Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS (2017)
F. Jaillet and F. Zara (Editors)

Implicit Mesh Generation using Volumetric Subdivision

C. Altenhofen[1,2] F. Schuwirth[1,2] A. Stork[1,2] D. Fellner[1,2,3]

[1]TU Darmstadt, Germany
[2]Fraunhofer IGD, Germany
[3]Graz University of Technology, Institute of Computer Graphics and Knowledge Visualization, Austria

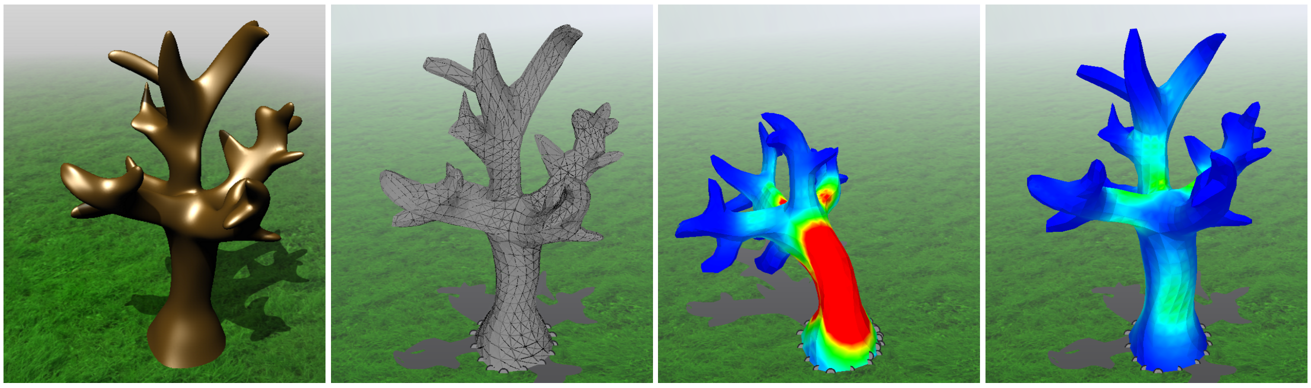


Figure 1: *A cartoon-style tree model created with our volumetric subdivision modeling application, converted into a tetrahedral mesh with our implicit meshing approach and simulated under gravity using a dynamic FEM solver. The fast transition between design and simulation allows for adapting the model to change its physical behavior. The simulation mesh is updated implictly during this process.*

Abstract

In this paper, we present a novel approach for a tighter integration of 3D modeling and physically-based simulation. Instead of modeling 3D objects as surface models, we use a volumetric subdivision representation. Volumetric modeling operations allow designing 3D objects in similar ways as with surface-based modeling tools. Encoding the volumetric information already in the design mesh drastically simplifies and speeds up the mesh generation process for simulation. The transition between design, simulation and back to design is consistent and computationally cheap. Since the subdivision and mesh generation can be expressed as a precomputable matrix-vector multiplication, iteration times can be greatly reduced compared to common modeling and simulation setups. Therefore, this approach is especially well suited for early-stage modeling or optimization use cases, where many geometric changes are made in a short time and their physical effect on the model has to be evaluated frequently. To test our approach, we created, simulated and adapted several 3D models. Additionally, we measured and evaluated the timings for generating and applying the matrices for different subdivision levels. For comparison, we also measured the tetrahedral meshing functionality offered by CGAL for similar numbers of elements. For changing topology, our implicit meshing approach proves to be up to 70 times faster than creating the tetrahedral mesh only based on the outer surface. Without changing the topology and by precomputing the matrices, we achieve a speed-up of up to 2800.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling | Curve, surface, solid, and object representations

1. Introduction

In computer graphics and animation, subdivision surfaces are widely used to create visually appealing 3D models. When aiming for a plausible physical behavior of those models for 3D animation or games, physically-based simulation comes into play. In many cases, it takes several loops of design and simulation to adjust a 3D geometry so that it shows the intended behavior in the

DOI: 10.2312/vriphys.20171079

simulation. Although subdivision surfaces have proven to be useful tools in animation, they show some of the same hurdles as other representation schemes to be overcome for simulation. Usually, the geometric mesh has to be transformed into a volumetric mesh to enable simulation (hereinafter referred to as *meshing* process). As the surface mesh just describes the outer boundary of the object, volumetric meshing tends to be a time-consuming process that might even require manual interaction, often to be re-done with every geometric change. Since there is no direct correlation between the design mesh and the simulation mesh, feedback and conclusions from the simulation results have to be derived manually to improve the design.

While the engineering community tries to solve this problem with *Iso Geometric Analysis (IGA)* [HCB05], we propose a method more suited for computer animation. By using subdivision volumes instead of subdivision surfaces for creating the initial 3D object, we encode the volumetric information into the model directly in the design phase. Volumetric modeling operations allow the manipulation of the geometry similarly to existing modeling tools, which use subdivision surfaces, while at the same time keeping the volumetric representation consistent underneath. For its diversity and its ability to handle control meshes of arbitrary topology, we chose the Catmull-Clark solid subdivision scheme [JM99] for our approach. Since the outer limit surface is identical to the limit surface of Catmull-Clark subdivision surfaces [CC78], there is no visual difference when designing 3D models with either Catmull-Clark surfaces or solids. For simulating, the volumetric subdivision scheme is applied to the control mesh multiple times until the desired mesh resolution is reached. Afterwards, the mesh can be converted into a purely tetrahedral mesh if necessary. To run the simulation, we use a custom GPU-based FEM solver based on [WBS*13]. However, as we are able to create a hexahedral or tetrahedral mesh, it does not require a special solver to be simulated and could also be fed into open-source or commercial solvers as well. Due to the initial volumetric representation, many vertices are shared between the design mesh and the simulation mesh and the simulation results can be visualized (also volumetrically) directly on the design mesh. This allows for clear hints on where to adapt/improve the model if necessary.

Since both the subdivision steps along with the conversion into a tetrahedral mesh can be expressed as one precomputable matrix-vector multiplication, our meshing process is much faster than those of commonly used meshing tools such as CGAL [AJR*16] or TetGen [Si15] and therefore allows much faster iterations of design and simulation. Additionally, due to the volumetric structure and the choice of the volumetric modeling operations, the mesh is guaranteed to be manifold (except for self-intersections).

Our main contributions are:

- A new approach for generating 3D models suited for design and simulation alike.
- A tighter integration of modeling and simulation with shorter iteration times and more insightful feedback.

These lead to the following benefits:

- Fast and consistent meshing due to an existing volumetric structure and precomputable mesh generation matrices.
- A partly relation between the geometric model and the simulation results.
- An ensured manifold mesh representation while modeling (except for self-intersections).

The paper is structured as follows: Section 2 summarizes existing work that is related or fundamental to our approach. Section 3 explains the approach in detail, showing its individual components and their connections. In Section 4, we describe our prototypical implementation. Section 5 shows the accomplished results, as well as benchmarks for the major steps and comparison with other algorithms. Section 6 wraps up the paper, reflects on our results and points out the advantages and possibilities for improvements of our approach. Finally, Section 7 gives an outlook on future improvements.

2. Related Work

This section overviews existing methods that are related to our approach and briefly discusses their benefits and drawbacks.

2.1. Modeling

In the field of modeling three-dimensional objects, the main categories are computer-aided design software (CAD) like Solid Works [Sys17], Rhino [MA17] or Fusion 360 [Aut17b] and polygonal modeling known from tools like Maya [Aut17a] or Blender3D [Fou17].

CAD software in general uses implicit volumetric representations such as BReps or parametric surface descriptions such as Bézier, B-spline or NURBS surfaces. The latter define smooth surfaces by a set of control points and are well suited for engineering applications. However, they are hard to use when aiming for organic shapes such as often required in computer graphics and animation.

In the area of design for computer graphics, animation and games, polygonal modeling tools are very common because of their easy-to-use modeling techniques. In contrast to CAD software, polygonal modeling tools like Blender3D offer subdivision surface algorithms to design organic 3D models. They provide a control mesh with a relatively low amount of degrees of freedom to model a smooth limit surface (see Section 2.2). A commercial example for heavily using subdivision techniques in different areas of 3D modeling and computer animation is Pixar [Stu17]. Nevertheless, all these tools typically offer surface modeling only.

In terms of volumetric modeling, there are only few approaches. Fairly new modeling techniques arise with the field of additive manufacturing, using modeling techniques on a voxel basis. An example is the software Monolith [Aut17c] which is a voxel-based modeling engine for multi-material 3D printing. It can describe inner structures and material properties. Another volumetric modeling technique is sculpting [MQW01], which allows the designer to initially model a rough shape and then define details like a sculptor by adding and removing parts locally from the shape. Analogously to subdivision surfaces, subdivision volumes exist, but they are mostly used in the context of simulation as described in Section 2.2.

2.2. Subdivision

Subdivision surfaces are widely used in computer graphics and computer animation. They provide smooth surfaces while at the same time only having a small number of degrees of freedom to define those. Due to the iterative or precomputable refinement process, memory consumption for a subdivision-based 3D object is much lower than for a finely tessellated model. It also allows for dynamic tessellation and level-of-detail approaches to improve the performance for rendering. For many years, different subdivision schemes have been developed. Some of them require purely triangular control meshes [Loo87, DLG90], while others work on quad-based meshes or are able to handle control meshes with arbitrary topology [Doo78, CC78]. Depending on the subdivision scheme and the topology of the control mesh, the limit surface has different continuity C (C^0, C^1, C^2).

As an extension to the existing subdivision schemes for surfaces, volumetric subdivision algorithms have been developed. They are mostly used in the engineering environment for global or local refinement of the simulation mesh [BHU10a]. Similar to subdivision surfaces, different volumetric subdivision schemes have different requirements to the (in this case volumetric) control mesh and result in different levels of continuity for their limit representation. Joy et. al [JM99] presented a volumetric extension to the Catmull-Clark surface subdivision scheme. As for the surface version, Catmull-Clark solids can handle control meshes with arbitrary topology as long as they are manifold. Chang et. al [CMQ03] as well as Schaefer et. al [SHW04] present alternative volumetric subdivision approaches based on tetrahedral meshes.

Subdivision surfaces as well as subdivision volumes struggle with so-called *irregular vertices*. These are vertices with a valence different to the valence intended for the subdivision scheme. E.g. Catmull-Clark surface subdivision [CC78] works best with quadrilateral patches and vertices of valence 4. The Butterfly scheme [DLG90] requires a triangular mesh with vertices of valence 6. Although most schemes define a limit surface/volume for these vertices, the drawback comes with reduced continuity (e.g. C^0 instead of C^1 or even C^2 for regular vertices) and/or a more difficult mathematical evaluation.

In the past years, several subdivision surface schemes have been improved and extended in order to overcome the problems with irregular vertices, e.g. [ZSS96, HS10, LFS16] and to even bridge the gap to spline-based representations [CADS09, RSAF16]. However, for subdivision volumes, those are still open points.

2.3. Meshing and Simulation

In terms of simulation techniques and especially in the domain of structural mechanics, the finite element method (FEM) [ZTT77] is widely used. As the FEM has evolved in the engineering field, adapted and specialized approaches have been developed for computer animation and video games for creating physically-based animations [MDM*02, MS06, NMK*06]. The basic idea is to divide the simulation domain into a set of discrete elements, to solve the physical problem for each element and then combine the local solutions to solve the global problem. In 3D, typical finite elements are tetrahedra or hexahedra. Every element has its degrees of freedom and its basis functions that are used to solve the partial differential equations. Over the years, several approaches have been presented to improve the method e.g. by adding a co-rotational term [HS04] or by introducing higher order basis functions [WKS*11, WMRA*15].

FEM approaches of any kind share the same requirement when simulating objects that are only represented by a surface model: *mesh generation*. *Mesh generation* or *meshing* describes the process of converting a (manifold) surface mesh into a volumetric mesh for simulation.

Different approaches exist on how to convert a surface representation into a fully volumetric, most often tetrahedral mesh [AJR*16, Si15, ACSYD05]. As most of them are designed for engineering applications, mesh quality is a very important requirement for the mesh generation process. [She02] as well as [Fie00] give good overviews over the different factors of mesh quality and how they can be satisfied.

Iso-geometric analysis IGA tries to avoid the problems of mesh generation in the engineering domain by using tri-variate spline representations for both design and simulation [HCB05]. However, these tri-variate spline meshes are not generated automatically and have also to be created from surface models in the first place. This implies also to use the spline basis functions for solving the partial differential equations. As an alternative to splines/NURBS, some approaches exist that work on subdivision surfaces [GCSO99, CSA*02, WHP11] and even subdivision volumes [BHU10b]. Due to the spline or subdivision basis functions, IGA-based methods need a specialized solver to run the simulation and have thereby only limited compatibility with common systems.

As an alternative to FEM, mass-spring systems are widely used in computer animation for cloth and hair animation but also for deformation [P*95, LBOK13]. To generate visually plausible animation high computation power is still needed.

Another physically-based animation approach can be found in the area of position based dynamics [MHHR07]. The basic idea is to use independent particles and simulate their behavior based on the properties of the neighborhood of each particle. A good overview can be found in [BMO*14].

Both methods can provide plausible results while having lower requirements to model quality than FEM. However to achieve visually appealing animations, proper constraints for the particles or springs have to be found manually for every use case.

The approach presented in this paper combines the well-known design properties of subdivision surfaces with a simplified and efficient mesh generation process for physically-based FEM simulations. Utilizing the volumetric Catmull-Clark subdivision scheme, it is possible to use the volumetric representation directly for both, modeling and simulation. Our volumetric modeling operations allow for versatile mesh generation and manipulation. Therefore, no additional step is needed to generate a simulation mesh. This creates a tighter integration between design and simulation.

3. Concept

In this section, we present our approach for integrated modeling and simulation using implicit mesh generation from volumetric subdivision.

3.1. Volumetric Modeling

Common tools for polygonal modeling used in design and computer animation only support surface modeling. Although closed surface meshes implicitly describe a volumetric object, they are not considered as volumetric meshes. In addition to vertices, edges and faces, a volumetric mesh consists of cells and has inner faces to separate these cells. A closed surface mesh could however be considered as a volumetric mesh consisting of just one polyhedral cell.

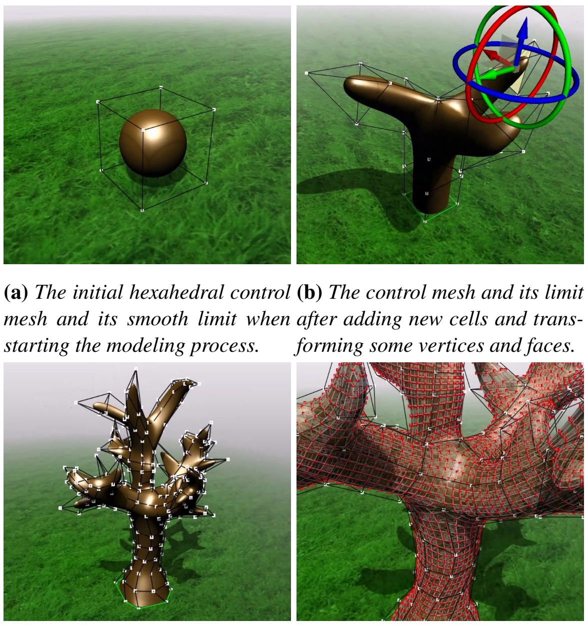
We present a modeling approach for fully volumetric meshes. Therefore, we developed a set of volumetric modeling operations similar to the ones found in the polynomial modeling functionality offered by common tools like Blender3D [Fou17] and Maya [Aut17a]. To form the geometry/the shape of the 3D object, one or multiple vertices, edges, faces or cells can be translated, rotated and scaled. To edit the topology of the object, new cells can be added to any outer face. To add more detail, additional vertices can be inserted into edges, faces or cells, splitting these into two or more edges/faces/cells. Alternatively, the entire mesh can be refined. During all these operations, the internal volumetric structure of the mesh is maintained. As the modeling operations are not restricted to the outer vertices/edges/faces of the model, the inner structures can be modified as well. Our system allows for example to shape the borders of inner cells or to e.g. encode material parameters on internal vertices for animation. Figure 2 shows an example modeling process from the initial geometry to a complete 3D model using our volumetric modeling approach.

3.1.1. Volumetric Catmull-Clark Subdivision

In order to keep the number of degrees of freedom low for modeling, we use a coarse control mesh to define the 3D object and apply a subdivision algorithm to create the actual model. This gives high control over the shape of the object while at the same time creates visually appealing surfaces. For our approach, we use Catmull-Clark subdivision since it is not restricted to a special kind of control mesh (e.g. triangular or quad-based). As we work with a volumetric mesh instead of a surface mesh, we apply the volumetric Catmull-Clark subdivision rules.

These can be summarized as follows:

1. For every cell, add a *cell point* at its centroid.
2. For every face, add a *face point* at the weighted average of its centroid and the centroids of all adjacent cells.
3. For every edge, add an *edge point* at the weighted average of its midpoint and the centroids of all adjacent cells and faces.
4. Move every original vertex to its new location at the weighted average of its original location and the centroids of all adjacent cells, faces and edges.
5. Connect each *cell point* with its corresponding *face points* and every *face point* with its corresponding *edge points* to create new edges, faces and cells.



(a) *The initial hexahedral control mesh and its smooth limit when starting the modeling process.* **(b)** *The control mesh and its limit after adding new cells and transforming some vertices and faces.*

(c) *The final tree model with a subdivided control mesh and several details.* **(d)** *The volumetric inner structures of the tree model after two steps of subdivision.*

Figure 2: *Creation of a 3D model with our volumetric modeling application. (a) to (c) show the progress from the initial hexahedral control mesh to the final design. (d) shows the volumetric inner structures for our implicit mesh generation approach that are maintained during the entire modeling process.*

A detailed description on the subdivision rules can be found in [JM99] and [BHU10b]. For elements on the outer surface, the subdivision rules for Catmull-Clark Surfaces apply. As for Catmull-Clark surfaces, sharp edges and features can be defined as well. Internally, we use a volumetric half-face data structure to efficiently calculate the adjacencies of cells, faces, edges and vertices needed for the subdivision process (see Section 4.1 for more details). An example of three subdivision steps applied on a hexahedral control mesh as well as its smooth limit can be seen in Figure 3.

3.1.2. Ensuring consistent Topology

Using our consistent modeling operations, the resulting volumetric subdivision meshes are manifold by design. There is no way to break them except for self-intersection. In general, these meshes have the following topological properties:

1. Every face on the boundary is connected to one cell.
2. Every face inside the mesh is connected to two neighboring cells.
3. Every edge connects two vertices and is shared by at least two faces.

However, self-intersection is a major problem. Allowing the user to arbitrarily manipulate the geometry of the object might lead to situations, where vertices are moved in a way that faces intersect

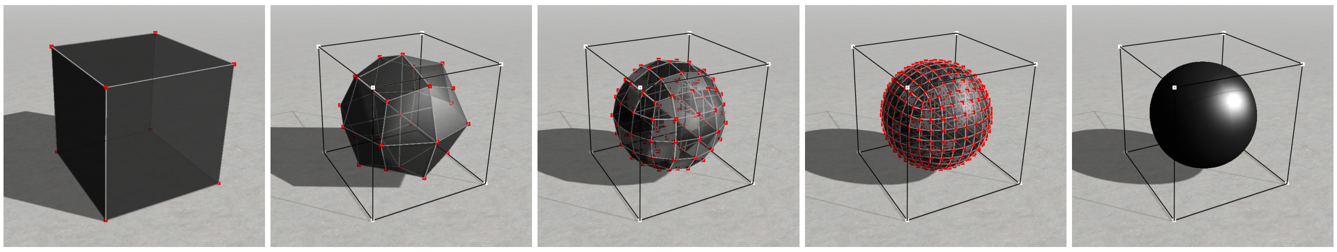

(a) *The original control mesh* **(b)** *One level of subdivision* **(c)** *Two levels of subdivision* **(d)** *Three levels of subdivision* **(e)** *The evaluated limit*

Figure 3: *Iterative subdivision with volumetric Catmull-Clark subdivision rules on a hexahedral control mesh. (a) shows the control mesh. (b) to (d) show the result of the subdivision process. (e) shows its smooth limit. Please note that in addition to the boundary faces, the volumetric cells and the inner faces are subdivided accordingly.*

other faces of the mesh. As this might happen with a surface mesh as well, it is much more likely to happen for volumetric meshes due to their interior structure. If inner vertices are not moved accordingly when manipulating the outer surface, they will at some point intersect with the mesh boundary while the user keeps modeling. For approximating subdivision schemes such as the Catmull-Clark surfaces and solids, self-intersections of the control mesh do not necessarily lead to self-intersections of the subdivided mesh, but they can be used as an approximation as it is a necessary but not sufficient criterion. For simplicity reasons, we avoid self-intersections of our volumetric representation by denying self-intersections for the control mesh. Section 7 shows an idea on how to handle this problem more accurately.

3.2. Meshing

To prepare the model for simulation, the volumetric control mesh gets subdivided a certain amount of times until the desired mesh resolution is reached. For an approximating but fast simulation, one or two subdivision steps might be enough, whereas for a very precise simulation, three or more steps could be needed. The subdivision is performed by using the volumetric Catmull-Clark subdivision rules as explained in Section 3.1.1. One step of subdivision from level $k-1$ to level k can be expressed as a matrix-vector multiplication of the control points p_{k-1} of level $k-1$ and the corresponding subdivision matrix S_k (see Equation 1).

$$p_k = S_k * p_{k-1} \tag{1}$$

The subdivision matrix S_k contains the factors needed to calculate p_k from p_{k-1}, according to the given subdivision rules. As one step of volumetric Catmull-Clark subdivision creates one additional vertex per edge, face and cell of the control mesh, the number of new vertices $|p_k|$ equals the number of vertices $|p_{k-1}|$ plus the number of edges $|e_{k-1}|$, faces $|f_{k-1}|$ and cells $|c_{k-1}|$ of the previous level. Therefore S_k is of size $|p_k| \times |p_{k-1}|$. Subsequent subdivision steps can be performed by either using the resulting control points of the previous level or by chaining the subdivision matrices together as shown in Equation 2.

$$\begin{aligned} p_{k+1} &= S_{k+1} * p_k \\ &= S_{k+1} * S_k * p_{k-1} \end{aligned} \tag{2}$$

This matrix representation of the subdivision rules allows computing a new combined subdivision matrix $\hat{S}_n$ that directly calculates the subdivided mesh of level n from the original control points p_0 (see Equation 3).

$$p_n = \hat{S}_n * p_0 \tag{3}$$

$\hat{S}_n$ is thereby the product of the subdivision matrices S_k of all subdivision levels k from 1 to n (see Equation 4).

$$\hat{S}_n = \prod_{k=1}^{n} S_k \tag{4}$$

If the simulation environment can deal with hexahedral meshes, the volumetric Catmull-Clark mesh can be used directly as a finite element mesh for the simulation. If a tetrahedral mesh is required, it can be easily generated from the volumetric subdivision mesh in the following way:

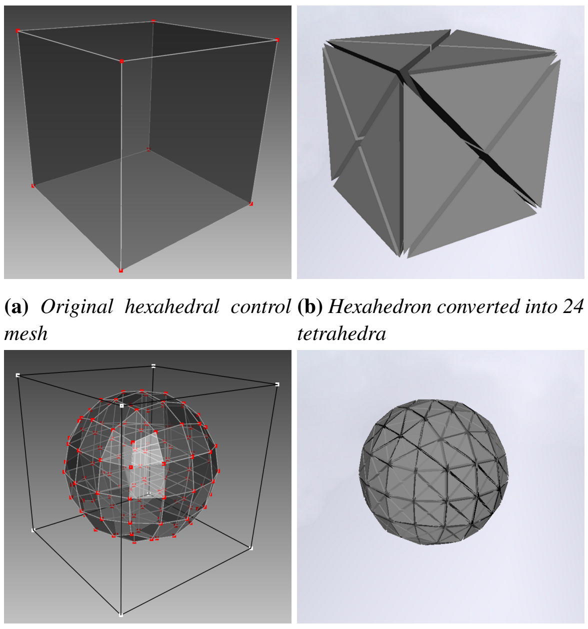
1. Insert a point at the center of each face
2. Insert a point at the center of each cell
3. Connect the original vertices of each face with the new center point of the face
4. Connect all vertices (including the faces center points) with the new point at the center of the cell

The result can be seen in Figure 4. This will split every hexahedral cell into 24 tetrahedral cells, sharing eight vertices with the original model. Other methods to split a hexahedron into tetrahedra exist [MT03,Lab17] that create a different amount of cells per hexahedron, but the one chosen here provides a very symmetric and unbiased mesh, although it creates a large amount of elements.

The conversion process to create the vertices of the tetrahedral mesh t_n can also be represented as a matrix-vector multiplication using the subdivided control vertices p_n of level n and the so called *tet generation matrix* T_n as shown in Equation 5. The vertices t_n can also be expressed in dependence of the original control points p_0 and the subdivision matrix $\hat{S}_n$.

$$\begin{aligned} t_n &= T_n * p_n \\ &= T_n * \hat{S}_n * p_0 \end{aligned} \tag{5}$$

Given by the tetrahedralization approach, the number of vertices of the tetrahedral mesh $|t_n|$ equals to the number of vertices $|p_n|$



(a) *Original hexahedral control mesh* **(b)** *Hexahedron converted into 24 tetrahedra*

(c) *Hexahedral control mesh with two levels of subdivision* **(d)** *Tetrahedral mesh generated from the subdivided hexahedron*

Figure 4: *Exemplary meshing process. One hexahedron (a) is being meshed into 24 tetrahedra (b). The same hexahedron has been subdivided twice (c) and then converted into a tetrahedral mesh (d).*

plus the number of faces $|f_n|$ and cells $|c_n|$ of the subdivided mesh. Therefore, T_n is of size $|t_n| \times |p_n|$.

Creating the simulation mesh in this fashion provides a very fast meshing process that allows for high feedback rates. When only changing the geometry of the original model (i.e. changing positions of the control points p_0), the matrices $\hat{S}_n$ and T_n remain unchanged and can be re-used. This allows to precompute these matrices and even their product $T_n * \hat{S}_n$ for a given subdivision level n, resulting in even faster mesh generation. However, changing the topology of the model by adding or removing vertices, edges, faces or cells or changing the subdivision level leads to a re-computation of $\hat{S}_n$ and T_n. Section 5.2 provides some timings on creating the matrices $\hat{S}_n$ and T_n for multiple subdivision levels as well as applying them to the control mesh to create the simulation mesh.

3.3. Simulation and Feedback

As described in Section 1, we run the simulation with a custom GPU-based FEM solver to further increase the integration between modeling and simulation. However, any other FEM solver could be used that accepts either a hexahedral or a tetrahedral mesh for its simulations. With our implicit meshing approach, the relation between the geometric model and the simulation model can be kept to some extent, making it easier to provide feedback of the simulation results for modeling. When converting every subdivided cell into 24 tetrahedra as shown in Figure 4, the vertices of the cell are shared with the tetrahedra. This keeps partial correspondence between the original subdivision mesh and the tetrahedral mesh and allows for a volumetric visualization of the simulation result directly on the subdivision model.

4. Implementation

In this section, we describe our prototypical system, which consists of the following components:

1. A 3D modeling environment based on Catmull-Clark subdivision solids
2. A GPU-based finite element solver for Computational Structural Mechanics (CSM)

4.1. Modeling Application

For designing 3D objects, we use our prototypical modeling application implemented in C++. It allows object generation and manipulation in the concept of polygonal modeling, similar to Blender3D or Maya while also maintaining the volumetric representation during modeling. We implemented a set of volumetric modeling operations to shape the subdivision control mesh. These operations include the transformation (translation, rotation and scale) of single or multiple cells, faces, edges and vertices as well as the ability to add new cells to the mesh. To create loops or holes, it is also possible to connect two boundary faces with one another by either inserting a new cell between them or by merging them directly. Analogously to Catmull-Clark subdivision surfaces, sharp features can be created by defining crease edges on the volumetric control mesh. The control mesh can be refined or subdivided (with and without smoothing) iteratively to increase the mesh resolution and to add details on lower levels. For the internal storage of the Catmull-Clark subdivision solids, we use a pointer-based half-face data structure. It can be seen as an extension to a half-edge data structure, by representing faces as a pair of two half-faces and adding additional entities to represent cells. As usually for half-edges, every half-face holds a reference to its opposing half-face and to the next half-face inside the same cell. Iterating over these references in different ways, allows for fast computation of all relevant neighboring relations needed for our modeling operations as well as for the subdivision process. When modifying the mesh topology by inserting or deleting elements, the references only have to updated locally; unaffected parts of the mesh remain unchanged. However, storing all this information results in a very high memory footprint, especially for complex models.

For interaction with the CSM solver, the subdivision matrix and the *tet generation matrix* are applied to the control mesh as described in Section 3.2. The simulation results can be visualized directly on the subdivision mesh using the partial correlation between itself and the tetrahedral mesh. The stress is shown color-coded on the respective vertices. Although this is just an approximation, it helps to identify the weak points of the 3D object and to improve the design as described in Section 3.3.

4.2. CSM Solver

As a fluent workflow is essential for a good integration, we use a custom GPU-based finite element solver based on [WBS*13] to simulate the physical behavior of our 3D object. The solver is written in C++ using NVIDIA CUDA for performing GPU computations. It requires a purely tetrahedral mesh as input, but is therefore able to run the simulation with either standard or corotational finite elements with linear or higher order basis functions [WMRA*15]. The simulation itself can be run as either a static or a dynamic simulation. While static simulations try to find an equilibrium between internal elastic forces and external forces (boundary conditions), dynamic simulations are time dependent and can deal with dynamically changing boundary conditions as well. To control the simulation parameters, material parameters can be defined in terms of Young's modulus and Poisson's ratio per object or on tetrahedral level. Boundary conditions such as fixation and external forces can be defined per vertex. As its output, the application provides the deformation and the computed stress for each vertex of the simulation mesh. Instead of feeding the results back to the modeling application and visualizing them on the subdivision mesh, they can also be visualized directly on the tetrahedral mesh inside the solver application. Displacement is shown geometrically on the vertices whereas the stress is visualized using an interactive color map. The dynamic simulation additionally shows the full animation of the object.

5. Results

In this section, we present the results produced with our method. We show performance measurements for the implicit tetrahedral mesh generation of several volumetric models compared to the generation of a tetrahedral mesh with CGAL, using only the surface model.

5.1. Hardware Setup

The measurements presented here were taken on a Windows 10 desktop PC with an Intel Core i7-3770 CPU with a clock rate of 3.4 GHz and 16 Gigabytes of DDR3 1600 system memory.

5.2. The meshing process

We used single-threaded CPU implementations written in C++ to evaluate both our meshing approach as well as the tetrahedral meshing provided by CGAL. Our test set consists of grid-based procedurally generated models in sizes from one to $40 \times 40 \times 1$ subdivision cells (see Figure 5).

To evaluate our approach, we subdivide every model two times and measure the computation time to generate the subdivision matrices S_1 and S_2, as well as the combined matrix $\hat{S}_2$ and the *tet generation matrix* T_2. Figure 6 shows the results of these measurements. Generating the matrices S_n and T_n is a process with linear complexity in the size of the control mesh (and therefore indirectly also in the number of vertices of the resulting tetrahedral mesh). This can be directly derived from the subdivision rules as they are also defined linearly for all cells, faces, edges and vertices of the control mesh. However, the time for calculating the combined subdivision matrix $\hat{S}_n$ depends on the subdivision level n since all matrices S_k for $k \leq n$ have to be multiplied (see Equation 4). Additionally, we measure the time to apply the matrices $\hat{S}_2$ and T_2 to

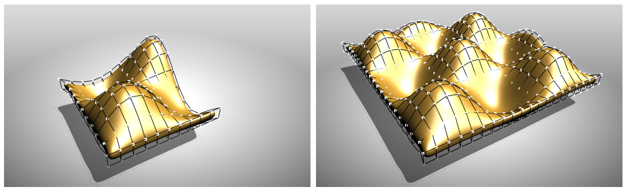

(a) *Procedurally generated subdivision volume with* $10 \times 10 \times 1$ *cells.* **(b)** *Procedurally generated subdivision volume with* $20 \times 20 \times 1$ *cells.*

Figure 5: *Procedurally generated volumetric subdivision models of various sizes for performance testing. The surfaces are created with different numbers of cells by evaluating the function* $z = sin(x) * sin(y)$.

the control mesh of each model. As can be seen in Figure 7 multiplication times also grow linearly in the number of vertices. This is also expected, since the computations are linear matrix-vector multiplications $\hat{S}_n * p_0$ and $T_n * p_n$ (see Equation 5). Nevertheless, the connectivity of the control mesh defines the sparsity of $\hat{S}_n$ and T_n and therefore has an effect on the complexity of the multiplications. For the evaluation of CGAL, we use a target cell size of 0.0625 in order create approximately as many tetrahedral elements as with our approach for subdivision level $n = 2$. As shown in Figure 8, tetrahedral meshing in CGAL is also a process with linear complexity for fixed parameters.

To further analyze the impact of the subdivision level on the computation times of our implicit meshing approach, we started with a simple cube and subdivided it iteratively up to six times. Table 1 shows the computational time for generating the matrices S_n, $\hat{S}_n$ and T_n. Keep in mind that the matrices have only to be regenerated if the topology of the design mesh has changed. Table 2 shows the computational time for applying the matrices $\hat{S}_n$ and T_n in order to subdivide and convert the model into a tetrahedral mesh of a specific resolution. Again, matrix generation as well as matrix multiplication show a linear complexity in the number of resulting vertices. The growth in relation to the subdivision level n is exponential, however. This is expected, since the number of resulting vertices is also exponentially related to the subdivision level. For comparison, Table 2 also shows the time needed to create a comparable tetrahedral mesh using CGAL. We chose the cell size c in a way to approximately generate a similar number of vertices than for the tetrahedral mesh created with our method.

5.2.1. Mesh Quality

Mesh quality is an important factor when it comes to simulation. As a general rule, mesh quality can be derived from the uniformity of the mesh elements [She02]. This means that all mesh elements (all tetrahedra/hexahdra) should have approximately the same size and a preferably uniform aspect ratio. The mesh generation of CGAL attempts to create optimized tetrahedral meshes in order to meet these quality requirements as good as possible. In contrast, our approach relies directly on the volumetric subdivision control mesh that is used to design the 3D object. Larger patches on the control mesh will result in larger simulation elements and smaller patches will result in smaller ones. The same applies to the aspect ratio

Subdivision Level n	**Control Vertices**	**Res. Vertices**	**Time [ms]** Generation of S_n	Generation of $\hat{S}_n$	Generation of T_n
0	8	15	-	-	0
1	27	71	1	1	1
2	125	429	1	2	1
3	729	2969	3	4	2
4	4913	22065	27	31	20
5	35937	170081	347	397	184
6	274625	1335489	3089	3610	1665

Table 1: *Timings for generating the partial subdivision matrix S_n, the combined subdivision matrix $\hat{S}_n$ and the tet generation matrix T_n for an increasing subdivision level n. As $n = 0$ corresponds to the original control mesh, subdivision is not necessary and only the tet generation matrix T_n has to be computed.*

	Our Method				**CGAL Meshing**		
		Time [ms]					**Time [ms]**
n	**Res. Vertices**	**Subdivision**	**Tet-Conversion**	**Complete Meshing**	**Comparable** c	**Res. Vertices**	**CGAL Meshing**
0	15	-	0	0	0.6	14	6
1	71	1	0	1	0.25	75	16
2	429	1	1	2	0.135	431	81
3	2969	2	1	3	0.0665	2935	452
4	22065	2	4	6	0.0328	22092	3742
5	170081	8	7	15	0.0163	170363	34144
6	1335489	26	22	48	0.0081	1348846	303994

Table 2: *Timings for subdividing the original control mesh ($\hat{S}_n * p_0$) and converting it into a tetrahedral mesh ($\hat{T}_n * p_n$) compared to the tetrahedral meshing provided by the CGAL library. We choose dedicated cell sizes c for CGAL in order to approximately create the same number of vertices as we do with our approach for subdivision level n. As for Table 1, the subdivision step is not needed for $n = 0$.*

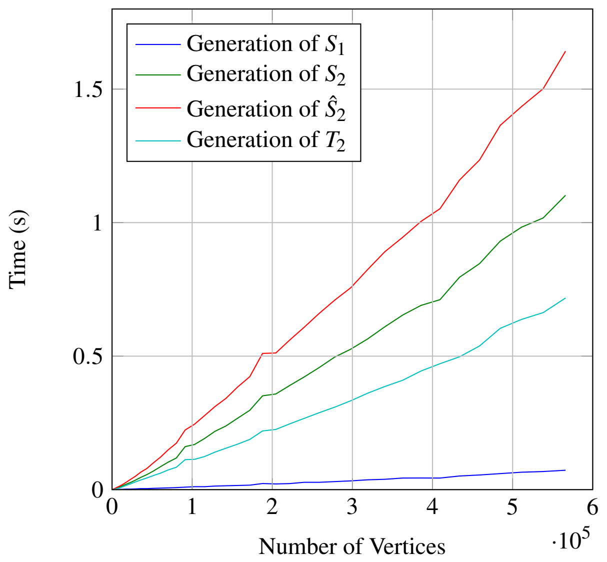


Figure 6: *Matrix generation of the first level subdivision matrix S_1, the second level matrix S_2, the combined subdivision matrix $\hat{S}_2$ and the tet generation matrix T_2 for procedurally generated models of different sizes. The x axis shows the number of vertices of the resulting tetrahedral mesh.*

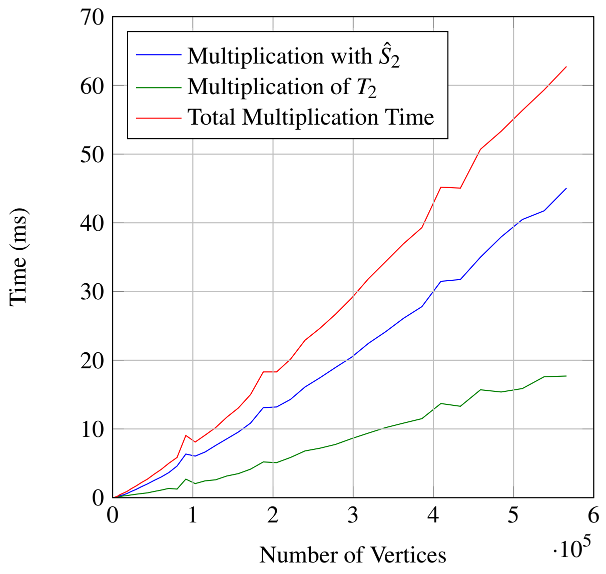


Figure 7: *Matrix multiplication of the original control points p_0 with the combined subdivision matrix $\hat{S}_2$, the tet generation matrix T_2 and the complete mesh generation matrix $T_2 * \hat{S}_2$. The x axis shows the number of vertices of the resulting tetrahedral mesh.*

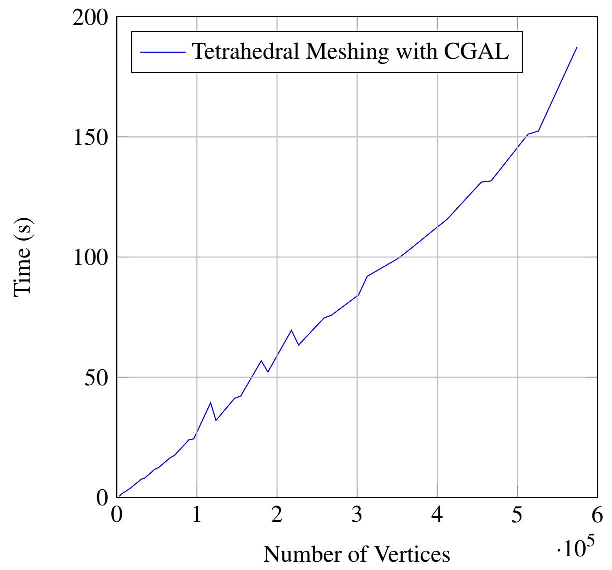


Figure 8: *Mesh generation times of CGAL. The tetrahedral meshes are generated from the outer surfaces of the same models used in Figure 6 and Figure 7. The x axis shows the number of vertices of the resulting tetrahedral mesh.*

of the elements. However, given by the subdivision scheme, these deficits get smaller with each level of subdivision. Additionally for computer animation, visually plausible results are often sufficient and fully detailed, physical simulations are not required. Unfortunately, having ill-shaped elements in the mesh might also lead to unstable simulations. Too large deviations in angles or aspect ratios of the elements might endanger the overall convergence of the simulation.

In order to analyze the mesh quality of our tetrahedral meshes compared to the ones created with CGAL, we calculated a set of quality indicators such as the volume, circumradius and connectivity of the mesh elements. The results are shown in Table 3. In general, CGAL creates tetrahedral meshes with much more uniform elements. This can be seen especially when comparing the min-max ratio for the circumradius. As this effect is not that big on the sine surface meshes due to their uniform procedural generation, it becomes much more visible when analyzing manually created free-form models such as the coyote or the tree model. However, our approach creates meshes with much more homogeneous connectivity. Even if the minimum and maximum number of adjacent vertices is similar for both approaches, the median is much closer to the minimum value compared to CGAL. This also leads to a sparsification of the matrices and a speed-up in simulation time.

To improve the mesh quality and obtain a more homogeneous simulation mesh, local refinement or smoothing could be used as described in Section 7.

5.3. Result visualization

As described in Section 3.3, the design mesh and the simulation mesh partially share their vertices. On the one hand, this leads to a geometrical and visual correspondence between the subdivision mesh and the tetrahedral mesh, while on the other hand it allows for visualization of the simulation results (e.g. the absolute stress) not only on the tetrahedral mesh, but on the subdivided design mesh as well. Figures 1 and 9 show two examples of how our complete system can be used, including all stages of modeling, implicit tetrahedral meshing, simulation and result visualization.

6. Conclusion

In this paper, we present a novel approach to strengthen the integration of design and simulation. By using a volumetric representation based on Catmull-Clark subdivision solids, we decrease the complexity of the meshing process and thereby the iteration times for simulation-based design. For designing the 3D model, volumetric modeling operations have been developed to keep the volumetric representation consistent and to preserve the internal structures. The simulation mesh is obtained by applying the volumetric subdivision scheme onto the model several times and if necessary converting the result into a purely tetrahedral mesh. This approach creates a partial correspondence between the design mesh and the simulation mesh that allows feedback from simulation results to be incorporated directly into the design. Additionally, the subdivision itself as well as the generation of the tetrahedral mesh can be expressed as matrix-vector multiplications. If the topology of the design mesh is unchanged and only the geometry is modified, the matrices can be precomputed to further speed up the mesh generation process. For changing topology, we achieve a speed-up of up to 70 compared to creating the tetrahedral mesh from the outer surface using CGAL. When only modifying the geometry of the design mesh, we even achieve a speed-up of up to 2800, reducing the re-meshing time to e.g. 55 milliseconds for a tetrahedral mesh of 500.000 vertices. However, the quality of our simulation meshes depends directly on the configuration of the control mesh and therefore might be worse than the meshes generated with CGAL. Furthermore, self-intersections might occur if the inner vertices are not adapted according to changes on the outer surface.

Our approach offers consistent volumetric modeling, fast meshing and tight correlation between the design and the simulation mesh, while at the same time keeping compatibility with standard FEM solvers for simulation.

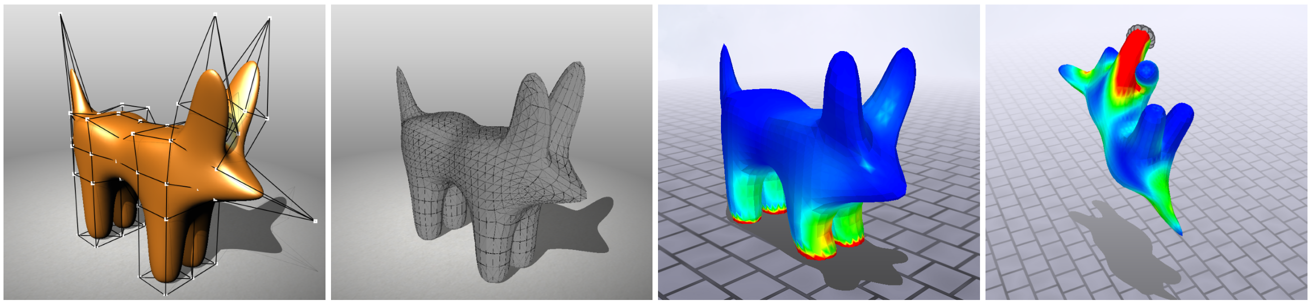
7. Future Work

The presented volumetric subdivision approach only creates manifold and watertight meshes by design, as long as no self-intersections occur. This holds especially for the inner structures of the volumetric mesh. To avoid modifying the inner structures manually, an adapted version of Laplacian smoothing or a mass-spring approach could provide an automatic adjustment to the inner control points while the designer works on the outer surface as he would in common modeling tools.

With our presented approach, mesh quality depends on the uniformity of the control mesh. Local refinement of badly shaped elements could help to increase the overall mesh quality and to better resolve high-frequency physical effects in specific areas of the mesh. For quad-based subdivision schemes like Catmull-Clark,

Model	Method	Tet Volume			Circumradius			Adjacent Vertices		
		min	max	ratio	min	max	ratio	min	max	median
Sine Surface	Ours	0.0001096	0.0005208	4.751824818	0.0654446	0.585578	8.947690107	5	26	6
$5 \times 5 \times 1$	CGAL	0.0000720	0.0009538	13.24722222	0.0674559	0.130755	1.938377518	3	20	11
Sine Surface	Ours	0.0001563	0.0007213	4.61484325	0.0718737	0.560796	7.802520254	5	26	6
$20 \times 20 \times 1$	CGAL	0.0004904	0.0010268	2.093800979	0.0605184	0.152669	2.522687315	3	23	11
Tree	Ours	0.0000493	0.0206962	419.801217	0.0597049	3.924601	65.73331502	4	31	6
	CGAL	0.0000615	0.0071105	115.6178862	0.0517199	0.270421	5.228567727	3	24	9
Coyote	Ours	0.0004643	0.0112495	24.2289468	0.1169941	2.656621	22.70730746	5	34	6
	CGAL	0.0000923	0.0080642	87.36944745	0.0576763	0.277445	4.810381387	3	25	11

Table 3: *Different indicators for mesh quality calculated for tetrahedral meshes generated with our approach compared to CGAL. We calculated the volume of each tetrahedron as well as the radius of its circumsphere. The table shows the minimum, maximum and the ratio between minimum and maximum for these values. Additionally, we analyzed the connectivity of every vertex to its neighboring vertices, showing the minimum, maximum and median value for each mesh. Subdivision levels and CGAL cell sizes for the selected models were chosen in a way that the number of vertices of the tetrahedral meshes match as good as possible.*



(a) *The coyote model created with our volumetric subdivision modeling application.*

(b) *The tetrahedral mesh of the coyote model generated with our implicit meshing approach.*

(c) *Dynamic CSM simulation of the coyote model under gravity. The absolute stress is color coded.*

(d) *Applying external forces to the model by creating a dynamic boundary condition on one of its legs.*

Figure 9: *Another example showing our approach. The coyote model is being designed with Catmull-Clark subdivision solids (a), subdivided twice and converted into a tetrahedral mesh with our implicit meshing approach (b) and simulated under gravity with the dynamic FEM solver (c). The dynamic simulation allows for defining dynamic boundary conditions during the simulation (d).*

local refinement is not possible without introducing T-junctions. Therefore, tetrahedral subdivision schemes such as [BHU10a] would be better suited. Extending our approach with such a scheme would allow local refinements and improve the quality of the simulation results without increasing the number of degrees of freedom as much as with global subdivision. As an alternative, a similar approach as for the inner control points could be envisaged to increase mesh uniformity. However, to not change the design, movement of the outer control points should to be constrained in a way that the outer limit surface of the mesh stays the same.

To further integrate modeling and simulation, modeling of physical properties is envisaged. Defining additional attributes such as Young's modulus or Poisson's ratio at the control points would enable different material distributions across the mesh and create smooth gradients in material transitions when subdividing. Crease edges and sharp features could be used to also model discrete material boundaries. This way, the stiffness and physical behavior of different parts of the model could be easily defined and modified.

Acknowledgements

This work was supported by the EU-project CAxMan (project number 680448) which is co-funded by the *Horizon 2020* Framework Program H2020-FoF-2015 of the European Union. More information can be found at https://www.caxman.eu/.

References

[ACSYD05] ALLIEZ P., COHEN-STEINER D., YVINEC M., DESBRUN M.: Variational tetrahedral meshing. In *ACM Transactions on Graphics (TOG)* (2005), vol. 24, ACM, pp. 617–625. 3

[AJR*16] ALLIEZ P., JAMIN C., RINEAU L., TAYEB S., TOURNOIS J., YVINEC M.: 3D mesh generation. In *CGAL User and Reference Manual*, 4.9 ed. CGAL Editorial Board, 2016. URL: `http://doc.cgal.org/4.9/Manual/packages.html#PkgMesh_3Summary`. 2, 3

[Aut17a] AUTODESK: Autodesk maya, jan 2017. URL: `http://www.autodesk.de/products/maya/overview`. 2, 4

[Aut17b] AUTODESK: Fusion360, jan 2017. URL: `https://www.rhino3d.com/`. 2

[Aut17c] AUTODESK: monolith, jan 2017. URL: `http://www.monolith.zone/`. 2

[BHU10a] Burkhart D., Hamann B., Umlauf G.: Adaptive and feature-preserving subdivision for high-quality tetrahedral meshes. In *Computer Graphics Forum* (2010), vol. 29, Wiley Online Library, pp. 117–127. 3, 10

[BHU10b] Burkhart D., Hamann B., Umlauf G.: Iso-geometric finite element analysis based on catmull-clark: Subdivision solids. In *Computer Graphics Forum* (2010), vol. 29, Wiley Online Library, pp. 1575–1584. 3, 4

[BMO*14] Bender J., Müller M., Otaduy M. A., Teschner M., Macklin M.: A survey on position-based simulation methods in computer graphics. In *Computer graphics forum* (2014), vol. 33, Wiley Online Library, pp. 228–251. 3

[CADS09] Cashman T. J., Augsdörfer U. H., Dodgson N. A., Sabin M. A.: Nurbs with extraordinary points: high-degree, non-uniform, rational subdivision schemes. In *ACM Transactions on Graphics (TOG)* (2009), vol. 28, ACM, p. 46. 3

[CC78] Catmull E., Clark J.: Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-aided design 10*, 6 (1978), 350–355. 2, 3

[CMQ03] Chang Y.-S., McDonnell K. T., Qin H.: An interpolatory subdivision for volumetric models over simplicial complexes. In *Shape Modeling International, 2003* (2003), IEEE, pp. 143–152. 3

[CSA*02] Cirak F., Scott M. J., Antonsson E. K., Ortiz M., Schröder P.: Integrated modeling, finite-element analysis, and engineering design for thin-shell structures using subdivision. *Computer-Aided Design 34*, 2 (2002), 137–148. 3

[DLG90] Dyn N., Levine D., Gregory J. A.: A butterfly subdivision scheme for surface interpolation with tension control. *ACM transactions on Graphics (TOG) 9*, 2 (1990), 160–169. 3

[Doo78] Doo D.: A subdivision algorithm for smoothing down irregularly shaped polyhedrons. In *Proceedings on Interactive Techniques in Computer Aided Design* (1978), vol. 157, Bologna, p. 165. 3

[Fie00] Field D. A.: Qualitative measures for initial meshes. *International Journal for Numerical Methods in Engineering 47*, 4 (2000), 887–906. 3

[Fou17] Foundation B.: Blender, jan 2017. URL: https://www.blender.org/. 2, 4

[GCSO99] Grinspun E., Cirak F., Schröder P., Ortiz M.: *Non-Linear Mechanics and Collisions for Subdivision Surfaces*. Tech. rep., 1999. 3

[HCB05] Hughes T. J., Cottrell J. A., Bazilevs Y.: Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement. *Computer methods in applied mechanics and engineering 194*, 39 (2005), 4135–4195. 2, 3

[HS04] Hauth M., Strasser W.: Corotational simulation of deformable solids. 3

[HS10] Huang J., Schröder P.: \ sqrt {3}-based 1-form subdivision. In *International Conference on Curves and Surfaces* (2010), Springer, pp. 351–368. 3

[JM99] Joy K. I., MacCracken R.: *The refinement rules for Catmull-Clark solids*. Tech. rep., Citeseer, 1999. 2, 3, 4

[Lab17] Laboratories S. N.: Htet - cubit 15.2 user documentation, jan 2017. URL: https://cubit.sandia.gov/public/15.2/help_manual/WebHelp/cubithelp.htm#mesh_generation/meshing_schemes/conversion/htet.htm. 5

[LBOK13] Liu T., Bargteil A. W., O'Brien J. F., Kavan L.: Fast simulation of mass-spring systems. *ACM Transactions on Graphics (TOG) 32*, 6 (2013), 214. 3

[LFS16] Li X., Finnigan G. T., Sederberg T. W.: G1 non-uniform catmull-clark surfaces. *ACM Trans. Graph. 35*, 4 (July 2016), 135:1–135:8. 3

[Loo87] Loop C.: Smooth subdivision surfaces based on triangles. 3

[MA17] McNeel R., Associates: Rhino, jan 2017. URL: https://www.rhino3d.com/. 2

[MDM*02] Müller M., Dorsey J., McMillan L., Jagnow R., Cutler B.: Stable real-time deformations. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2002), ACM, pp. 49–54. 3

[MHHR07] Müller M., Heidelberger B., Hennix M., Ratcliff J.: Position based dynamics. *Journal of Visual Communication and Image Representation 18*, 2 (2007), 109–118. 3

[MQW01] McDonnell K. T., Qin H., Wlodarczyk R. A.: Virtual clay: A real-time sculpting system with haptic toolkits. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics* (New York, NY, USA, 2001), I3D '01, ACM, pp. 179–190. 2

[MS06] Mezger J., Strasser W.: Interactive soft object simulation with quadratic finite elements. In *International Conference on Articulated Motion and Deformable Objects* (2006), Springer, pp. 434–443. 3

[MT03] Mueller M., Teschner M.: Volumetric meshes for real-time medical simulations. In *Bildverarbeitung für die Medizin 2003*. Springer, 2003, pp. 279–283. 5

[NMK*06] Nealen A., Müller M., Keiser R., Boxerman E., Carlson M.: Physically based deformable models in computer graphics. In *Computer graphics forum* (2006), vol. 25, Wiley Online Library, pp. 809–836. 3

[P*95] Provot X., et al.: Deformation constraints in a mass-spring model to describe rigid cloth behaviour. In *Graphics interface* (1995), Canadian Information Processing Society, pp. 147–147. 3

[RSAF16] Riffnaller-Schiefer A., Augsdörfer U., Fellner D.: Isogeometric shell analysis with {NURBS} compatible subdivision surfaces. *Applied Mathematics and Computation 272, Part 1* (2016), 139 – 147. Subdivision, Geometric and Algebraic Methods, Isogeometric Analysis and Refinability. 3

[She02] Shewchuk J.: What is a good linear finite element? interpolation, conditioning, anisotropy, and quality measures (preprint). *University of California at Berkeley 73* (2002), 12. 3, 7

[SHW04] Schaefer S., Hakenberg J., Warren J.: Smooth subdivision of tetrahedral meshes. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (2004), ACM, pp. 147–154. 3

[Si15] Si H.: Tetgen, a delaunay-based quality tetrahedral mesh generator. *ACM Trans. Math. Softw. 41*, 2 (Feb. 2015), 11:1–11:36. 2, 3

[Stu17] Studios P. A.: Pixar in a box, jan 2017. URL: http://www.pixar.com/about/Pixar-In-A-Box. 2

[Sys17] Systems D.: Solidworks, jan 2017. URL: http://www.solidworks.de/. 2

[WBS*13] Weber D., Bender J., Schnoes M., Stork A., Fellner D.: Efficient gpu data structures and methods to solve sparse linear systems in dynamics applications. In *Computer Graphics Forum* (2013), vol. 32, Wiley Online Library, pp. 16–26. 2, 6

[WHP11] Wawrzinek A., Hildebrandt K., Polthier K.: Koiter's thin shells on catmull-clark limit surfaces. In *VMV* (2011), pp. 113–120. 3

[WKS*11] Weber D., Kalbe T., Stork A., Fellner D., Goesele M.: Interactive deformable models with quadratic bases in bernstein–bézier-form. *The Visual Computer 27*, 6 (2011), 473. 3

[WMRA*15] Weber D., Mueller-Roemer J., Altenhofen C., Stork A., Fellner D.: Deformation simulation using cubic finite elements and efficient p-multigrid methods. *Computers & Graphics 53* (2015), 185–195. 3, 7

[ZSS96] Zorin D., Schröder P., Sweldens W.: Interpolating subdivision for meshes with arbitrary topology. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), ACM, pp. 189–192. 3

[ZTT77] Zienkiewicz O. C., Taylor R. L., Taylor R. L.: *The finite element method*, vol. 3. McGraw-hill London, 1977. 3