

Transient Dip Paint Simulation of Entire Car Bodies within One Day

Martin Schifko

Engineering Center Steyr GmbH & Co. KG

Hans Steiner

Caelynx

Daniela Fellhofer, Xinghua Song, and Kevin Verma

Engineering Center Steyr GmbH & Co. KG

Christoph Bauinger

Technical University Graz

ABSTRACT

The development of entire car bodies benefits from simulations, especially if they are performed at an early stage of development. Simulations lower the costs for required car body modifications. This paper focuses on dip painting simulation and describes the simulation process by detecting badly painted areas and liquid carry overs. A new hydrodynamic method has been benchmarked to CFD and real-life results. Results will be shown together with case examples. The solver is similar to, and as accurate as, standard CFD solvers; it is faster in its computation speed by a factor of at least 1000. The electrophoretic deposition (ELPO) of an entire car body can be simulated overnight with ALSIM. How is this possible? This paper points out the reason for all of this.

CITATION: Schifko, M., Steiner, H., Fellhofer, D., Song, X. et al., "Transient Dip Paint Simulation of Entire Car Bodies within One Day," *SAE Int. J. Mater. Manf.* 8(2):2015, doi:10.4271/2015-01-0731.

INTRODUCTION

The motivation for the software project ALSIM came in 2003 from AUDI's need for a simulation tool which could predict poorly and unpainted surfaces during ELPO. The first version of ALSIM did not use a CFD based solution - see [4], since AUDI's focus was not only accuracy, but also a very short simulation time. Hence, a pure geometric approach was applied, which worked well. Gas bubbles and their behaviour could be predicted and analysed within days. The next step was to embed the detection of liquid carry over when a car body dips out of its bath - see [7] - this also worked well. Gas bubble predictions fit nicely with reality and liquid carry over problems could also be simulated. However, the size of the underlying volumes was not exact, since at that time, ALSIM was not able to consider flow restriction and time.

PROBLEM DESCRIPTION AND OUTLINE

As production volumes increased, the total process time for draining off in ELPO was reduced. Hence, accurately predicting time related liquid carry-over became more important. Therefore, physical properties needed to be considered during the computation. As a

result, using a CFD based code as the basis for computation became inevitable. How could a CFD code be used and be very fast, i.e. compute 36 angle steps of the ELPO of an entire car body over night?

In this paper we will present methods and achievements within ALSIM, how a CFD code can be used for simulation by delivering accurate results within hours - see [1]. The section titled "Volumetric Representation" describes a new approach for an exact volumetric description of a 3D object - see [3]. In comparison to tetrahedral meshes, this new representation consists of fewer elements (by a factor of 2000), and furthermore, is just as accurate. CFD based code can be applied to this representation. For the consideration of flow restriction, ALSIM only applies it to those areas which are small enough (e.g. holes and gaps - see [5]) to need such consideration. The section "Bottleneck Consideration" will explain the method behind this. A tailgate simulation illustrates results and design modifications in the section titled "Example". The section "Verification" compares real life results with CFD results and ALSIM results - see [2]. A performance table - see [6] - is described in the section "Performance". Finally, current and future tasks are mentioned at the end of this paper.

VOLUMETRIC REPRESENTATION

In CFD and Finite Element Analysis, the computational domain is usually divided into a large number of small, fundamental, volumetric units, like tetrahedrons or hexahedrons. If the units are sufficiently small, computations can be simplified inside each unit without introducing substantial errors. In ALSIM software, the main idea for improving computation times is the use of fewer, larger volume units. In order to preserve accuracy, computations within these large volumetric units should no longer be approximated. Thus the simulation of fluid inside a volumetric unit needs to be relatively simple.

One approach ensuring this is a *convex decomposition* of the computational domain: Within a convex, volumetric unit a liquid's surface is exposed to uniform pressure conditions, such that the liquid will always be able to balance. Therefore a unique filling height can be computed for a volumetric unit given the amount of contained liquid, taking into account only information about the volumetric unit's geometry. The convex decomposition is computed by a sweep plane algorithm and consists mostly of a large number of volumetric units, which can be reduced by allowing small concavities in the units. However, these nearly complex decompositions still contain more units than necessary for the discussed purpose.

So a new kind of decomposition of the computational domain is introduced, namely a decomposition into so called *flow volumes*. They are defined such that they only meet along horizontal boundary patches. While sweeping from bottom to top of the object, *important points* are identified, building a so called *Reeb Graph* which is explained later. An important point can be either a local minimum, local maximum or a saddle point. In [Figure 1](#) they are marked by black dots. Every important point ends the former flow volume and starts a new one. Therefore, these flow volumes are in some way monotonic: In the planar case, a polygon is called monotonic with respect to a straight line L if the intersection of the polygon with any line orthogonal to L is connected. As a generalization, we could call a spatial area (like the interior of a polyhedron) *monotonic* with respect to a straight line L if every intersection of the area with a plane orthogonal to L is connected. In this sense, flow volumes are monotonic with respect to the z -axis. This form of monotonicity, however, is even weaker than the concept of weak monotonicity of polyhedral which requires the intersection to be bounded by a simple polygon. This is summarized in the following

Definition 1. A connected part F of the three-dimensional space is called *flow volume* for a given triangulated solid if

- The boundary of F consists of triangles of the triangulated solid and of parts of horizontal planes on top and bottom.
- F does not intersect the interior of the solid.
- Every horizontal slice of F is connected.
- F is maximal among all volumes satisfying the previous conditions.

Due to the third property in Definition 1, there can be only one filling level within a flow volume without violating physical laws. For example, the inside and outside of a bowl cannot belong to the same flow volume, because the filling level inside the bowl might be

different than the filling level on the outside. On the other hand, the inside of a simple bowl is contained in one flow volume, as any liquid in the bowl will balance such that a unique filling level is achieved. The last property makes sure that the computed volumetric units are not too small, as otherwise arbitrarily small volume slices would be allowed. [Figures 2 and 3](#) show the volume decomposition for a simple object. Note that flow volumes need not be topologically simple, as they may contain holes.

Reeb graphs are an important concept in Morse theory for gathering topological information. In Morse theory, the topology of a manifold is analyzed using critical points of a differentiable function f over the surface, the *Morse function*. A Reeb graph's vertices represent these critical points, its edges correspond to parts of the manifold of which all points with the same f value are connected. The flow volume decomposition motivated by the application of dip coating simulations corresponds to the Reeb graph of the computational domain, considered as 3-manifold with boundary and using the height function as Morse function. Each edge of this Reeb graph represents one flow volume. There exist several algorithms for computing Reeb graphs (or the closely related *contour trees*) for 3-manifolds. These algorithms use volumetric descriptions of the object, e.g. tetrahedral meshes, or voxel-based representations, and they are able to deal with general Morse functions. In the case of flow volumes, where the Morse function is the height function, using a boundary representation is sufficient and simplifies the computation. There exist also algorithms operating on surfaces. These compute the Reeb graph of the surface, considered as 2-manifold without boundary. As shown in [Figure 1](#), the Reeb graphs of the surface, the interior or exterior volume are in general different. Reeb graphs (or contour trees) of a surface with respect to the height function are typically computed by tracking intersection components while sweeping the surface with a horizontal intersection plane. Similarly the surface can be dissected into horizontal slices of a certain thickness, using refinements until no topologically important features are lost. However, these setups typically use many horizontal intersections. For computing flow volumes for an object represented by a boundary surface mesh, we adapt algorithms for surface Reeb graphs. Additional information about the relative spatial position of intersection polygons is required in order to determine intersection polygons' connections through the computational domain. To this end, the object is swept with a horizontal plane, computing global intersections, but only in critical points where they are needed for the flow volume computation. Between those, intersection components are traced by monotone paths.

The decomposition of free volume into flow volumes is unique and always possible. To visualize this, consider a horizontal plane sweeping through the object from bottom to top. Each connected area where the plane intersects free volume corresponds to one flow volume. Whenever such an area appears in the intersection plane, a new flow volume gets started. The disappearance of such an area from the intersection plane marks the end of a flow volume. When such areas are joined or separated, the corresponding flow volumes are closed and new ones are started, as otherwise the horizontal intersections would no longer be connected [2].



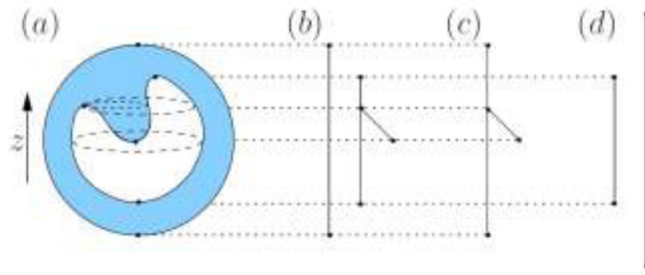


Figure 1. Reeb graphs with respect to the height function. (a) Sample object: sphere with a bowl-shaped hole. (b) Reeb graph of the surface. (c) Reeb graph of the solid's interior (shaded area). (d) Reeb graph of the solid's exterior. In this example the three Reeb graphs are different. The flow volumes correspond to the edges in (d).

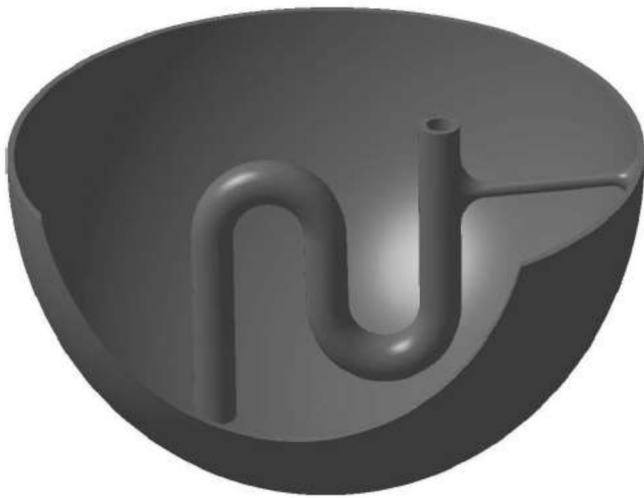


Figure 2. A given object for demonstration of the decomposition of free volume - see [Figure 3](#).

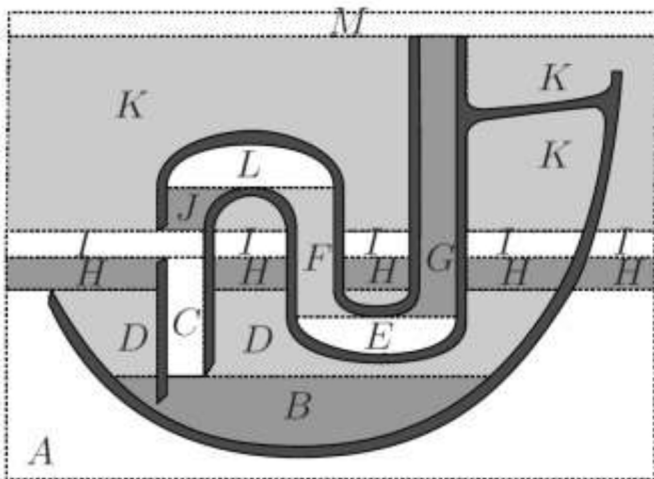


Figure 3. Decomposition of free volume of the object in [Figure 2](#). Vertical cut through the object with the flow volumes marked by letters. Flow volume H is an example for a flow volume with holes (one hole for each pipe).

BOTTLENECKS CONSIDERATION

Detection

In early releases of ALSIM, liquid flowed in the simulation without consideration of time. After dipping in, liquid immediately touched every reachable surface of the object. Equally, redundant liquid was gone as soon as the object was dipped out. That is not realistic as liquid needs time to spread, especially if it has to go through narrow channels. This caused inaccuracy and simulation errors, which should be avoided.

Thus, an algorithm was developed which detects all bottlenecks. The user can designate, which narrow channels qualify as a bottleneck by choosing the maximal value of a bottleneck's circumstance. The basic idea of this algorithm is to create a distance field using the famous Dijkstra Algorithm of finding the shortest path between two nodes in a graph. Starting from one vertex of the mesh, its neighbors are added to the distance field, along with their neighbors, until the given circumference is reached. So called "loops" are found whenever the neighborhood of a newly inserted vertex fulfills a certain criterion. A loop is built by a chain of vertices and can also go round the material or indicate a bottleneck which can be represented by a shorter chain. A bottleneck is defined as the shortest loop round a narrow channel. Examples for loops which are not bottlenecks can be seen in [Figure 4](#). To distinguish bottlenecks from loops, so called filters are applied. The first one uses the Formula of Gauss to drop loops which go around material, like the longish loop on the left side of [Figure 4](#). Another one shortens chains of vertices that are too long. Looking at the second loop in [Figure 4](#) one can recognize an outcome which could be avoided to represent the narrow channel by the shortest path. The start vertex of the distance field is always the first one in the loop chain. Simply said, this filter takes a vertex out of the middle of the chain, creates a new distance field and compares the resulting loops. After a second run, which is done in the case of not according loops, a stable loop is accepted as a bottleneck [5].

Of course it is not enough to do this procedure for one vertex, but it isn't necessary to do it for every vertex. By using a special selection method, some critical vertices which are likely to be located at a bottleneck are chosen. From the remaining vertices, a certain percentage is chosen randomly.



Figure 4. Examples of loops. The left one goes round material and has to be dropped. The second loop has an outcome which has to be shortened to get a bottleneck. Both loops were found by creation of a distance field which started from the light blue vertex on the left side.

Integration

As explained in the previous section, the flow volumes and their relations are represented by the Reeb graph. Bottlenecks are found and represented as lists of vertices. For the simulation, their position with respect to the flow volumes is relevant. So the bottlenecks get integrated into the Reeb graph, as explained below.

First of all there has to be a distinction whether a bottleneck is vertical (when liquid flows in horizontal direction through it) or horizontal (when liquid flows vertically through it), because they have to be treated differently. Examples of different kinds of bottlenecks can be seen in [Figure 5](#).

In the case of horizontal bottlenecks, there is a second classification whether the bottleneck can be projected to the upper or lower boundary of the corresponding volume, or if it is in the middle of it (imagine the narrow part of a sandglass). For integration into the Reeb graph, every horizontal bottleneck must be located “between” two volumes. Then, the edge between the appropriate volumes is marked as bottleneck connection. If it lies in the middle of a volume, this volume must be split into two new volumes, which also influences the structure of the Reeb graph. Then, once again, the edge between the two new volumes is marked as bottleneck connection.

Vertical bottlenecks can only appear in the middle of a volume, so a volume split is indispensable. Again, the edge between the volume parts is marked as bottleneck connection.

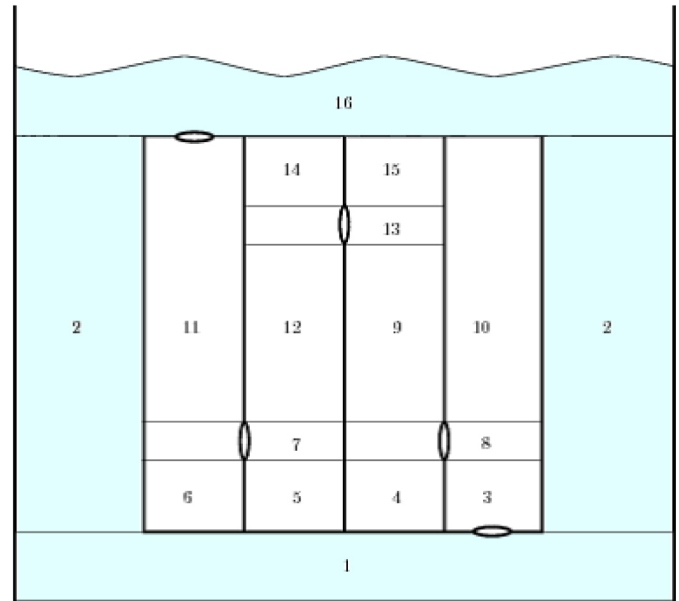


Figure 5. Example of an object (full object in 3D “validation cube” is shown in the next section) in a paint bath. The bottlenecks and the original flow volumes are plotted.

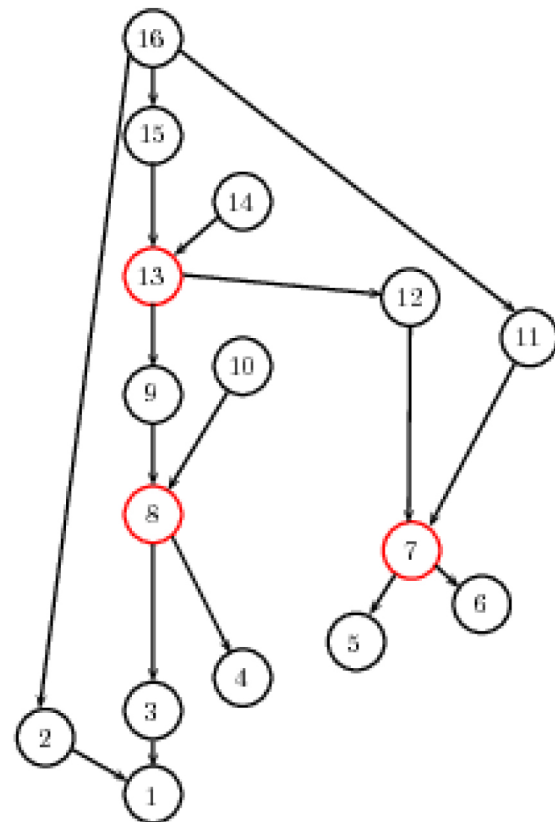


Figure 6. Corresponding Reeb graph of the object in [Figure 5](#). The red marked nodes represent the volumes which have to be split, because they contain a vertical bottleneck.