Moving meshes to fit large deformations based on centroidal Voronoi tessellation (CVT)

Witalij Wambold[1], Günter Bärwolff[2], Hartmut Schwandt[2]

[1] Volkswagen AG, Component Development, Letter box 7359, Salzgitter 38231, Germany and TU Berlin, Institute of Mathematics
[2] Technische Universität Berlin, Institute of Mathematics, Straße des 17. Juni 136, 10623 Berlin, Germany

Abstract. The essential criterion for stability and fast convergence of CFD-solvers (CFD - computational fluid dynamics) is a good quality of the mesh. Based on results of [29] in this paper we use the so-called centroidal Voronoi tessellation (CVT) not only for mesh generation and optimization. The CVT is applied to develop a new mesh motion method. The CVT provides an optimal distribution of generating points with respect to a cell density function. For a uniform cell density function the CVT results in high-quality isotropic meshes. The non-uniform cases lead to a trade-off between isotropy and fulfilling cell density function constraints. The idea of the proposed approach is to start with the CVT-mesh and apply for each time step of transient simulation the so-called Lloyd's method in order to correct the mesh as a response to the boundary motion. This leads to the motion of the whole mesh as a reaction to movement. Furthermore, each step of Lloyd's method provides a further optimization of the underlying mesh, thus the mesh remains close to the CVT-mesh. Experience has shown that it is usually sufficient to apply a few iterations of the Lloyd's method per time step in order to achieve high-quality meshes during the whole transient simulation. In comparison to previous methods our method provides high-quality and nearly isotropic meshes even for large deformations of computational domains.

Key words: Mesh Motion; Centroidal Voronoi Tessellation; Finite Volume Method

1 Introduction

Currently there are numerous areas of applications in which the shape of the solution domain is variable for every time step of the simulation. Examples for such cases are prescribed boundary motion in pumps, internal combustion engines, free-rising bubbles in water as well as wind turbine simulations. One problem of such simulations is the propagation of the displacement at the surfaces into the volume mesh. It is well known that the essential criterion for stability and fast convergence of CFD-solvers is a good quality of the mesh. Maintaining these criteria for the internal mesh is a quite difficult task if the domain suffers from large deformations. In this paper we present a method based on centroidal Voronoi

tessellation that provides a high-quality mesh even for large boundary motions. The algorithm is implemented as an extension to the OpenFOAM® framework [15].

2 Previous work

Several methods for mesh deformation have been presented in the literature during the past decades. In [29] we discussed the properties, advantages and disadvantages for example of Laplacian smoothing [22], the spring analogy method of [4, 2, 6, 3, 10, 27], a finite element method proposed in [11], interpolation of the boundary displacements to the interior mesh by radial basis functions proposed by [24], and a new method based on a disk relaxation algorithm recently developed by Xuan Zhou [28].

Regardless of all methods described above, each is just suitable for certain degrees of deformations, because these approaches describe the mesh motion simply through relocation of mesh vertices. The cell topology, however, remains unchanged.

3 Contribution

Up to now the CVT has been used primarily for mesh generation and optimization [18, 7, 17, 13]. CVT provides an optimal distribution of generating points with respect to a given cell density function. For a uniform cell density function the CVT results in high-quality isotropic meshes. The non-uniform cases lead to a trade-off between isotropy and fulfilling of the cell density function constraints.

The idea of the approach is to start with a CVT-mesh and to apply for each time step of transient simulation the so-called Lloyd's method [20, 21, 19] to correct the mesh with respect to the boundary motion. This leads to the motion of the whole mesh if the boundary is moved. Furthermore, each step of Lloyd's method provides a further optimization of the underlying mesh, thus the mesh remains close to the CVT-mesh. In order to create the initial CVT-mesh from a given arbitrary mesh, we also apply Lloyd's method. An integral part of our work is to develop an efficient CVT implementation that allows a mesh generation close to CVT at each simulation step. Our code is written in C++ as extension to the OpenFOAM® framework. This package enables the developer through, a convenient class hierarchy to extend the built-in code without great effort.

Compared to previous approaches our technique provides nearly isotropic polyhedral meshes even for large boundary deformation. Another advantage of our approach is the operation on already existing cells, so the interpolation of fields between two different meshes is avoided.

The algorithm affects solely the cell topology and thus allows to keep the field affiliation to each cell. Only the face fluxes must be calculated for a new generated CVT-mesh. Moreover, our algorithm can be run in parallel, because

the computation of each Voronoi cell is carried out independently from other cells.

4 Theoretical background of the CVT

We restrict our treatment of the Voronoi tessellation to 3D-space. Given an open set $\Omega \subseteq \mathbb{R}^3$, the set $\{V_i\}_{i=1}^n$ is called a tessellation of Ω if $V_i \cap V_j = \emptyset$ for $i \neq j$ and $\cup_{i=1}^n \overline{V_i} = \overline{\Omega}$. Let $\|\cdot\|$ denote the Euclidean norm on $\mathbb{R}^3$. Given a set of points $\{x_i\}_{i=1}^n$ belonging to $\overline{\Omega}$, the Voronoi region $\hat{V}_i$ corresponding to the point x_i is defined by

$$\hat{V}_i = \{x \in \Omega \quad | \quad \|x_i - x\| < \|x_j - x\| \text{ for } j = 1, \ldots, n \text{ with } j \neq i\}. \tag{1}$$

The points $\{x_i\}_{i=1}^n$ are called generating points or generators. The set $\{\hat{V}_i\}_{i=1}^n$ is a Voronoi tessellation of Ω, and each $\hat{V}_i$ corresponds to the Voronoi region of the generator x_i. Given a region $V \subseteq \mathbb{R}^3$ and a cell density function ρ, defined on V, the centre of mass x^* of V is defined by

$$x^* = \int_V x\rho(x)dx / \int_V \rho(x)dx. \tag{2}$$

A centroidal Voronoi region V_i^* is a Voronoi region $\hat{V}_i$ with the property:

$$x_i = x_i^*. \tag{3}$$

A Voronoi tessellation where all regions satisfy the condition in (3) is called centroidal Voronoi tessellation.

One of the algorithms for CVT computation is the Lloyd's method. This is an iterative algorithm consisting of the following simple steps: starting from an initial Voronoi tessellation corresponding to an old set of generators, a new set of generators is defined by the centres of mass of the old Voronoi regions. A mathematical scheme of the Lloyd's method is given in algorithm 1. In the

Algorithm 1 Lloyd algorithm

For a given domain Ω with cell density function ρ defined on Ω and the initial set of generators $\{x_i\}_{i=1}^n$ perform following iterations:
for k=1,2,...,nIterations **do**
Construct the Voronoi tessellation $\{V_i^{(k-1)}\}_{i=1}^n$ of Ω with generators $\{x_i^{(k-1)}\}_{i=1}^n$.
Take the centres of mass of $\{V_i^{(k)}\}_{i=1}^n$ as the new set of generators $\{x_i^{(k)}\}_{i=1}^n$.
Break if some stopping criterion is met.
end for

sequel we give a short mathematical argumentation for Lloyd's method. For a

detailed mathematical treatment we refer to [20]. Let us define the set

$$\mathcal{M} := \{ \left(x_1, x_2, \cdots, x_n\right)^T \mid x_i \in \Omega \text{ for } i = 1, \cdots, n \}. \tag{4}$$

Then we know that considering (1) for a fixed boundary $\partial\Omega$ each V_i depends on x_i and a neighbourhood of x_i. Therefore we can say that, V_i depends on $X := \left(x_1, x_2, \cdots, x_n\right)^T \in \mathcal{M}$. Then for each step of the Lloyd's iteration we have:

$$x_i^{(k+1)} = \frac{\int\limits_{V_i(X^{(k)})} x\rho(x)dx}{\int\limits_{V_i(X^{(k)})} \rho(x)dx}. \tag{5}$$

Further we define the map

$$T : \mathcal{M} \mapsto \mathcal{M}, \;\; X^{(k)} \mapsto X^{(k+1)}, \quad \text{such that } T_i(X^{(k)}) := x_i^{(k+1)}. \tag{6}$$

Considering (2) and (3) we obtain in case of CVT the following equality:

$$X^{(k)} = T(X^{(k)}) \text{ or } X = T(X). \tag{7}$$

In view of (7) Lloyd‘s method may be viewed as a fixed point iteration. This shows that, Lloyd‘s method has a linear convergence rate. And according to [19] this convergence rate decreases as the number of generators gets large. Some accelerating techniques like the Lloyd-Newton method are given in [19]. Apart from this fact Lloyd's method is very well suited for the mesh motion, because the time step of the simulation should be set smaller while the number of cells gets larger. This is done in order to bound the Courant number by one.

Unfortunately, and despite great advantages of the CVT, its computation becomes difficult for complicated domain boundaries. The most widely used approach to construct the Voronoi tessellation in 3D-space is the computation of a dual data structure referred as Delaunay triangulation. See for example [18, 16, 8]. We propose an alternative approach for the generation of Voronoi tessellations.

5 Centroidal Voronoi Generator

There are numerous preprocessors for the efficient generation of tetrahedral meshes. We presume that the fluid domain is already meshed. We are interested in construction of the CVT from a given tetrahedral or polyhedral mesh. Now we use Lloyd's algorithm as described above. The sum of the absolut values of the differences between old generators and new generators serves as convergence criterion for Lloyd's method. One step of Lloyd's method includes the following tasks:

1. Calculate the cell density distribution in the whole domain.
2. Compute the new generators with respect to the cell density function.

3. Compute the tessellation of the cuboid containing the underlying domain.
4. Clip the cuboid-mesh with the given domain boundaries.

In general there does not exist a pure CVT for the domains with non-flat boundaries. We are also forced to distinguish between internal cells and patch cells. We think with patch on the set of the faces, which compound the domain boundary. In general the mesh patches can get or loose some faces during the simulation. For this purpose, we use additional bounding surfaces, which bound the domain. So we have the boundary part of the volume mesh (the set of the patch faces) as well as the bounding surfaces. The boundary motion means in our work the motion of the bounding surfaces. This allows us to keep the topology of the boundary representation fixed.

In general we can handle both parametric surfaces and triangulated surfaces as boundary representation of the domain. Here we focus on managing triangulated surfaces. In this work we use two different techniques to handle the boundary cells described in subsection 5.2. The next subsection shows how the cell density function can be calculated.

5.1 Computation of the cell density function

As described above we start with a discretized model. This means that the user has defined an appropriate mesh edge size for each edge of the model. Further we refer to the mesh edge size as feature size. Qiang Du [18] proposed to take for the cell density function the inverse 5th power of the feature size. Some experiments have confirmed that this strategy provides the best description of the cell density function. Let e_{ij} and n_i be edges and number of edges of the Voronoi cell V_i, then we define its feature size as

$$s_i = \frac{\sum_{j=1}^{n_i} \|e_{ij}\|}{n_i}. \tag{8}$$

The cell density ρ_i of V_i is defined by:

$$\rho_i = \frac{1}{s_i^5}. \tag{9}$$

In order to determine the distribution of the cell density in the interior of the domain we first compute the distribution of the feature size in the whole domain, and than we get the cell density using the equation (9). For the distribution of the feature size in the interior we solve the Laplace equation:

$$\nabla \cdot (d^2 \nabla s) = 0, \tag{10}$$

where d square denotes the diffusion coefficient for the feature size. In order to solve the equation (10) we employ the Finite-Volume method using the mesh from the previous Lloyd's iteration. So we get the feature edge size on the cell centres of the mass. After that we use linear interpolation in order to get the feature edge size on the vertices of the mesh.

An appropriate mesh grading can be achieved by variation of the diffusion coefficient. It turned out that, the inverse distance from the boundaries often provides an optimal mesh grading. The algorithm for computing of the distance to the nearest patch was implemented in OpenFOAM® [15]. With the cell density function on the cell centres we interpolate it to the mesh vertices.

5.2 Computation of generators

For the internal cells we take the centres of mass as the new generators. The centre of mass for a polyhedron V_i with respect to the cell density function ρ can be computed as follows. First, V_i is decomposed into n tetrahedra as proposed by [18]. This straightforward decomposition technique is very efficient and always feasible for convex polyhedra. Let x_j and m_j be the centres of mass and the masses of the tetrahedra forming the polyhedron V_i. Than the centre of mass y_i of V_i can be computed from

$$y_i = \int_{V_i} y\rho(y)dy / \int_{V_i} \rho(y)dy = \sum_{j=1}^{n} x_j m_j / \sum_{j=1}^{n} m_j.$$

The centre of mass and the mass of each tetrahedron can be computed by any quadrature rule. We now distinguish between so-called constrained centroidal Voronoi tessellation (CCVT) and unconstrained (only clipped) CVT. In our work we use both methods depending on the given geometry.

CCVT By the CCVT we will understand a CVT, where generators of the patch cells are constrained to be located on the surface. The CCVT for surfaces in $\mathbb{R}^n$ was originally proposed by Qiang Du, see [**?**]. In his further work [18] he suggests to project the centre of mass of the patch cell onto the bounding surface. Unfortunately, this procedure does not have a unique solution. The projection of sphere centre to the sphere surface, for example, leads to an infinite number of results. This could occur if the centre of mass is located on the centre of the osculating circle of the surface. In order to avoid this problem we use another method. Depending on the number of patch faces of the patch cell we perform an appropriate step:

1. One patch face: The centre of mass is taken as the generator of the appropriate patch face. We decompose the underlying face into a number of triangles and use the same technique as for a polyhedron mentioned above.
2. Two patch faces: When both faces contain a common edge we reduce the problem to a one dimensional problem. Hence, the generator is computed as the centre of mass lying on the common edge. If both patch faces do not contain a common edge, we perform the step 1 for the patch face with the centre nearest to the centre of mass of the cell.
3. At least three patch faces: If we can find three patch faces, which contain a common vertex, we set the generator to the common vertex. In case of

several common vertices, we take the vertex with the nearest distance to the centre of mass of the cell. If there are no common vertices, then we search for the faces with common edges, and perform step 2 for the common edge, which centre located nearest to the centre of mass of the cell. If we can find neither a common vertex nor a common edge, we perform step 1 for the patch face with the centre nearest to the centre of mass of the cell.

It is clear, that the generator produced by the described technique does not reside exactly on the surface, but is located very close to it. The technique described here is very simple and leads always to a unique solution.

Clipped CVT Another technique to manage patch cells should be described in the sequel. It is not necessary that, the generators are constrained to the boundary. It is also possible to simply clip the boundary cells by the bounding surface. In such case the generator can be inside as well as outside of the domain. This has some disadvantages:

- The domain boundary gets new cells or loses cells during the simulation. As a consequence small patch faces arise on the boundary. This reduces the mesh quality.
- In comparison to CCVT, there is no optimization of the patch cells.
- For large deformations of the domain boundary per simulation step it can happen, that cells lie completely outside of the domain.
- The volume of the discretisation domain changes slightly at each step of the simulation.

A major advantage of the clipped CVT compared to CCVT is the exact fulfilment of the cell density function constraints. In case of the CCVT the cells stick to the surface and can not leave this surface. If we have too many cells on the surface and not enough in the interior, we can not fulfil the required cell density function constraints.

5.3 Tessellation of the cuboid

With the new generators, we can proceed with the tessellation of the underlying domain. First, we compute the Voronoi tessellation of the cuboid containing the whole domain and after that we clip this mesh with domain boundaries. For the computation of the CVT of the cuboid we opted to use the voro++ library [23]. This library deals directly with Voronoi cells and computes a Voronoi tessellation by the cuts with perpendicular-bisector planes. The cells are handled and saved independently by this library. We use only one function from this library to compute the plane cuts. Now we describe the computation of a new Voronoi cell.

- First, we initialize each Voronoi cell as cuboid so that the whole domain is contained in it. The dimensions of the cuboid are cumputed as dimensions of the domain plus offset. Each neighbouring generator creates with own

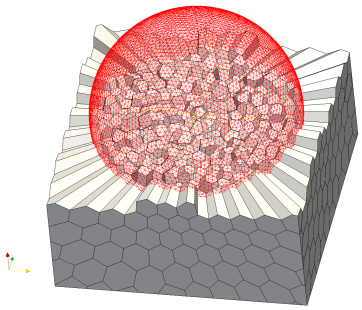
generator the corresponding perpendicular-bisector. Our task is to find the correct indices of the generators, which perpendicular-bisectors would contain a face of the new Voronoi cell. In order to determine these indices we make use of the connectivity information of the previous mesh.
- We cut the initial cuboid-cell using the perpendicular-bisectors created by generator indices of the direct neighbours of the previous mesh.
- We use multiple levels of the old neighbouring indices, because each cell of the new mesh can get new neighbours if the mesh is moved. By neighbours we understand here all cells, which contain at least one common vertex with the current cell.
- After each cut the cell gets smaller. So we use the maximal radius of the own cell in order to determine whether the perpendicular-bisector created by the relevant neighbouring index cuts the cell. This can significantly speed up the cutting routine.
- We stop the recursive routine at a level where the previous level of the old neighbouring indices did not create the perpendicular-bisector, which intersects the cell. The experiments have shown that this technique works well and is very efficient.

After performing the described procedure for all cells, we achieve a decomposition of the cuboid which has the same dimensions as the cuboid used for the initial cell. It turned out that for each Voronoi cell it is usually sufficient to visit two levels of the neighbours from the previous mesh. Therefore, we get for each cell an average computational amount t_c, that only depends on the number of the visited levels of the neighbours. That means that t_c is independent of the whole number of the cells n. Therefore, for the n cells the decomposition of the whole domain takes the expected time $t_c \cdot n \approx \mathcal{O}(n)$.

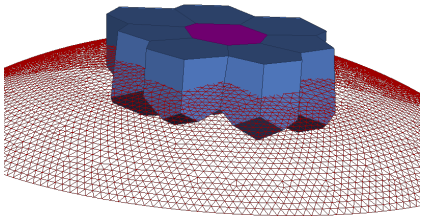
5.4 Clipping with boundaries

Once we have the decomposition of a cuboid covering the whole domain, we start clipping the underlying cells with bounding surfaces. Usually we have a lot of surfaces bounding the domain. Suppose we have found for each surface at least one intersected cell. So we can perform the cuts for remaining cells recursively using neighbourhood relations saved during tessellation of the cuboid. The tessellated cuboid is shown in Fig 1a. Fig. 1b shows the first intersected cell and its neighbours. Using the example in Fig. 1b we are explaining our procedure. During the cutting of each cell (magenta cell in Fig. 1b) the next intersected cell (blue cells in Fig. 1b) is determined by the face containing the edge which is currently being intersected. So we push the neighbour index into the so-called FIFO queue and search for the next intersected edge of the cell. Once the cut of the current cell is completed we pop out the cell index from the queue and compute the intersection for this cell. This process is terminated when all edges, that intersect the current surface, have been visited.

As mentioned above for clipping of the underlying mesh with each surface we need at least one cell intersecting this surface. For that we make use of the



(a) The mesh of the cuboid is coloured in grey. Boundary surface is coloured in red.



(b) Boundary surface is coloured in red. The first intersected cell is coloured in magenta and its neighbours are coloured in blue.

Fig. 1: Clipping with domain boundaries.

patch cells from the previous mesh. For each patch we choose an appropriate cell and compute intersections for each edge of this cell. Now we still need to know which part of the clipped cell remains in the domain and which must be removed. That can be done by means of the surface normal.

A similar approach for recursive cutting of the boundary cells with the corresponding surface was proposed by [8]. In comparison to this approach we handle the Voronoi cells directly instead of using the dual data structure known as the Delaunay triangulation.

5.5 Merging to the global mesh

After some iterations of Lloyd's method we achieve a mesh close to CVT within the limit of the predefined tolerance. Since all cells are computed independently, we need to merge these to a global mesh. Hence, we need to create the global vertices and faces from those local entities. The global faces can be created easily using the fact that each global face belongs to exactly two local faces. In order to construct the global vertices we march through the local faces and correlate the vertices for both corresponding faces. We mark the visited local vertices of both adjacent cells with new created global labels. Only the non-visited local vertices create new global vertex labels.

This technique works as long as the two corresponding local faces are equal. As a result of rounding errors there are neighbours with non-equal corresponding faces. Hence, the number of vertices contained in both faces is not equal. In such cases these faces are corrected. We compute correlations for both faces and remove the redundant vertex from the face with greater number of vertices. Since each vertex belongs to at least three faces the procedure also modifies other faces from the handled cell. We also have to check the modified local faces for equality to the corresponding local faces. In case of inequality, we correct the

underlying faces too. Each removal of a vertex leads to non-flat faces. Therefore the position of the affected vertices is computed by the least squares method, i.e. each affected vertex forms the smallest distance to the original planes containing it. Although the described method causes an increasing number of arithmetic operations the whole computational effort increases only very slightly, because only very few cells are affected. The proposed technique works reliably and is substantiated by a series of test cases.

6 Mesh motion cases

This section shows three simple examples with prescribed boundary motion. We emphasize that, the generated meshes have a very high quality and fulfil the quality-check criteria of OpenFOAM® at each time step of the simulations.

Fig. 2 shows a cuboid which top wall is moved down during the simulation. The cell density function is uniform. The simulation starts with a tetrahedral mesh (Fig. 2a). After some iterations of the Lloyd's method we achieve a mesh close to CVT (Fig. 2b). The cells partition the whole domain perfectly, because the generators are not constrained to lie on the boundaries (CVT technique). It can be observed that, the cell density of the cells increases for smaller volumes, because the number of the cells remains constant during the simulation. In the last step (Fig. 2e) the simulation arrives at a single thin layer of cells.

Fig. 3 illustrates a volume between two concentric spheres. The radius of the inner sphere is increased during the simulation. The cell density function is computed as the inverse third power of the sphere radius. Here we used a very fine triangle surface mesh, in order to reach a very small gap between both spheres. See Fig. 3f.

The next example, illustrated in Fig. 4, shows a cylinder with an enclosed sphere. The cell density function is explicit defined on the boundary and computed in the interior by Laplacian Smoothing as described in subsection 5.1. The generators are fixed on the parametric surfaces. This keeps the volume of the interior domain constant, because the boundary-mesh remains unchanged during any affine transformations of the enclosed sphere surface.

7 Validation of usability within the Finite-Volume Method

Obviously employing of the centroidal Voronoi meshes for the Finite-Volume method (FVM) can be very beneficial [**?**]. As already mentioned above, the proposed mesh motion approach do not affect number of cells during the simulation. We just move the cells like particles through space. In case of mesh motion the physical phenomena are described by so-called arbitrary Lagrangian-Eulerian Formulation. For further details please refer to [12]. For an incompressible and divergence-free flow the momentum equation and continuity equation is of the

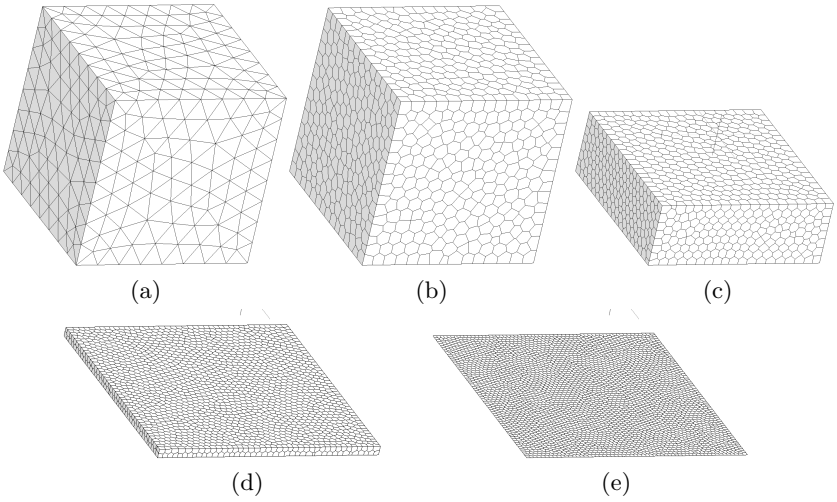


Fig. 2: Cuboid, top wall moved down (consequent steps (a)-(e))

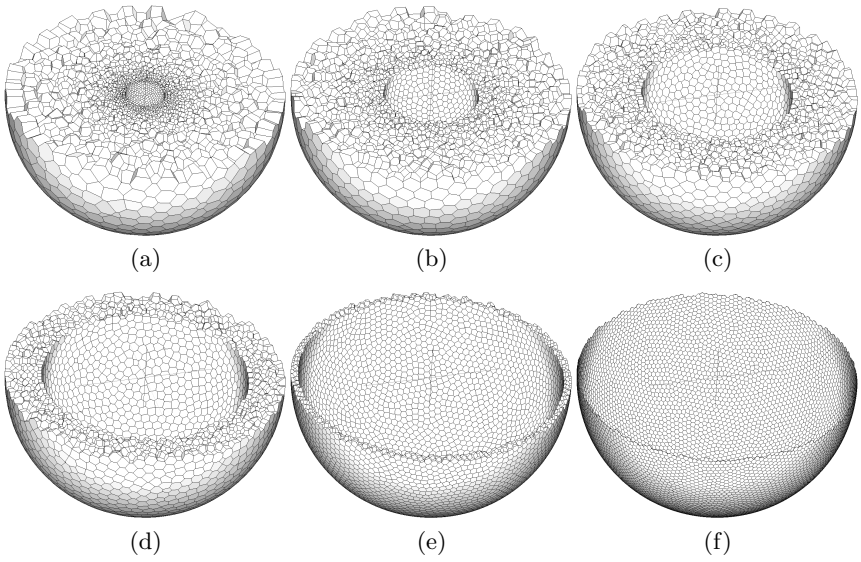
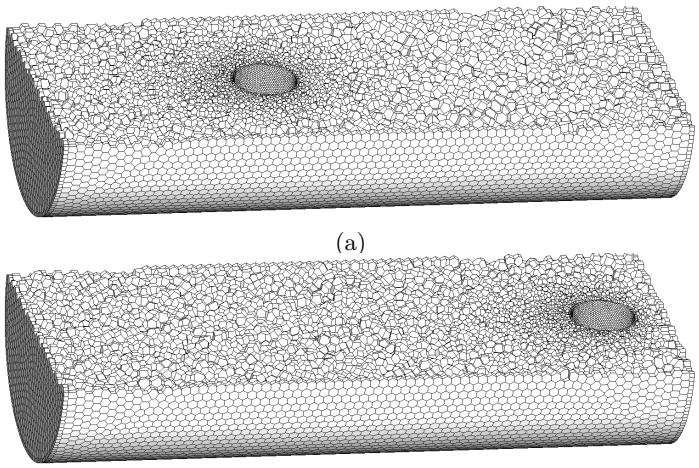


Fig. 3: Volume between two concentric spheres (consequent steps (a)-(f)).



(a)

(b)

Fig. 4: A cylinder with enclosed sphere (consequent steps (a)-(b)).

form:

$$\left.\frac{\partial U}{\partial t}\right|_{\chi} + \nabla \cdot ((U - U_m)U) - \nabla \cdot (\nu \nabla U) = -\nabla p \tag{11}$$

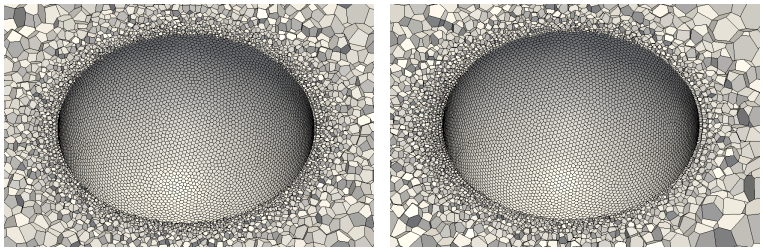
$$\nabla \cdot U = 0 \tag{12}$$

Where U, U_m, ν and p denote respectively the velocity, the mesh velocity, the kinematic viscosity and the pressure. In order to solve the system of the differential equations (11)-(12) we use the so-called Semi-implicit Method for Pressure Linked Equations (SIMPLE). For further details we refer to [1, 9].

For a validation of the developed mesh motion solver we need a simple model, which has an analytical solution. We decided to simulate the free-falling sphere in a viscous fluid. Such a case can be also validated by a steady state solution. For the purpose of comparisons we have to transform the steady state solution to the reference coordinate system of the transient case. For both simulations the following parameters were used:

- $v = 0.01\ m/s$ - sphere relative velocity
- $r = 0.001\ m$ - sphere radius
- $\eta = 0.000885$ - dynamic viscosity
- $R = 0.05\ m$ - radius of cylinder
- $H = 0.4\ m$ - length of cylinder

Before the start of the transient simulation we have to create a quasi-CVT mesh. After round 60 iteration of the Lloyd's method we got a quasi CVT-mesh, which is shown in the Fig. 5a. Considering the Fig. 5b we can note that, the cell density in the downwind region is significantly less than the cell density in the upwind region. This phenomenon will be explained in the section 8.



(a) The initial mesh (b) The mesh at the time 0.3585

Fig. 5: The quasi-CVT mesh in the area of the sphere. The whole mesh contains 775380 Voronoi cells. The x-axis points to the right.

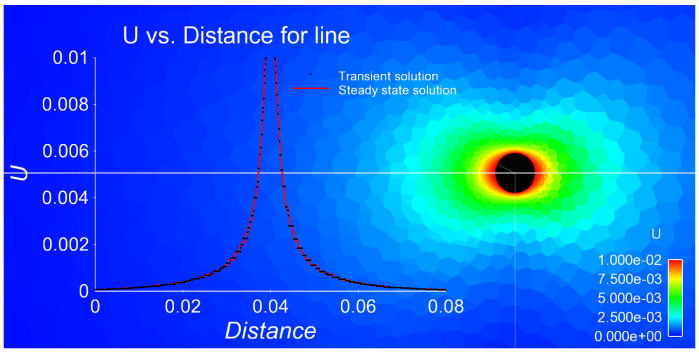


Fig. 6: Velocity field of the steady state and the transient simulations of free-falling sphere in a viscous fluid. The solutions are plotted over the distance on the line (0.16, 0., 0.)-(0.24, 0., 0.). The zero point on the x-axis corresponds to the point (0.16, 0., 0.) in the 3D-space. The steady state case is computed with 9000 iterations of the simpleFoam. The transient case is simulated up to 0.3585 seconds.

Fig. 6 shows the simulation results of both the steady state and the transient cases. Except for small differences, we get a good agreement of both fields. In order to make a quantitative comparison, we computed the drag force on the sphere:

$$F_D = \oint_S p\, n\, dS - \oint_S \tau \cdot n\, dS, \tag{13}$$

where p, n and τ denote pressure, outward-pointing normal to face vector and shear stress tensor. Using the formula (13) for both cases we get following results:

$$F_D^{st} \approx 1.7451e-7N \quad \text{(with 9000 simpleFoam steps)} \tag{14}$$

$$F_D^{tr} \approx 1.7358e-7N \quad \text{(transient up to 0.3585 sec.)} \tag{15}$$

$$|(F_D^{tr} - F_D^{st})\,/F_D^{tr}\,| \approx 0.53\% \tag{16}$$

We recognise that the deviation in the drag force is within the acceptable range. Furthermore, for the considered geometry the drag force can be calculated by means of the Stokes' Law:

$$F_D = 6\,\pi\,\eta\,v\,r\,\lambda \tag{17}$$

$$\lambda = \lambda_R \lambda_H = (1 + 2.1\,r/R)\,(1 + 3.3\,r/H), \tag{18}$$

where λ - Ladenburg-correctors for finite vessel dimensions with ($r \ll R$, $r \ll H$). Using the equations (15), (17) and (18) we get the following deviation in the drag force:

$$F_D^{Stoke} \approx 1.7526e-07N. \tag{19}$$

$$|(F_D^{tr} - F_D^{Stoke})\,/F_D^{tr}\,| \approx 0.96\% \tag{20}$$

Summarising the above we can say that the proposed mesh motion method can be used by Finite-Volume methods. We observed the mesh quality values during the whole transient simulation. For a detailed description of the mesh quality criteria please refer to [**?**]. The maxima of the critical mesh values are shown in table 1 and compared to a polyhedral mesh of a commercial preprocessing tool. Table 1 indicates that the CVT provides better quality meshes than classical

Table 1: Mesh quality results

	Max aspect ratio	Max non-orthogonality	Max skewness
CVT mesh	2.33210739	15.20106280	3.14209776
Commercial preprocessor	6.23371	55.1742	1.7585

mesh generators. The mesh skewness in case of the CVT is higher, because we got higher gradients of the cell density function in the area of the sphere. The mesh skewness can be improved varying the mesh diffusion coefficient as defined in the equation (10).

8 Conclusions and future Work

Up to now the CVT has been used for mesh generation and optimization. This paper shows the possibility to use the CVT for mesh motion. We use the term mesh motion, because the number of cells remains constant during the simulation. Previous approaches relating to retopologization usually lead to a change of the number of cells, see [25].

The computational effort of the developed mesh motion method is comparable with one iteration of the `pimpleDyMFoam` solver (See [15]). We are currently working on the development of an efficient solution for a treatment of the non-convex boundaries. To our knowledge, in all previous works, the non-convex boundaries lead to non-convex star-shaped Voronoi cells, see [8, 14]. A further

decomposition of such cells leads to convex cells, but these decomposed cells are not centroidal Voronoi cells.

As already mentioned in section 7 we get non-symmetric cell density distribution in the area of the moving sphere. This is due to the fact that, our cell motion approach is exclusively based on the CVT. The cells move because we try to reconstruct the CVT mesh. For that we just perform a few steps of Lloyd's method. In case of the sphere motion there are no motion condition for the laterally placed cells. The cells on the sphere surface just float through the surrounding cells. This leads to a high cell density in the back of the sphere. This problem can be solved by adding an additional displacement to the newly computed generator, which can be determined by the Laplacian of the boundary displacement. Using this strategy we have already performed some experiments, which shows that the cell density remains constant in the area of the sphere.

It seems interesting to use error estimators for the construction of appropriate cell density distribution. But here we are with Finite-Volume methods not in such a good situation as in the case of Finite-Element methods. But some results of [5] should be helpful.

References

1. Benjamin de Foy and William Dawes. "Unstructured pressure-correction solver based on a consistent discretization of the Poisson equation". In: International journal for numerical methods in fluids 34 (1999), pp. 463-478.
2. C. Farhat, C. Degand, B. Koobus, M. Lesoinne. "Torsional springs for twodimensional dynamic unstructured fluid meshes". In: Computer Methods in Applied Mechanics and Engineering 163 (1-4 Sept. 1998), pp. 231-245.
3. Carlo L. Bottassoa, Davide Detomib, Roberto Serra. "The ball-vertex method: a new simple spring analogy method for unstructured dynamic meshes". In: Computer Methods in Applied Mechanics and Engineering 194 (39-41 Oct. 2005), pp. 4244-4264.
4. Christoph Degand, Charbel Farhat. "A three-dimensional torsional spring analogy method for unstructured dynamic meshes". In: Computers & Structures 80 (3-4 Feb. 2002), pp. 305-316.
5. Robert Eymard, J.-M. Herard. "Finite Volumes for Complex Applications V". Wiley 2008.
6. Dehong Zeng, C. Ross Ethier. "A semi-torsional spring analogy model for updating unstructured meshes in 3D moving domains". In: Finite Elements in Analysis and Design 41 (11-12 June 2005), pp. 1118-1139.
7. Desheng Wang and Qiang Du. "Mesh optimization based on the centroidal voronoi tessellation". In: International Journal of Numerical Analysis and Modeling 2 (2005), pp. 100-113.
8. Dong-Ming Yan, Wenping Wang, Bruno Levy, Yang Liu. "Efficient Computation of Clipped Voronoi Diagram for Mesh Generation". In: Computer-Aided Design 45 (Apr. 2013), pp. 843-852.
9. Fue-Sang Lien. "A pressure-based unstructured grid method for all-speed flows". In: International journal for numerical methods in fluids 33 (1999), pp. 355-375.
10. George A. Markou, Zacharias S. Mouroutis, Dimos C. Charmpis, Manolis Papadrakakis. "The ortho-semi-torsional (OST) spring analogy method for 3D mesh moving boundary problems". In: Computer Methods in Applied Mechanics and Engineering 196 (4-6 Jan. 2007), pp. 747-765.

11. Hrvoje Jasak, Zeljko Tukovic. Automatic mesh motion for the unstructured finite volume method. Nov. 2006.
12. J. Donea, Antonio Huerta, J.Ph. Ponthot and A. Rodriguez-Ferran. In: Encyclopedia of Computational Mechanics, "Chapter 14, Arbitrary Lagrangian-Eulerian Methods" (2004).
13. Long Chen. Mesh smoothing schemes based on optimal delaunay triangulations. Math Department, The Pennsylvania State University, State College.
14. Mohamed S. Ebeida and Scott A. Mitchell. Uniform Random Voronoi Meshes. In: Proceedings of the 20th International Meshing Roundtable, Paris, France (2011), pp. 273-290. Sandia National Laboratories, P.O. Box 5800, Albuquerque, NM 87185-1318 msebeid@sandia.gov.
15. OpenFOAM C++ Documentation. url: http://foam.sourceforge.net/do cs/cpp/.
16. Pierre Alliez, David Cohen-Steiner, Mariette Yvinec, Mathieu Desbrun. "Variational Tetrahedral Meshing". In: ACM Transactions on Graphics. Proceedings of ACM SIGGRAPH 2005 24 (July 2005), pp. 617-625.
17. Qiang Du and Desheng Wang. "Anisotropic centroidal voronoi tessellations and their applications". In: SIAM Journal on Scientific Computing 26.3 (2005), pp. 737-761.
18. Qiang Du and Desheng Wang. "Tetrahedral mesh generation and optimization based on centroidal Voronoi tessellation". In: International journal for numerical methods in engineering 56 (2003), pp. 1355-1373.
19. Qiang Du, Maria Emelianenko. "Acceleration schemes for computing centroidal Voronoi tessellations". In: Numerical linear algebra with applications 0 (2005), pp. 1-19.
20. Qiang Du, Maria Emelianenko, and Lili Ju. "Convergence of the Lloyd algorithm for computing centroidal voronoi tessellations". In: SIAM Journal Numerical Analysis 44.1 (2006), pp. 102-119.
21. Qiang Du, Vance Faber, Max Gunzburger. "Centroidal Voronoi Tessellations: Applications and Algorithms". In: SIAM REVIEW 41.4 (1999), pp. 637-676.
22. R. Löhner, and C. Yang. "Improved ALE mesh velocities for moving bodies". In: Communications in Numerical Methods in Engineering 12 (1996) pp. 599-608.
23. Chris H. Rycroft. Voro++: a three-dimensional Voronoi cell library in C++. 2009.
24. S. Jakobsson, O. Amoignon. "Mesh deformation using radial basis functions for gradient-based aerodynamic shape optimization". In: Computers & Fluids 36 (2007), pp. 1119-1136.
25. Sandeep Menon and David P. Schmidt. "Conservative interpolation on unstructured polyhedral meshes: An extension of the supermesh approach to cell-centered finite-volume variables". In: Computer Methods in Applied Mechanics and Engineering, 200 (2011), pp. 2797-2804.
26. Sina Arabi, Ricardo Camarero, Francois Guibault. "Unstructured meshes for large body motion using mapping operators". In: Mathematics and computers in simulation 106 (2014) pp. 26-43.
27. Xia-ping Zhang, Dai Zhou, Yan Bao. "Mesh motion approach based on spring analogy method for unstructured meshes". In: Journal of Shanghai Jiaotong University 15 (2010) pp. 138-146.
28. Xuan Zhou, Shuixiang Li. "A new mesh deformation method based on disk relaxation algorithm with pre-displacement and post-smoothing". In: Journal of Computational Physics 235 (Feb. 2013) pp. 199-215.
29. Witalij Wambold, Günter Bärwolff. "New mesh motion solver for large deformations based on CVT". In: Procedia Engineering 82 (Oct. 2014) pp. 390-402.