

PARALLEL GENERATION OF UNSTRUCTURED SURFACE GRIDS

Udo Tremel¹ Frank Deister¹ Oubay Hassan² Nigel P. Weatherill²

¹*EADS, Munich, Germany {udo.tremel|frank.deister}@m.eads.net*

²*University of Wales Swansea, Swansea, U.K. {o.hassan|n.p.weatherill}@swansea.ac.uk*

ABSTRACT

In this paper a new grid generation system is presented for the parallel generation of unstructured triangular surface grids. The object-oriented design and implementation of the system, the internal components and the parallel meshing process itself are described. Initially in a rasterisation stage, the geometry to be meshed is analysed and a smooth distribution of local element sizes in 3-D space is set up automatically and stored in a Cartesian mesh. This background mesh is used by the advancing front surface mesher as spacing definition for the triangle generation. Both the rasterisation and the meshing are MPI-parallelised. The underlying principles and strategies will be outlined together with the advantages and limitations of the approach. The paper will be concluded with examples demonstrating the capabilities of the presented approach.

Keywords: unstructured surface mesh generation, geometry rasterisation, MPI-parallel, automatic, object-orientation

1. INTRODUCTION

The generation of unstructured surface grids is still one of the most interactive tasks to be performed during a numerical CFD analysis requiring a considerable amount of human effort. For the complex configurations currently in a routine use, the time to get a well suited surface grid tends to become a bottleneck[1] compared to the later stages such as volume mesh generation and numerical simulation. Two main reasons can be identified causing this.

One problem is the preparation of the geometry definition itself. Time constraints prohibit the detailed examination and clean up of the full geometry. Due to undetected and unrepaired features, failures generally occur during the meshing process. These errors stipulate human repair and modification steps until the geometry definition can be meshed successfully. To accelerate this process, the response time of the surface grid generator should be as short as possible.

The second reason is the, mostly interactive, defini-

tion of the local element sizes to be applied for different parts of the geometry. For example, for a well suited aerodynamic surface grid the leading and trailing edges have to be resolved accurately from the beginning, compared to other regions such as fuselage surfaces, etc. Even in the case of mesh adaptation cycles during the simulation, the initial surface grid should be able to resolve the major flow characteristics, so that only a few adaptation steps are applied in order to obtain an acceptable result. This implies that reasonable element sizes have to be defined for the (initial) surface grid. Standard means are tetrahedral background grids and sources[2] together with octree-based approaches [3, 4]. For complex configurations such as complete aircrafts, hundreds of sources are not uncommon, imposing on the user hours of tedious work.

In the following sections a recently developed surface mesh generation system is presented. Its objective is to reduce the above mentioned problems as far as possible. The object-oriented design and implementation

is described first, followed by a section focusing on the automatic definition of the local element sizes in 3-D space. In section 4 the parallelisation of the time-consuming tasks is illustrated and in section 5 examples demonstrate the capabilities of the approach. The paper is concluded by a summary.

2. OBJECT-ORIENTED DESIGN AND IMPLEMENTATION

Based on the existing FLITE-ST surface mesh generator[5] from the School of Engineering of the University of Wales Swansea, a new object-oriented (OO) surface mesh generator system, the ST++-system, has been developed at EADS M. The OO-methodology has been chosen because the OO concepts of polymorphism, inheritance and encapsulation[6] [7] inherently enable and support the building and maintenance of large and complex software systems. Compared to the procedural structured programming in, for example, Fortran77 or C, the resulting code is generally better understandable, maintainable, extensible and reusable. This is caused by the powerful features supported in OO languages such as strongly typed interfaces, templates, design patterns, etc.[8] [9] C++ has been selected as implementation language due to the rich OO features the language offers¹. Another reason is the downward compatibility to C which enables the integration and reuse of already available, validated and highly optimised C and Fortran77/90 routines.

2.1 ST++-System Overview

In Figure 1 the three major components of the ST++-system are shown. These objects interact together and

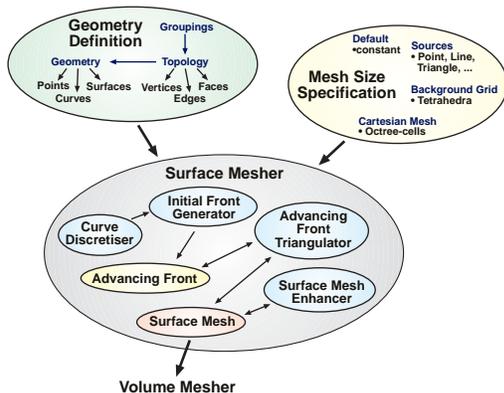


Figure 1: Surface mesh generator design.

¹Strong type checks, single and multiple inheritance, templates, abstract classes and interfaces, streams, exceptions, the standard template library (STL), ...

perform the following steps to generate a surface mesh:

1. The geometry definition imports the geometrical and topological items from CAD data.
2. The mesh size specification initialises the prescribed means to define the spacings.
3. The surface mesher starts the advancing front triangulation of the geometry, controls the mesh enhancements and prepares and exports the surface mesh later to be used in the volume mesh generation.

2.2 Geometry Definition

All the geometrical and topological entities are encapsulated in the geometry definition object illustrated in Figure 2. The implemented boundary representa-

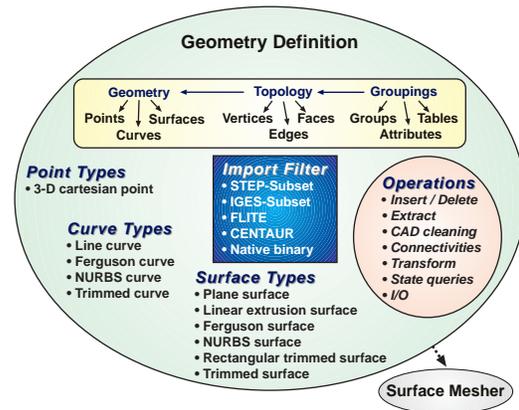


Figure 2: Geometry definition component.

tion (B-Rep) structure consists of geometrical points, curves and surfaces referenced by topological vertices, edges and faces respectively, defining the outer hull of the body to be meshed. All types of items can be grouped together and the grouping information can be kept in multiple tables to allow non-unique identifiers². Specific attributes can be assigned to each group, which enables, for example, the use of different tolerances for the configuration and the outer farfield parts. Import filters are available for the industrial relevant data exchange formats STEP[10], NASA-IGES[11] and the traditional FLITE format[5].

All the geometrical entities are derived from a generic point, curve or surface object defining a common interface for all derived types. For example, the following methods are part of the interface for curves:

²Depending on the CAD database the geometry was imported from.

- Evaluate(doubleT u, doubleT *outXYZ, doubleT *outTangent=0)
- CalculateProjection(const doubleT *xyz, doubleT &outU)
- CalculateArcLength(doubleT u1, doubleT u2, doubleT &outL)

Algorithms working only with such an abstract interface are independent from the real mathematical representation of the underlying parametric spline composite curves and tensor-product surfaces. Implemented are various types of curves and surfaces such as Ferguson-splines[12], Bezier-splines[13] or NURBS[14].

For dynamic modifications the geometry definition object offers methods for the insertion and deletion of geometric entities, for the transformation of existing entities, etc. One important capability is the extraction of arbitrary subgeometries, which is heavily used by the parallel mesher to extract subparts transmitted to and meshed by another process.

2.3 Mesh Size Specification

Another important aspect of mesh generation, besides the handling of geometrical CAD data, is the control over the spatial distribution of size and shape of the elements to be generated. Inside the ST++-system, this is the task of the mesh size specification object shown in Figure 3. The local element size at a cer-

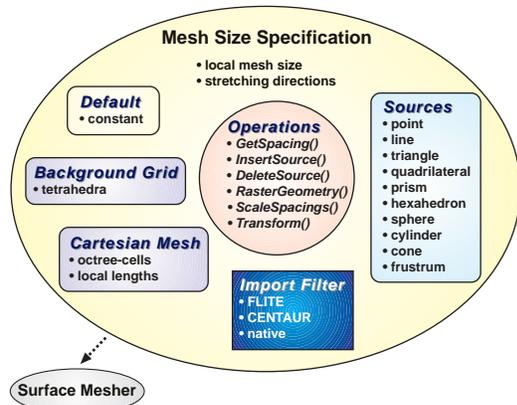


Figure 3: Mesh size specification component.

tain point in 3-D space is determined by the minimum size defined by all active objects. For that purpose, a Cartesian background grid, a tetrahedral background grid and sources are available, which can be used independently from each other depending on the requirements of the user. Additional types of application specific sources can easily be added, because all sources are derived from a generic source object defining the following common lean interface for all sources:

- GetMinSpacing(doubleT &outMinSpacing)

- GetSpacing(const doubleT *xyz, doubleT &outSpacing)

These methods only have to be specialised and implemented to add a new source type transparently to the objects working with the mesh size specification object.

Several methods are offered to achieve dynamic modifications of the local edge lengths. Sources can be added, deleted or modified dynamically, all spacings can be scaled in common and spatial transformations can be applied to all entities. These operations are intended to be used during dynamic mesh modification processes originating from transient simulations, design optimisations, etc.

The *automatic* determination of a well suited Cartesian background mesh based on a rasterisation of the CAD geometry is presented in section 3. This approach differs from the octree based approaches presented in [3, 4], because the sizes of the Cartesian cells are largely independent from the local element sizes calculated. Only the scalar quantities defined in each Cartesian cell are used to derive the mesh size at a certain point in space. Experience has shown that the Cartesian cell size is generally two to eight times larger than the calculated lengths, hence, the Cartesian meshes created are much smaller than those grids generated in [3, 4].

2.4 Surface Mesher

The kernel of the ST++-system is the surface mesher component illustrated in Figure 4. Its task is the

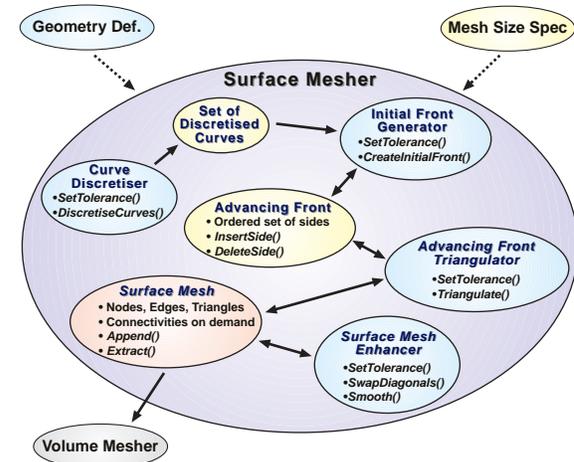


Figure 4: Surface mesher component.

control and the exchange of data between the different objects responsible for the different steps of the

meshing procedure. Based on the advancing front algorithm[15] [16] the different steps are spread across the curve discretiser, the initial front generator and the advancing front triangulator object. For each topological face to be meshed, the curve discretiser discretises all the topological edges connected to the face if not already done. This set of straight sides is given to the initial front generator which builds up the initial advancing front. The triangulator uses this starting front to generate the triangles added to the surface mesh. When no sides are left in the front, the triangulation is finished and the surface mesh is transferred to the mesh enhancer for optimisation. After optional postprocessing steps, such as removal of duplicate nodes and edges, correcting the orientation of the facets, checks, etc., the merged surface meshes can act as the starting point for a volume triangulation.

3. GEOMETRY RASTERISATION

A fully automatic and parallel feature-based rasterisation of native CAD data has been developed. The local curvature and characteristic length are investigated along CAD curves and inside trimmed CAD surfaces in order to define local sample lengths. A locally refined Cartesian background mesh (octree data-structure) is constructed to prolongate and therewith smooth the sample lengths. Additionally, Cartesian cells inside the geometry may be blanked out in order to avoid length prolongation through solid bodies. During the surface mesh generation, the Cartesian background mesh serves as the mesh size specification. Details are provided in [17].

3.1 Rasterisation of Native CAD Curves

The rasterisation of CAD curves are controlled mainly by three sampling parameters specified by the user: minimal arc length L_{min} , maximal arc length L_{max} and maximal curvature angle α_{max} . Now a CAD curve is subdivided into consecutive curve segments applying the following three sampling criteria: the arc length of a curve segment must not be smaller than the specified minimal arc length L_{min} . Conversely, the curve segment length must be smaller than the maximal arc length L_{max} . Finally, the curvature angle must be smaller than the maximal curvature angle α_{max} . This last sampling criterion is only applied, if the arc length is larger than the minimal arc length L_{min} . The curvature angle is taken as the angle between the tangential vectors at the two end points of a curve segment. The sample length cannot be larger than the length of the corresponding CAD curve. Finally, all curve segments are approximated by straight lines. For each straight line a bounding box is determined, termed raster box, which controls the local resolution of the later to be generated Cartesian mesh.

3.2 Rasterisation of Native CAD Surfaces

The rasterisation of CAD surfaces requires the same three user specified sampling parameters L_{min} , L_{max} and α_{max} , which are used also for curve rasterisation. Because only the part inside a trimmed CAD surface is considered, a scan-line algorithm[18] [19] from computer graphics is applied. As a first step, the trimming CAD curves are approximated by sequences of straight lines. For this, they are rasterised as described earlier. The resulting straight lines in physical (Cartesian) space are transformed to the (u-v)-parameter space of the CAD surface, in which the remaining computation takes place. This discrete representation of corresponding trimming curves must not intersect each other because of the scan-line algorithm.

The second step consists of computing the stencil point distribution, where the local surface curvature will be investigated later. For both u- and v-direction equally distributed iso-curves (probes) of the CAD surface are rasterised applying again the previous curve rasterisation algorithm. However, this time the curvature angle is defined as the angle between the surface normal vectors at the end points of a curve segment. The end points of the evaluated curve segments are taken as the desired stencil points. For each direction, the final stencil point distribution is extracted from these probes. In Figure 5 the final stencil point distribution in u- and v-direction are represented by the circles.

Now the scan-line algorithm is applied separately for the u- and v-iso-curves, which are defined by the stencil points (third step). It identifies the parts of the iso-curves which are inside the polygon constituted by the sequences of straight lines and thus inside the trimmed surface. The demarcations are drawn as squares in Figure 5. However, these inner curve parts are rasterised again with the presented approach. Afterwards, the computed sample lengths are related to the stencil points located inside the trimmed surface (circles drawn with thick lines in Figure 5). Every stencil

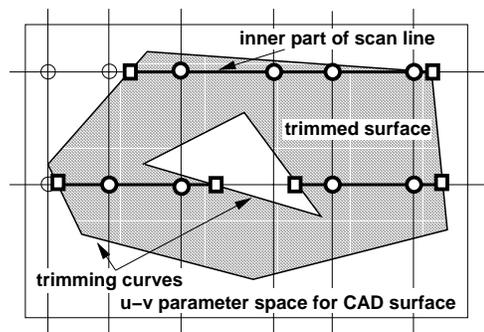


Figure 5: Rasterisation of trimmed surface.

point inside the trimmed surface gets a raster box according to the stored sample length. Additional raster boxes are created if the distance between two stencil points is larger than their sample lengths. In this way, the trimmed surface is completely enclosed by raster boxes, which are used for the generation of the Cartesian background mesh.

3.3 Cartesian Background Mesh

The locally refined Cartesian background mesh specifies the mesh size required by the surface mesh generator. It is based on the hierarchical octree-data structure describing the connectivity between the Cartesian cells [20] [21] [22]. The raster boxes along the CAD curves and CAD surfaces determine the local resolution of the Cartesian background mesh: all Cartesian cells which intersect a raster box are identified. These Cartesian cells must not be larger than the current raster box. Besides, the sample length of the corresponding curve segment, or rather, the surface stencil point is stored in every intersected Cartesian cell. At the end, the Cartesian background mesh is smoothed accordingly to a one-level difference rule: it is not allowed that two neighbouring Cartesian cells differ by more than one refinement level.

The stored sample lengths are prolonged through the Cartesian mesh. The rate of change between adjacent Cartesian cells is limited by an user-defined slope. In this way, a smoothed sample length distribution is achieved in the complete flow domain. Moreover, the user is able to control the rate of coarsening of the triangulation by modifying this slope parameter. Finally, the gradient of the sample length is calculated using a least square method.

During generation of the surface triangle mesh, the Cartesian background mesh specifies the local mesh size. For each point, the local mesh size is required. First, the Cartesian cell is identified applying the hierarchical octree data-structure enclosing this point. Then the sought mesh size is interpolated linearly using the sample length and its gradient, which is stored in the Cartesian cell. The smoothed mesh size specification is also available in space and therefore usable by a volume mesh generator.

3.4 Blanking out of Solids

The penetration of the sample length through solids is avoided. This means, that the local mesh size of the lower side of a thin geometry does not influence the mesh size on the upper side. Therewith, unneeded refinement is avoided and the resulting surface mesh is locally more homogenous. Especially this affects the quality of a possible quasi-prismatic mesh, because the local prismatic mesh height strongly depends on the

corresponding (underlying) surface triangle. Figures 6 and 7 show the impact of blanking out solids for the nacelle of a generic transport aircraft: the smaller sample lengths of the engine (Figures 15 and 16) penetrate through the solid nacelle coating and reduce the sample lengths there, Figure 7. In contrast, Figure 6 illustrates that this does not occur, if the prolongation through solids is avoided. For thick geometries, blanking out of solids is not necessary.

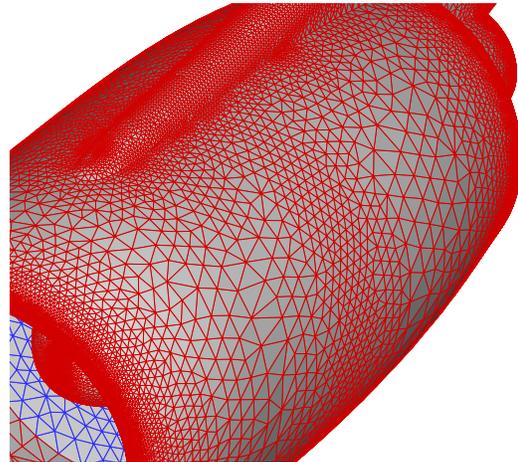


Figure 6: Without blanking out solids for nacelle (generic transport aircraft).

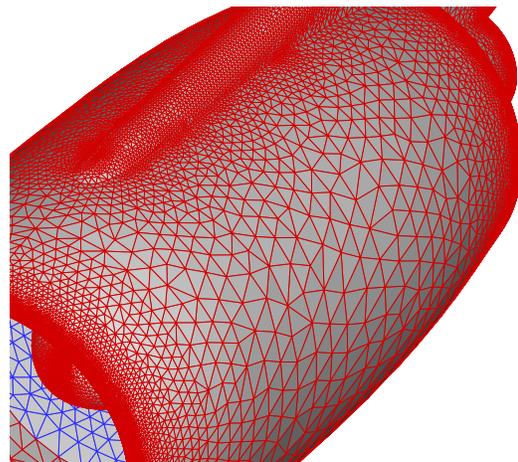


Figure 7: With blanking out solids for nacelle (generic transport aircraft).

In order to prevent sample length prolongation through solids, all Cartesian cells inside solids are blanked out: these cells simply are not considered for the length prolongation procedure. As pre-requisite, a

closed initial surface mesh of the geometry is required. If an initial surface mesh is not available (for example STL output format of the CAD system), the initial surface mesh is generated without blanking out solids. Here - after rasterisation of the geometry and before length prolongation - the smoothed Cartesian mesh together with the initially stored sample lengths are stored. Therewith, it is avoided to raster the geometry again for generating the final surface mesh.

After finishing the initial surface mesh, the sample length is prolonged again using the Cartesian mesh previously stored. But this time, all Cartesian cells inside the body are not considered for the length prolongation. In order to blank out inner cells, all Cartesian cells are identified, which are intersected by the initial surface mesh. The locations (inside / outside) of the remaining cells are found using a ray-tracing and a coloring algorithm[19] [22]: first the location of a cell (with undefined location) is determined by ray tracing. Afterwards, all neighbor cells obtain the same location recursively, which are not marked to be intersected by the geometry. Figure 8 presents all Cartesian cells which are inside the generic transport aircraft, whereas the intersected cells are drawn in Figure 9. Here, the small pictures show the regions marked by the black circles.

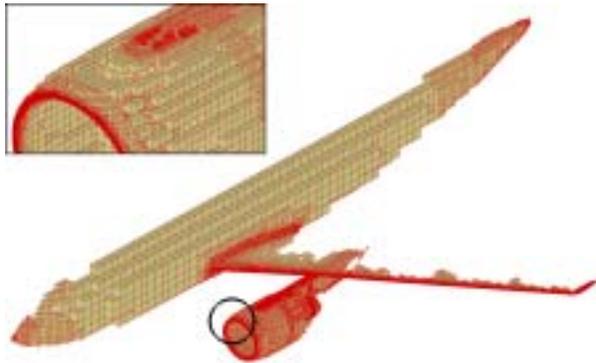


Figure 8: Cartesian cells inside the geometry (generic transport aircraft).

4. PARALLELISATION

To reduce the response time of the ST++ surface grid generator, the most time-consuming tasks are parallelised based on the message passing programming model by the use of the MPI-standard[23] [24]. This enables the efficient utilisation of both shared and distributed memory systems, which would not be the case for a shared memory parallelisation based on multithreading. Due to this distributed memory approach also cost-effective PC-cluster type hardware can be

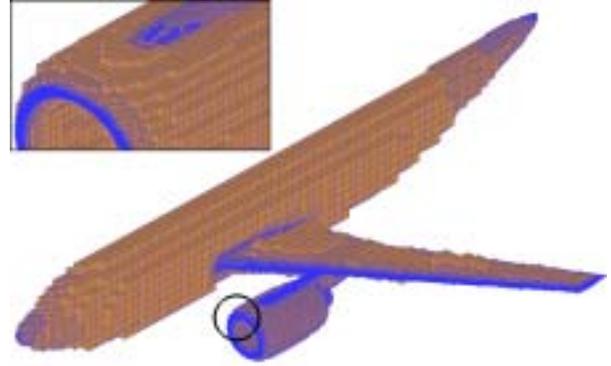


Figure 9: Cartesian cells intersected by the geometry (generic transport aircraft).

used successfully for parallel surface meshing as will be shown in section 5.

4.1 Parallel Geometry Rasterisation

The first computational intensive part is the rasterisation of the geometry. Here the main loop over all edges and faces is parallelised in a pipeline approach, which is illustrated in Figure 10. At the beginning the n par-

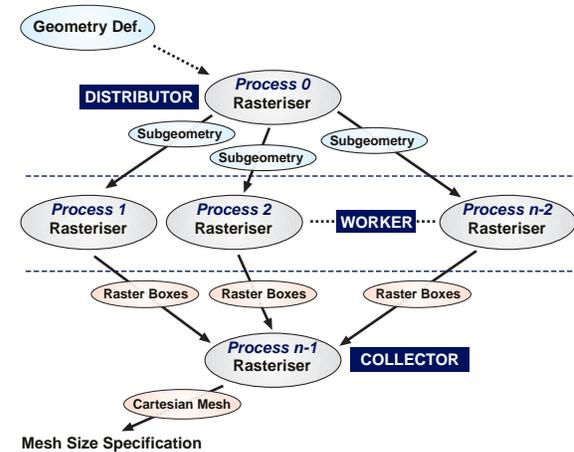


Figure 10: Parallel rasterisation of the geometry.

ticipating processes are subdivided into one distributor, one collector and $n - 2$ workers performing the analysis. To achieve an automatic load balancing the workers first have to ask for a new set of edges and/or faces to be rastered. By this 'work on demand'-strategy the load imbalance caused by different geometric entities, parameters, etc., is minimised. When a worker process finished the rasterisation of its current subgeometry as described in section 3, the computed raster boxes are

sent to the collector process for an adaptation of the Cartesian mesh. At the end the collector process performs all further postprocessing operations (smoothing, length prolongation, I/O, etc.) on the Cartesian mesh in sequential mode. Hence, the scalability of the parallelisation is limited, but up to a modest number of processes a sufficient speedup of more than one order of magnitude is obtained as demonstrated in section 5.

4.2 Parallel Surface Meshing

For the parallel surface meshing the loop over the faces to be meshed is parallelised similar to the parallel rasterisation. This is shown in Figure 11. After each

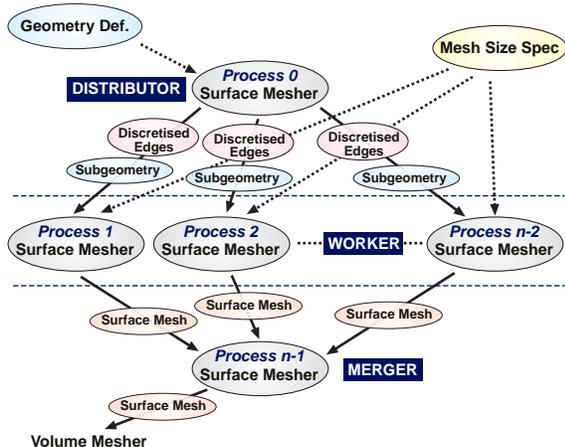


Figure 11: Parallel surface meshing.

process has determined its role (distributor, merger or worker), the necessary initialisation steps are executed. The distributor reads in the complete geometry and all workers are initialising the mesh size specification. Then the parallel meshing starts. Each worker queries for the next set of faces to be meshed to enable an automatic load balancing. Depending whether or not the distributor pre-discretises the face boundary edges, the next set of geometric entities is sent to the corresponding worker with or without the (already) pre-discretised edges. This pre-discretisation is mandatory due to floating point roundoff errors, which may cause an edge to be discretised with one side more or less, which prevents the recombination of the sub-meshes into one consistent mesh due to non-matching boundaries. Heterogeneous clusters running different types of processors are candidates for this parallel pitfall. When the worker has finished the advancing front triangulation and the mesh enhancements, the surface mesh is transmitted to the merger process. After all faces have been meshed, the merger assembles the complete mesh by combining all the received sub-meshes. On the final grid further postprocessing op-

erations (unifying the orientation, etc.) are performed in sequential mode.

As with any pipelining strategy, it is important to:

- keep the pipeline filled,
- select the optimal length of the pipeline,
- optimise the chunks of data to be processed between different stages.

Hence, a CAD model of reasonable size should be used if also many processors are involved, otherwise a reasonable scalability will not be obtained. Additionally, faces requiring more computational effort should be processed first during the parallel meshing to balance the throughput. Especially at the end, such faces should not occur, otherwise some processors continue working while most of the others have already finished their work, which can heavily limit the scalability of the approach. Therefore a heuristic meshing weight is calculated for each face during the rasterisation. This weight is estimated by the sum over all inverse edge lengths times the surface areas of the corresponding raster boxes. Such a weight is approximately proportional to the number of triangles to be generated for the face and enables the above mentioned optimisations concerning the order in which the faces are meshed.

The size of the subgeometries can be adapted accordingly to the capabilities of the underlying communication network. Smaller subsets can be used for low-latency/high-bandwidth networks, whereas larger subgeometries might be more favourable for less performant interconnects reducing the number of messages to be exchanged. However, this also depends to a large extent on other factors such as the size of the geometry, the processors, etc.

It is clear that the maximum speedup achievable is inverse proportional to the maximum time needed to raster/mesh a single face. If some faces consume a large amount of computational time, the parallel execution will not perform as expected. In such cases a fine-grain parallelisation of the underlying algorithms will have to be used to further speed up the process. Nevertheless, if the meshing time can be reduced more than an order of magnitude as will be shown in section 5, the result is worth the effort inherent with the presented approach.

4.3 Parallel Surface Remeshing

The parallel surface remeshing presented in Figure 12 is characterised by a three stage approach:

1. Parallel surface mesh analysis

2. Parallel curve discretisation
3. Parallel surface remeshing

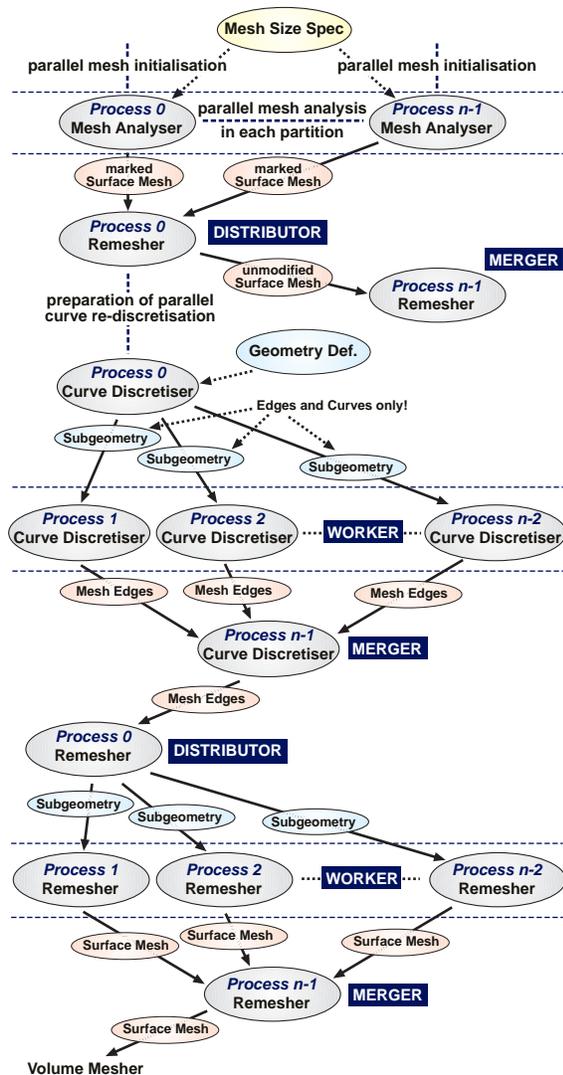


Figure 12: Parallel surface remeshing.

It starts with the initialisation of the modified mesh size specification in every process. Modifications can be caused by the addition/removal of sources, by a modified raster length prolongation, etc. For a parallel analysis the existent surface mesh is partitioned across the available processes with one element overlap across the domains by means of the *MeshLib*-library³. In the first stage all elements (triangles) containing edges not respecting the mesh size are marked within

³An OO library developed by the first author for the MPI-parallel handling of hybrid unstructured meshes including partitioning, decomposition, communication, etc.

each partition. To ensure that both the sequential and the parallel marking algorithms result in the same set of markers, the states of the external elements⁴ have to be exchanged after each marking loop in the parallel version. After this parallel analysis step, the partitions including the markers are collected again on the single distributor process, which extracts the parts to be remeshed and forwards the unselected parts directly to the merger process. In the second stage, the parallel discretisation of all internal edges lying on curves of the geometry is performed. Start- and endpoints of such segments are determined by the distributor and sent, together with the underlying geometric entity, to the next worker waiting for data. The discretised segments are collected and recombined by the merger and returned back to the distributor when all discretisations have been performed. In the third stage the distributor extracts the outer boundary of each hole and sends away these edges, together with the underlying surface definitions, to the next worker querying for data. The remeshed holes are collected at the merger and inserted again into the existent mesh.

It depends on the application area whether a complete regeneration from scratch or a local remeshing based on the already existent surface mesh is the better approach. Concerning dynamic CFD simulations the remeshing will be the alternative of choice because most of the volume mesh is normally kept fixed and only a small subset of the mesh has to be modified. With a complete new surface mesh also the entire volume mesh would have to be regenerated causing much more overhead compared to the local remeshing. During the generation of the initial surface mesh a complete regeneration might be the best option due to the normally similar time required and the slightly better quality.

5. EXAMPLES

5.1 Generic Transport Aircraft

The presented concept of fully automatic surface mesh generation is demonstrated with a generic transport aircraft. The CAD description is imported via the STEP format and consists of 864 NURBS curves and 346 NURBS surfaces. The only input parameters (for CAD surfaces) specified by the user are the minimal arc length L_{min} (default value 0.25 [mm], for wing 0.1 [mm], for nacelle 0.3 [mm]), the maximal arc length L_{max} (default value 50 [mm], for nacelle 20 [mm], for farfield/symmetry plane 1 [m]) and the maximal curvature angle α_{max} (default 10 [°]). Additionally, the following CAD curves are rastered:

⁴Duplicated elements in the overlap area of a partition owned from another process

- wing trailing edge ($L_{min} = 0.75$ [mm], $L_{max} = 1.5$ [mm]),
- wing root edge ($L_{min} = 2.5$ [mm], $L_{max} = 5.0$ [mm]),
- and wing tip edge ($L_{min} = 0.5$ [mm], $L_{max} = 1.0$ [mm]).

We emphasize that not a single source is specified for this case. Figure 13 presents the surface mesh of the generic transport aircraft: the half configuration consists of 322204 triangles (final surface mesh). The initial surface mesh without blanking out solids contained 330320 triangles. Details of the wing tip region are shown in Figure 14. Furthermore, in Figures 15 and 16 the nacelle coating is opened, thus the inner part of the engine is revealed. Here, Figure 16 shows the detail of Figure 15, which is marked by the black circle. In Figure 17 timing measurements are given

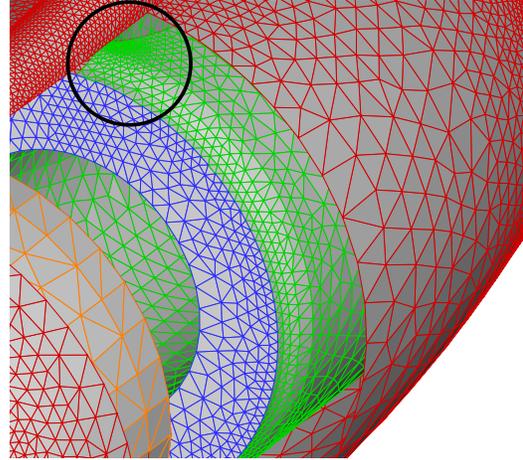


Figure 15: Inside view of nacelle for generic transport aircraft.



Figure 13: Surface triangulation for generic transport aircraft.

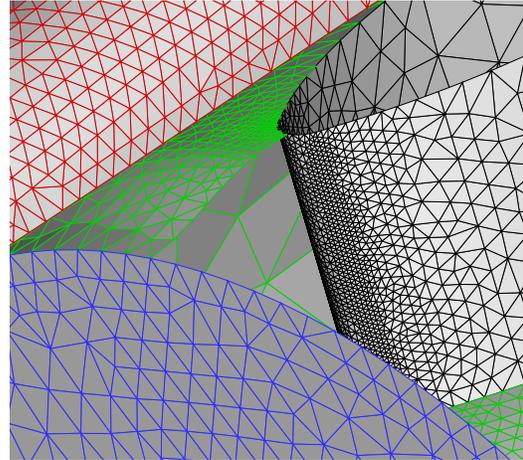


Figure 16: Detail of inside view of nacelle for generic transport aircraft.

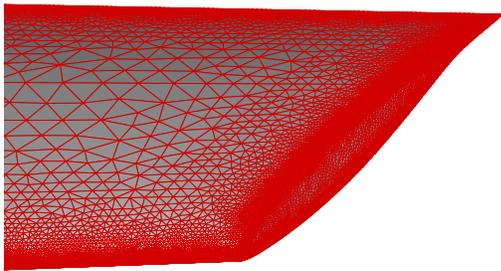


Figure 14: Wing tip of generic transport aircraft.

for parallel rasterisation and surface meshing. All runs had been performed on a PC cluster system running with XEON 2.67 GHz processors connected via gigabit ethernet. Although only a parallel meshing speed-up of about four can be obtained for 16 processors, the

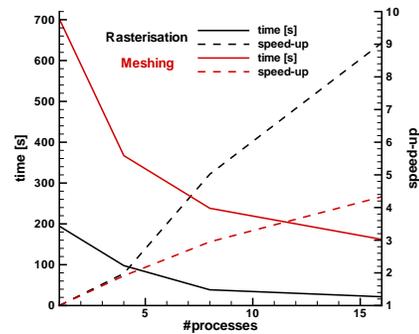


Figure 17: Performance figures for generic transport aircraft.

important total time (also including I/O operations, ...) to get a surface mesh for the complete configuration starting from a geometry without any sources is reduced down to less than five minutes.

5.2 Advanced Fighter-Type Aircraft

The geometry definition of this example consists of 6788 NURBS curves and 2749 NURBS surfaces and is imported via the STEP-format. Input parameters (for CAD surfaces) specified by the user were the minimal arc length L_{min} (default value 4 [mm], for air data sensors 0.5 [mm], for nearfield 0.5 [m]), the maximal arc length L_{max} (default value 100 [mm], for air data sensors 5 [mm], for nearfield 5 [m]) and the maximal curvature angle α_{max} (default 15 [°]). Again, no sources were used. The surface grid (containing 1.6 million triangles) shown in shaded mode in Figure 18 is thus solely based on the edge lengths calculated by the rasterisation process. In Figure 19 timing mea-

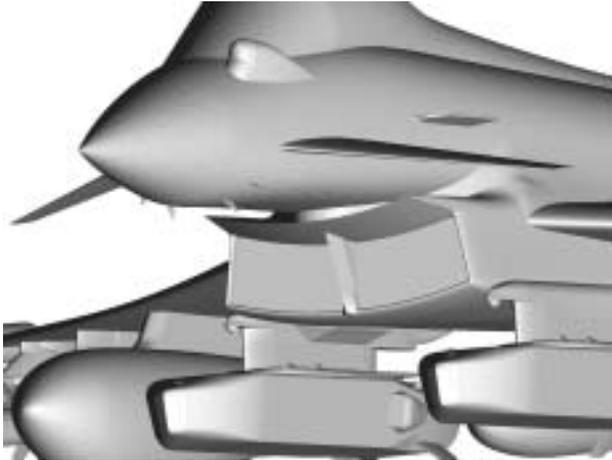


Figure 18: Front view of advanced fighter-type aircraft.

surements are given for parallel rasterisation and surface meshing. All runs had been performed on a PC cluster system running with XEON 2.67 GHz processors connected with a QUADRICS network. Speed-ups of more than one order of magnitude are obtained for both the rasterisation and the surface meshing. The most important result is the reduction of the total time needed to get a surface grid starting from the watertight geometry. For 32 processors, the turnaround time can be reduced from about 1:10 hours down to 10 minutes.

6. CONCLUSIONS

The recently developed ST++-system has been presented. The OO design and implementation of the

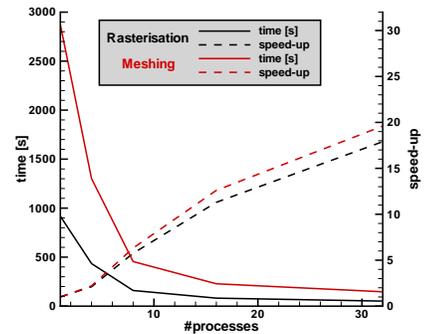


Figure 19: Performance figures for an advanced fighter-type aircraft.

system was described together with the three major components, the geometry definition, the mesh size specification and the surface mesher itself. Based on a rasterisation of the geometry an automatic way of determining a smooth distribution of the element sizes in 3-D space was highlighted. To achieve fast turnaround times, the computationally intensive parts can be executed in parallel. Especially for large complex configurations containing thousands of geometric entities the turnaround time can be reduced greatly by the presented approach as shown in the examples.

ACKNOWLEDGEMENTS

We would like to thank EADS Military Aircraft, Ottobrunn, and all colleagues in the numerical simulation department for the support.

Special acknowledgements have to be given to Luciano Fornasier, Stephan M. Hitzel, Kaare A. Sørensen and Herbert Rieger for the great support concerning different topics of this work.

References

- [1] Fornasier L., Deister F., Tremel U., Hassan O., Weatherill N.P. "Robust and Efficient Generation of Unstructured Surface Grids about Geometrically Complex Configurations Using Real-Design CAD Data." *41th AIAA Aerospace Sciences Meeting and Exhibit*. AIAA, Jan. 2003
- [2] Thompson J.F., Soni B.K., Weatherill N.P., editors. *Handbook of Grid Generation*. CRC Press LLC, 1999. Chapter 26
- [3] McMorris H., Kallinderis Y. "Octree-Advancing Front Method for Generation of Unstructured Surface and Volume Meshes." *AIAA Journal*, vol. Vol. 35, no. No. 6, Jun. 1997

- [4] Aftosmis M.J., Delanaye M., Haimes R. "Automatic Generation of CFD-Ready Surface Triangulations from CAD Geometry." *AIAA Paper 99-0776*, 1999
- [5] Computational Dynamics Research (CDR), Innovation Centre, University College, Singleton Park, Swansea SA2 8PP, U.K. *FLITE-3D User Manual*
- [6] Gamma E., Helm R., Johnson R., Vlissides J. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995
- [7] Stroustrup B. *The C++ Programming Language*. Addison-Wesley, 3rd edn., 1997
- [8] Balzert H. *Lehrbuch der Software-Technik*. Spektrum Akademischer Verlag, 1998. Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung
- [9] Balzert H. *Lehrbuch der Software-Technik*. Spektrum Akademischer Verlag, 1998. Software-Entwicklung
- [10] ISO (International Organisation for Standardization), Geneva. *STandard for the Exchange of Product model data (STEP), ISO 10303*
- [11] NIST (National Institute of Standards and Technology). *Initial Graphics Exchange Specification (IGES), Version 5.3*, 1990
- [12] Thompson J.F., Soni B.K., Weatherill N.P., editors. *Handbook of Grid Generation*. CRC Press LLC, 1999. Chapter 27
- [13] Farin G.E. *Curves and Surfaces in Computer Aided Geometric Design*. Academic Press, Inc., 4th edn., 1997
- [14] Piegl L., Tiller W. *The NURBS Book*. Springer, 2nd edn., 1997
- [15] Thompson J.F., Soni B.K., Weatherill N.P., editors. *Handbook of Grid Generation*. CRC Press LLC, 1999. Chapters 17, 19
- [16] Löhner R. *Applied CFD Techniques*. John Wiley & Sons Ltd, 2001
- [17] Deister F., Tremel U., Hirschel E.H., Rieger H. "Automatic Feature-Based Sampling of Native CAD Data for Surface Grid Generation." *Numerical Notes on Fluid Mechanics*, 2003. To appear
- [18] Piegl L., Richard A. "Tessellating trimmed NURBS surfaces." *Computer Aided Design*, vol. 27, no. No. 1, 16–26, 1995
- [19] Foley J.D., van Dam A., Fisher S.K., Hughes J.F. *Computer Graphics: Principles and Practice*. Addison-Wesley, 2nd edn., 1990
- [20] Deister F. *Selbstorganisierendes hybrid-kartesisches Netzverfahren zur Berechnung von Strömungen um komplexe Konfigurationen*. Ph.D. thesis, Universität Stuttgart, 2002
- [21] Deister F., Hirschel E. "Self-Organizing Hybrid Cartesian Grid/Solution System with Multigrid." *AIAA-2002-0112*. 2002
- [22] Aftosmis M.J. "Solution Adaptive Cartesian Grid Methods for Aerodynamic Flows with Complex Geometries." *Lecture Series CFD*, vol. 2. VKI, Mar. 1997
- [23] Snir M., et al. *MPI — The Complete Reference*. The MIT Press, 2nd edn., 1998. Vol. 1, The MPI Core
- [24] Gropp W., et al. *MPI — The Complete Reference*. The MIT Press, 1998. Vol. 2, The MPI Extensions