Known-authors Automatic Multi-Material Mesh Generation for the Simulation of Metal Casting Processes

Robert Schneiders, Jan Thomsen, Andreas Schulz[*]

Abstract

We describe a robust algorithm for the automatic generation of mixed-element meshes for multi-material domains. Geometries are given as a set of overlapping triangulations, with minimal requirements to geometric quality. The algorithm overlays a geometry with a octree-like base mesh which is then adapted to the interfaces between materials and to external boundaries. Invalid elements are dealt with by smoothing, splitting, topology changes and optimization so that only valid elements remain. No user intervention is required. It was written by Schneiders, Thomsen and Schulz.

1 Introduction

MAGMA GmbH was founded in 1989 in Aachen and develops the program MAGMASOFT© for the simulation of metal casting processes: Mold filling, thermal solidification and shrinkage, residual stresses and distortion[MAG].It is a dedicated software solution to analyze the foundry processes, with a high focus on usability and ease of use for solving complex industrial problems.

1.1 Problem Specification

A cast part is manufactured by carving its negative form out of a mold. That also includes the runners, overflows and other parts of the casting system. Fig. 1 shows a cast part with runner and overflows, Fig.2 the corresponding cavity in the mold.

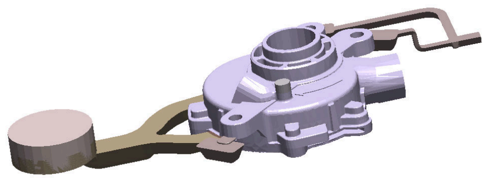
Figure 1: Part, runner and overflows

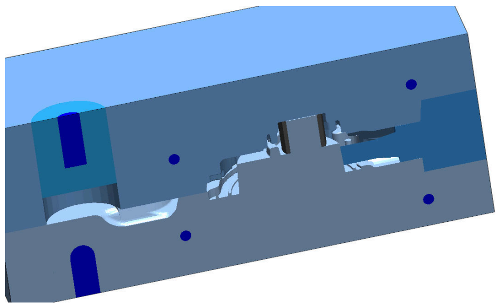
Figure 2: Mold, cooling channels and cavity

The full casting layout with part geometries, cores, inserts and cooling channels leads to a complex geometrical setup. This geometry must be meshed with a mixed-element mesh that conforms to the material interfaces and external boundaries, with only minimum user interaction allowed.

[*]MAGMA Giessereitechnologie GmbH.

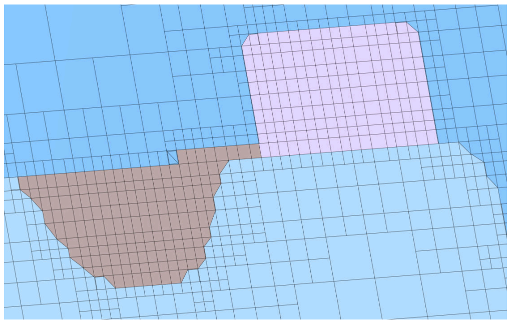

Figure 3: Cut through background mesh

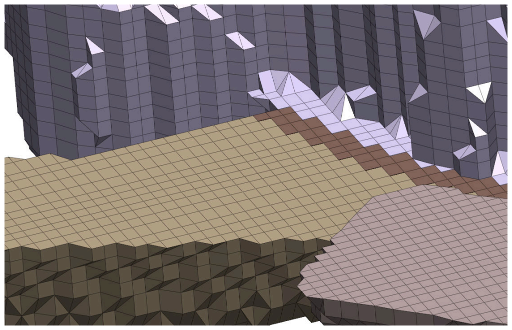

Figure 4: Background mesh with splits at boundary for the cavity

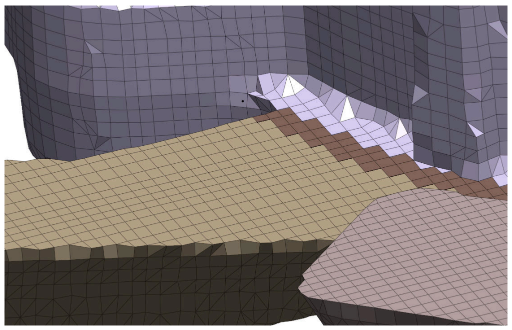

Figure 5: Projected and smoothed surface mesh

2 The Meshing Algorithm

2.1 Background mesh generation

The geometry to be meshed is given as a sequence of overlapping triangulations $T_0 ... T_{n-1}$, with materials $M_0 ... M_{n-1}$, T_0 usually being the mold and M_0 being mold material. The part of the mold to be meshed is defined as

$$T_0 \backslash (T_1 \cup T_2 \cup ... \cup T_{n-1}) \tag{1}$$

and its elements have material M_0. In general, the non-overlapped part of a triangulation $\overline{T}_i$ which is assigned material M_i is

$$\overline{T}_i = T_i \backslash (T_{i+1} \cup T_{i+2} \cup ... \cup T_{n-1}) \tag{2}$$

Each $\overline{T}_i$ is thus not overlapped by any $T_k, k > i$. To determine whether a given point p is inside a triangulation $\overline{T}_i$ without actually computing $\overline{T}_i$, ray casting is used. The material m a given point is assigned to is defined as M_m, where

$$M_m = \max_{k=0...n-1} : p \in T_k \tag{3}$$

The domain to be meshed is overlayed by a balanced octree-based background mesh. Cells at material boundaries are diagonally split, if necessary, and assigned materials according to the position of their barycenter in the triangulation (3). Fig. 3 shows a cut through the resulting mesh, fig. 4 shows part of the background mesh with interface elements split [Tim].

The quality requirements on the triangulations S_i are minor, small gaps are allowed. Triangulations may overlap or have gaps between them. This makes the algorithm robust and removes much of the requirements to geometry preparation for the user.

2.2 Capturing material interfaces

For triangulation T_i with material $m = M_i$, we consider the elements with material m. They define a submesh. The faces at the outer boundary of the submesh represent the material interfaces and constitute a polygonal surface mesh S_i that roughly corresponds to the triangulation T_i. To fit S_i to T_i, first the nodes of S_i are projected onto T_i. The surface mesh is then smoothed in a volume-preserving manner [Tau], and its nodes are re-projected on T_i. Fig. 5 shows the result.

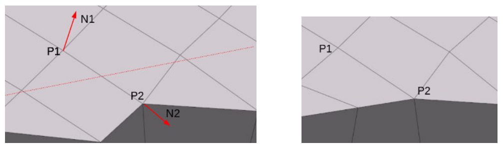

Figure 6: Edge capturing

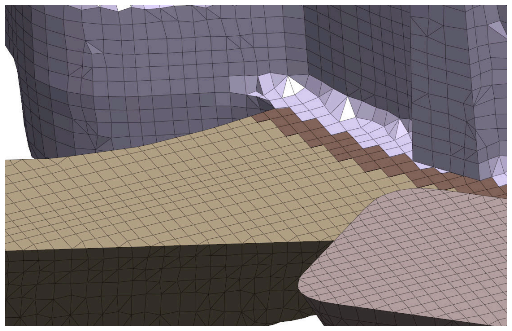

Figure 7: Optimized surface mesh

The smoothed surface mesh S_i in many places cuts off the edges of the triangulation T_i (fig. 6). To capture the edges, an algorithm proposed in [Bid] is used. In Fig. 6, the edge (P_1, P_2) cuts off the triangulation edge. P_1 and P_2 are on the triangulation T_i, with N_1 and N_2 being the normals on T_i. The normals planes intersect at line g, and P_2 is then projected onto g. The algorithm manages to capture most of the edges of T_i.

Edge capturing may deform triangles and quadrilaterals. Collapsed triangles are dealt with in a special step, and quadrilaterals with obtuse edges later.

The mesh surface nodes on T_{n-1} are now fixed for the rest of this part of the algorithm. The procedure is repeated for all nodes not yet fixed, for triangulations T_{n-2}, ..., T_0.

This part of the algorithm takes a mesh surface S_i and fits it to a triangulation T_i. This is a special case of remeshing algorithms which take a polygonal mesh and fit it to another one. There exists extensive research on this topic [All] [Kha].

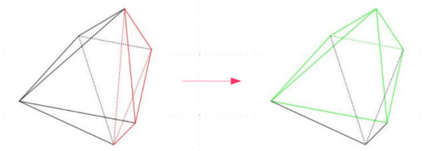

Figure 8: Edge Swapping

2.3 Guarantee element quality of the volume mesh

When projecting to and optimizing at material boundaries, internal element quality is is affected and often compromised. Several steps are done (see [Lo]):

- Volume weighted Laplacian smoothing is applied to the internal nodes.
- Elements with degenerated quads are split.
- Clusters of internal elements are improved by edge swapping.

At this stage, some invalid elements remain and must be dealt with. To do so, we first need to define the geometrical quality of an element E. The following properties are taken into account:

- Element Jacobian $J(E)$: For each node N of an element E, normalize the adjacent edges $e_j, j = 0, 1, 2$, and calculate the triple product $J(N) = e_0 \times (e_1 \cdot e_2)$. Element jacobian is then defined as $J(E) = \max_{N \in E} J(N)$. We require that $J(E) \geq J_{min}$.
- Face warpage: $W(E) \leq W_{max}$
- Element diameter $D(E) = \sqrt[3]{Vol(E)}/c$, where c is a characteristic edge length of the background mesh. We require that $D(E) \geq D_{min}$.

The limits can be set by the user.

The values are normalized and combined into a single quality value $Q(E) \geq 0$ so $Q(E) = 0$ for a perfect element, $Q(E) = 1$ being the largest accepted value. If $Q(E) > 1$, E is considered degenerate.

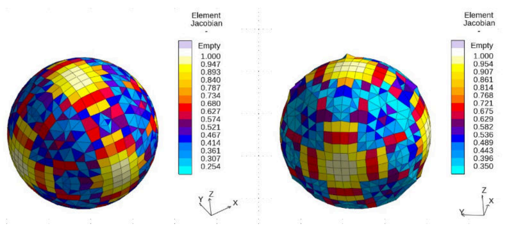

Figure 9: Accept deviation from geometry

For a mesh node N, node quality is defined as

$$Q(N) = \max_{E:N\in E} Q(E) \tag{4}$$

For a node N with $q = Q(N) > 1$, the algorithm tries to improve its position by moving it to 6 new positions; if it finds a position where $Q(N) \le q-\epsilon$, it moves the node to that position. This procedure is repeated, until no better position is found ($\epsilon = 0.001$ chosen for performance reasons).

Nodal optimization is then used as follows:

- Interior nodes are optimized
- Interior and boundary nodes are optimized; boundary nodes are constraint to stick to the corresponding geometry S_i (they are projected on S_i before quality evaluation.

This procedure improves many of the remaining bad elements. However, for non-trivial geometries, some remain. These are then improved by node optimization, boundary nodes being allowed to move away from the geometry. After this step, all elements fulfill the quality requirements.

Fig. 9 shows the principle: For a simple geometry, we want to generate a mesh with minimum jacobian value $J_{min} = 0.35$, a severe limit. This cannot be achieved by using the standard optimization procedure (fig. 9 left). After moving some nodes away from the geometry, all elements fulfill $J(E) > J_{min}$ (for lower values of J_{min}, no geometry deviation would be necessary).

In practice, all bad elements are healed, with the number and amount of geometry deviations being small (fig. 10). This optimization step acts as a safety valve: In almost all cases, meshes of numerically valid elements are generated, and the visual quality of the mesh is not compromised. The whole process does not require any user intervention!

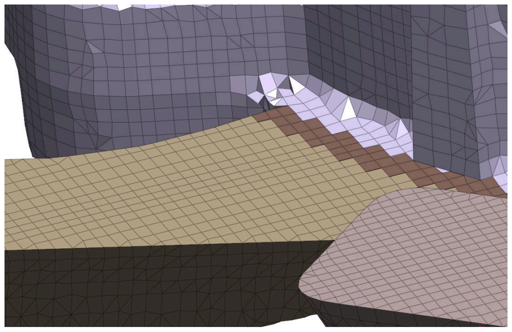

Figure 10: Final mesh

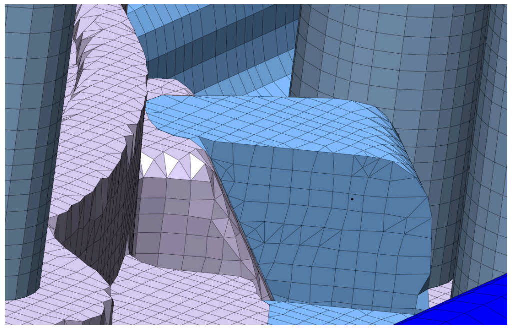

Figure 11: Final mesh, detail

3 Conclusions

We have presented an algorithm for meshing complex multi-material geometries for use in metal casting simulation. The algorithm is very robust and works without user intervention, making it easy to use for people in industry.

References

[MAG] *www.magmasoft.com.*

[Tim] A. Timalsina, M. Knepley, *Tetrahedralization of a Hexahedral Complex*, Proceeding of the 23th International Meshing Roundtable (2022)

[Tau] G. Taubin, *A Signal Processing Approach to Fair Surface Design*, Computer Graphics Proceedings, pp. 351–358 (2005)

[Bid] K. Bidmon and T. Ertl, *Generation of Mesh Variants via Volumetrical Representation and Subsequent Mesh Optimisation*, Proceedings of the 14th International Meshing Roundtable, 8 (2005), pp. 275–286

[All] P. Alliez, G. Ucelli, C. Gotsman, M. Attene, *Recent Advances in Remeshing of Surfaces*, In: De L. Floriani, M. Spagnuolo, (eds) Shape Analysis and Structuring. Mathematics and Visualization. Springer, Berlin, Heidelberg (2008), pp. 53—82.

[Kha] Khan et al., *Surface Remeshing: A Systematic Literature Review of Methods and Research Directions*, IEEE Transactions on Visualization and Computer Graphics. PP. 1-1. (2020) 10.1109/TVCG.2020.3016645.

[Lo] S.H. Lo, *Finite Element Mesh Generation*, CRC Press (2015)