises the generation of the mesh, providing efficient access to the required information on the desired shape and size of the mesh elements at each point of the discretized domain. The structures are compared in terms of computational and memory complexity as well as regarding the accuracy of the approximation of the set of discrete metric sources in the created control space structure.*

1. Introduction

Meshes are used in many applications – the simulation of processes, visualization, object reconstruction, image analysis, etc. In each case, a mesh must meet a number of criteria related to the geometry of the domain and other requirements specific to the issue. In the process of generating and adapting meshes, it is important to properly determine the desired size and shape of the elements at each point of the modeled object. Geometric- and problem-specific information may be inconsistent and introduce different descriptions of the desired mesh element size in some sub-domains. At the preprocessing stage, it is necessary to gather and unify various bits of information; at that point, the appropriate methods dedicated to smoothing the sizing should be applied. Automated determination of the proper size and shape specifications plays a key role in effective mesh generation.

There are two ways to adjust element size and mesh gradation. The first approach is based on an a posteriori and iterative remeshing evaluation [8, 9]. Other methods that are more widespread nowadays correct and smooth the so-called sizing field before the actual mesh generation. A separate problem in this second approach is how to store, retrieve, and process metric information for this field. The auxiliary structures dedicated to this purpose may have the form of a Cartesian mesh (either homogeneous [22] or adapted [3]), background mesh [7, 19, 21, 24], or quadtree/octree [18, 20, 23]. Recently, the application of a kd-tree was also proposed by the authors of [16]. Each of these structures has its advantages and disadvantages. It is necessary to assess the size of the memory required by the structure as well as the time it takes to create and access the information. It should not be forgotten that, in the mesh-generation process, the structure will be continuously searched to determine the appropriate size field at the specified point of the domain. While a background mesh is more easily adapted locally and generally takes up less memory, for example, finding the item containing the queried point of space takes a lot of time. Tree structures (octree and kd-tree) may need more memory, but they are more efficient with respect to time. In the methods developed by the authors, an octree as well as kd-tree control space[1] is used.

Some authors rely on the sizing field mainly for the length of the mesh edges, which (depending on the method) may result in the limitation of the adaptation to isotropic meshes. In our approach, the basic value stored in the control space is the metric, which also makes it easy to generalize the problem of mesh adaptation to anisotropic meshes. The use of metrics in the process of constructing meshes is now widespread [1, 2, 5, 6, 17].

---

[1]The control space (CS) is defined here as an auxiliary structure extending the concept of a sizing field responsible for providing the necessary or beneficial information at any point in the meshing domain (e.g., the required element size and shape, local metric gradation) during the mesh-generation process.

We use a Riemannian metric $\mathcal{M}$ that varies locally in different subdomains and which is then introduced into the generator as the operator defining the desired size and shape of the elements. Additionally, the concept of metric transformation tensor $\mathbf{M}$ was introduced in order to increase the efficiency of using the metric in the generator. The relationship between metric $\mathcal{M}$ and tensor $\mathbf{M}$ has been described in detail in previous works (e.g., [12–14]). The sources of the metric are different depending on the nature of the area and the specific application of the mesh, so they may have an unacceptably large discrepancy. A metric taken from many sources must therefore be properly introduced and unified in the control space. The methods used to create and adapt the octree control space are described in greater detail in [10, 13, 15]. In [16], a kd-tree-based control space structure was proposed and compared with octree. Tests were based on the adaptation of a control space structure to continuous metric sources. This paper focuses on the problem of adapting a control space to discrete metric sources combined with adjusting the metric gradation in the control space.

2. Kd-tree and octree structures

The considered structure of a discrete control space is based on a three-dimensional tree. The proposed kd-tree structures were created based on a general form of kd-tree [4] with modifications introduced by the authors (with respect to the kd-tree application as a control space structure for mesh generation and adaptation).

The single main node of the tree is the size of the meshing domain bounding box. During adaptation, the tree nodes (called *tree elements* for consistency with the meshing terminology) are successively split, forming new cuboid tree elements. The leaf nodes (*tree leaves*) are mainly used to store metric-related information; other (non-leaf) tree elements contain information about the split. The information about the metric is stored in the form of a *control node* structure consisting of two fields: the metric transformation tensor and coordinates associated with this node.

Three variations of kd-trees and three variations of octrees were implemented:

- **kd-tree-L** ($\mathcal{K}_L$) – kd-tree structure with control nodes stored in leaves,
- **kd-tree-Li** ($\mathcal{K}_{Li}$) – kd-tree structure with control nodes stored in leaves and additional interpolation of metric based on adjacent leaves using formulas derived from 27-nodes quadratic hexahedral shape functions,
- **kd-tree-V** ($\mathcal{K}_V$) – kd-tree structure with control nodes stored in vertices of leaves and interpolation of metric using 8-node linear shape functions,
- **octree-L** ($\mathcal{K}o_L$) – octree structure with control nodes stored in leaves,
- **octree-LB** ($\mathcal{K}o_{LB}$) – balanced octree structure with control nodes stored in leaves,
- **octree-V** ($\mathcal{K}o_V$) – octree structure with control nodes stored in vertices of leaves and interpolation of metric using linear shape functions.

On the main level, each tree structure contains a pointer to the root of the tree. Vertex-based trees ($\mathcal{K}_V$ and $\mathcal{K}o_V$) also have an additional container for metric control nodes, which are referenced in the leaves. The connectivity organization of the vertices in the tree leaves is shown on Figure 1.
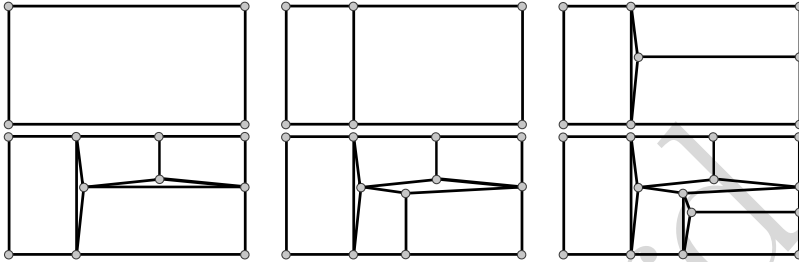

Figure 1. Local connectivity of leaf edges with subsequent kd-tree splits (2D case for clarity)

The procedure of retrieving the metric value at any point within the meshing domain using the tree-based control space structures begins with finding the tree leaf containing the given point (starting from the main tree element and traversing the elements following the splitting information stored in the non-leaf elements). Then, depending on the tree type, the metric is either returned directly from the control node stored in this leaf ($\mathcal{K}_L$, $\mathcal{K}o_L$ and $\mathcal{K}o_{LB}$) or the resultant metric value is calculated using a set of control nodes (from the leaf vertices or leaf neighbors) and a set of appropriate shape functions ($\mathcal{K}_{Li}$, $\mathcal{K}_V$ and $\mathcal{K}o_V$).

The elements of the trees may contain the following data (depending on the type of tree and the type of tree element):

- **metric values** – stored only in the leaves in the form of control nodes. For leaf-based trees, one control node is stored in each leaf with coordinates equal to the middle of the leaf box. For vertex-based trees, the control nodes are stored in the container defined on the main level of the tree; each leaf stores eight references to control nodes. In this way, the control nodes may be shared between some neighboring tree elements (Fig. 1).
- **splitting data** – used only in the non-leaf tree elements. For kd-trees, the axis and value of the splitting hyperplane is stored together with two references to the children elements. For octree structures, the splitting (middle) point is stored together with eight references to the children elements.
- **neighboring tree elements** – stored only in the leaves in the form of references to the adjacent tree elements. This information is required in order to facilitate the sharing references to the control nodes for the vertex-based trees, the interpolation of the metric values for $\mathcal{K}_{Li}$, balancing $\mathcal{K}o_{LB}$, and the smoothing of the gradation of the leaf-based trees.

3. Adaptation of control space

Two families of control space structures based on an octree and kd-tree are described in this article. The presented work concentrates on the adaptation of a control space structure resulting from the insertion of additional discrete metric sources. These types of sources provide only partial information about the metric available at discrete points. Such metric information can also be introduced in various time frames; e.g., resulting from the subsequent stages of creating the mesh. The area of influence of individual sources depends on their locations and the metrics associated with them; however, it should be noted that the difficulty is also due to the fact that they are potentially unevenly distributed. Discrete sources can define very diverse metrics; as a consequence, this requires the smoothing of the metric field and, thus, the control space. Regarding complexity, it is also important that there may be a potentially large amount of source data. All of these factors make it necessary to modify the procedures of construction and adaptation of the control space (as compared to the case of continuous sources).

3.1. Introduction of discrete metric sources

For each point-source Q with associated metric $(Q, \mathbf{M}_Q)$, the following general procedure is used first:

1. Retrieve current metric $\mathbf{M}_{\mathrm{CS}}(Q)$ from the control space (CS).
2. If metric difference $\delta_{\mathcal{M}}$ is very small (Eq. 1), no adaptation is necessary (STOP)

$$\delta_{\mathcal{M}}(\mathbf{M}_Q, \mathbf{M}_{\mathrm{CS}}(Q)) < \epsilon_\delta \ , \tag{1}$$

   where $\delta_{\mathcal{M}}$ is a metric non-conformity measure[2] [17].
3. Calculate new metric transformation tensor $\mathbf{M}'_Q$ as an intersection[3] ($\min_{\mathcal{M}}$) [15] of the metric tensor defined in the point-source and the metric tensor calculated from the current CS:

$$\mathbf{M}'_Q = \min_{\mathcal{M}}(\mathbf{M}_Q, \mathbf{M}_{\mathrm{CS}}(Q)) \ . \tag{2}$$

4. If the modification is negligible (Eq. 3), no adaptation is necessary (STOP)

$$\delta_{\mathcal{M}}(\mathbf{M}'_Q, \mathbf{M}_{\mathrm{CS}}(Q)) < \epsilon_\delta \ . \tag{3}$$

5. If the size of the current leaf (calculated as $\max(l_{\mathcal{M}}^x, l_{\mathcal{M}}^y, l_{\mathcal{M}}^z)$ with respect to the metric stored in the point-source where $l_{\mathcal{M}}^d$ is the metric length of a leaf along dimension d) is larger than the prescribed threshold ($l_{\min}$), the leaf is split, and the procedure is recursively run for the leaf containing the new point-source. The maximum depth of the tree is also enforced.

---

[2]The metric non-conformity coefficient of two metric tensors $\mathcal{M}_1$ and $\mathcal{M}_2$ is calculated as the Euclidean norm of total residual $||\mathcal{M}_1^{-1}\mathcal{M}_2 + \mathcal{M}_2^{-1}\mathcal{M}_1 - 2I||$

[3]The intersection of the metric tensors is calculated as a simultaneous reduction of the quadratic forms and can be geometrically interpreted as an ellipsoid, with the largest volume contained within the ellipsoids associated with the two metric tensors.

6. After the structure has been adapted, the intersection operation for each control vertex V_i of the leaf containing the new point-source is used:

$$\mathbf{M}_{V_i} \leftarrow \min_{\mathcal{M}}(\mathbf{M}_{V_i}, \mathbf{M}_Q) \ .$$

The above procedure is general, but the fundamental operation of adapting the tree structure – the splitting of the leaves – is different between the implemented tree variations following the different data stored in the leaves of these structures. However, before the cuboid element of the kd-tree can be split, it is necessary to select the splitting plane. The splitting methods should be adapted to the discrete distribution of the metric sources. Compared to the procedures for continuous sources, two other splitting methods were chosen in this case that are better-suited to the available discrete metric information and less computationally expensive. The following methods were investigated:

1. **Longest Axis** (s_L). The element is split in the middle of the longest axis with a plane perpendicular to this axis (Fig. 2(a)).
2. **Golden Ratio** (s_G). The element is split by a plane perpendicular to the longest axis using the golden ratio formula, where the point-source should lie in the smaller part of the division if possible (Fig. 2(b)).
3. **Minimum Element** (s_M). The element is split by a plane perpendicular to the longest axis, setting the splitting plane close to the point-source (with a length margin of half of $l_{\min}$) (Fig. 2(c)).

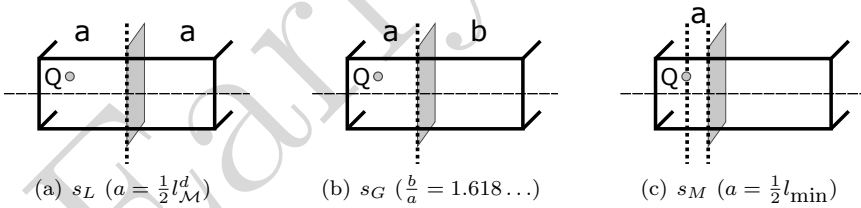In all cases, the lengths of the nodes' dimensions are calculated in metric space.

(a) s_L ($a = \frac{1}{2}l_{\mathcal{M}}^d$)   (b) s_G ($\frac{b}{a} = 1.618\ldots$)   (c) s_M ($a = \frac{1}{2}l_{\min}$)

Figure 2. Methods of choosing splitting plane

3.2. Adjusting metric gradation

The metric field stored in the control space structure may come from different sources. In order to properly guide the meshing procedure, the metric field may need to be adjusted in a way that results in the desired transition of elements in the created meshes.

The procedure of enforcing the desired gradation of the mesh in control space is based on an elementary operation of adjusting the metric for a pair of points according to the defined gradation ratio parameter [13]. The general procedure consists of the following steps:

1. Gather all control nodes from the whole tree.
2. Construct the adjacency information, storing all of its neighboring nodes for each control node:
   - for trees with control nodes stored in the tree vertices – the set of neighboring nodes consists of all control nodes from the vertices that are connected to the vertex of the selected node with a leaf edge (Fig. 3(b)),
   - for trees with control nodes stored in the tree elements – the set of neighboring nodes is gathered by inspecting all six faces of the selected leaf and including the control nodes from the nearest neighboring leaves for each face of the selected leaf (Fig. 3(a)).

   Due to the locality of the connections between the tree vertices (Fig. 1), the maximum number of adjacent control nodes is limited to six in each case.
3. Create an active queue initially containing all control nodes.
4. While the active queue is not empty:
   (a) remove control node c_* from the beginning of the queue,
   (b) for each node c_i adjacent to c_*, perform the elementary operation of smoothing a pair of nodes (c_*, c_i),
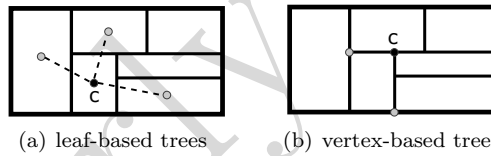   (c) insert all modified control nodes at the end of the queue.

(a) leaf-based trees   (b) vertex-based trees

Figure 3. Determining adjacency of control nodes (2D case for clarity)

4. Tests

The tests were designed to measure the quality of the created trees and the performance of the procedure of the control space adaptation. In all cases, the control space was created in domain $\mathcal{D}_1 = [-1, 1] \times [-1, 1] \times [-1, 1]$ and initialized as a single tree leaf with a metric set of $\mathcal{M}^{\max}$, which defines an isotropic metric with lengths of elements equal to half of the domain diagonal. Then, a number of discrete metric point-sources were generated and successively inserted into the control space. After inserting all point-sources, the gradation of the control space was adjusted respecting the prescribed parameter of the maximum gradation ratio.

In order to evaluate the required density of the discrete metric sources and its influence on the adaptation quality, the set of metric sources was created by sampling the predefined surfaces with the selected density. Different surfaces were tested as the base of the generated points, among which were the following:

- sphere with center at point [0,0,0] and radius of 0.8,
- intersecting planes described by following equations:
    - $x = -0.5, y = u, z = v$
    - $x = u, y = -0.2 + au, z = v$ with $a = 0.0$ or $a = 0.2$.

For each selected point on the probed surface, the metric is assigned as

$$
\mathcal{M} = \begin{pmatrix} h_1^{-2}(x,y,z) & 0 & 0 \\ 0 & h_2^{-2}(x,y,z) & 0 \\ 0 & 0 & h_3^{-2}(x,y,z) \end{pmatrix}, \tag{4}
$$

where h_1, h_2, h_3 are the lengths of the elements along the main directions. For greater flexibility in the tests, an additional parameter ($s_{an} = [s_a, s_n]$) was used; this allows us to introduce anisotropic metric sources by means of scaling the metric by s_a in all directions and then by s_n in the direction normal to the probed surface.

4.1. Test design

The adaptation process of all of the presented tree structures was analyzed. The tests were executed with varying densities of the input metric sources ($\rho_m \in \{1.0, 0.5, 0.2\}$ – the lower the value, the higher the density of the sources). Other parameters changing during the tests were the minimum length of leaf edges l_{min}, maximum depth of the tree, approximation accuracy threshold δ_τ, gradation ratio g_r, and coefficient s_{an}. Three methods for splitting the tree element boxes were tested: the longest axis (S_L), golden ratio (S_G), and minimum element (S_M). For each case, the following quantities were measured:

- average access time ($\bar{t}_a$) – a uniform grid of points in $\mathcal{D}$ was created; for each point, the metric value was retrieved from the tree structure, then the average value was calculated,
- creation time (t_c) – total time required for adaptation of the tree structure to the given set of discrete metric sources,
- approximation error ($\delta_{\max}$) – maximum value of difference $\delta_\mathcal{M}$ between the kd-tree's/octree's approximation and the value computed directly from the set of metric sources,
- tree size – the number of tree elements, number of metric values stored in the tree, and total memory usage of the tree structure (m_u).

5. Analysis of results

A presentation of all of the obtained results is not possible due to their size, which is a consequence of the many parameters taken into account. Because of this, only selected results are presented. The tables below are limited to selected cases where the point sources were generated on a sphere. The other tested examples gave comparable characteristics of the results and led to similar conclusions. In the presented case,

structures were created with coefficients $g_r = 1.25$ and $s_{an} = [2.0, 1.0]$, and the maximum tree depth for the kd-trees is equal to 42; for the octree structures, this is equal to 14.

The analysis was mainly carried out while paying particular attention to the computational and memory efficiency of various versions of the kd-tree and octree structures as well as the approximation accuracy for these structures.[4]

5.1. Time and memory efficiency

Table 1 contains selected results showing the effect of given approximation accuracy threshold δ_τ on the size of the trees for the two chosen densities of the metric sources.

The size of the trees $\mathcal{K}o_{LB}$ is remarkably larger than the rest. On the other hand, the kd-trees with splitting s_M are too small. As a detailed further analysis has shown, these trees have been insufficiently adapted to the metric field. For the remaining trees, the prescribed δ_τ for a given source density has a significant impact on the size of the trees regardless of their type. The density of the metric sources affects the size of the trees, but not that much. It can be seen that the sizes of the vertex-based trees are larger, which is understandable in view of how the metric information is stored in them. Comparing splitting methods s_L and s_G, it is worth noting that splitting s_G results in kd-trees of smaller sizes in most cases; however, with the high source density and high required accuracy, the proportions change to the disadvantage of this technique.

Table 1

Tree-size [MB] created for selected sphere model

tree	split	$\rho_m = 1$			$\rho_m = 0.2$		
		$\delta_\tau = 1$	$\delta_\tau = 0.5$	$\delta_\tau = 0.2$	$\delta_\tau = 1$	$\delta_\tau = 0.5$	$\delta_\tau = 0.2$
$\mathcal{K}_L$	s_L	17.07	41.32	67.56	20.64	78.85	285.82
	s_G	8.16	33.13	62.29	9.42	59.32	341.10
	s_M	5.27	5.64	4.28	6.45	8.90	13.78
$\mathcal{K}_V$	s_L	51.10	127.15	210.21	61.15	234.30	853.81
	s_G	24.81	102.72	194.87	28.24	179.42	1041.96
	s_M	16.37	17.69	13.50	19.21	26.56	41.37
$\mathcal{K}_{Li}$	s_L	17.07	41.32	67.56	20.64	78.85	285.82
	s_G	8.16	33.13	62.29	9.42	59.32	341.10
	s_M	5.40	5.64	4.28	7.17	9.48	13.87
$\mathcal{K}o_L$		16.36	52.35	94.23	18.76	72.33	269.14
$\mathcal{K}o_{LB}$		528.22	599.33	635.08	856.71	998.83	1033.88
$\mathcal{K}o_V$		42.89	143.35	282.76	47.83	186.04	701.23

Table 2 illustrates the results for the creation and adaptation time t_c of the control spaces. Except for $\mathcal{K}o_{LB}$ trees (for which the t_c time is far too long), the

---

[4]The tests were performed using the computational resources of the Zeus supercomputer: `https://www.top500.org/system/177388`

creation time is slightly shorter for the trees based on leaves. This time naturally increases with the density of sources ρ_m and required accuracy δ_τ (but differently for each type of structure).

Table 2

Adaptation time t_c $[\mu s]$ for tree structures created for selected sphere model

| tree | split | $\rho_m = 1$ | | | $\rho_m = 0.2$ | | |
		$\delta_\tau = 1$	$\delta_\tau = 0.5$	$\delta_\tau = 0.2$	$\delta_\tau = 1$	$\delta_\tau = 0.5$	$\delta_\tau = 0.2$
$\mathcal{K}_L$	s_L	6.38	9.24	15.32	6.52	20.17	98.00
	s_G	2.39	10.70	12.30	4.58	14.45	63.90
	s_M	1.41	1.96	1.73	9.83	7.35	9.42
$\mathcal{K}_V$	s_L	8.62	28.90	63.91	13.64	57.05	548.15
	s_G	3.26	21.79	53.54	7.39	37.21	856.65
	s_M	4.09	4.83	3.87	9.42	11.99	15.85
$\mathcal{K}_{Li}$	s_L	3.71	15.56	15.33	11.81	25.41	65.62
	s_G	1.74	7.03	13.36	9.82	20.13	68.65
	s_M	1.78	2.34	2.19	11.98	17.19	19.58
$\mathcal{K}o_L$		4.57	14.35	25.81	6.70	20.58	67.03
$\mathcal{K}o_{LB}$		184.71	211.18	217.12	558.58	597.29	597.52
$\mathcal{K}o_V$		10.72	28.28	75.01	9.64	34.11	235.39

The access time is presented in Table 3. It can be seen that the tree size has no significant effect on the access time; the access time is more related to the type of tree. The method of creating $\mathcal{K}_{Li}$ determines the longer access times. Octree structures are large in size; on the other hand, they offer rather short access times. Longer access times for the kd-trees with the s_M splitting method are again associated with their incorrect adaptation to the given metric.

Table 3

Access time $(\bar{t}_a)$ $[\mu s]$ for tree structures created for selected sphere model

| tree | split | $\rho_m = 1$ | | | $\rho_m = 0.2$ | | |
		$\delta_\tau = 1$	$\delta_\tau = 0.5$	$\delta_\tau = 0.2$	$\delta_\tau = 1$	$\delta_\tau = 0.5$	$\delta_\tau = 0.2$
$\mathcal{K}_L$	s_L	2.96	1.48	1.11	1.48	1.11	1.85
	s_G	2.22	2.22	1.11	1.11	1.48	1.48
	s_M	2.96	3.70	3.70	5.56	3.70	4.07
$\mathcal{K}_V$	s_L	1.85	2.22	1.85	2.22	1.85	1.85
	s_G	2.22	1.85	1.85	1.85	1.85	3.33
	s_M	4.07	3.70	4.07	4.07	4.81	4.44
$\mathcal{K}_{Li}$	s_L	7.41	12.59	7.78	7.78	7.78	7.78
	s_G	7.41	7.41	9.26	7.78	8.15	8.15
	s_M	10.74	13.33	12.96	10.37	14.07	13.70
$\mathcal{K}o_L$		0.74	0.74	0.74	0.74	0.37	0.74
$\mathcal{K}o_{LB}$		1.48	2.22	2.22	3.33	3.33	2.96
$\mathcal{K}o_V$		1.85	1.11	1.11	1.11	1.11	1.11

5.2. Approximation accuracy

The evaluation of the accuracy of the approximation of the metric field was performed by means of two coefficients. Coefficient $\delta_{\max}^S$ was calculated as the maximum metric difference between the original data (the set of discrete metric sources) and the metric field in the control space tree structure (measured for all coordinates of the discrete metric sources). $\delta_{\max}^R$ is the maximum metric difference computed via a regular probing of the whole testing domain $\mathcal{D}$. In this case, the metric retrieved from the adapted control space was compared to the metric value calculated directly from all discrete metric sources using gradation adjustment for each tested point.

The only non-zero values of $\delta_{\max}^S$ are observed for $\mathcal{K}_{Li}$ because of the way this tree is created (where complex metric interpolation is used). It can be concluded that the application of complex metric interpolation methods for control space tree structures does not significantly improve the quality of these structures.

Table 4 shows the values of coefficient $\delta_{\max}^R$ for the selected model of metric source distribution. The results are satisfactory except for $\mathcal{K}o_{LB}$ structures and kd-trees with s_M splitting. The lowest values were noted for $\mathcal{K}_V$ trees with splitting s_L. Although the size of these types of trees are slightly larger and the times of their creation are slightly longer than the others, they give acceptable access times and provide the best accuracy.

Table 4
Accuracy $\delta_{\max}^R$ for tree structures created for selected sphere model

tree	split	$\rho_m = 1$			$\rho_m = 0.2$		
		$\delta_\tau = 1$	$\delta_\tau = 0.5$	$\delta_\tau = 0.2$	$\delta_\tau = 1$	$\delta_\tau = 0.5$	$\delta_\tau = 0.2$
$\mathcal{K}_L$	s_L	1.78	1.83	2.31	1.74	1.69	1.70
	s_G	1.62	1.68	1.70	1.62	1.66	1.69
	s_M	6.48	8.06	8.06	6.48	8.06	8.06
$\mathcal{K}_V$	s_L	0.30	0.37	0.43	0.30	0.37	0.35
	s_G	1.67	1.59	1.42	1.64	1.63	1.59
	s_M	6.48	8.06	8.06	7.05	8.06	8.06
$\mathcal{K}_{Li}$	s_L	1.20	1.12	1.15	1.20	1.20	1.20
	s_G	1.11	1.02	1.04	0.89	0.83	0.86
	s_M	6.70	8.06	8.06	6.53	8.06	8.06
$\mathcal{K}o_L$		1.83	1.90	1.94	1.80	1.70	1.64
$\mathcal{K}o_{LB}$		6.88	6.00	5.99	5.20	4.70	5.97
$\mathcal{K}o_V$		1.79	1.62	1.04	1.96	1.76	1.75

An analysis of all of the results leads to one more conclusion regarding the permissible tree depth. This depth cannot be too small, as this results in the creation of structures that do not sufficiently approximate the metric field (especially the varied one). Setting a maximum depth for kd-trees to 20 made refining the approximation accuracy threshold almost pointless, as the trees were not able to sufficiently adapt to the higher quality requirements. On the other hand, if the permissible depth is

too high, the tree may be too large and unnecessarily increase the time and memory cost of the mesh-generation process. In the performed tests, the selected depth of 42 levels for the kd-trees and 14 for the octrees was sufficient to observe the effect of modifying the other parameters in most cases (the approximation accuracy threshold or density of the input metric sources). Still, for some types of trees (mostly octrees and vertex-based kd-trees), the possibility of deeper tree refinement (enforced with a lower value of the approximation accuracy threshold) had the undesired effect of substantially increasing the tree size while only slightly improving the $\delta_{\max}^R$ value and actually decreasing the resultant mesh quality (as mentioned in the next paragraph).

5.3. Impact on created meshes

In order to verify the correctness of the construction procedures for the control spaces, appropriate meshes were created for all of the considered cases with the anisotropic mesh generator developed by the authors. The generator uses an iterative Delaunay triangulation technique with local non-Eculidean metric transformation [11].

The quality of the mesh was measured using coefficient L_R (among others), which determines the ratio of the number of edges with a metric length within a range of [0.8, 1.25] to the total number of edges in the mesh (the ideal metric length of the edges is equal to 1). The obtained values of this coefficient for the selected case are included in Table 5. It can be seen that, for kd-trees with s_M splitting, the quality of the mesh is low. Not much better results were obtained for the $\mathcal{K}o_{LB}$ trees. Other results are comparable; $\mathcal{K}_V$ trees with s_L splitting guarantee the best mesh quality.

Table 5

Mesh quality coefficient L_R for tree structures created for selected sphere model

tree	split	$\rho_m = 1$			$\rho_m = 0.2$		
		$\delta_\tau = 1$	$\delta_\tau = 0.5$	$\delta_\tau = 0.2$	$\delta_\tau = 1$	$\delta_\tau = 0.5$	$\delta_\tau = 0.2$
$\mathcal{K}_L$	s_L	0.67	0.66	0.66	0.67	0.67	0.67
	s_G	0.66	0.66	0.66	0.66	0.67	0.67
	s_M	0.21	0.13	0.12	0.21	0.13	0.12
$\mathcal{K}_V$	s_L	0.72	0.69	0.65	0.72	0.72	0.70
	s_G	0.66	0.65	0.62	0.67	0.67	0.66
	s_M	0.22	0.14	0.12	0.22	0.14	0.12
$\mathcal{K}_{Li}$	s_L	0.69	0.68	0.68	0.70	0.69	0.69
	s_G	0.69	0.68	0.67	0.69	0.69	0.69
	s_M	0.22	0.13	0.14	0.22	0.13	0.12
$\mathcal{K}o_L$		0.68	0.68	0.68	0.68	0.69	0.68
$\mathcal{K}o_{LB}$		0.45	0.47	0.49	0.40	0.43	0.45
$\mathcal{K}o_V$		0.66	0.69	0.69	0.65	0.67	0.69

It should also be noted that it is not necessary to excessively increase the required accuracy of the control space (by decreasing coefficient δ_τ). Often, better results can

be achieved with $\delta_\tau = 0.5$ rather than $\delta_\tau = 0.2$ (with the addition of a smaller tree size).

The figures below show examples of meshes (in cross-sections) generated using the developed control space structures. The mesh from Figure 4(a) was generated on the basis of the $\mathcal{K}_V$ tree, which was created for discrete sources on the sphere sampled with density $\rho_m = 1$. The threshold δ_τ used in this example was 1, with g_r equal to 1.25. The tree was split by the s_L method. The maximum permissible depth of the tree was set at 30. The metric sources was set with some anisotropy ($s_{an} = [4.0, 0.25]$), which caused the effects seen in the figure – the mesh elements in the area of the sphere are slightly elongated in a certain direction. The obtained L_R-value was 0.7, and the generated mesh has 291,570 elements.

(a) $\mathcal{K}_V$, $L_R = 0.7$, NT=291,570   (b) $\mathcal{K}o_V$, $L_R = 0.79$, NT=350,293
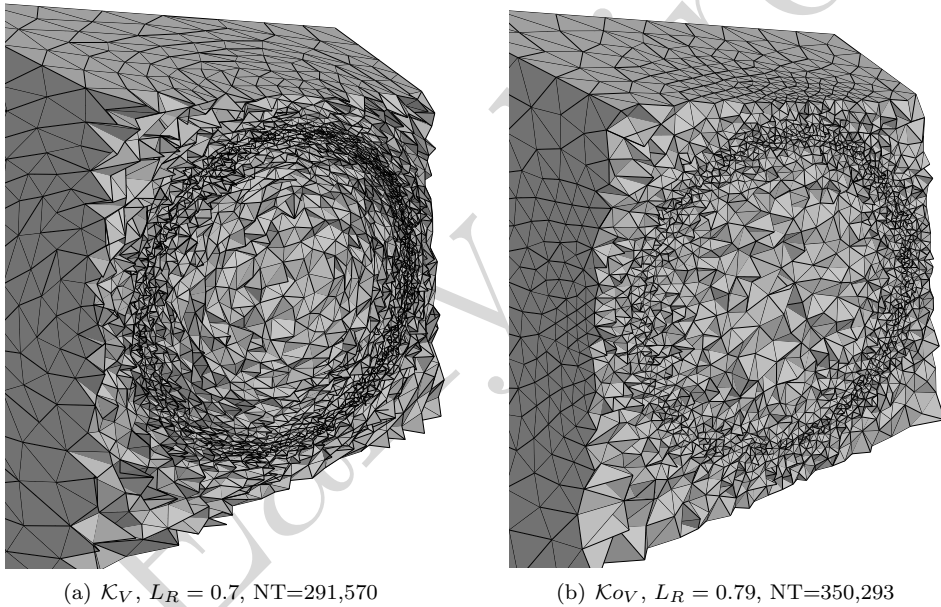
Figure 4. Examples of meshes generated based on different control space structures

Figure 4(b) presents a mesh created using octree $\mathcal{K}o_V$. For the density of the sources on the sphere, coefficients δ_τ and g_r were assumed as with the example from Figure 4(a). The elements visible in the vicinity of the sphere are smaller and denser, as no anisotropy was introduced into the source metric in this case. The obtained L_R-value was 0.79 in this case, and the number of mesh elements is 350,293.

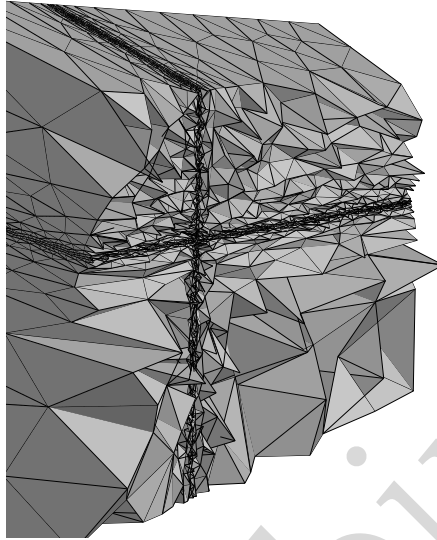Figure 5 shows the mesh generated for the intersecting planes created using anisotropic metric sources.

Figure 5. Example of mesh generated for metric sources distributed along intersecting planes

6. Conclusions

As in [16], it has been shown that kd-trees are useful for constructing control space structures. This time, the focus was on the introduction of discrete metric sources into these structures. In most test cases, the kd-tree structures resulted in better meshes produced by the mesh generator at a lower memory cost for storing the control space structures. The leaf-based kd-tree seems to be a good overall candidate – simple, fast, and memory efficient. Leaf-based kd-trees with interpolation may result in a better quality of the generated meshes in some cases but at the cost of the increased complexity and lower speed. Vertex-based kd-trees are able to provide the best results in term of the resultant mesh quality; however, the downside is the increased size of the structure.

These tests were performed using evenly spaced metric sources. Further studies will provide for tests using more-anisotropic metric sources and irregular locations. It is also necessary to analyze the combination of both continuous and discrete metric sources in a single control space structure in order to properly assess the suitability of particular structures.

References

[1] Alauzet F.: Size Gradation Control of Anisotropic Meshes. In: *Finite Elem. Anal. Des.*, vol. 46(1-2), pp. 181–202, 2010.

[2] Alauzet F., Loseille A., Dervieux A., Frey P.: *Multi-Dimensional Continuous Metric for Mesh Adaptation*, pp. 191–214. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[3] Aubry R., Karamete K., Mestreau E., Dey S., Löhner R.: Linear Sources for Mesh Generation. In: *SIAM Journal on Scientific Computing*, vol. 35(2), pp. A886–A907, 2013.

[4] de Berg M., Cheong O., van Kreveld M., Overmars M.: *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2008.

[5] Borouchaki H., George P.L., Hecht F., Laug P., Saltel E.: Delaunay mesh generation governed by metric specifications. Part I. Algorithms. In: *Finite Elements in Analysis and Design*, vol. 25, pp. 61–83, 1997.

[6] Bottasso C.L.: Anisotropic mesh adaption by metric-driven optimization. In: *International Journal for Numerical Methods in Engineering*, vol. 60(3), pp. 597–639, 2004. ISSN 1097-0207. http://dx.doi.org/10.1002/nme.977.

[7] Chen J., Xiao Z., Zheng Y., Zheng J., Li C., Liang K.: Automatic sizing functions for unstructured surface mesh generation. In: *International Journal for Numerical Methods in Engineering*, vol. 109(4), pp. 577–608, 2017. ISSN 1097-0207. http://dx.doi.org/10.1002/nme.5298. Nme.5298.

[8] Cunha A., Canann S., Saigal S.: Automatic Boundary Sizing For 2D And 3D Meshes. In: *In Trends in unstructured mesh generation, volume AMD-220*, pp. 65–72. 1997.

[9] Frey P.J.: About Surface Remeshing. In: *Proc. 9th Int. Meshing Roundtable*, pp. 123–136. Sandia National Laboratories, 2000.

[10] Głut B., Jurczyk T.: Preparation of the Sizing Field for Volume Mesh Generation. In: *Proc. 13th Int. Conf. on Civil, Structural and Environmental Engineering Computing*. Chania, Crete, Greece, 2011. Paper 115.

[11] Głut B., Jurczyk T., Kitowski J.: Anisotropic Volume Mesh Generation Controlled by Adaptive Metric Space. In: *Proc. of Int. Conf. NUMIFORM'07*, pp. 233–238. Porto, Portugal, 2007.

[12] Jurczyk T.: *Efficient Algorithms of Automatic Discretization of Non-Trivial Three-Dimensional Geometries and its Object-Oriented Implementation*. Ph.D. thesis, AGH University of Science and Technology, Kraków, Poland, 2007.

[13] Jurczyk T., Głut B.: Adaptive Control Space Structure for Anisotropic Mesh Generation. In: *Proc. of ECCOMAS CFD 2006 European Conference on Computational Fluid Dynamics*. Egmond aan Zee, The Netherlands, 2006.

[14] Jurczyk T., Głut B.: Metric 3D Surface Mesh Generation Using Delaunay Criteria. In: *Lecture Notes in Computer Science*, vol. 3992, pp. 302–309, 2006.

[15] Jurczyk T., Głut B.: The Insertion of Metric Sources for Three-dimensional Mesh Generation. In: *Proc. 13th Int. Conf. on Civil, Structural and Environmental Engineering Computing*. Chania, Crete, Greece, 2011. Paper 116.

[16] Jurczyk T., Głut B.: Tree Structures for Adaptive Control Space in 3D Meshing. In: *Computer Science*, vol. 17(4), p. 541, 2017. ISSN 2300-7036. `https://journals.agh.edu.pl/csci/article/view/2126`.

[17] Labbé P., Dompierre J., Vallet M.G., Guibault F., Trépanier J.Y.: A universal measure of the conformity of a mesh with respect to an anisotropic metric field. In: *Int. J. Numer. Meth. Engng*, vol. 61, pp. 2675–2695, 2004.

[18] Miranda A.C.O., Martha L.F.: Mesh generation on high-curvature surfaces based on a background quadtree structure. In: *Proceedings, 11th International Meshing Roundtable*, pp. 333–342. 2002.

[19] Owen S.J., Saigal S.: Surface mesh sizing control. In: *International Journal for Numerical Methods in Engineering*, vol. 47(1-3), pp. 497–511, 2000.

[20] Persson P.O., Staten M.L., Xiao Z., Chen J., Zheng Y., Zeng L., Zheng J.: 23rd International Meshing Roundtable (IMR23) Automatic Unstructured Element-sizing Specification Algorithm for Surface Mesh Generation. In: *Procedia Engineering*, vol. 82, pp. 240 – 252, 2014.

[21] Pippa S., Caligiana G.: GradH-Correction: guaranteed sizing gradation in multi-patch parametric surface meshing. In: *International Journal for Numerical Methods in Engineering*, vol. 62(4), pp. 495–515, 2005. ISSN 1097-0207. `http://dx.doi.org/10.1002/nme.1177`.

[22] Pirzadeh S.Z.: Structured Background Grids for Generation of Unstructured Grids by Advancing-Front Method. In: *AIAA Journal*, vol. 31(2), pp. 257–265, 1993.

[23] Quadros W.R., Vyas V., Brewer M., Owen S.J., Shimada K.: A computational framework for automating generation of sizing function in assembly meshing via disconnected skeletons. In: *Engineering with Computers*, vol. 26(3), pp. 231–247, 2010.

[24] Xiao Z., Chen J., Zheng Y., Zeng L., Zheng J.: Automatic Unstructured Element-sizing Specification Algorithm for Surface Mesh Generation. In: *Procedia Engineering*, vol. 82, pp. 240 – 252, 2014. ISSN 1877-7058. `http://dx.doi.org/http://dx.doi.org/10.1016/j.proeng.2014.10.387`.

Affiliations

Barbara Głut
Department of Computer Science, AGH, glut@agh.edu.pl,
ORCID ID: https://orcid.org/0000-0003-1522-873X

Tomasz Jurczyk
Department of Computer Science, AGH, jurczyk@agh.edu.pl,
ORCID ID: https://orcid.org/0000-0003-3557-6985