Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2016)
Ladislav Kavan and Chris Wojtan (Editors)

Hierarchical *hp*-Adaptive Signed Distance Fields

Dan Koschier[†], Crispin Deul[‡] and Jan Bender[§]

Graduate School for Computational Engineering
TU Darmstadt, Germany

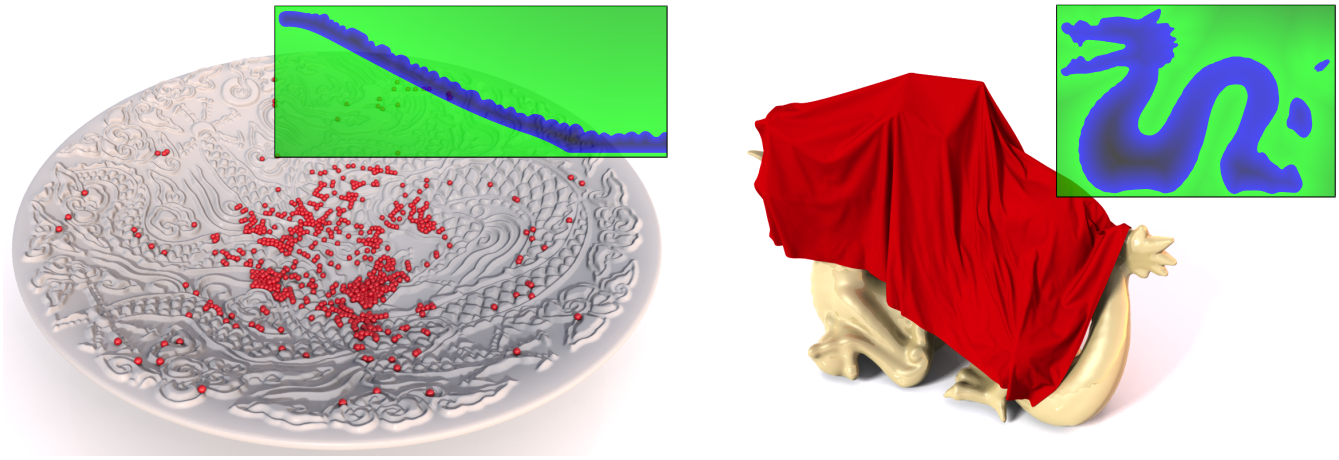


Figure 1: *Left: Collisions of 1000 marbles dropped into a bowl with highly complex structures are accurately resolved using our SDF representation. Right: A sheet of cloth represented by 320k triangles is dropped on the Stanford dragon. The mesh's characteristic features are outlined due to our accurate SDF representation serving as collision detector.*

Abstract

In this paper we propose a novel method to construct hierarchical hp-adaptive Signed Distance Fields (SDFs). We discretize the signed distance function of an input mesh using piecewise polynomials on an axis-aligned hexahedral grid. Besides spatial refinement based on octree subdivision to refine the cell size (h), we hierarchically increase each cell's polynomial degree (p) in order to construct a very accurate but memory-efficient representation. Presenting a novel criterion to decide whether to apply h- or p-refinement, we demonstrate that our method is able to construct more accurate SDFs at significantly lower memory consumption than previous approaches. Finally, we demonstrate the usage of our representation as collision detector for geometrically highly complex solid objects in the application area of physically-based simulation.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

Signed distance fields are a frequently used tool in the field of computer graphics and serve a wide range of applications including surface reconstruction [CT11], rendering [Jam10], geometrical modeling [FP06] or collision detection [MASS15]. Given a three-dimensional spatial domain $\mathcal{B} \subset \mathbb{R}^3$ the distance function is usually defined as the Euclidian distance from a given point $\boldsymbol{\xi} = (\xi, \eta, \zeta)^T$ to the nearest point which lies on the domain's boundary $\partial\mathcal{B}$. Moreover, a signed distance function augments the distance function by the information whether the point in question lies inside or outside the domain. From a mathematical point of view the signed distance

† koschier@gsc.tu-darmstadt.de
‡ deul@gsc.tu-darmstadt.de
§ bender@gsc.tu-darmstadt.de

function $\Phi : \mathbb{R}^3 \mapsto \mathbb{R}$ is defined as

$$\Phi(\boldsymbol{\xi}) = s(\boldsymbol{\xi}) \inf_{\boldsymbol{\xi}^* \in \partial\mathcal{B}} \|\boldsymbol{\xi} - \boldsymbol{\xi}^*\|,$$
$$s(\boldsymbol{\xi}) = \begin{cases} -1 & \boldsymbol{\xi} \in \mathcal{B} \\ 1 & \text{otherwise.} \end{cases} \quad (1)$$

While analytic representations for signed distance functions of rather simple shapes (e.g. spheres, tori, boxes etc.) are efficient to evaluate, the computation of the signed distance to arbitrary polyhedral shapes is very expensive. For that reason, it is common practice to discretize the signed distance function in order to evaluate the function more efficiently. In the following, we will refer to a discretized signed distance function as Signed Distance Field (SDF).

The most common approach to construct an SDF is to sample the signed distance at the vertices of a regular hexahedral grid and to trilinearly interpolate within each cell, as e.g. proposed by Xu and Barbič et al. [XB14b]. However, for complex objects this discretization strategy either consumes a large amount of memory or is not sufficiently accurate while the sampling additionally may suffer from aliasing effects. More elaborate approaches sample the function adaptively in order to increase the accuracy in highly detailed regions and to reduce memory consumption, e.g. using an octree as proposed by Frisken et al. [FPRJ00]. Especially in regions near curved or sharp features, strong subdivision is demanded yielding very memory consuming SDF representations.

In this paper, we propose a novel method to efficiently construct a grid-based SDF using hierarchical *hp*-refinement based on piecewise polynomial fitting. Besides spatial adaption using octree subdivision to refine the cell-size (h), we adapt the approximation's polynomial degree (p). We employ an orthonormal polynomial basis using shifted, normalized Legendre polynomials that enables us to hierarchically construct higher order polynomials without any rejection of previously computed coefficients. Using a novel *hp*-decision criterion we steer the refinement process to decide whether to apply h- or p-adaption. We demonstrate on complex objects that our method generates highly accurate SDFs while keeping the memory consumption at a minimum. Based on nearness weighting, we provide the user the possibility to focus the refinement during construction on regions near the underlying object's surface. Finally, we show in several experiments that our *hp*-adaptive SDFs are well-suited for the robust detection of collisions in dynamic simulations (cf. Figure 1). Besides pure detection of contacts, the field provides information about penetration depth and contact normals. Additionally, we would like to mention that our method is not limited to this application.

2. Related Work

Numerous approaches in computer graphics use Signed Distance Fields (SDF). See Jones et al. [JBS06] for an overview.

In the field of physics-based animation, SDFs are especially well-suited for collision detection. SDFs allow rapid distance queries between possibly colliding objects. Furthermore, the gradient of the SDF, which defines the shortest path to the surface, can be used as contact normal in the collision response. Bridson et al. [BMF03] as well as Fuhrmann et al. [FSG03] use SDFs to resolve collisions between cloth and rigid objects. Haptic rendering, which involves data provided by SDFs, is presented by Barbič and James [BJ08]. Rigid body collisions are detected by methods of Kaufman et al. [KSP07], Glondu et al. [GSM*12], and Xu and Barbič [XZB14]. Furthermore, Xu et al. [XB14a] present a continuous collision detection for rigid bodies. In this paper, we also demonstrate the usage of our new SDF representation in the application area of rigid body collision detection. However, with the modifications presented by McAdams et al. [MZS*11], our SDF approach can also be applied to rigid-deformable and deformable-deformable collision detection.

Since the introduction of SDFs to computer graphics by Rosenfeld and Pfaltz [RP66], many methods have been presented to accelerate the exact evaluation of signed distance functions based on meshes (see e.g. [SFP12]). Although these methods got faster and faster, the computation times are still far too long than required for applications like interactive simulation or haptic rendering. Moreover, approximations to signed distance functions using precomputed SDF are often sufficiently accurate. Due to the fact that discretizations of increasingly complex scenes are very memory consuming, various methods focusing on a reduction of the memory consumption were developed. One of the most popular methods is the adaptively sampled distance fields (ADFs) approach introduced by Frisken et al. [FPRJ00]. ADFs construct an octree of the SDF. During construction, a cell of the octree is divided into finer cells as long as the sampled error of the distance approximation of the current cell at the new corners of the finer cells is above a given threshold. It follows that ADFs require many fine cells in regions where the trilinear discretization does not adequately represent the distance to the surface. The construction time of ADFs as well as the memory consumption have been improved by Perry and Frisken [PF01]. Recently, Liu and Kim [LK14] presented a method to compute ADFs on GPUs. Another popular approach to reduce memory consumption is to discretize only a narrow band close to the objects surface as proposed by Bærentzen [Bær02] and Erleben and Dohlmann [ED08]. Our *hp*-adaptive SDFs could also be generated for a narrow band in order to additionally save memory. However, we are generally interested in a high quality representation of the SDF on the whole predefined domain in order to quickly exclude possible contacts when testing against bounding spheres.

A hybrid approach for meshes between exact evaluation of distance fields and precomputation of distance information is presented by Huang et al. [HLC*01]. This approach works with a regular grid where each cell stores data providing all triangles that influence the distance values inside the cell. As a result, exact distance evaluation is available for all cells. While this method decouples grid size and accuracy successfully, it is difficult to find the right trade-off between the grid size and the number of triangles per cell. Other approaches augment the grid with additional data to represent sharp features like corners or edges without subdividing the grid unnecessarily. For example, Ju et al. [JLSW02] store hermite data on the grid, Qu et al. [HNR*04] store an additional curvilinear grid and Bærentzen [Bae05] uses a point cloud in addition to the grid. A very cache efficient spatial subdivision scheme for volumetric data on a grid has been presented by Museth [Mus13]. This approach is tailored for very large sparse data sets with a domain of at least 8192^3 cells. In order to handle large data sets the

approach uses a structure similar to a B+-tree to find the cells containing data. Instead of dividing space into hexahedral cells, Wu and Kobbelt [WK03] propose to use a binary space partition (BSP-tree) where the distance field inside the cells is approximated by a linear function. The main advantage of this method compared to grids is the adjustment of splitting planes to the geometry. In opposition, our method fits a function to the distance field. In contrast to spatial subdivision schemes, Jones [Jon04] transforms the distance field with a vector distance transform and a specially defined predictor to be able to use entropy compression on the field. By reducing the SDF data to a 2D height field projected onto a proxy geometry Otaduy et al. [OJSL04] as well as Moustakas et al. [MTS07] reduce the consumed memory. The main problem of both approaches is to define a suitable proxy geometry.

Mitchell et al. [MASS15] present multivalued signed distance fields where several cells might occupy a single volume of space. This allows to represent non-manifold features that cannot be represented by standard grid-based representations. As such, the approach is orthogonal to the aforementioned methods to represent more detail with less memory. Therefore, the method should also be compatible with our grid-based method. Image-based volume contacts proposed by Faure et al. [FBAF08] and Allard et al. [AFC*10] are an alternative approach of capturing detailed contact geometry but require high resolution sampling for precise contact handling. As a result, Wang et al. [WFP12] apply, similar to our method, an error estimation based on polynomials to guide the refinement of the spatial sampling.

Like most of the aforementioned methods, our approach uses spatial subdivision to increase the accuracy. But in addition to that, we apply higher-order polynomial fitting to approximate the distance field inside the grid cells. As a result, our method can use the best fitting tool depending on the current distance field data to reduce memory consumption.

3. Signed Distance Field Construction

In this section we will describe how to construct the hierarchical *hp*-adaptive SDF. The algorithm expects a rectangular domain Ω in the form of an axis-aligned bounding box, an initial grid resolution and the function Φ, as defined in Equation (1), that maps a query point to the exact signed distance of the underlying geometry, e.g. a polygonal mesh. The construction consists of several steps. In the initialization step we construct a coarse signed distance field using the initial grid resolution by fitting low-order polynomials into each cell serving as an initial guess. Subsequently, we estimate the error in terms of the quadratic distance between the polynomial approximation and its embedded lower order approximation. We select the cell contributing the largest residual and decide whether to perform *h*- or *p*-refinement by means of a novel criterion, followed by the actual refinement. Finally, we repeat the previous step as long as the residual exceeds a certain threshold. For a more compact overview we outlined our method in Algorithm 1. All variables contained in the algorithm will be explained the following sections.

Algorithm 1: *hp*-adaptive SDF construction.

```
Data: n_x, n_y, n_z, τ, Ω, p_max, l_max
ε ← 0
n ← n_x n_y n_z
pending ← priority_queue{}
for e ← 0 to n do
    fit_polynomial(e, 2)                 // Fit polynomial of lowest order 2 to each base cell e. Equation (5)
    ε_e ← estimate_error(e)              // Equation (6)
    ε ← ε + ε_e
    pending.push({e, ε_e})
end
while not pending.empty() and ε > τ do
    {e, ε_e} ← pending.pop()
    {p, l} ← {degree(e), level(e)}
    μ_e ← estimate_improvement_p(e)      // Equation (8)
    ν_e ← estimate_improvement_h(e)      // Equation (9)
    refine_p ← p < p_max and ( l == l_max or μ_e > ν_e )
    refine_h ← l < l_max and not refine_p
    if refine_p then
        fit_polynomial(e, p + 1)         // Equation (5)
        ε ← ε − ε_e
        ε_e ← estimate_error(e)          // Equation (6)
        ε ← ε + ε_e
        pending.push({e, ε_e})
    end
    if refine_h then
        children ← subdivide(e)          // Octree subdivision.
        ε ← ε − ε_e
        for j ∈ children do
            fit_polynomial(j, p)         // Equation (5)
            ε_j ← estimate_error(j)      // Equation (6)
            ε ← ε + ε_j
            pending.push({j, ε_j})
        end
    end
end
```

3.1. Exact Signed Distance Computation

In order to determine the exact signed distance Φ from a given point to the input mesh, we first determine the unsigned distance. This is done by finding the nearest triangle within the mesh and by computing the distance to the triangle. In order to accelerate the nearest triangle search, we build a bounding sphere hierarchy in combination with a special traversal algorithm according the method proposed by Sanchez et al. [SFP12]. Afterwards, we determine the sign of the minimal distance using the angle-weighted pseudo-normal test which solely consists of a single dot product as proposed by Bærentzen and Aanæes [BA05]. For further details on the pseudo-normal test, we would like to refer the reader to their publication.

3.2. Polynomial Fitting

In this section we describe how to efficiently fit a multivariate polynomial of degree p to the exact signed distance function for a single cell. Given an arbitrary polynomial basis, we are interested in the coefficient set that minimizes the quadratic distance between the polynomial and the signed distance function. Mathematically, this results in the following quadratic minimization problem:

$$\begin{gathered}\min_{\mathbf{c}_e^p} R_e^p(\mathbf{c}_e^p) \\ R_e^p = \int_{\Omega_e} \frac{1}{2}(f_e^p - \Phi)^2 d\boldsymbol{\xi}, \quad f_e^p = \mathbf{c}_e^p \cdot \mathbf{P}_e^p \\ \mathbf{P}_e^p = \left\{P_e^{\boldsymbol{\rho}}\right\},\ \mathbf{c}_e^p = \left\{c_e^{\boldsymbol{\rho}}\right\} \\ \boldsymbol{\rho} = (\rho_\xi, \rho_\eta, \rho_\zeta),\ 0 \leq \rho_\xi + \rho_\eta + \rho_\zeta \leq p,\end{gathered} \tag{2}$$

where f_e^p represents the polynomial degree p approximation, R_e^p the half squared error to the exact signed distance function and where the vectors $\mathbf{P}_e^p$ and $\mathbf{c}_e^p$ denote the polynomial basis and corresponding coefficients, respectively. Furthermore, $\boldsymbol{\rho}$ describes the polynomial degree in each direction of the corresponding basis polynomial. This problem can then be reformulated as a dense linear equation system:

$$\begin{gathered}\mathbf{A}_e^p \mathbf{c}_e^p = \mathbf{b}_e^p, \\ \mathbf{A}_e^p = \int_{\Omega_e} \mathbf{P}_e^p \left(\mathbf{P}_e^p\right)^T d\boldsymbol{\xi}, \quad \mathbf{b}_e^p = \int_{\Omega_e} \mathbf{P}_e^p \Phi d\boldsymbol{\xi}.\end{gathered} \tag{3}$$

In order to finally interpolate using the fitted polynomial, $f_e(\boldsymbol{\xi})$ has to be evaluated.

3.3. Hierarchical p-Refinement

Unfortunately, the matrix $\mathbf{A}_e^p$ can be badly conditioned for arbitrary polynomial bases, e.g. a monomial basis. Another problem is that the system has to be reassembled and reevaluated if the basis is augmented by polynomials of higher degree. To overcome these issues, we construct a tensor product basis, based on shifted normalized Legendre polynomials:

$$\begin{aligned}P_e^{\boldsymbol{\rho}}(\boldsymbol{\xi}) &= \prod_{x \in \{\xi,\eta,\zeta\}} \sqrt{\frac{2\rho_x + 1}{b_e^x - a_e^x}} L_{\rho_x}(x') \\ L_p(x) &= \frac{1}{2^p} \sum_{l=0}^{p} \binom{p}{l}^2 (x-1)^{p-l} (x+1)^l \\ &= \frac{1}{p}\left((2p-1)\, x\, L_{p-1}(x) - (p-1)\, L_{p-2}(x)\right),\end{aligned} \tag{4}$$

where a_e^x and b_e^x are the minimum and maximum coordinate of the cell e in x-direction while $x' = \frac{2}{b_e^x - a_e^x} x - \frac{b_e^x + a_e^x}{b_e^x - a_e^x}$ is the shifted coordinate. Thanks to the basis' orthonormality, i.e. $\int_{\Omega_e} P_e^{\boldsymbol{\rho}} P_e^{\boldsymbol{\rho}^*} \delta\boldsymbol{\xi} = \delta_{\rho_\xi \rho_\xi^*} \delta_{\rho_\eta \rho_\eta^*} \delta_{\rho_\zeta \rho_\zeta^*}$, the matrix of Equation (3) becomes the identity matrix, i.e. $\mathbf{A}_e = \mathbf{I}$, where δ_{ij} denotes the Kronecker-δ. As a consequence, the formula for the coefficients reduces to

$$\mathbf{c}_e^p = \int_{\Omega_e} \mathbf{P}_e^p \Phi d\boldsymbol{\xi}. \tag{5}$$

Please note that besides the vanished requirement to compute the matrix $\mathbf{A}_e$, there remains no coupling between the coefficients. This is especially advantageous as only new coefficients have to be computed when increasing the polynomial degree. Based on this fact, we consider the basis *hierarchical*, as the coefficient set $\mathbf{c}_e$ can simply be augmented by computing the missing, desired entries without affecting existing ones. By definition of Equation (2), the number of entries contained in vectors $\mathbf{P}_e^p$ and $\mathbf{c}_e^p$ is then $n_c(p) = \frac{1}{6}(6 + 11p + 6p^2 + p^3)$. Note that it would also be possible to use the complete set of polynomials up to order p, such that $0 \leq \max(\rho_\xi, \rho_\eta, \rho_\zeta) \leq p$ instead of $0 \leq \rho_\xi + \rho_\eta + \rho_\zeta \leq p$. However, the number of resulting vector entries would grow faster ($n_c(p) = (p+1)^3$) which would result in a less granular refinement.

The biggest challenge during the described fitting step is to evaluate the integral in Equation (5). A common approach to solve integrals with a-priori unknown integrand is to apply locally or globally adaptive, multi-dimensional numerical integration rules, e.g. adaptive Gauss quadrature or Monte-Carlo integration using importance sampling. However, an application of these methods to Equation (5) would heavily suffer from two issues. The first issue is the smoothness of the integrand. While the polynomials are smooth and therefore infinitely often differentiable, Φ is only guaranteed to be continuous for two-manifold geometries and usually contains discontinuities in its derivatives. A possible source of these 'kinks' are sharp features in the underlying geometry. Moreover, they may also arise when smooth surfaces are considered as, e.g., the signed distance function in the center of a sphere is non-differentiable. This results in a high number of required Φ evaluations during numerical integration. Even worse, an evaluation of Φ as defined in Equation (1) is computationally expensive, hence the number of evaluations should be kept to a minimum. In order to compute the integral sufficiently well and in an acceptable amount of time, we heuristically approximate it using multi-dimensional Gauss quadrature of order $4p$, where p is the highest polynomial degree contained in $\mathbf{P}_e$. Using this heuristic, we experienced no artifacts or major issues. Moreover, our results demonstrate the quality of the generated SDFs.

3.4. Hierarchical h-Refinement

Besides incrementing the polynomial degree, a spatial subdivision is also desirable. More specifically it is especially useful in regions where the underlying signed distance function has low regularity and is therefore not very smooth. In this case the benefit of higher order polynomials is marginal while h-adaption, typically using low-order polynomials, resolves 'kinks' in the function better. Based on a regular subdivision we generate an octree for each of the base cells. If a cell is about to be h-refined, we again fit polynomials to the exact signed distance function by means of solving Equation (5) for the finer cells and reject the coarser approximation. Please note that the construction of the coarser approximation was not redundant as it is essential in order to decide if the cell should be h- or p-refined in the further process.

3.5. Error Estimation

The previously described refinement methods must be steered in order to control where the approximation should be improved and to decide whether to apply h- or p-refinement. As the function Φ may be evaluated at any point in space, it is theoretically possible

to directly compute the exact quadratic error $\varepsilon = \sum_e R_e(\mathbf{c}_e)$ using the sum of each cell e's individual quadratic error. However, it is computationally very expensive to approximate this exact quadratic error which is due to two reasons. First, every evaluation of the exact signed distance function Φ is rather expensive depending on the complexity of the underlying mesh, so we aim to keep the number of evaluations to a minimum. Second, the integrand is in general locally non-smooth which results in either poor accuracy if static numerical integration is used or an unacceptably high number of function evaluations for adaptive numerical integration rules. Please note that we initially intended to use an approximation to the exact error, but discovered that either the accuracy or the runtime was not acceptable.

As a robust alternative, we estimate the remaining error using the currently available approximation. More specifically, our estimation is based on the difference of the current degree p approximation compared to a lower order approximation of degree $p-1$:

$$\begin{aligned}
\varepsilon_e^p &= \int_{\Omega_e} \left(\Delta_e^p\right)^2 d\boldsymbol{\xi} \\
&= \int_{\Omega_e} \left(\sum_{i+j+k=p} c_e^{(i,j,k)} P_e^{(i,j,k)} \right)^2 d\boldsymbol{\xi} \\
&= \sum_{i+j+k=p} \sum_{\alpha+\beta+\gamma=p} c_e^{(i,j,k)} c_e^{(\alpha,\beta,\gamma)} \int_{\Omega_e} P_e^{(i,j,k)} P_e^{(\alpha,\beta,\gamma)} d\boldsymbol{\xi} \\
&= \sum_{i+j+k=p} |c_e^{(i,j,k)}|^2, \\
\Delta_e^p &= \mathbf{c}_e^p \cdot \mathbf{P}_e^p - \mathbf{c}_e^{p-1} \cdot \mathbf{P}_e^{p-1}
\end{aligned} \tag{6}$$

Thanks to the choice of the orthonormal polynomial basis, the estimation is computationally very efficient due to two reasons. First, the lower order approximation is directly embedded in the current approximation and can simply be computed by using only the computed coefficients $\mathbf{c}$ up to the requested order. Second, the constant coefficients can be factored out while the integrals of the polynomials of equal degree become exactly 1 and all others 0. This finally results in simple sum of squared coefficients.

3.6. Construction Algorithm

In this section we will give a detailed description of Algorithm 1. The approach can be considered a globally adaptive refinement following a top-down strategy. The main idea of the construction algorithm is to maintain a priority queue yielding the next cell designated to be refined based on the individual error contributed by the cell.

In line 1 to 3 we initialize the total error variable and the priority queue and compute the number of base cells in the initial, user-defined grid resolution. In the initialization loop (line 4 to 9) we fit a polynomial of degree 2 into each cell, estimate the error contributed by the cell, accumulate the error in the total error variable and insert the cell index based on its error contribution into the priority queue. The core part of the algorithm is the refinement loop described in line 10 to 34. We aim to refine the approximation as long as the error exceeds a certain threshold τ and refinable cells exist. After retrieving the element contributing the highest individual error, we have to decide whether to geometrically refine the cell or to increase its approximation's polynomial degree. If the cell has reached its maximum refinement level, only the degree may be increased and vice versa, such that no further criterion is required. Otherwise, we estimate the improvement that a p- or an h-adaption would yield. This is done by trying out both refinement strategies and measuring the remaining error on the h-adaption induced subcells. In order to make a final decision we developed the hp-decision criterion:

$$\begin{cases} \text{adapt } p & \text{if } \mu_e > \nu_e \\ \text{adapt } h & \text{otherwise,} \end{cases} \tag{7}$$

$$\mu_e = \frac{1}{n_c(p+1) - n_c(p)} \left(\varepsilon_e^p - 8\,\varepsilon_e^{p+1} \right), \tag{8}$$

$$\nu_e = \frac{1}{7 n_c(p)} \left(\varepsilon_e^p - 8 \max_{c \in \mathcal{C}_e} \varepsilon_c^p \right), \tag{9}$$

where $\mathcal{C}_e$ is the set of child cells resulting from the octree subdivision of e. The criterion decides in favor of a p-adaption if the improvement per additional degree of freedom based on the scaled error of the $(p+1)$-polynomial defined on the coarse cell is greater than the improvement per additional degree of freedom due to the scaled maximum error of the spatially subdivided order p polynomial measured on each of the subdomains on the finer octree level. The reason for preferring the criterion over simply measuring which adaption would result in the greater improvement is the following. We aim to favor an h-adaption if the approximation on any of the potential subcells gains more accuracy from h-adaption compared to p-adaption. Additionally, the scaling balances the refinement to counteract over-refinement in one dimension. Otherwise, the algorithm tends to drastically increase the polynomial degree in the first few steps as this improves the approximation on average over the coarse cell very well while there is potentially only a small improvement on some of the octree subdomains. This forces at least the same degree on the subcells resulting from subsequent h-adaptions. Consequently, many unnecessary degrees of freedom arise resulting in a higher memory consumption and computational effort for both construction and interpolation. Following the criterion, we either refine the grid or increase the polynomial degree according to lines 18-22 or 25-32, respectively. Moreover, the error is updated and the resulting cells' indices with the respective individual errors are inserted into the queue.

Please note that we accumulate the residual during the whole construction process. To overcome concerns regarding an increasing numerical error, we store the residual of each individual cell and recompute the total residual $\varepsilon = \sum_e \varepsilon_e$ every 1000 steps.

3.7. Nearness Weighting

For some applications of SDFs a comparably higher accuracy near the surface may be desired while regions far away from the surface are less interesting. In order to achieve the respective refinement behavior, we extend Equation (6) by a weighting factor κ_e depending on the nearness of the cell to the surface:

$$\begin{aligned}
\varepsilon_e^* &= \left(1 - \frac{1}{V_e d} \left| \int_{\Omega_e} \Phi d\boldsymbol{\xi} \right| \right)^\theta \varepsilon_e \\
&\approx \left(1 - \frac{1}{V_e d} \left| \int_{\Omega_e} f_e d\boldsymbol{\xi} \right| \right)^\theta \varepsilon_e = \kappa_e \varepsilon_e,
\end{aligned} \tag{10}$$

Figure 2: *Legend. Each color in this legend represents either the polynomial degree or the octree refinement level in other plots.*

where V_e, d and θ are the cell e's volume, the construction domain's diagonal and the nearness exponent, respectively. By replacing Φ with f_e we avoid the expensive exact signed distance evaluations and use the current approximation instead. The factor's integral part divided by the cell's volume represents the average signed distance value of the approximation. We additionally divide the cell by d as the maximum possible average distance must be smaller than the length of the domains diagonal if we presume that the underlying object is completely contained in the domain, i.e. $\mathcal{B} \subseteq \Omega$, such that we can guarantee $0 \leq \kappa_e \leq 1$. However, due to potentially strong deviations of the approximation, κ_e may lie outside of the interval. In this case we simply clamp the factor to $[0,1]$. Please note that if nearness weighting is used, the criterion described by Equations (7)-(9) must be modified accordingly.

4. Results and Discussion

All computations presented in this section were performed on two Intel Xeon E5-2697 processors with 2.7GHz, 30MB Cache, 12 cores per processor and 64GB RAM. We parallelized the SDF construction using Intel TBB while we always executed the code with 48 threads. All deformable and rigid body simulations with contacts are based on the approaches proposed by Bender et al. [BKCW14] and Deul et al. [DCB14] implemented in the open-source library *PositionBasedDynamics* [Ben16]. In summary our results cover four types of experiments. Firstly, we analyzed the convergence of the proposed method with respect to the number of coefficients. Secondly, we generated SDFs for a variety of meshes and summarized the key data in Table 1. Thirdly, we simulated various scenarios including rigid and deformable objects demonstrating the practical applicability of our approach for physically-based simulation. Finally, we measured the average time required to compute distance values with our SDFs. In the following paragraphs each of these experiments will be described in detail.

Convergence and Refinement Analysis Figures 3 and 4 show the convergence behavior with respect to the number of required coefficients (*#DOF*) measured using the estimated error (residual) described in Equation (6) on logarithmic scale. We compared our *hp*-adaptive approach to pure octree-subdivision with linearly (*h*1-adaptive) and quadratically (*h*2-adaptive) fitted polynomials. Both examples show the superiority of our approach as we require a fraction of the number of coefficients compared to the other methods. The curves' 'kinks', mostly visible in the curve according to *h*1-adaption, appear when all cells of a certain octree level are subdivided such that the decrease in the residual becomes suddenly smaller. We would like to put additional emphasis on the fact that we constructed the polynomials in all cases using the fitting approach (cf. Equation (2)) which yiels the optimal solution in terms of the measured error. Using the standard approach of previous works of sampling distance values within each cell would yield

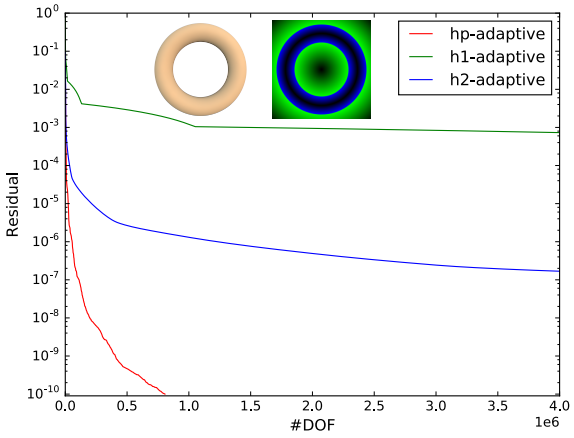


Figure 3: *Comparison of the convergence for a torus model of our hp-adaptive method with a pure octree-subdivision using linearly and quadratically fitted polynomials. #DOF encodes the number of polynomial coefficients required to enforce the corresponding residual.*

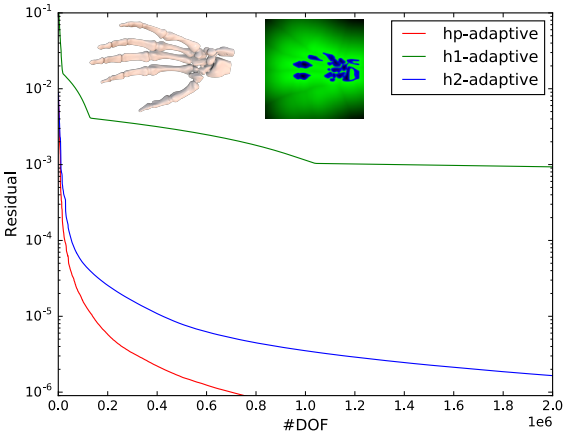


Figure 4: *Convergence study of SDF construction for a skeleton hand.*

even worse results for the *h*1- and *h*2-adaptions. For further investigation, we visualized the leaf cells and their polynomial degree and their according polynomial degree of an example slice as depicted in Figures 5 and 6. Notable is that the *h*-refinement with low-order polynomials was primarily used in regions where Φ is non-differentiable while smooth regions are mainly represented by large cells with high polynomial degree. This exactly correlates with our assumptions about the refinement behavior and demonstrates the meaningfulness and applicability of our *hp*-decision criterion (cf. Equation 7).

Construction Statistics Table 1 summarizes statistics on the input triangle meshes and the according SDF construction results. All in-

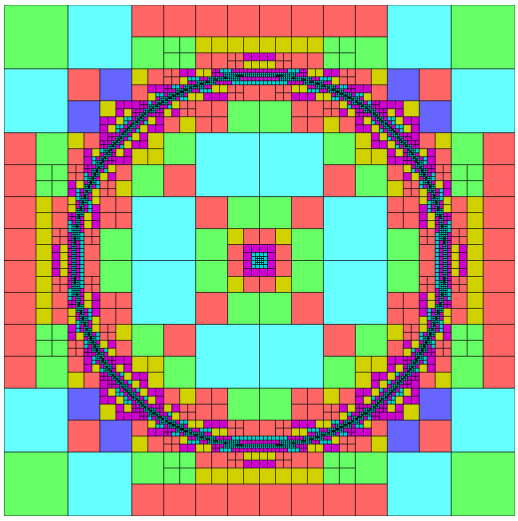


Figure 5: *Torus degree plot. Visualization of octree cells with corresponding polynomial degrees according to the legend depicted in Figure 2.*

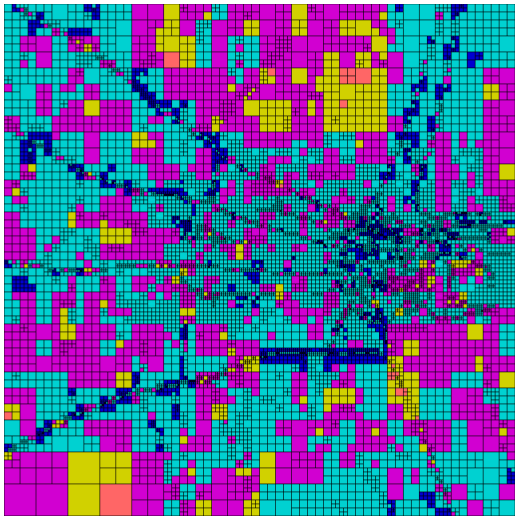


Figure 6: *Skeleton hand degree plot. Visualization of octree cells with corresponding polynomial degrees according to the legend depicted in Figure 2.*

put meshes were scaled to the unit box $[-1,1]^3$ and the construction domain was enlargened by 10% while we globally chose $p_{\max} = 30$ and $l_{\max} = 10$. Additionally, we used a nearness exponent of $\theta = 4$ for all examples. The mesh column shows the name of the mesh and its number of containing vertices and faces. The SDF column further contains the resolution of the intial construction grid, the time required for construction, the number of the resulting octree leaf cells, the per-volume distributions of degree and octree depth with colors according to the legend in Figure 2, the target error and the final memory consumption as well as a visualization of an exemplary slice of the SDF. We store the SDF in a data structure which mainly

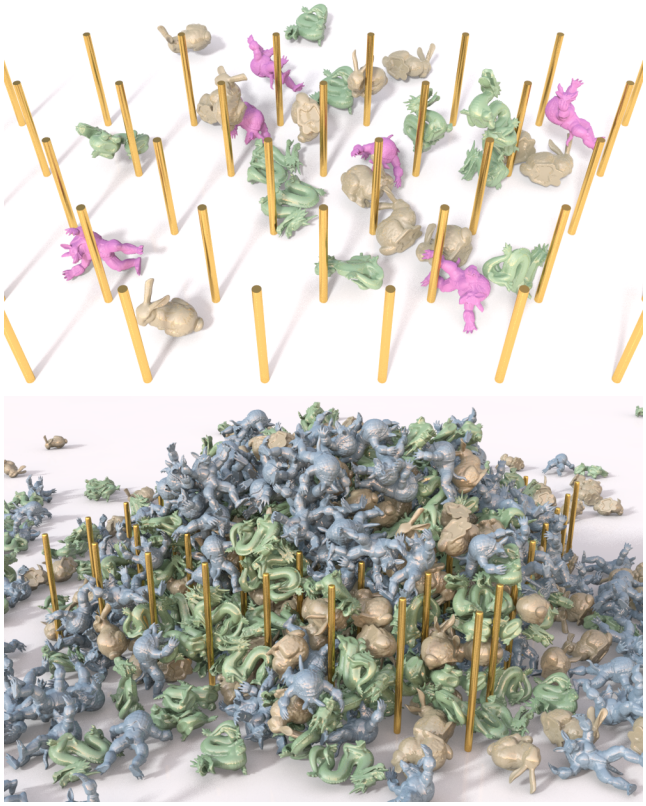


Figure 7: *Top: Complex rigid and deformable bodies slide down an inclined plane with obstacles. Bottom: 800 rigid bodies fall onto a set of 64 poles having several thousand contacts per simulation step.*

consists of four arrays. The first two arrays contain the polynomial coefficients in double-precision and a prefix-sum stating at which point in the coefficient array the coefficients of each cell start and how many coefficients belong to the respective cell. The remaining two arrays represent a child node index list containing the index of the corresponding octree nodes stored in the last array.

Collision Detection In order to demonstrate the applicability of our SDFs to collision detection in dynamics simulations, we simulated several scenarios. The surface of each object was sampled using points which were organized in a bounding-sphere hierarchy (BSH). The BSH was constructed and traversed similar to the approach described by Sanchez et al. [SFP12]. Furthermore, we generated an SDF for each (non-deformable) object offline and serialized the SDF. The field was then loaded by the simulation code and in each time-step tested against the point samples of the other objects. Figure 7 shows two experiments where dynamic bodies collide with a couple of poles. Each body has 10k surface points which were used for the distance queries. The collision detection was accelerated by performing the collision tests for all pairs of bodies in parallel. In the first simulation (top) rigid and deformable bodies slide down an inclined plane. While the rigid dragons and bunnies have their own signed distance field, the collisions of the

Figure 8: *Dynamic simulation of a marble run with subsequent armadillo bowling.*

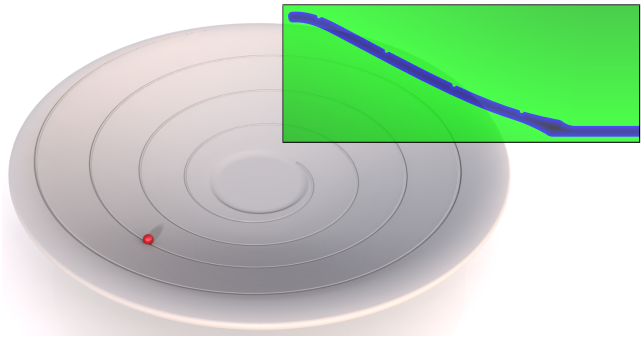


Figure 9: *Dynamic simulation of a marble following a highly-detailed, helix shaped groove in a bowl.*

deformable armadillos are only detected using the distance fields of the other bodies and obstacles. In the second experiment 800 rigid armadillo, bunny and dragon models were dropped onto a set of 64 poles. In a simulation time of 25 seconds the maximum and average number of contacts per step were 15050 and 8007, respectively. Within the finally resting body pile we observed more than 11600 contacts in each simulation step. Our collision detection including the BSH traversal and the signed signed distance field queries required an average computation time of 158 ms per simulation step. Note that such an efficient contact computation would not have been possible using a collision detection with triangle-triangle tests as our distance fields accurately represent all geometric models in the scenario with a total number of more than 132M triangles.

Figure 9 and 1 (left) show bowls with very fine structures. Regarding the first bowl, a marble runs down a helix-formed groove while we dropped 1000 marbles into the second bowl. In both scenarios the contact information between the marbles and the bowls is very accurate while the highly-detailed surfaces are flawlessly represented by the SDF. Figure 8 shows a marble rolling on a marble run. Please note that the geometry is very thin and small compared to its bounding box. Our method was still able to construct a very accurate SDF while the memory consumption was surprisingly small (cf. Table 1). In a final simulation scenario depicted in Figure 1 (right), we covered the Stanford dragon with a sheet of cloth. The features of the dragon surface are still clearly visible as they are silhouetted against the sheet.

Distance Query Performance Finally, we measured the time to query the distance using the SDF. We randomly sampled the field with several thousand points and averaged the resulting measured values. For the armadillo and structured bowl this took approximately 4.76×10^{-4} ms and 7.16×10^{-4} ms, respectively. If the SDF-gradient was additionaly requested, the queries took 7.34×10^{-4} ms and 7.84×10^{-4} ms. Thanks to the recursive form of the Legendre polynomials (cf. Equation (4)) their evaluation was accelerated by reusing redundant terms from order 0 to order p within the distance evaluation as well as the gradient computation.

5. Conclusion

In this paper we presented a novel hierarchical method to construct SDFs. We introduced an approach based on shifted, orthonormalized Legendre polynomials to efficiently fit polynomials to the exact signed distance function in a hierarchical manner. Spatial adaptivity was realized using octree subdivision. We developed a new *hp*-decision criterion based on an estimated error in order to steer the refinement. The criterion-controlled subdivision heavily improves the convergence compared to traditional pure spatial refinement. Moreover, a nearness weighting approach was presented that modifies the error measure such that the refinement is focused near the underlying object's surface. We demonstrated that our method is able to produce very accurate SDFs for complex geometries consuming only a small amount of memory and that these are very well-suited to detect contacts and collisions, implicitly providing depth and contact normal, in physically-based simulations.

Our method also has some limitations. Generally, the SDFs resulting from our approach are discontinuous over cell borders. However, the discontinuities vanish for small target errors τ. In order to produce meaningful results for coarse approximations we plan to extend our method in order to provide sufficiently smooth transitions between cells. Another limitation is that for two-dimensional objects embedded in three dimensions such as cloth or shells it can not be distinguished between 'inside' and 'outside'. For that reason our method is not useful to handle cloth-cloth contacts or self-intersections of cloth. Surfaces exactly touching each other as in cutting or fracture simulations cannot be easily resolved for collision handling by simply discretizing the signed distance. Therefore, we plan to investigate if we can extend our method to a non-manifold SDF similar to the work of Mitchell et al. [MASS15]. This would potentially allow us to apply the approach to collision detection in cutting and fracture simulations.

Acknowledgements

We would like to thank Manuel Scholz for his kind support in providing geometrical models for our results.

This work is supported by the 'Excellence Initiative' of the German Federal and State Governments and the Graduate School of Computational Engineering at Technische Universität Darmstadt.

References

[AFC*10] ALLARD J., FAURE F., COURTECUISSE H., FALIPOU F., DURIEZ C., KRY P. G.: Volume contact constraints at arbitrary resolution. *ACM Transactions on Graphics 29*, 4 (2010), 82:1–82:10. 3

[BA05] BÆRENTZEN J. A., AANÆS H.: Signed distance computation using the angle weighted pseudonormal. *IEEE Transactions on Visualization and Computer Graphics 11*, 3 (2005), 243–253. 3

[Bae05] BAERENTZEN A. J.: Robust Generation of Signed Distance Fields from Triangle Meshes. *International Workshop on Volume Graphics* (2005), 167–239. 2

[Bær02] BÆRENTZEN J. A.: *Manipulation of volumetric solids with applications to sculpting*. Phd thesis, Technical University of Denmark, 2002. 2

[Ben16] BENDER J.: PositionBasedDynamics Library. https://github.com/InteractiveComputerGraphics/PositionBasedDynamics, 2016. 6

[BJ08] BARBIČ J., JAMES D. L.: Six-DoF Haptic Rendering of Contact Between Geometrically Complex Reduced Deformable Models. *IEEE Transactions on Haptics 1*, 1 (2008), 39–52. 2

[BKCW14] BENDER J., KOSCHIER D., CHARRIER P., WEBER D.: Position-Based Simulation of Continuous Materials. *Computers & Graphics 44*, 0 (2014), 1–10. 6

[BMF03] BRIDSON R., MARINO S., FEDKIW R.: Simulation of clothing with folds and wrinkles. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (2003), Eurographics Association, pp. 28–36. 2

[CT11] CALAKLI F., TAUBIN G.: SSD: Smooth Signed Distance Surface Reconstruction. *Computer Graphics Forum 30*, 7 (2011), 1993–2002. 1

[DCB14] DEUL C., CHARRIER P., BENDER J.: Position-based rigid-body dynamics. *Computer Animation and Virtual Worlds* (2014). 6

[ED08] ERLEBEN K., DOHLMANN H.: Signed Distance Fields Using Single-Pass GPU Scan Conversion of Tetrahedra. In *GPU Gems 3*. Addison-Wesley Professional, 2008, ch. 34, pp. 741–763. 2

[FBAF08] FAURE F., BARBIER S., ALLARD J., FALIPOU F.: Image-based collision detection and response between arbitrary volume objects. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (2008), pp. 155–162. 3

[FP06] FRISKEN S. F., PERRY R. N.: Designing with Distance Fields. In *ACM SIGGRAPH Courses* (2006), pp. 60–66. 1

[FPRJ00] FRISKEN S. F., PERRY R. N., ROCKWOOD A. P., JONES T. R.: Adaptively sampled distance fields: a general representation of shape for computer graphics. *ACM Transactions on Graphics* (2000), 249–254. 2

[FSG03] FUHRMANN A., SOBOTTKA G., GROSS C.: Distance Fields for Rapid Collision Detection in Physically Based Modeling. In *Computer Graphics and Vision* (2003), pp. 1–8. 2

[GSM*12] GLONDU L., SCHVARTZMAN S. C., MARCHAL M., DUMONT G., OTADUY M. A.: Efficient Collision Detection for Brittle Fracture. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (2012), pp. 285–294. 2

[HLC*01] HUANG J., LI Y., CRAWFIS R., LU S. C., LIOU S. Y.: A complete distance field representation. In *IEEE Visualization* (2001), pp. 247–254. 2

[HNR*04] HUAMIN QU, NAN ZHANG, RAN SHAO, KAUFMAN A., MUELLER K.: Feature preserving distance fields. In *IEEE Symposium on Volume Visualization and Graphics* (2004), pp. 39–46. 2

[Jam10] JAMRIŠKA O.: Interactive Ray Tracing of Distance Fields. In *Central European Seminar on Computer Graphics* (2010), vol. 2. 1

[JBS06] JONES M., BAERENTZEN J., SRAMEK M.: 3D distance fields: a survey of techniques and applications. *IEEE Transactions on Visualization and Computer Graphics 12*, 4 (2006), 581–599. 2

[JLSW02] JU T., LOSASSO F., SCHAEFER S., WARREN J.: Dual Contouring of Hermite Data. *ACM Transactions on Graphics 21*, 3 (2002), 339–346. 2

[Jon04] JONES M. W.: Distance field compression. *Journal of WSCG 12*, 2 (2004), 199—-204. 3

[KSP07] KAUFMAN D. M., SUEDA S., PAI D. K.: Contact trees: Adaptive Contact Sampling for Robust Dynamics. In *ACM SIGGRAPH Sketches* (2007). 2

[LK14] LIU F., KIM Y. J.: Exact and Adaptive Signed Distance Fields Computation for Rigid and Deformable Models on GPUs. *IEEE Transactions on Visualization and Computer Graphics 20*, 5 (2014), 714–725. 2

[MASS15] MITCHELL N., AANJANEYA M., SETALURI R., SIFAKIS E.: Non-manifold Level Sets: A multivalued implicit surface representation with applications to self-collision processing. *ACM Transactions on Graphics 34*, 6 (2015), 247:1–247:9. 1, 3, 8

[MTS07] MOUSTAKAS K., TZOVARAS D., STRINTZIS M. G.: SQ-Map: Efficient Layered Collision Detection and Haptic Rendering. *IEEE Transactions on Visualization and Computer Graphics 13*, 1 (2007), 80–93. 3

[Mus13] MUSETH K.: VDB: High-resolution Sparse Volumes with Dynamic Topology. *ACM Transactions on Graphics 32*, 3 (2013), 27:1–27:22. 2

[MZS*11] MCADAMS A., ZHU Y., SELLE A., EMPEY M., TAMSTORF R., TERAN J., SIFAKIS E.: Efficient elasticity for character skinning with contact and collisions. *ACM Transactions on Graphics 30*, 4 (2011), 37:1–37:12. 2

[OJSL04] OTADUY M. A., JAIN N., SUD A., LIN M. C.: Haptic display of interaction between textured models. In *IEEE Visualization* (2004), pp. 297–304. 3

[PF01] PERRY R. N., FRISKEN S. F.: Kizamu: A System for Sculpting Digital Characters. In *Computer Graphics and Interactive Techniques* (2001), pp. 47–56. 2

[RP66] ROSENFELD A., PFALTZ J. L.: Sequential Operations in Digital Picture Processing. *Journal of the ACM 13*, 4 (1966), 471–494. 2

[SFP12] SANCHEZ M., FRYAZINOV O., PASKO A.: Efficient Evaluation of Continuous Signed Distance to a Polygonal Mesh. In *Spring Conference on Computer Graphics* (2012), pp. 101–108. 2, 3, 7

[WFP12] WANG B., FAURE F., PAI D. K.: Adaptive image-based intersection volume. *ACM Transactions on Graphics 31*, 4 (2012), 97:1–97:9. 3

[WK03] WU J., KOBBELT L.: Piecewise Linear Approximation of Signed Distance Fields. In *Vision, Modeling and Visualization* (2003), pp. 513–520. 3

[XB14a] XU H., BARBIČ J.: Continuous Collision Detection Between Points and Signed Distance Fields. In *Proceedings of Virtual Reality Interactions and Physical Simulations (VRIPhys)* (2014). 2

[XB14b] XU H., BARBIČ J.: Signed Distance Fields for Polygon Soup Meshes. In *Graphics Interface* (2014), pp. 1–7. 2

[XZB14] XU H., ZHAO Y., BARBIČ J.: Implict Multibody Penalty-based Distributed Contact. *IEEE Transactions on Visualization and Computer Graphics 20*, 9 (2014), 1266–1279. 2

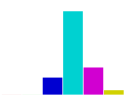
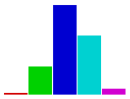
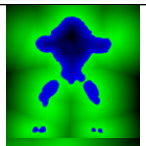
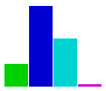
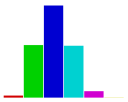
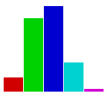
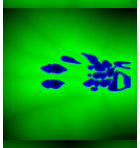
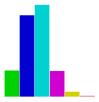
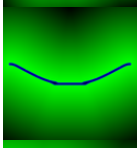
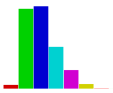
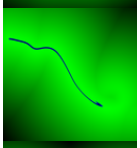
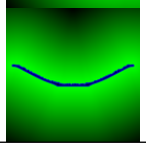
Mesh			Signed Distance Field							
Name	#Vert.	#Faces	Base Grid	Constr. Time	#Cells	Degree	Depth	ε, τ	Memory	Field
Armadillo	173k	346k	6^3	270s	71k			10^{-6}	10.6 MB	
Bunny	34.8k	69.6k	6^3	301s	55k			10^{-6}	8.4 MB	
Dragon	40k	80k	10^3	1245s	299k			10^{-7}	46.9 MB	
Hand	66.2k	132.5k	10^3	833s	159k			10^{-7}	25 MB	
Helix Bowl	164.9k	329.8k	4^3	423s	159k			5×10^{-9}	34.9 MB	
Marble Run	27k	53k	4^3	312s	87k			5×10^{-9}	18.4 MB	
Structured Bowl	2.05M	4.1M	4^3	4121s	1.25M			5×10^{-9}	250 MB	

Table 1: *Construction statistics. Mesh column shows object names and corresponding number of vertices and faces. The SDF column contains initial grid resolution, required time for construction, number of octree leaf cells, normalized histograms capturing the volume-fraction of the domain occupied by cells of the corresponding degree or octree depth, the (enforced) target error and the final memory consumption as well as a slice image of the SDF. The colors of the histograms are encoded as depicted in Figure 2 according to the degree or octree depth.*