An hp-Adaptive Discretization Algorithm for Signed Distance Field Generation

Dan Koschier, Crispin Deul, Magnus Brand, and Jan Bender

Abstract—In this paper we present an hp-adaptive algorithm to generate discrete higher-order polynomial Signed Distance Fields (SDFs) on axis-aligned hexahedral grids from manifold polygonal input meshes. Using an orthonormal polynomial basis, we efficiently fit the polynomials to the underlying signed distance function on each cell. The proposed error-driven construction algorithm is globally adaptive and iteratively refines the SDFs using either spatial subdivision (h-refinement) following an octree scheme or by cell-wise adaption of the polynomial approximation's degree (p-refinement). We further introduce a novel decision criterion based on an error-estimator in order to decide whether to apply p- or h-refinement. We demonstrate that our method is able to construct more accurate SDFs at significantly lower memory consumption compared to previous approaches. While the cell-wise polynomial approximation will result in highly accurate SDFs, it can not be guaranteed that the piecewise approximation is continuous over cell interfaces. Therefore, we propose an optimization-based post-processing step in order to weakly enforce continuity. Finally, we apply our generated SDFs as collision detector to the physically-based simulation of geometrically highly complex solid objects in order to demonstrate the practical relevance and applicability of our method.

Index Terms—Signed distance field, adaptive discretization, higher-order polynomials, physically based simulation, collision detection.

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 [1], rendering [2], geometrical modeling [3] or collision detection [4]. For a three-dimensional spatial domain $\mathcal{B} \subset \mathbb{R}^3$ the distance function is usually defined as the Euclidean distance from a given point $\boldsymbol{\xi} = (\xi, \eta, \zeta)^T$ in space to the nearest point on the boundary $\partial\mathcal{B}$ of the domain. The sign of the distance function additionally provides information about whether the point in question lies inside or outside the domain. Mathematically, the signed distance function $\Phi : \mathbb{R}^3 \to \mathbb{R}$ is defined as

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

Signed distance functions can be evaluated efficiently if an analytic form exists for the associated object. This is the case for simple geometric shapes such as spheres, tori, boxes, etc. For arbitrary polyhedral shapes the evaluation of the signed distance function is, however, computationally 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 rest of the paper, a discretized signed distance function is referred to as a Signed Distance Field (SDF).

Most commonly, an SDF is constructed by sampling the signed distance at the vertices of a regular hexahedral grid and by trilinearly interpolating within each cell, as e.g. proposed by Xu and Barbič et al. [5]. However, for complex objects this discretization strategy either consumes a large amount of memory or is not sufficiently accurate. The sampling may additionally suffer from aliasing effects. More elaborate approaches sample the function adaptively in order to increase the accuracy in regions with fine details and to reduce the overall memory consumption. The adaptive sampling can be realized, e.g. as an octree-like scheme, as proposed by Frisken et al. [6]. Especially in regions near curved or sharp features, strong subdivision is required resulting in 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 polynomial degree (p) of the local discretization. We employ an orthonormal polynomial basis using shifted, normalized Legendre polynomials. This enables us to hierarchically construct higher order polynomials without having to discard and recompute any of the previously computed coefficients. Using a novel hp-decision criterion our algorithm estimates whether h- or p-adaption is more beneficial in each individual refinement step. By constructing SDFs for complex surfaces, we demonstrate that our method generates highly accurate discretizations while memory consumption remains at a minimum. Using our novel nearness weighting approach, the user can choose to focus the refinement efforts on regions close to the surface of the associated object. Finally, we show in several

- *Dan Koschier - Computer Animation Group, RWTH Aachen University E-mail: koschier@cs.rwth-aachen.de*
- *Crispin Deul - Graduate School CE, TU Darmstadt E-mail: deul@gsc.tu-darmstadt.de*
- *Magnus Brand - Computer Animation Group, RWTH Aachen University E-mail: brand@cs.rwth-aachen.de*
- *Jan Bender - Computer Animation Group, RWTH Aachen University E-mail: bender@cs.rwth-aachen.de*

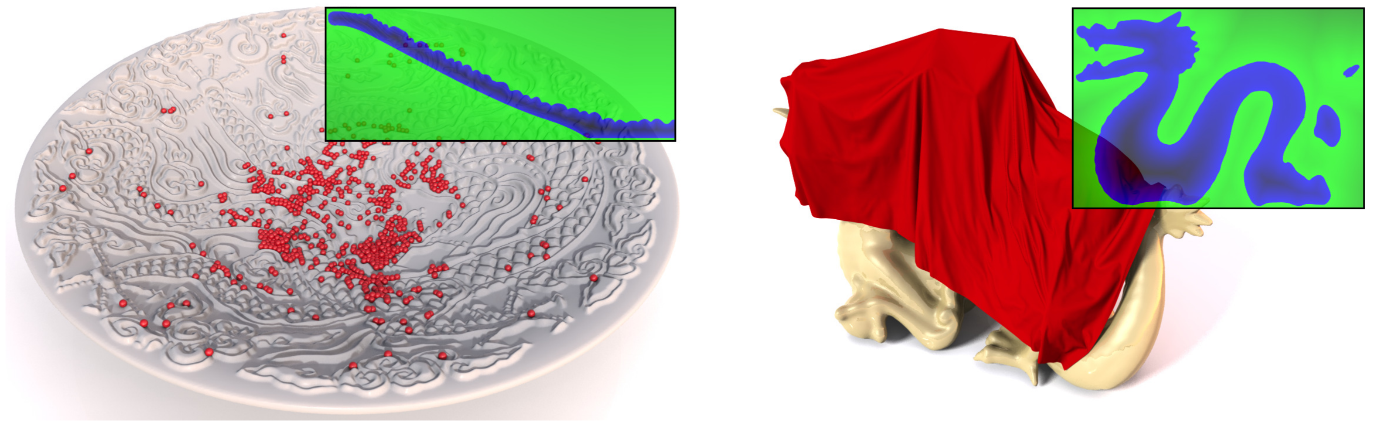


Fig. 1: Left: Collisions of 1000 marbles dropped into a bowl with highly complex structures are accurately resolved using our SDF representation. Right: A sheet of cloth represented by 320k triangles is dropped on the Stanford dragon. The mesh's characteristic features are outlined due to our accurate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

experiments that our *hp*-adaptive SDFs are well-suited for the robust detection of collisions in dynamic simulations (cf. Figure 1). In addition to the detection of contacts, the SDF also provides information about the penetration depth and contact normals. Our method is however not limited to this application.

2 Related Work

Since the introduction of SDFs by Rosenfeld and Pfaltz [7] to the computer graphics community, numerous approaches to construct and use SDFs have been proposed. For a general overview, we would like to refer the reader to the survey of Jones et al. [8].

Signed Distance Fields. In recent years, many methods have been presented to accelerate the exact evaluation of signed distance functions based on polygonal representations (see e.g. [9]). Even though the computational efficiency was drastically improved, the computation time still can not fulfill the strict time constraints of applications such as interactive simulation or haptic rendering. Moreover, approximations to signed distance functions using precomputed SDFs can be efficiently queried and, therefore, serve as an excellent alternative. Due to the fact that discretizations of increasingly complex surfaces are very memory consuming, various methods focusing on a reduction of memory consumption were developed. One of the most popular methods is the adaptively sampled distance fields (ADFs) approach introduced by Frisken et al. [6]. During the ADF construction the underlying signed distance function is first sampled on a coarse grid and recursively refined using octree subdivisions as long as the deviation of trilinearly interpolated signed distance samples from the exact value exceed a certain threshold. The method was later improved in terms of memory consumption and construction time by Perry and Frisken [10]. As a consequence of the refinement strategy, a large number of cells is required in regions where a trilinear discretization does not accurately represent the signed distance function. In order to further reduce the memory requirements narrow band approaches were proposed by Bærentzen [11] and Erleben and Dohlmann [12] that discretize only regions close to the object surface. A very cache efficient narrow-band discretization approach for volumetric data on grid-structures has been presented by Museth [13]. This method is tailored to very large sparse data sets with grid resolutions 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. At this point, we would like to mention that a narrow band construction is directly applicable to our proposed *hp*-adaptive SDFs. However, we are generally interested in a high quality representation of the SDF on the whole domain in order to quickly exclude possible contacts in the demonstrated application.

In contrast to using purely scalar valued SDFs several approaches were proposed that augment the discretization by additional geometric information. Huang et al. [14] propose a hybrid approach for distance field representation of polygonal meshes. Using a regular hexahedral grid, they store a list of triangles and the respective scalar distance value from the cell center to each triangle for each individual cell. This allows for an exact signed distance computation without any discretization errors as the polygonal representation is explicitly stored within the data structure. However, it is still necessary to perform expensive computations to evaluate distances to explicit polygons when querying the SDF. Moreover, explicitly storing the triangle lists results in a substantial memory overhead. Mitchell et al. [4] 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. In order to improve the representation of sharp features such as corners or hard edges while avoiding unnecessary refinement, several approaches were proposed. Ju et al. [15] store additional hermite data on the grid that represents exact intersection points and normals. Using an additional curvilinear offset grid Qu et al. [16] align the mesh features with the second grid to improve the discretization. Similarly, Bærentzen [17] uses an additional point cloud when reconstructing a mesh from an SDF in order to recover sharp features.

As opposed to discretizing the signed distance function using an axis-aligned hexahedral grid, Wu and Kobbelt [18] present a discretization approach based on binary space partitioning (BSP-tree). In each subdivision step the splitting planes are aligned with geometric features in the input data in order to optimize the approximation. Jones [19] completely avoids spatial subdivision but transforms the distance field with a vector distance transform using a specially defined predictor in order apply entropy compression to the distance data. By reducing the SDF data to a 2D height field projected onto a proxy geometry Otaduy et al. [20] as well as Moustakas et al. [21] reduce the consumed memory. The main problem of both approaches is to define a suitable proxy geometry.

SDF-based Collision Handling. In the field of physics-based animation, SDFs allow for rapid distance queries between potentially colliding objects and are therefore especially well-suited for collision detection. Moreover, a contact normal for collision response can be directly deduced by computing the SDF's gradient as it points in the direction of the closest point on the object surface. Several works adopt the concept of SDF-based collision detection for rigid body simulations, i.e. Kaufman et al. [22], Glondu et al. [23], and Xu and Barbič [24]. Bridson et al. [25] as well as Fuhrmann et al. [26] use SDFs to resolve collisions between cloth and rigid objects. Barbič and James [27] presented a method for haptic rendering involving data provided by SDFs. Image-based volume contacts proposed by Faure et al. [28] and Allard et al. [29] are an alternative approach of capturing detailed contact geometry but require high resolution sampling for precise contact handling. Therefore, Wang et al. [30] apply, similar to our method, an error estimation based on polynomials to guide the refinement of the spatial sampling. In order to improve efficiency and robustness of rigid body collision handling Xu et al. [31] developed an SDF-based continuous collision detector. In this paper, we demonstrate the applicability of our new SDF representation in rigid body simulations with contacts. Please note, that our collision handling using our SDF representation is not limited to rigid body simulations and can also be applied for rigid-deformable and deformable-deformable collision detection when considering the modifications presented by McAdams et al. [32].

Similar to several aforementioned methods, we propose an adaptive construction algorithm for SDF generation. In contrast to all previous approaches, we fit polynomials to the underlying signed distance function and are the first to not only spatially subdivide (h-adaption) the grid but also improve the discretization by hierarchical augmentation of the polynomial basis with higher order polynomials (p-adaption). As a consequence, our globally-adaptive, error-driven construction algorithm is able to generate memory efficient but highly accurate SDFs as we show in our results.

***hp*-Adaptivity in Numerical Methods for PDEs**. While this paper is, to the best of our knowledge, the first approach on hp-adaptive generation of SDFs, hp-adaptivity is well-investigated in the field of numerical methods for partial differential equations (PDEs), especially in the context of finite element solvers. In an early work, Babuška et al. [33] introduced the concept of p-adaptivity for a finite element solver for one- and two-dimensional PDEs and proved that its convergence rate is not worse (and in some cases even better) than h-adaptive approaches. The concept was then combined with an h-adaptive approach and analyzed by Babuška and Suri [34] building the first hp-adaptive method in finite element analysis. For a discussion on further developments in the field until 1994 we would like to refer the reader to the survey of Babuška and Suri [35]. As hp-adaptive approaches refine the approximation in two distinct dimensions a suitable criterion is required to decide whether to refine in h- or p-direction. In this regard, Mitchell and McClain [36] compare several strategies to guide the hp-refinement. A very effective strategy to decide which refinement direction is likely to improve the approximation best was first proposed by Schmidt and Siebert [37] for one-dimensional problems. They apply a refinement in p- and h-direction and choose the refinement direction that reduces the estimated remaining error the most. Unfortunately, for higher-dimensional problems this strategy is computationally very expensive as the numerical approximation has to be recomputed for each refinement step on the whole domain. In the application of SDF generation this problem is not present as the (discretized) field has only to be reconstructed in the currently considered cell which makes this strategy effective yet efficient. Similar to the basis polynomials in our approach, Houston et al. [38] use Legendre polynomials as shape functions in an hp-adaptive finite element solver for hyperbolic conservation laws. They analyze the decay rate of the polynomial's Legendre expansion coefficients to estimate local regularity of the PDE's solution in each finite element and use this information to guide the refinement directions.

In contrast to methods for numerical solvers for PDEs our approach is targeted towards construction of SDFs. We employ a normalized, shifted Legendre polynomial basis for discretization. The orthonormality property of the basis enables us to efficiently fit the basis functions to the exact input signed distance function without the requirement to solve a (usually required) linear equation system. Using the proposed polynomial basis we also develop an efficient degree-based error estimator and propose a novel refinement direction criterion by estimating the individual improvement of an h- or p-refinement step.

3 Signed Distance Field Construction

In this section, we present our novel hierarchical hp-adaptive SDF construction algorithm. Given an initial grid on an axis-aligned bounding box representing a rectangular domain Ω the method aims to discretize a signed distance function Φ (cf. Equation 1) implied by a corresponding surface descriptor, e.g. a polygonal mesh. The algorithm can be divided into the following steps. In the first step, a coarse SDF is constructed by fitting low-order polynomials to Φ on each individual cell serving as an initial guess. In the second step, we estimate the error contributed by each cell by computing the quadratic distance between the approximating polynomial and its embedded lower order polynomial. Following a globally-adaptive top-down strategy, we select the cell that contributes the largest residual as candidate for refinement in the third step and and apply our novel decision criterion in order to determine whether to apply h- or p-refinement. Finally, the third step is repeatedly performed until the aggregate residual of all cells tracked over the refinement process falls below the target error threshold. In the following sections we will give a detailed explanation of each step and discuss the mechanisms and mathematical foundations to perform the discretization.

3.1 Exact Signed Distance Computation

Given a triangular input mesh we require a method to determine the exact signed distance to the surface. Therefore, we first compute the unsigned distance by finding the closest triangle of the mesh and subsequently evaluating the distance to the individual polygon. As a naïve search for the nearest triangle has linear complexity, we accelerate the procedure by construction of a bounding sphere hierarchy with a special traversal algorithm as proposed by Sanchez et al. [9]. In order to determine the sign of the minimal distance we follow the approach of Bærentzen and Aanæes [39] by using the angle-weighted pseudo-normal test which only requires the evaluation of a single dot product with a precomputed surface normal.

3.2 Polynomial Fitting

In order to locally discretize the signed distance function we fit a multivariate polynomial of degree p to Φ on a single cell. Given an arbitrary polynomial basis, we minimize a quadratic distance measure between the polynomial approximation and the underlying signed distance function in order to find an optimal coefficient set for the basis polynomials. Mathematically, this results in the following quadratic minimization problem:

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

where R_e represents the half squared error to the exact signed distance function, Ω_e the domain of the eth cell, f_e the polynomial approximation of order p, respectively. Furthermore, the $\mathbf{P}_e$ and $\mathbf{c}_e$ denote the polynomial basis vector and cell coefficient vector, respectively. Furthermore, $\boldsymbol{\rho}$ describes the polynomial degree in each direction of the corresponding basis polynomial. The solution to the quadratic minimization problem corresponds to the solution of the linear equation system

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

which yields the desired coefficient set $\mathbf{c}_e$. The SDF can then be queried at point $\boldsymbol{\xi}$ by evaluating the fitted approximation $f_e(\boldsymbol{\xi})$.

3.3 Polynomial Basis and Hierarchical p-Refinement

Obviously, the underlying polynomial basis affects the structure and condition number of matrix $\mathbf{A}$. Moreover, the dense linear equation system grows when the degree of the polynomial basis is increased. For these reasons, we aim to employ a basis that is orthogonal on the corresponding cell which diagonalizes the matrix in Equation (3). The fact that Legendre polynomials have the property to be orthogonal on the interval $[-1, 1]$ makes them attractive for the construction of a higher-dimensional orthogonal basis. In order to generalize the Legendre basis to be orthogonal on an arbitrary interval we shift the coordinates accordingly. Consequently, we construct a polynomial tensor-product basis based on shifted normalized Legendre polynomials that keeps the system well-conditioned and diagonalizes the generally dense linear system. The polynomial basis is then defined by

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

where a_e^x and b_e^x are the minimum and maximum coordinate of the cell e in x-direction with shifted coordinate $x' = \frac{2}{b_e^x - a_e^x} x - \frac{b_e^x + a_e^x}{b_e^x - a_e^x}$. The one-dimensional portions of the Legendre basis are depicted in Figure 2. Due to the orthonormality of the basis, 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 solver matrix becomes the identity matrix, i.e. $\mathbf{A}_e = \mathbf{I}$, where δ_{ij} denotes the Kronecker-δ. Consequently, the linear equation system (3) reduces to

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

Besides the fact that the requirement to compute and assemble $\mathbf{A}_e$ vanished, the diagonality implies that there is no coupling between the coefficients. This is especially

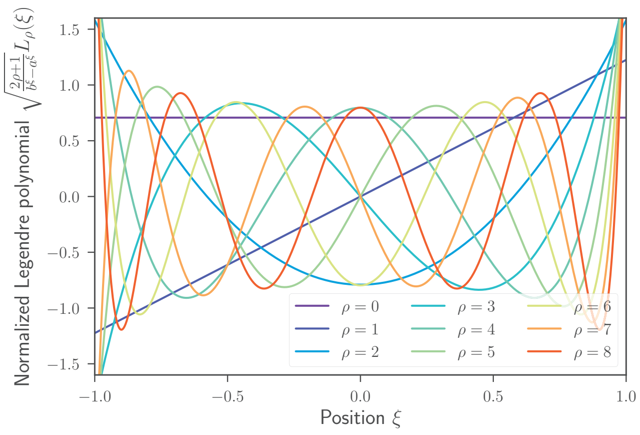


Fig. 2: First nine (one-dimensional) shifted, normalized Legendre polynomials used for discretizing the signed distance function on cell interval $[a_\xi, b_\xi] = [-1, 1]$.

advantageous as only new coefficients have to be computed when the approximation's polynomial degree is increased. As the approximation's polynomial degree can simply be augmented by computation of desired coefficients in $\mathbf{c}_e$ we consider the basis *hierarchical*. By definition of Equation (2), the number of entries contained in vectors $\mathbf{P}_e$ and $\mathbf{c}_e$ is $n_c(p) = \frac{1}{6}(6 + 11p + 6p^2 + p^3)$. Please note that instead of a polynomial basis fulfilling $0 \leq \rho_\xi + \rho_\eta + \rho_\zeta \leq p$ the complete set of polynomials such that $0 \leq \max(\rho_\xi, \rho_\eta, \rho_\zeta) \leq p$ can be used for discretization. However, this strategy turned out to be less efficient as the number of resulting vector entries then grows faster ($n_c(p) = (p+1)^3$) resulting in less granular refinement steps.

In order to finally fit the basis polynomials to the underlying signed distance function the integral equation describing the coefficient vector (cf. Equation (5)) has to be evaluated. Unfortunately, the only information on the properties of the integrand we have is that it is a continuous (but not necessarily smooth) function. Common approaches for the numerical integration of a-priori unknown functions include locally or globally adaptive, multi-dimensional numerical integration rules, e.g. adaptive Gauss quadrature or Monte-Carlo integration using importance sampling. However, the convergence of these methods for the application to Equation (5) may suffer from two major 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. Possible sources for these 'kinks' are sharp features in the underlying geometry or ambiguities in the signed distance function when the closest point on the surface is not unique. An example for the second case is the non-differentiability in the signed distance function at the center of a sphere where all surface points have the exact same distance to its center. This, finally, results in a large number of expensive Φ evaluations during numerical integration. In order to compute the integral sufficiently well and in an acceptable amount of time, we heuristically approximate it using multi-dimensional Gauss-Legendre 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 that we are able generate very accurate SDFs using the described strategy.

3.4 Hierarchical h-Refinement

In contrast to increasing the polynomial order of the approximation spatial subdivision can be an effective alternative for refining the SDF. Especially in regions where the underlying signed distance function is not smooth and therefore has low regularity h-adaption is known to be more effective (cf. [36]). A simple but very reasonable explanation for this is that the very smooth polynomials are not suitable to represent 'kinks' in the function while more low-order polynomials are better at capturing these features. In order to realize the spatial subdivision we maintain an octree for each of the base cells. After subdividing a cell into eight subcells corresponding to the next octree level we again fit polynomials to the exact signed distance function by means of solving Equation (5) and reject the coarse approximation. At this point we would like to stress the fact that the coarser approximation was not unnecessary as it is essential for error estimation and the decision whether to apply h- or p-refinement in the further process.

3.5 Error Estimation

In order to steer the error-driven refinement process we need to compute the discretization error ϵ_e over the domain of each individual cell e. Theoretically, it is possible to directly approximate the exact quadratic error $R_e(\mathbf{c}_e)$ using numerical integration since we are able to evaluate Φ at any point on the domain. This, however, results in a significant computational effort for two reasons. Firstly, exact signed distance evaluations are expensive because they require to traverse the acceleration data structure and to compute geometric distances to multiple triangles. Therefore, we aim to keep the number of Φ evaluations to a minimum. Secondly, a numerical computation of $R_e(\mathbf{c}_e)$ with sufficient accuracy is hard as the integrand is general locally non-smooth. Therefore, static numerical integration rules result in poor accuracy while adaptive techniques require an unacceptably large number of function evaluations. Please note that we initially intended to approximate the exact error, but discovered that neither accuracy nor performance were acceptable.

As a robust alternative, we estimate the cell-wise 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 $p - 1$:

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

where $\mathbf{c}_e^* = \{c_e^{\boldsymbol{\rho}^*}\}$ and $\mathbf{P}_e = \{P_e^{\boldsymbol{\rho}^*}\}$ are coefficient and polynomial vector of the embedded approximation of order $p - 1$ of cell e with $0 \leq \rho_\xi^* + \rho_\eta^* + \rho_\zeta^* \leq p - 1$. Due to the hierarchical basis the $p - 1$ approximation is directly embedded in the current approximation. Moreover, the integral error measure can be evaluated analytically as all coefficients can be factored out while the remaining integral factors become either exactly one or zero as a consequence of the orthonormal basis. This finally results in a simple sum over the squared constant coefficients of the degree p basis polynomials. Please note that using smaller p approximations for a-posteriori error estimation is common practice in the finite element community and has proven to be an effective approach (cf. [36]).

3.6 Construction Algorithm

In this section we will present our novel construction algorithm based on the previously introduced error estimator

Algorithm 1: hp-adaptive SDF construction.

```
   Data: $\Phi, n_\xi, n_\eta, n_\zeta, \tau, \Omega, p_{\max}, l_{\max}$
 1 $\epsilon \leftarrow 0$
 2 $n \leftarrow n_\xi n_\eta n_\zeta$
 3 pending ← priority_queue{}
 4 for $e \leftarrow 0$ to $n$ do
 5     fit_polynomial($e$, $\Phi$, 2)          // Fit polynomial of order 2 to each base cell $e$. Eq. (5)
 6     $\epsilon_e \leftarrow$ estimate_error($e$)         // Eq. (6)
 7     $\epsilon \leftarrow \epsilon + \epsilon_e$
 8     pending.push({$e$, $\epsilon_e$})
 9 end
10 while not pending.empty() and $\epsilon > \tau$ do
11     {$e$, $\epsilon_e$} ← pending.pop()
12     $\{p, l\}$ ← {degree($e$), level($e$)}
13     $\mu_e \leftarrow$ estimate_improvement_p($e$)
         // Eq. (8)
14     $\nu_e \leftarrow$ estimate_improvement_h($e$)
         // Eq. (9)
15     refine$_p \leftarrow p < p_{\max}$ and ( $l == l_{\max}$ or $\mu_e > \nu_e$ )
16     refine$_h \leftarrow l < l_{\max}$ and not refine$_p$
17     if refine$_p$ then
18         fit_polynomial($e$, $\Phi$, $p+1$)     // Eq. (5)
19         $\epsilon \leftarrow \epsilon - \epsilon_e$
20         $\epsilon_e \leftarrow$ estimate_error($e$)     // Eq. (6)
21         $\epsilon \leftarrow \epsilon + \epsilon_e$
22         pending.push({$e$, $\epsilon_e$})
23     end
24     if refine$_h$ then
25         children ← subdivide($e$)         // Octree subdiv.
26         $\epsilon \leftarrow \epsilon - \epsilon_e$
27         for $j \in$ children do
28             fit_polynomial($j$,$p$)          // Eq. (5)
29             $\epsilon_j \leftarrow$ estimate_error($j$)     // Eq. (6)
30             $\epsilon \leftarrow \epsilon + \epsilon_j$
31             pending.push({$j$, $\epsilon_j$})
32         end
33     end
34 end
```

and refinement strategies. In general, the approach can be interpreted as an error-driven globally adaptive construction following a top-down strategy. The abstract procedure is outlined in Algorithm 1 and we will guide the reader through each step of the construction process.

The algorithm expects the exact signed distance function Φ, an initial grid consisting of $n_\xi \times n_\eta \times n_\zeta$ cells on a rectangular domain Ω, the maximum refinement degree $p_{\max}$, the maximum octree depth $l_{\max}$, and the target error threshold τ as input. Further, the global error on the domain will be tracked using the total error ϵ in the course of the construction (cf. line 1).

The main idea of the construction algorithm is to maintain a priority queue which yields the index of the cell contributing the largest individual error in each iteration as a candidate for refinement. In line 1 to 3 we initialize the total error variable ϵ and the priority queue and compute the total number of base cells in the coarse initial grid. In the initialization loop (lines 4 to 9) we fit a polynomial of degree two to the underlying signed distance function on each individual cell, estimate the contributed error, 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 lines 10 to 34. The loop refines the discretization until the error falls below the target error threshold τ and as long as refinable cells exist. After retrieving the element contributing the highest individual error from the priority queue, we have to decide whether to spatially subdivide 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 either p- or h-adaption yields. This is done by individually applying both refinement strategies and estimating the remaining error on the h-adaption induced subcells. Following this strategy, we developed the following hp-decision criterion:

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

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

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

where $\mathcal{C}_e$ is the set of child cells resulting from the octree subdivision of e and $\alpha > 0$ an error scaling parameter. The error improvement per additional degree of freedom μ_e for p-refinement is computed based on the scaled error of the $(p+1)$-polynomial defined on the coarse cell. Analogously, a measure ν_e for the improvement corresponding to h-refinement is computed based on the scaled maximum error of the spatially subdivided order p polynomial measured on each of the subdomains on the finer octree level. The criterion decides in favor of a p-adaption if the former improvement is greater than the latter. 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. Moreover, as the used error measure is degree based the algorithm tends to underestimate the remaining error for p-refinement. Therefore, we additionally bias the decision towards h-adaption using $\alpha = 8$ in order to counteract over-refinement in p-direction. 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 would force at least the same degree on the subcells resulting from subsequent h-adaptions. Consequently, many unnecessary degrees of freedom arise leading to high memory consumption and computational effort for both construction and interpolation. At this point we would like to point out that the strategy to scale the p-error estimate for balancing the refinement is also common practice in the field of hp-adaptive finite element analysis (cf. [36]). Finally, lines 18-22 and 25-32 describe how we increase the polynomial degree and spatially subdivide the current cell, respectively. Furthermore, the total error is updated and the resulting cells' indices with the respective individual errors are inserted into the priority queue.

Please note that we accumulate the total error over the whole construction process. To avoid numerical errors due to the accumulation we store the error estimate of each individual cell and recompute the total residual $\epsilon = \sum_e \epsilon_e$ every 1000 iterations.

3.7 Nearness Weighting

For some applications of SDFs a comparably higher accuracy near the object's surface may be desired while regions far away from the surface are less interesting. We propose a weighting factor to compute a new transformed nearness weighted error estimate

$$\epsilon_e^* = \kappa_e \epsilon_e \tag{10}$$

that will artificially decrease the hp-refinement in regions far away from the object's surface. In order to find a suitable cell-wise coefficient κ_e a measure encoding the distance to the object surface is required. Building on the fact that Φ provides the point-wise shortest distance to the surface the average distance represented by a cell e is $(1/V_e)\int_{\Omega_e}\Phi d\boldsymbol{\xi}$ where V_e denotes the cell volume. As we aim to avoid expensive Φ evaluations we use the current approximation f_e instead. On this basis we model a polynomial weighting factor

$$\kappa_e = \left(1 - \frac{1}{V_e d}\left|\int_{\Omega_e} f_e d\boldsymbol{\xi}\right|\right)^\theta, \tag{11}$$

where d and θ denote the construction domain's diagonal and weighting exponent, respectively. Additionally dividing the distance measure by d normalizes it and ensures that $\kappa_e \in [0,1]$ as long as the domain fully contains the object surface, i.e. $\mathcal{B} \subseteq \Omega$. 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 $[0,1]$. If an even stronger decreasing weighting is desired an exponential factor can be used instead:

$$\kappa_e = \exp\left(-\frac{\theta}{V_e d}\left|\int_{\Omega_e} f_e d\boldsymbol{\xi}\right|\right). \tag{12}$$

Please note that if nearness weighting is used, the criterion described by Equations (7)-(9) must be modified accordingly.

4 Enforcing Weak Continuity

The previously presented algorithm describes a method to fit multivariate polynomials to the underlying signed distance function for each individual cell such that the global error is minimized efficiently. However, we have no guarantee that the resulting SDF is continuous over shared faces (interfaces) of neighboring cells. This is usually not a problem if the target error τ is chosen sufficiently small. If, however, a rather coarse approximation is desired the jumps in the discretization may pose a problem. Therefore, we present an approach using quadratic programming to enforce weak continuity over cell interfaces, in this section.

Generally, it is possible to project the discontinuous discretization onto a conforming discrete space. A method for the construction of a conforming hp-adaptive discretization was for example proposed by Di Stolfo et al. [40]. Our cell-wise decoupled approximation could be projected onto such a conforming discrete space, e.g. using an L_2 projection. But since the polynomials are already optimally fit to the signed distance function with respect to the quadratic error measure R_e, we aim to stick to the initial discretization as much as possible. Therefore, such a projection would potentially result in an uncontrollable loss of accuracy in order to directly enforce continuity. In contrast, we gain control of how much of the initial accuracy is lost to improve continuity by weak enforcement of continuous transitions.

4.1 Interface Error Measure

We first define an integral error measure in order to quantify the discontinuity of the SDF between two neighboring cells. Let i be an interior face of the adaptive grid with i_l and i_r denoting the indices of its incident cells. Then, we define the integral error measure as

$$E_i(\mathbf{c}_{i_l}, \mathbf{c}_{i_r}) = \int_{\Gamma_i} \frac{1}{2}\left(f_{i_\text{l}} - f_{i_\text{r}}\right)^2 dA$$
$$= \frac{1}{2}\begin{pmatrix}\mathbf{c}_{i_\text{l}}^T & \mathbf{c}_{i_\text{r}}^T\end{pmatrix} \underbrace{\int_{\Gamma_i}\begin{pmatrix}\mathbf{P}_{i_l}\mathbf{P}_{i_l}^T & -\mathbf{P}_{i_l}\mathbf{P}_{i_r}^T \\ -\mathbf{P}_{i_r}\mathbf{P}_{i_l}^T & \mathbf{P}_{i_r}\mathbf{P}_{i_r}^T\end{pmatrix} dA}_{\mathbf{M}_i} \begin{pmatrix}\mathbf{c}_{i_\text{l}} \\ \mathbf{c}_{i_r}\end{pmatrix}, \tag{13}$$

where $\Gamma_i = \Omega_{i_\text{l}} \cap \Omega_{i_r}$ and where dA denotes the integration variable for surface integration. The matrix integral $\mathbf{M}_i$ in Equation (13) can be exactly evaluated using Gaussian quadrature of corresponding order. Moreover, due to the chosen polynomial basis $\mathbf{M}_i$ can be evaluated analytically when the cells adjacent to an interface have the same cell size and therefore correspond to the same h-refinement depths. For an elaboration on how the analytic solution to the integral can be computed in this case we would like to refer the reader to Appendix A. Further, we define the global discontinuity error as sum of the cell-individual errors, i.e. $E = \sum_{i\in\mathcal{I}} E_i$ where $\mathcal{I}$ denotes the set of all inner faces.

4.2 Regularization

It is obvious that an minimization of the error energy E in the global coefficient vector $\mathbf{c}$ minimizes the "jumps" in the discretization. However, the solution of the optimization problem is not unique as there is an infinite number of global coefficient vectors minimizing the function. Therefore, we need to regularize the problem to guarantee uniqueness and to moreover find a meaningful solution.

In order to regularize the problem, we want to keep the optimized solution as close to the initial discretization as possible. Therefore, we define a per-cell regularization energy

$$\begin{aligned}\Psi_e(\mathbf{c}_e, \mathbf{c}'_e) &= \int_{\Omega_e} \frac{1}{2} \left(f_e - f'_e\right)^2 d\boldsymbol{\xi} \\ &= \frac{1}{2} \left(\mathbf{c}_e - \mathbf{c}'_e\right)^T \underbrace{\int_{\Omega_e} \mathbf{P}_e \mathbf{P}_e^T d\boldsymbol{\xi}}_{\mathbf{I}} \left(\mathbf{c}_e - \mathbf{c}'_e\right) \\ &= \frac{1}{2} \left(\mathbf{c}_e - \mathbf{c}'_e\right)^T \left(\mathbf{c}_e - \mathbf{c}'_e\right),\end{aligned} \tag{14}$$

where f'_e, $\mathbf{c}'_e$ and $\mathbf{I}$ denote the initial (discontinuous) approximation, the according initial coefficient vector and the identity matrix, respectively. Once again, choosing an orthonormal polynomial basis pays off as the integral part of the regularization energy vanishes. The regularization energy over the whole domain is then defined by $\Psi = \sum_{e\in\mathcal{E}} \Psi_e = \frac{1}{2}(\mathbf{c} - \mathbf{c}') \cdot (\mathbf{c} - \mathbf{c}')$ where $\mathcal{E}$ denotes the set of cells contained in the grid.

4.3 Optimization

Using the previously defined interface error measure and regularization energy we formulate a convex minimization problem in the coefficient vector $\mathbf{c}$ in order to compute an SDF with improved continuity

$$\min_{\mathbf{c}} \left(E(\mathbf{c}) + \beta\Psi(\mathbf{c}, \mathbf{c}')\right) \tag{15}$$

$$\Leftrightarrow$$

$$\left(\mathbf{M} + \beta\mathbf{I}\right)\mathbf{c} = \beta\mathbf{c}', \tag{16}$$

where $\beta > 0$ denotes a regularization parameter and $\mathbf{c}'$ the coefficient vector of the discontinuous approximation. Furthermore, $\mathbf{M} = \partial^2 E/\partial\mathbf{c}^2$ is the second derivative of the interface error term and can be assembled from the block matrices $\mathbf{M}_i$. An interpretation of β is simple as we get a more or less arbitrary (but continuous) SDF for $\beta \to 0$ whereas we converge towards the initial discretization induced by the initial coefficient vector $\mathbf{c}'$ for $\beta \to \infty$. As the objective function is quadratic in the coefficient vector $\mathbf{c}$ and convex, the (unique) solution to the minimization problem is equivalent to the solution of the linear equation system given by Equation (16). The problem can finally be solved using an arbitrary (sparse) linear solver. As the number of coefficients, i.e. degrees of freedom, is very large for detailed SDFs the memory requirements for direct solvers using matrix factorization can easily exceed the available memory capacity. Therefore, we used a conjugate gradient descent solver in combination with an incomplete Cholesky factorization for preconditioning in our results.

5 Results and Discussion

All computations in this section were carried out on two Intel Xeon E5-2697 processors with 2.7GHz, 30MB Cache, 12 cores per processor and 64GB RAM. We parallelized the SDF construction with Intel TBB and used 48 threads in all computations. All deformable and rigid body simulations with contacts are based on the approaches proposed by Bender et al. [41] and Deul et al. [42] implemented in the open-source library *PositionBasedDynamics* [43].

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 2. 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.

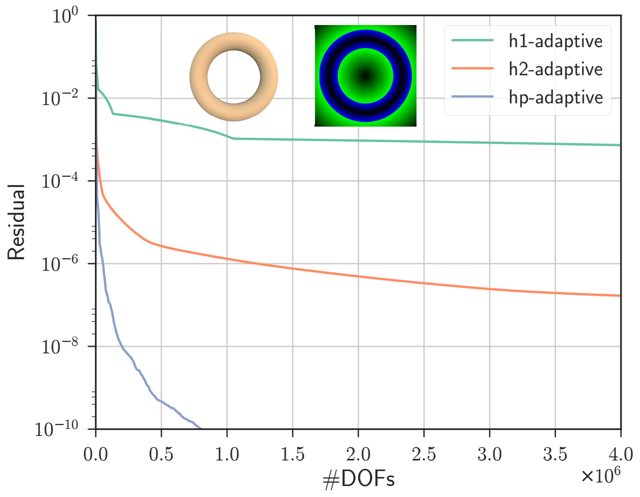


Fig. 3: Comparison of the convergence for a torus model of our *hp*-adaptive method with a pure octree-subdivision using linearly and quadratically fitted polynomials. #DOF encodes the number of polynomial coefficients required to enforce the corresponding residual.

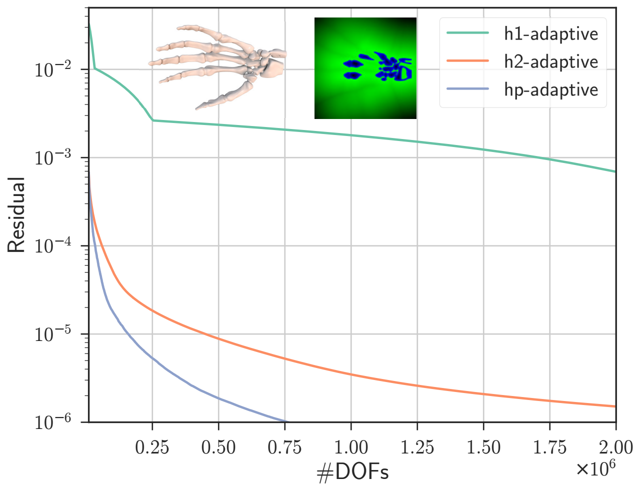


Fig. 4: Convergence study of SDF construction for a skeleton hand.

Convergence and Refinement Analysis. Figures 3 and 4 illustrate the convergence graph during SDF construction for a torus and a skeleton hand model. The number of

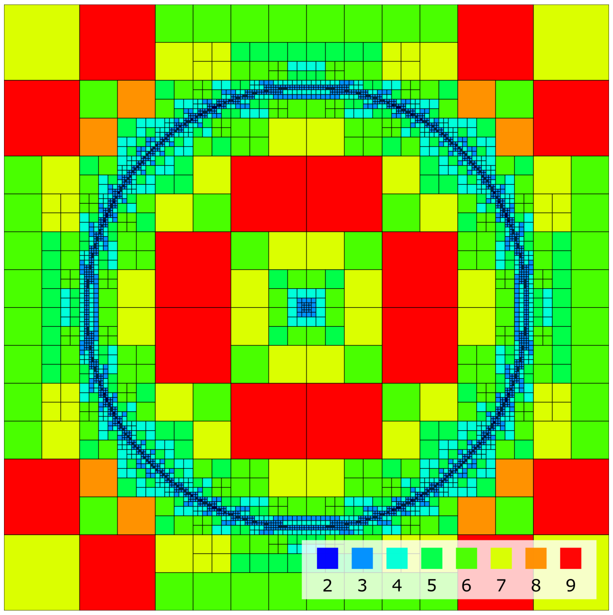


Fig. 5: Torus degree plot. Visualization of octree cells with corresponding polynomial degrees according to the legend depicted in the figure.

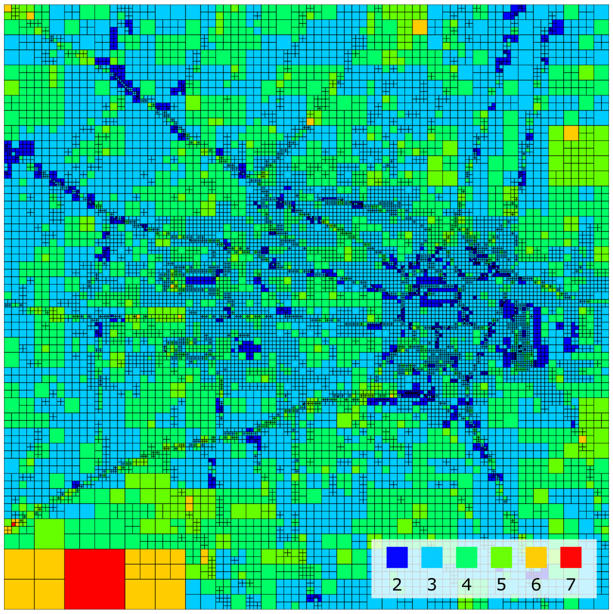


Fig. 6: Skeleton hand degree plot. Visualization of octree cells with corresponding polynomial degrees according to the legend depicted in the figure.

required coefficients ($\#DOF$) is shown on the abscissa while the estimated error (residual) from Equation (6) is displayed on the ordinate in logarithmic scale. We compared our hp-adaptive approach to pure octree-subdivision with linear ($h1$-adaptive) and quadratic ($h2$-adaptive) 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', most visible in the curve of the $h1$-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 stress the fact that we constructed the polynomials in all cases using the fitting approach (cf. Equation (2)) which yields the optimal solution in terms of the measured error. Using the traditional approach of sampling distance values within each cell would yield even worse results for the $h1$- and $h2$-adaptions. For further investigation, we visualized the leaf cells and their polynomial degree for an example slice as depicted in Figures 5 and 6. It can be noticed that 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).

In order to demonstrate the effect of nearness weighting we additionally constructed an SDF for the skeleton hand with exponential nearness weighting with $\theta = 30$. The according degree plot is illustrated in Figure 7. The result clearly shows how regions close to the object surface are strongly refined while regions far away correspond to a coarse discretization in comparison to the unweighted result in Figure 6. Moreover, the reduced field required approximately 72% less memory compared to the unweighted result.

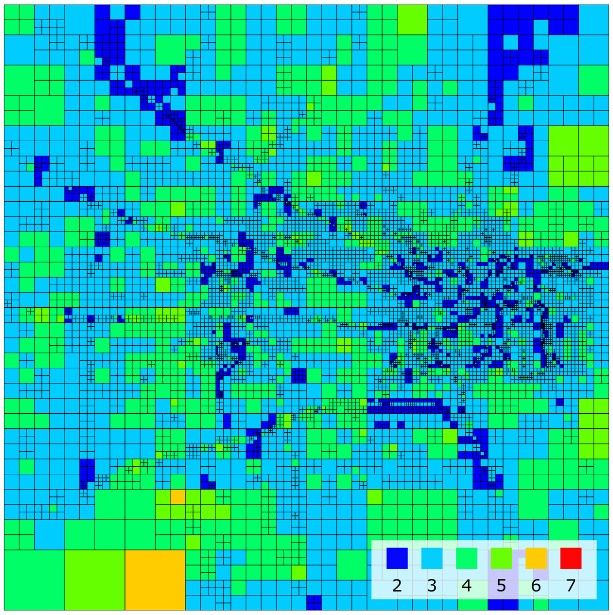


Fig. 7: Skeleton hand degree plot of exponentially nearness weighted SDF with $\theta = 30$.

Construction Statistics. Table 2 summarizes statistics on the input triangle meshes and the according SDF construction results. All input meshes were scaled to the unit box $[-1, 1]^3$ and the construction domain was enlarged by 10% while we globally chose $p_{\max} = 30$ and $l_{\max} = 10$. Additionally, we used polynomial nearness weighting with $\theta = 4$ for all examples. The mesh column shows the name of the mesh and its number of vertices and faces. The SDF column further contains the resolution of the initial construction grid, the time required for construction, the number of the resulting octree leaf cells, the distributions of degree and octree depth,

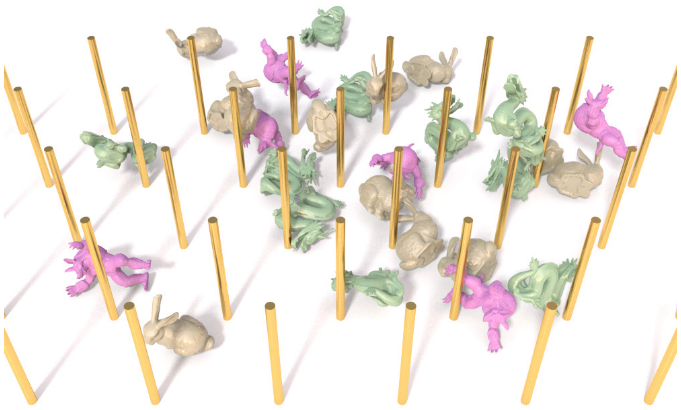


Fig. 8: Complex rigid and deformable bodies slide down an inclined plane with obstacles.

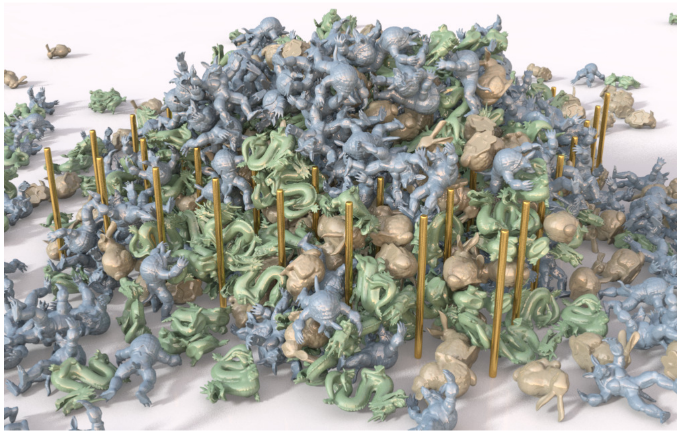


Fig. 9: 800 rigid bodies fall onto a set of 64 poles having several thousand contacts per simulation step.

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 essentially consists of four arrays. The first two arrays contain the polynomial coefficients in double-precision and a prefix-sum stating at which index 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 indices of the corresponding octree nodes stored in the last array.

Collision Detection. We used our novel SDF representation in several physics-based simulations as collision detector in order to demonstrate the practical relevance of our approach. To realize the collision detector we additionally point-sampled the involved objects' surfaces and organized the samples in a bounding-sphere hierarchy (BSH). The BSH was constructed and traversed similar to the approach described by Sanchez et al. [9]. As the SDF construction process can be interpreted as preprocessing step all generated SDFs were serialized. Finally, the relevant SDFs were loaded prior to each simulation scenario and tested against the point samples of the other objects. We conducted two experiments, depicted in Figures 8 and 9, where several dynamic bodies collide with a static grid of poles. Each

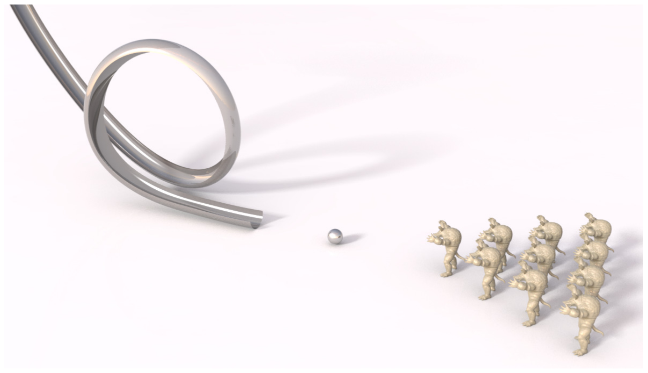


Fig. 10: Dynamic simulation of a marble run with subsequent armadillo bowling.

body was sampled with approximately 10k sample points in order to perform the distance queries. In order to reduce the number of distance queries we implemented a bounding volume hierarchy (BVH) with bounding spheres. We further accelerated the collision tests by parallelization of the collision test for each object pair. In the first scenario (Figure 8) deformable and rigid bodies slide on an inclined plane. While SDFs for the rigid dragons and bunnies was used, collisions of the deformable armadillos were 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. After simulating a time interval of 25 seconds the average and maximum number of contacts per step were 8007 and 15050, respectively. More than 11600 contacts per time step were observed in the finally resting state. The collision detection including signed distance field queries and BSH traversal required a computation time of 158 ms on average per time step. Here, we would like to stress the fact that an accurate detection based on geometric vertex-triangle and edge-edge tests would not have been feasible in a comparable computation time on the CPU as the scenario manages models with a total number of more than 132M triangles. In a further scenario, we simulated two finely structured bowls as illustrated in Figure 11 and 1 (left). In the first bowl a marble rolls on a helical groove whereas 1000 comparably small marbles were dropped 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 10 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 that required only a small amount of memory (cf. Table 2). Finally, we simulated a sheet of cloth covering the Stanford dragon as depicted in Figure 1 (right). The features of the dragon surface are still clearly visible as they are silhouetted against the sheet.

Continuity Optimization. We performed the post-processing optimization step for enforcing weak continuity on an SDF of the Stanford dragon ($\tau = 5 \times 10^{-7}$, exponential nearness weighting $\theta = 20$). As depicted in Figure 12 (left) we first generated the SDF with a moderate target error. While the discretization captures the macro-

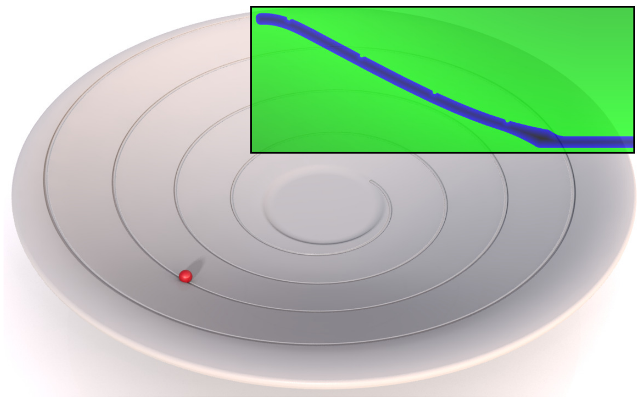


Fig. 11: Dynamic simulation of a marble following a highly-detailed, helix shaped groove in a bowl.

h-level	Query time $[ns]$
0	169
1	202
2	226
3	245
4	276
5	322
6	380
7	611
8	1157

(a)

p-level	Query time $[ns]$
1	169
2	194
3	238
4	276
5	336
6	397
7	483
8	577
9	697

(b)

TABLE 1: Performance measurements. A single cell was uniformly refined using either p- or h-adaption. The resulting discrete SDF was then queried using random samples and the average time for a single query was computed.

scopic shape of the underlying geometry well, especially regions with high-frequent features may suffer from the generally discontinuous piecewise approximation. In order to improve the continuity of the field we optimized the SDF as explained in Section 4 with regularization parameter $\beta = 3$. The SDF consisted of $1.3M$ coefficients and the sparse solver matrix contained $178M$ non-zero entries. The construction of the matrix took $47s$ and the incomplete Cholesky decomposition took $58s$. Both procedures were not parallelized. The actual solve of the equation system with a conjugate gradient solver took $44s$ with 88 iterations using a parallelized implementation of the matrix vector product. The result illustrated in Fig. 12 (right) clearly shows that the approximation continuity is greatly improved while the general shape and features are maintained.

Improving the continuity of the field can also be beneficial when the SDF is applied in collision handling. The marble run scenario described in the previous paragraph was simulated using an unoptimized but very accurately discretized SDF (cf. entry in Table 2). We resimulated the scenario using a less accurate but optimized SDF ($\tau = 10^{-8}$, #cells $= 22.3k$, 4.36MB, $\beta = 8$, exponential nearness weighting with $\theta = 20$). Using this more compact but post-processed representation we were able to reproduce the scenario with the marble rolling smoothly until it reaches the end of the track and finally resulting in a dynamic simulation of comparable quality.

Distance Query Performance. We measured the time to query the distance using our discrete SDF. Therefore, we randomly sampled the field with several thousand points and averaged the resulting measured values. For the armadillo and structured bowl this resulted in approximately 4.76×10^{-4} ms and 7.16×10^{-4} ms, respectively. If the SDF-gradient was additionally requested, the queries took 7.34×10^{-4} ms and 7.84×10^{-4} ms on average. In order to analyze the individual effect of h- or p-refinement on the query performance we measured the required query time for an SDF initially consisting of a single cell with linear polynomials that was zero to eight times h-refined or up to nine times p-refined. The results are depicted in Tables 1a and 1b. Thanks to the recursive form of the Legendre polynomials (cf. Equation (4)) their evaluation can be accelerated by reusing redundant terms from order 0 to order p during the distance evaluation as well as the gradient computation.

6 Conclusion

In this paper, a novel method to hierarchically construct higher-order SDFs was presented. The approach efficiently fits shifted, orthonormalized Legendre polynomials to the exact signed distance function in a hierarchical manner. Spatial adaptivity was implemented using octree subdivision. We developed a new hp-decision criterion using degree-based error estimation in order to steer the adaption in the refinement process. By comparing our method to traditional purely spatially adaptive approaches we demonstrated that our criterion-controlled refinement algorithm greatly improves convergence. We further introduced a nearness weighting approach modifying the estimated error in order to focus the refinement on regions close to the surface of the underlying object. Moreover, an optimization-based post-processing step was proposed that weakly enforces continuity over element interfaces. We demonstrated that our method is able to produce very accurate SDFs for complex geometries while consuming only a small amount of memory. Moreover, the SDFs were shown to be very well-suited for the detection of contacts and collisions in physically-based simulations as they implicitly contain information about the penetration depth and contact normal.

The proposed technique also has limitations. In its current version the method only features isotropic refinement. However, we think that anisotropic adaption strategies could significantly reduce the memory requirements while still maintaining an accurate approximation. This could potentially be realized by using a k-d-tree for spatial refinement and axis-dependent degree adaption for p-refinement. A fundamental problem is that for two-dimensional objects embedded in three dimensions, such as cloth or shells, inside and outside cannot be distinguished. For that reason our method is not useful for handling cloth-cloth contacts or self-intersections of cloth. Collisions between touching surfaces – a scenario which occurs in cutting or fracture simulations – cannot easily be handled by simply discretizing the signed distance. For that reason, we plan to investigate

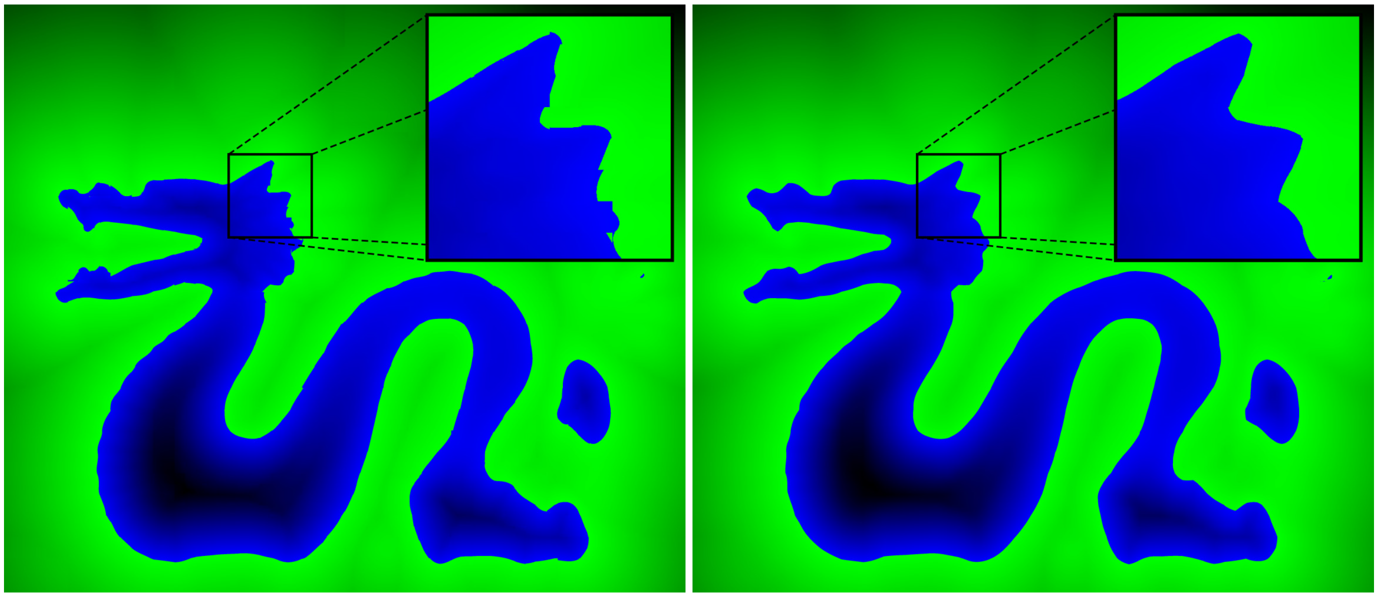


Fig. 12: Discretization of the Stanford dragon (approx. 5.4MB) with target error $\tau = 5 \times 10^{-7}$ and exponential nearness weighting ($\theta = 20$). Left: SDF with subtle artifacts due to discontinuities on element interfaces. Right: SDF after continuity optimization with regularization parameter $\beta = 3$.

if the proposed method can be extended to a non-manifold representation similar to the work of Mitchell et al. [4].

Appendix A
Analytic Evaluation of the Interface Error Matrix

The matrix integral $\mathbf{M}_i$ introduced in Section 4.1 can be evaluated analytically when the two neighboring cells correspond to the same h-refinement depth. Each block contained in $\mathbf{M}_i$ consists of the outer product of two polynomial basis vectors.

Let $\mathbf{P} = \{P_i\}$ and $\mathbf{P}^* = \{P_j^*\}$ be the normalized, shifted Legendre polynomial basis vectors of two neighboring cells, respectively. Since the polynomial basis vectors were constructed using a tensor-product, each component can be multiplicatively decomposed into factors only dependent on a single coordinate, i.e. $P_i(\xi, \eta, \zeta) = P_{i,\xi}(\xi)P_{i,\eta}(\eta)P_{i,\zeta}(\zeta)$. Without loss of generality, let us assume that the normal of the common face points in ξ-direction. Let us further assume that both cells have the same size and are aligned with each other in η- and ζ-direction. Each block entry in $\mathbf{M}_i$ is constructed using an integral over the outer product of the basis vectors (cf. Equation (13)):

$$\int_\Gamma \mathbf{P}\mathbf{P}^{*T} dA = \int_{a^\eta}^{b^\eta} \int_{a^\zeta}^{b^\zeta} \mathbf{P}\mathbf{P}^{*T} d\eta d\zeta.$$

Given the previously stated assumptions, we can analytically evaluate the entry in the ith row and jth column of the matrix block as follows:

$$\begin{aligned}\left[\int_\Gamma \mathbf{P}\mathbf{P}^{*T} dA\right]_{ij} &= \int_{a^\eta}^{b^\eta} \int_{a^\zeta}^{b^\zeta} P_{i,\xi}P_{i,\eta}P_{i,\zeta}P_{j,\xi}^*P_{j,\eta}^*P_{j,\zeta}^* d\eta d\zeta \\ &= P_{i,\xi}P_{j,\xi}^* \underbrace{\int_{a^\eta}^{b^\eta} P_{i,\eta}P_{j,\eta}^* d\eta}_{=\delta_{ij}} \underbrace{\int_{a^\zeta}^{b^\zeta} P_{i,\zeta}P_{j,\zeta}^* d\zeta}_{=\delta_{ij}} \\ &= P_{i,\xi}P_{j,\xi}^*\delta_{ij}.\end{aligned}$$

Acknowledgments

This work is supported by the *German Research Foundation (DFG)* under contract numbers BE 5132/4-1 the 'Excellence Initiative' of the German Federal and State Governments and the Graduate School of Computational Engineering at TU Darmstadt. We would further like to thank Manuel Scholz for his kind support in providing geometric models for the results.

References

[1] F. Calakli and G. Taubin, "SSD: Smooth Signed Distance Surface Reconstruction," *Computer Graphics Forum*, vol. 30, no. 7, pp. 1993–2002, 2011.
[2] O. Jamriška, "Interactive Ray Tracing of Distance Fields," in *Central European Seminar on Computer Graphics*, vol. 2, pp. 1–7, 2010.
[3] S. F. Frisken and R. N. Perry, "Designing with Distance Fields," in *ACM SIGGRAPH Courses*, 2006, pp. 60–66.
[4] N. Mitchell, M. Aanjaneya, R. Setaluri, and E. Sifakis, "Non-manifold Level Sets: A multivalued implicit surface representation with applications to self-collision processing," *ACM Transactions on Graphics*, vol. 34, no. 6, pp. 247:1–247:9, 2015.
[5] H. Xu and J. Barbič, "Signed Distance Fields for Polygon Soup Meshes," in *Graphics Interface*, 2014, pp. 1–7.
[6] S. F. Frisken, R. N. Perry, A. P. Rockwood, and T. R. Jones, "Adaptively sampled distance fields: a general representation of shape for computer graphics," *ACM Transactions on Graphics*, vol. 26, no. 3, pp. 249–254, 2000.

[7] A. Rosenfeld and J. L. Pfaltz, "Sequential Operations in Digital Picture Processing," *Journal of the ACM*, vol. 13, no. 4, pp. 471–494, 1966.

[8] M. Jones, J. Baerentzen, and M. Sramek, "3D distance fields: a survey of techniques and applications," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 4, pp. 581–599, 2006.

[9] M. Sanchez, O. Fryazinov, and A. Pasko, "Efficient Evaluation of Continuous Signed Distance to a Polygonal Mesh," in *Spring Conference on Computer Graphics*, 2012, pp. 101–108.

[10] R. N. Perry and S. F. Frisken, "Kizamu: A System for Sculpting Digital Characters," in *Computer Graphics and Interactive Techniques*, 2001, pp. 47–56.

[11] J. A. Bærentzen, "Manipulation of volumetric solids with applications to sculpting," Phd thesis, Technical University of Denmark, 2002.

[12] K. Erleben and H. Dohlmann, "Signed Distance Fields Using Single-Pass GPU Scan Conversion of Tetrahedra," in *GPU Gems 3*. Addison-Wesley Professional, 2008, ch. 34, pp. 741–763.

[13] K. Museth, "VDB: High-resolution Sparse Volumes with Dynamic Topology," *ACM Transactions on Graphics*, vol. 32, no. 3, pp. 27:1–27:22, 2013.

[14] J. Huang, Y. Li, R. Crawfis, S. C. Lu, and S. Y. Liou, "A complete distance field representation," in *IEEE Visualization*, 2001, pp. 247–254.

[15] T. Ju, F. Losasso, S. Schaefer, and J. Warren, "Dual Contouring of Hermite Data," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 339–346, 2002.

[16] H. Qu, N. Zhang, R. Shao, A. Kaufman, and K. Mueller, "Feature preserving distance fields," in *IEEE Symposium on Volume Visualization and Graphics*, 2004, pp. 39–46.

[17] A. J. Baerentzen, "Robust Generation of Signed Distance Fields from Triangle Meshes," *International Workshop on Volume Graphics*, pp. 167–239, 2005.

[18] J. Wu and L. Kobbelt, "Piecewise Linear Approximation of Signed Distance Fields," in *Vision, Modeling and Visualization*, 2003, pp. 513–520.

[19] M. W. Jones, "Distance field compression," *Journal of WSCG*, vol. 12, no. 2, pp. 199—-204, 2004.

[20] M. A. Otaduy, N. Jain, A. Sud, and M. C. Lin, "Haptic display of interaction between textured models," in *IEEE Visualization*, 2004, pp. 297–304.

[21] K. Moustakas, D. Tzovaras, and M. G. Strintzis, "SQ-Map: Efficient Layered Collision Detection and Haptic Rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 1, pp. 80–93, 2007.

[22] D. M. Kaufman, S. Sueda, and D. K. Pai, "Contact trees: Adaptive Contact Sampling for Robust Dynamics," in *ACM SIGGRAPH Sketches*, 2007.

[23] L. Glondu, S. C. Schvartzman, M. Marchal, G. Dumont, and M. A. Otaduy, "Efficient Collision Detection for Brittle Fracture," in *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 2012, pp. 285–294.

[24] H. Xu, Y. Zhao, and J. Barbič, "Implict Multibody Penalty-based Distributed Contact," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 9, pp. 1266–1279, 2014.

[25] R. Bridson, S. Marino, and R. Fedkiw, "Simulation of clothing with folds and wrinkles," in *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*. Eurographics Association, 2003, pp. 28–36.

[26] A. Fuhrmann, G. Sobottka, and C. Groß, "Distance Fields for Rapid Collision Detection in Physically Based Modeling," in *Computer Graphics and Vision*, 2003, pp. 1–8.

[27] J. Barbič and D. L. James, "Six-DoF Haptic Rendering of Contact Between Geometrically Complex Reduced Deformable Models," *IEEE Transactions on Haptics*, vol. 1, no. 1, pp. 39–52, 2008.

[28] F. Faure, S. Barbier, J. Allard, and F. Falipou, "Image-based collision detection and response between arbitrary volume objects," in *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 2008, pp. 155–162.

[29] J. Allard, F. Faure, H. Courtecuisse, F. Falipou, C. Duriez, and P. G. Kry, "Volume contact constraints at arbitrary resolution," *ACM Transactions on Graphics*, vol. 29, no. 4, pp. 82:1–82:10, 2010.

[30] B. Wang, F. Faure, and D. K. Pai, "Adaptive image-based intersection volume," *ACM Transactions on Graphics*, vol. 31, no. 4, pp. 97:1–97:9, 2012.

[31] H. Xu and J. Barbič, "Continuous Collision Detection Between Points and Signed Distance Fields," in *Proceedings of Virtual Reality Interactions and Physical Simulations (VRIPhys)*, pp. 1–7, 2014.

[32] A. McAdams, Y. Zhu, A. Selle, M. Empey, R. Tamstorf, J. Teran, and E. Sifakis, "Efficient elasticity for character skinning with contact and collisions," *ACM Transactions on Graphics*, vol. 30, no. 4, pp. 37:1–37:12, 2011.

[33] I. Babuška, B. A. Szabo, and I. N. Katz, "The p-Version of the Finite Element Method," *SIAM Journal on Numerical Analysis*, vol. 18, no. 3, pp. 515–545, 1981.

[34] I. Babuška and M. Suri, "Error estimates for the combined h and p versions of finite element method," *Numerische Mathematik*, vol. 37, pp. 252–277, 1981.

[35] I. Babuska and M. Suri, "The p and $h - p$ Versions of the Finite Element Method, Basic Principles and Properties," *SIAM Review*, vol. 36, no. 4, pp. 578–632, 1994.

[36] W. F. Mitchell and M. A. McClain, "A Comparison of hp -Adaptive Strategies for Elliptic Partial Differential Equations," *ACM Transactions on Mathematical Software*, vol. 41, no. 1, pp. 1–39, 2014.

[37] A. Schmidt and K. G. Siebert, "A posteriori estimators for the h-p version of the finite element method in 1D," *Applied Numerical Mathematics*, vol. 35, no. 1, pp. 43–66, 2000.

[38] P. Houston, B. Senior, and E. Süli, "Sobolev Regularity Estimation for hp-Adaptive Finite Element Methods," in *Numerical Mathematics and Advanced Applications*. Springer Milan, 2003, pp. 631–656.

[39] J. A. Bærentzen and H. Aanæs, "Signed distance computation using the angle weighted pseudonormal," *IEEE Transactions on Visualization and Computer Graphics*, vol. 11, no. 3, pp. 243–253, 2005.

[40] P. Di Stolfo, A. Schrder, N. Zander, and S. Kollmannsberger, "An easy treatment of hanging nodes in hp-finite elements," *Finite Elements in Analysis and Design*, vol. 121, pp. 101–117, 2016.

[41] J. Bender, D. Koschier, P. Charrier, and D. Weber, "Position-Based Simulation of Continuous Materials," *Computers & Graphics*, vol. 44, no. 0, pp. 1–10, 2014.

[42] C. Deul, P. Charrier, and J. Bender, "Position-based rigid-body dynamics," *Computer Animation and Virtual Worlds*, 2014.

[43] J. Bender, "PositionBasedDynamics Library," https://github.com/InteractiveComputerGraphics/\PositionBasedDynamics, 2017.

Dan Koschier is a PhD student at RWTH Aachen University and associate member of the Graduate School of Computational Engineering, TU Darmstadt. He received his MSc degree in Computational Engineering from TU Darmstadt, in 2014. His research interests include physically-based simulation of deformable solids, cutting, fracture and fluids.

Crispin Deul received his MSc in computer science from TU Darmstadt in 2011. He is a PhD student at the Graduate School of Excellence Computational Engineering at TU Darmstadt. His research focuses on interactive simulation techniques including motion control, physically-based skinning, and position-based dynamics.

Magnus Brand is a PhD student at RWTH Aachen University. He received his MSc in computer science from TU Darmstadt in 2016. His research interests include the physically based simulation of deformables and fluids using Eulerian discretization approaches.

Jan Bender is professor of computer science and leader of the Computer Animation Group at RWTH Aachen University. He received his diploma, PhD and habilitation in computer science from the University of Karlsruhe. His research interests include interactive simulation methods, multibody systems, deformable solids, fluid simulation, collision handling, cutting, fracture, GPGPU and real-time visualization.

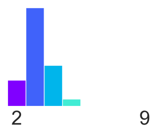
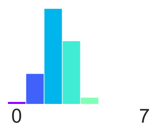
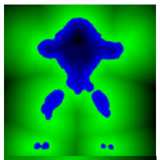
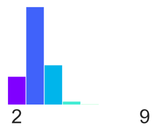
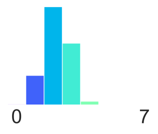
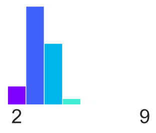
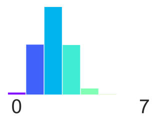
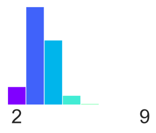
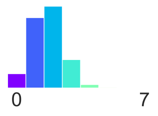
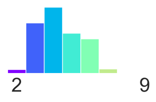
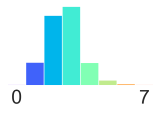
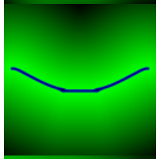
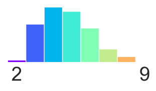
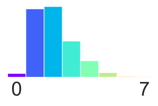
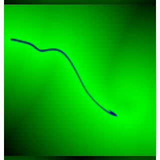
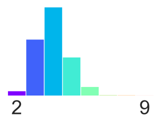
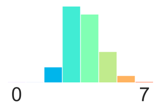
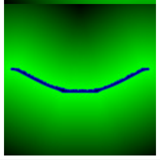
Mesh			Signed Distance Field							
Name	#Vert.	#Faces	Base Grid	Constr. Time	#Cells	Degree	Depth	ϵ, τ	Memory	Field
Armadillo	173k	346k	6^3	270s	71k			10^{-6}	10.6 MB	
Bunny	34.8k	69.6k	6^3	301s	55k			10^{-6}	8.4 MB	
Dragon	40k	80k	10^3	1245s	299k			10^{-7}	46.9 MB	
Hand	66.2k	132.5k	10^3	833s	159k			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 2: Construction statistics. The mesh columns show object names and corresponding number of vertices and faces. The SDF columns contain 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.