Vision, Modeling, and Visualization (2011)
Peter Eisert, Konrad Polthier, and Joachim Hornegger (Eds.)

Direct Visualization of Particle-Partition of Unity Data

M. Üffinger[1], M. A. Schweitzer[2], F. Sadlo[1], and T. Ertl[1]

[1]Visualization Research Center Universität Stuttgart (VISUS), Germany
[2]Institute of Parallel and Distributed Systems, Universität Stuttgart, Germany

Abstract

Direct visualization of higher-order data provides manifold advantages over the traditional approach, which is based on resampling and subsequent visualization by interpolation-based techniques. Most important, it avoids excessive computation and consumption of memory, and prevents artifacts by pixel-accurate visualization at interactive rates. This work addresses particle-partition of unity simulation data, where fields are modeled both using cell-based analytic representations together with enrichment functions centered at individual points. This combination of bases allows for superior simulation convergence rates and is able to capture high field variations with comparably small sets of basis functions. In this paper we propose direct visualization of such data from 2D simulations, providing accurate insight. We additionally visualize solver performance, allowing for more directed simulation design, and exemplify our technique using a GPU-based prototype on crack simulation examples.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

1. Introduction

A large part of scientific visualization deals with discretized field data. To account for their continuous nature, visualization techniques typically involve interpolation, rendering the data at least piecewise continuous. There is a multitude of available interpolation schemes—and often the choice would depend on the domain where the data originate from and on the concepts involved in the respective visualization technique. However, it is nowadays still common practice to neglect these details and to follow a simple but generic approach, often resulting in visualization techniques based on tensor-product linear interpolation.

There is, however, an alternative visualization approach for such data: direct visualization of piecewise analytic data. Instead of being present as a set of samples with some connectivity and leaving the interpolation question open, they provide a concise description of the field data. They can be based on scattered point sampling, such as in the case of smoothed particle hydrodynamics (SPH) or radial basis function (RBF) data where analytic kernel functions reside at discrete points, or may be based on grids such as higher-order finite element (FEM) or discontinuous Galerkin (DG) data where the field is analytically described for every cell of the grid. From a data-centric view, particle-partition of unity (PPUM) [Sch03, GS00] data, the topic of this paper, represent a combination of these two cases: they use a cell-based (local coordinate-based) analytic representation similar to higher-order FEM but at the same time use *enrichment functions* placed in a scattered manner in global coordinates. In this sense, our application relates to and uses ideas from direct visualization techniques presented for SPH data [SFBP09] as well as DG data [RCMG07, UFE10, SUP*11].

We show the utility of our interactive direct visualization technique by comparing it with the traditionally used resampling approach with subsequent visualization based on standard interpolation techniques. Further, we present dedicated visualization methods for investigating PPUM data, avoiding misleading visualizations, providing new insights in this type of data, and supporting efficient and effective PPUM simulation design. We exemplify our approach using crack simulation examples.

The paper is organized as follows: Section 2 refers to related work whereas Section 3 describes the PPUM simulation approach from the visualization point of view. Section 4 presents our interactive pixel accurate visualization method for 2D PPUM data, including details on its Open GL shader

implementation. Results are presented in Section 5 and the advantages of the method in comparison to traditional visualization techniques are discussed. Section 6 concludes the paper with a summary of the results and future work.

2. Related Work

Direct visualization of higher-order data has gained importance in recent years. Most work addressed either point-based data arising, e.g., from smoothed particle hydrodynamics, or grid-based data from, e.g., finite-element or discontinuous Galerkin simulations. Due to their basic difference in representation, these visualization techniques also employ different strategies. In the SPH context there is the work of Schindler et al. [SFBP09] regarding line-type feature extraction. One of the first FEM-related techniques are due to Gallagher [GN89] and Zumbusch [Zum94]. Haasdonk et al. [HOR*03] visualize polynomial higher-order field functions given on adaptively refined 2D grids. Another related approach is due to Nelson and Kirby, and Meyer et al. [NK06, MNKW07], who employ a raycasting and a particle-based technique to visualize isosurfaces in 3D flow simulations. More recently, in the field of discontinuous Galerkin flow simulations, Üffinger et al. presented a technique for volume rendering [UFE10], and Pagot et al. a method for isosurface extraction [PVS*10] and line-type feature extraction [POS*11]. Most techniques visualize polynomial field solutions only. In contrast, our technique targets the solution from generalized FEM simulations, which additionally feature arbitrary non-polynomial basis functions, e.g., discontinuous or singular functions.

3. Particle-Partition of Unity Method

The generalization of the classical finite element method is an active research field aimed to overcome the cumbersome issue of mesh-generation and to improve on the approximation properties especially for problems with micro structure, discontinuities, and singularities (see [FB10, Sch11a, DO96, BM96] and references therein). Since FEM basis functions are piecewise polynomial functions they are well-suited for the approximation of piecewise smooth functions only. Thus, adaptive mesh-refinement techniques must be employed in the FEM to attain acceptable convergence rates. In various generalizations of the FEM [BB99, BCO94] the restriction to piecewise polynomial shape functions is abolished, thereby allowing for an algebraic refinement of the approximation space by problem-dependent enrichment functions to account for specific behavior of the solution that is known a priori, e.g., by asymptotic analysis.

In fracture mechanics for instance we must cope with discontinuous displacement fields and stress distributions that are singular at the moving crack fronts. Here, the characteristic singularity is known analytically from an asymptotic expansion of the solution and the generalized finite element methods can utilize this information to substantially improve the efficient simulation of fracture processes. Note that this algebraic refinement approach however is not limited to analytic information but can also employ pre-computed handbook functions, or even experimental data.

The fundamental prerequisite for this so-called enrichment approach is the availability of a partition of unity (PU). In the extended finite element method (XFEM) [MDB99] or the generalized finite element method (GFEM) [SBC00] the employed PU comes from classical FEM shape functions whereas in the particle-partition of unity method the employed PU is constructed by a meshfree scattered data technique from independent points, as described in Section 3.1. The algebraic refinement of an approximation space obviously improves the approximation properties, however, it may also adversely affect the stability of the basis functions and thereby the efficient iterative solution of the arising linear system. In the PPUM these stability issues, which are observed in practice in the GFEM and XFEM, can be easily overcome via the use of a so-called flat top PU and a local preconditioning technique [Sch11b]. Moreover, there is a multilevel solver available for the PPUM [GS02] that can cope with arbitrary enrichment functions. Thus, the PPUM is currently the most stable and efficient approach to the generalization of the FEM. However, its implementation cannot be based on an available FEM code.

3.1. Data Model

In contrast to traditional FEM approaches, in PPUM, fields are primarily represented in a mesh-less manner, i.e., they are defined on a set of points $P = \{x_i \,|\, i = 1 \dots \hat{N}\}$ inside a domain Ω. The required continua are established by means of d-binary trees (we assume $d = 2$, i.e., quadtrees), constructed from a bounding-box $\mathcal{C}_\Omega \supset \Omega$ of P that is subdivided until each cell $\mathcal{C}_i$ with center (c_i^x, c_i^y) and size $(2h_i^x, 2h_i^y)$

$$\mathcal{C}_i = (c_i^x - h_i^x, c_i^x + h_i^x) \times (c_i^y - h_i^y, c_i^y + h_i^y)$$

associated with a leaf of the tree contains at most a single point $x_i \in P$, see Figure 1 (left). From this set of pairwise disjoint cells $\mathcal{C}_i$, a cover C_Ω; i.e., a collection of overlapping patches ω_i, is attained by simple scaling by $\alpha > 1$

$$\omega_i := (c_i^x - \alpha h_i^x, c_i^x + \alpha h_i^x) \times (c_i^y - \alpha h_i^y, c_i^y + \alpha h_i^y), \qquad (1)$$

see Figure 1 (right). Note that a cover patch ω_i is defined for leaf-cells $\mathcal{C}_i$ $(i = 1 \dots N)$ that contain a point $x_i \in P$ as well as for empty cells that do not contain any point from P.

On each cover patch ω_i a local approximation v_i of the field solution is computed and then blended smoothly by a partition of unity $\sum_{i=1}^{N} \varphi_i \equiv 1$ to form the global approximation; i.e., a global approximation v^{PU} defined on Ω in the PPUM is defined as the weighted sum

$$v^{\mathrm{PU}}(x,y) = \sum_{i=1}^{N} \varphi_i(x,y) v_i(x,y) \qquad (2)$$

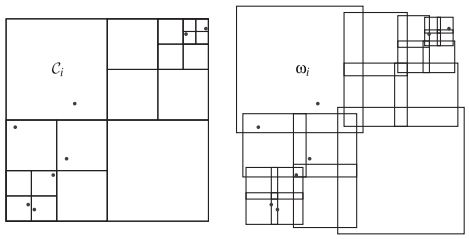


Figure 1: *Quadtree subdivision with cells $\mathcal{C}_i$ constructed by the simulation from a given set of points (left). The PPUM cover is then obtained by scaling the cells, resulting in the patches ω_i (right).*

of overlapping local approximations v_i defined on ω_i.

Each local approximation v_i in general consists of a smooth polynomial part p_i (similar to FEM) and an application-dependent enrichment part e_i, i.e.,

$$v_i(x,y) = p_i(x,y) + e_i(x,y),$$

which are described by the coefficients p_i^s and e_i^t and the associated basis functions ψ_i^s for the polynomials and η_i^t for the enrichments respectively. Thus, a local approximation is given by

$$v_i(x,y) = \sum_s p_i^s \psi_i^s(x,y) + \sum_t e_i^t \eta_i^t(x,y) \tag{3}$$

and with (3) put into (2) the respective global approximation is obtained:

$$v^{\mathrm{PU}}(x,y) = \sum_{i=1}^{N} \varphi_i(x,y) \left(\sum_s p_i^s \psi_i^s(x,y) + \sum_t e_i^t \eta_i^t(x,y) \right) \tag{4}$$

Note that the local polynomials $p_i(x,y)$ are spanned by a local basis ψ_i^s defined on ω_i. The enrichment basis functions η_i^t employed in the method are application dependent and are usually given as global functions η^t on the whole computational domain Ω since they are designed to capture special behavior of the solution at a particular location in Ω.

3.2. Fracture Enrichments

In fracture mechanics the most common choices in 2D are the following. Let $C \subset \Omega$ denote a crack that induces a discontinuous displacement field $\mathbf{u}$ across the crack line with particular singularities at the crack tips c_l and c_u. Thus on patches ω_i with

$$\omega_i \cap C \neq \emptyset \quad \text{and} \quad \{c_l, c_u\} \cap \omega_i = \emptyset$$

(red patches in Figure 2) the local polynomials $p_i(x,y) = \sum_s p_i^s \psi_i^s(x,y)$ are enriched by the additional basis functions

$$H_{\pm}^C(x,y)\psi_i^s(x,y) \tag{5}$$

where $H_{\pm}^C$ denotes the Haar function that is discontinuous at the crack C. Therefore, the respective local approximation v_i is given by

$$v_i(x,y) = \sum_s p_i^s \psi_i^s(x,y) + \sum_s q_i^s H_{\pm}^C(x,y)\psi_i^s(x,y),$$

with two sets of polynomial coefficients p_i^s and q_i^s. This type of enrichment is denoted as multiplicative. If a patch ω_i contains a crack tip ξ_{tip}, i.e., $c_l \in \omega_i$ or $c_u \in \omega_i$ (blue patches in Figure 2), the patch is enriched by the Westergaard functions

$$W_{\mathrm{tip}} := \{\sqrt{r}\cos\frac{\theta}{2}, \sqrt{r}\sin\frac{\theta}{2}, \sqrt{r}\sin\theta\sin\frac{\theta}{2}, \sqrt{r}\sin\theta\cos\frac{\theta}{2}\} \tag{6}$$

given in local polar coordinates with respect to the tip ξ_{tip}. These functions are derived from an asymptotic expansion of the solution and capture its dominant singularity. Here, the local approximation v_i is given by

$$v_i(x,y) = \sum_s p_i^s \psi_i^s(x,y) + \sum_t w_i^t \eta_i^t(x,y),$$

with the polynomial coefficients p_i^s and the four additional coefficients w_i^t associated with the four enrichment basis functions η_i^t of (6). This type of enrichment is referred to as additive. Figure 4 illustrates the four functions that are used to model the crack tip. The schematic view in Figure 2 shows how the different types of enrichments are employed in a simple crack simulation scenario.

The PU functions φ_i on a cover C_Ω are constructed by Shepard's approach in the PPUM. To this end a weight function $W_i : \Omega \to \mathbb{R}$ with $\mathrm{supp}(W_i) = \omega_i$ is defined on each cover patch ω_i by

$$W_i(x,y) = \begin{cases} \mathcal{W} \circ T_i(x,y) & (x,y) \in \omega_i \\ 0 & \text{else} \end{cases} \tag{7}$$

with an affine transform $T_i : \omega_i \to [0,1]^d$ and $\mathcal{W} : [0,1]^d \to \mathbb{R}$ denoting the reference d-linear B-spline. The Shepard functions

$$\varphi_i(x,y) := \frac{W_i(x,y)}{S_i(x,y)}, \quad \text{with} \quad S_i(x,y) := \sum_{l=1}^{N} W_l(x,y) \tag{8}$$

are then defined by simple averaging of these weight functions. The functions $\{\varphi_i\}$ with $i = 1 \ldots N$ form a partition of unity; i.e., they hold $0 \leq \varphi_i(x) \leq 1$ and $\sum_{i=1}^N \varphi_i \equiv 1$.

In summary, the evaluation of a global PPUM approximation (4) at a point $\mathbf{x}_w = (x,y) \in \Omega$ in global world coordinates involves the evaluation of the weight functions W_i and the basis polynomials ψ_i^s which are both defined in local coordinates on ω_i, and the evaluation of enrichment basis functions, i.e., the Haar function $H_{\pm}^C(x,y)$ of (5) and the Westergaard functions of (6), given in global coordinates.

4. Pixel-Accurate Visualization of PPUM Data

This section presents our pixel-accurate visualization framework that exemplifies the challenges involved in interactive visualization of 2D field solutions computed by particle-partition of unity simulation methods. The system harnesses

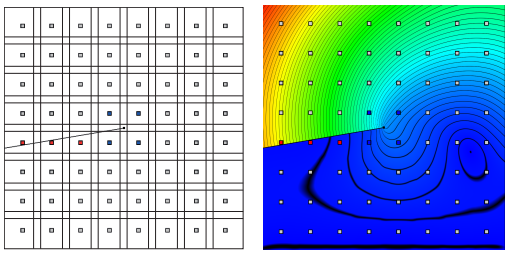


Figure 2: *Schematic view of a crack simulation with the PPUM. The centers of the patches ω_i are marked by squares whose color indicates the type of crack enrichment (blue: additive enrichment (6); red: multiplicative enrichment (5); gray: no enrichment, polynomial approximation). Additive enrichment is employed at the crack tip. Multiplicative enrichment based on Haar functions is used in regions that are completely cut by the crack.*

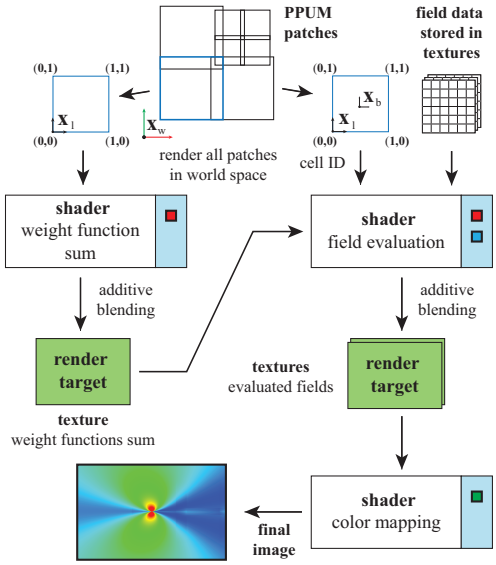


Figure 3: *Rendering pipeline for direct visualization of PPUM data. Its input are the overlapping PPUM patches according to Figure 1, and the field solution coefficients of the patches, which are stored in textures. The PPUM problem specific implementation of the function interface is indicated by the small colored boxes: PU weight function and its gradient (red), enrichment functions (blue), derived field functions (green).*

the Open GL rendering pipeline, and the flexibility of its shading language GLSL, to achieve accurate evaluation and visualization of the analytical field solution $v^{\mathrm{PU}}(x,y)$ with $\mathbf{x}_w = (x,y) \in \Omega$ given in world space coordinates within the simulation domain Ω. The enrichment functions of the simulation are often specified in an analytical manner. One example are the additive crack tip functions shown in Figure 4, which feature singularities that cannot be captured with resampled representations appropriately. Thus, a pixel-exact visualization technique also has to evaluate the true analytical representation of the enrichment functions. Therefore, our rendering pipeline provides a GLSL function interface which allows to easily replace the analytical function implementations that are specific to a particular PPUM solution. In future applications with dynamic enrichment function selection during simulation and a respective file format, this will allow for automatic GLSL implementation replacement.

Figure 3 illustrates the rendering pipeline. The higher-order field solution computed by the simulation is given on a number of overlapping rectangular PPUM patches covering the domain Ω. Our system is able to efficiently evaluate the field solution on a per-pixel level. Section 4.1 describes the evaluation core that handles the evaluation of the polynomial part of the solution local to a specific patch ω_i, computes the contribution of the patch enrichment functions by calling the exchangeable enrichment function implementations, and weights everything according to the cell's partition of unity function. Thereby, the spatially overlapping influence regions of the local solution functions have to be correctly accounted for. Finally, after the contributions of all patches have been summed up on a per-pixel level, the field values stored in the screen-sized field textures are mapped to color (Section 4.2).

In the following the field evaluation system is described at the example of a single scalar field solution. The extension to multiple scalar fields, vector fields, and their first order analytic derivatives is discussed in Section 4.3.

4.1. Field Evaluation

The global field solution $v^{\mathrm{PU}}(x,y)$, given in (2) and (4) , consists of multiple terms, the local solution functions $v_i(x,y)$ of the overlapping patches, and the PU part φ_i, which determines the contribution of the $v_i(x,y)$ in regions where multiple patches ω_i overlap. Consequently, multiple patches can contribute to the final field value of a pixel. The contribution of each patch at a pixel can be separated from each other if the sum of all weight functions $S = \sum W_i$ (8) is computed in screen space first.

To compute the weight function sum S on the domain Ω, the individual overlapping patches ω_i are rendered separately, and for each fragment the weight function W_i is evaluated at interpolated local patch coordinates $\mathbf{x}_l \in [0,1]^2$. Additive Open GL blending is employed to sum up the contributions of multiple patches in overlap regions in a floating point texture attached to the render target. Note, that in regions being covered by a single patch the weight sum S is equal to $W_i(\mathbf{x}_l)$.

Now, the contribution $\varphi_i v_i$ of each patch ω_i to a pixel can be computed separately by a fragment shader that fetches the

weight function sum S at the corresponding fragment from the texture computed in the previous stage. The contributions of multiple ω_i to a pixel can again be summed up with additive Open GL blending. To be able to evaluate the correct solution function v_i the individual patches ω_i are rendered with additional attribute values attached to each vertex. This includes a unique patch identification number that allows to access the patch's data stored in textures, like polynomial and enrichment function coefficients, from within the shader. Additionally, element-local coordinates $\mathbf{x}_l \in [0,1]^2$ are attached. For each pixel covered by the patch geometry the rasterization engine automatically generates a fragment and calculates its interpolated local coordinates $\mathbf{x}_l$ and corresponding world coordinates $\mathbf{x}_w \in \Omega$. The field contribution v_i (3) is then evaluated at the interpolated position.

Our system provides a main evaluation routine that includes the evaluation of the polynomial part of the solution in patch-local barycentric coordinates $\mathbf{x}_b = 2 \cdot (\mathbf{x}_l - 0.5)$. The polynomials are given in a monomial representation, which comes with the advantage that only the polynomial coefficients need to be stored. The structure and the order o of the corresponding basis functions $\psi_i = x_b^k y_b^l$ with $k+l <= o$ can be easily reconstructed during runtime, e.g., if one agrees to use a Morton order. Besides compact storage the monomial representation additionally allows for simple iterative evaluation of the polynomials on the GPU [UFE10]. If a patch is enriched by multiplicative or additive functions, those have to be evaluated and multiplied with their coefficients, too, to compute $\varphi_i v_i$.

The additive and multiplicative enrichment functions are problem specific. Therefore, we designed a simple GLSL function interface which is used by the evaluation core routine. For example if the core needs to evaluate the additive enrichment functions at $\mathbf{x}_w$ it calls the function `void EvalAddEnrichment(vec2 wCoord, int patchId, in sampler2D addEnrichData, out float values[numAdditiveFunctions])`. The system provides data required by the actual implementation, e.g., the orientation of the crack tip functions of our example problem, through a data texture. Note the enrichment functions are defined with respect to the global world space system. The function interface allows the problem specific parts of the PPUM function evaluation system to be replaced easily. With the GLSL compilation approach this can be done during runtime without the need to restart the application.

4.2. Color Mapping

The mapping of the field values to color is performed in a separate render pass. First, the system calls a predefined function that, e.g., selects a field component, or, maps the computed field values to a single scalar value, like, e.g., gradient field magnitude. Alternatively, a replaceable mapping function that can be implemented for a specific problem, is called. This was done for the von Mises field (Section 5.1). Finally, the scalar value is transferred to the color domain by using a 1D transfer function.

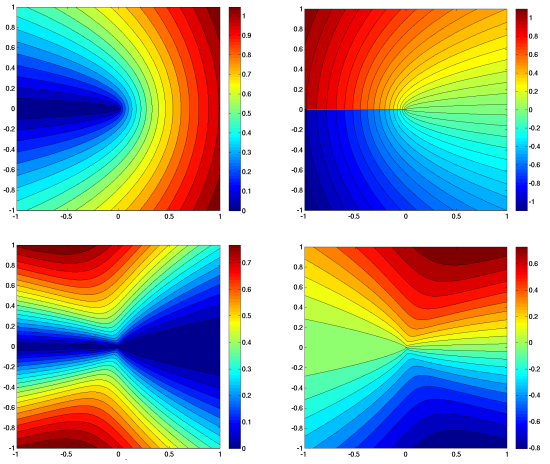


Figure 4: *Four additive crack tip enrichment functions represented with the analytical functions given in (6). For an adequate resolution of the singularities at the tip by piecewise polynomial basis functions or grid resampling techniques, strong adaptive refinement towards the tip would need to be employed. Moreover, the line of discontinuity, i.e., the crack would need to be resolved by the mesh.*

4.3. Vector Fields and Spatial Derivatives

Evaluating vector fields is a straightforward extension. Each component of the vector field comes with a full set of coefficient vectors, and thus, the components can be handled independently like a scalar field. The only difference is that the evaluation core needs to evaluate two scalar functions and the textures storing the result of the field evaluation in image space have to provide twice the storage space.

Often, the visualization of derived fields which build upon spatial field derivatives is of major interest. Therefore, the core system provides functionality to compute analytic first derivatives of the field solution. This includes the gradient field in the case of scalar, and the Jacobian field in the case of vector fields. Note, the according gradient function implementations need to be derived for the problem specific enrichment functions. In case of the polynomial solution the derivative can easily be computed, e.g., for a specific monomial term $x_b^k y_b^l$ the derivative in x_b direction simply is given as $k x_b^{k-1} y_b^l$. This enables the core to accurately compute the scalar gradient field with virtually no overhead. Note it is important to apply the chain rule, as the gradient field has to be evaluated with respect to world space, but the polynomial solution is defined in the patch's local barycentric space.

Computing the gradient in regions with overlapping patches is more complicated. Assume, the scalar field contribution of a single patch ω_i is given as $\varphi_i v_i$ with $\varphi_i = \frac{W_i}{\sum_l W_l}$ being the PU contribution of the patch, and v_i being the patch's

local field value. Its gradient

$$\nabla(\varphi v_i) = \nabla\varphi_i v_i + \varphi_i \nabla v_i \tag{9}$$

then not only involves ∇v_i, but also

$$\nabla\varphi_i = \nabla\frac{W_i}{\sum W_k} = \frac{\nabla W_i \sum W_k - W_i \sum \nabla W_k}{(\sum W_k)^2} \tag{10}$$

the gradient of the PU weighting. To be able to evaluate this, the weight function sum shader needs to compute the sum of the weight function gradients for each pixel and store it in the weights sum texture. The contributions of the individual patches can then be simply added up and stored in the field textures, similarly to the scalar field.

By using multiple RGBA floating point textures as render targets of the field evaluation shader, multiple scalar and vector fields and their gradients can be computed simultaneously and used as building blocks to construct more complex derived fields in the color mapping stage.

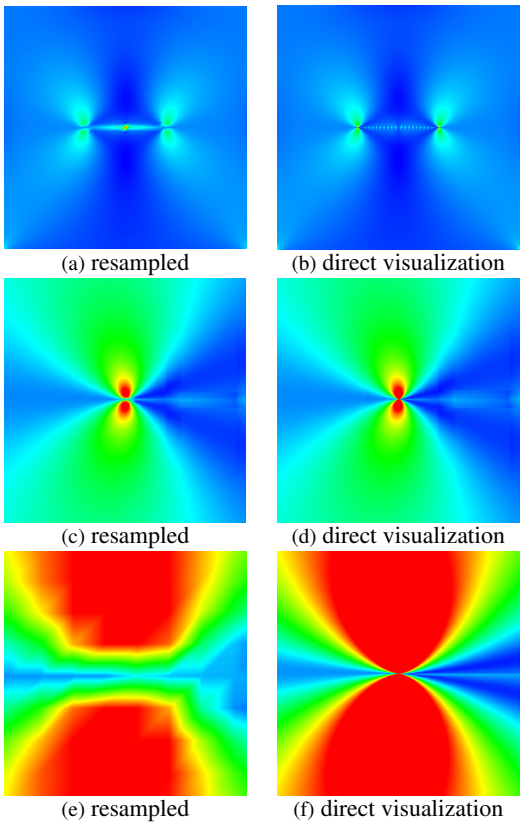
5. Results and Evaluation

We evaluate our method by visualizing the simulation results obtained with the PPUM for several fracture mechanics problems in two dimensions. In particular, we consider the static loading of a pre-cracked steel panel and the propagation of a crack. Thus, we approximate the equations of elasticity on a two-dimensional domain with internal traction-free boundaries. From a numerical point of view, the main issue in these simulations is the accurate approximation of the displacement field near the crack tip. The main question from an application point of view is if the material fails (near the crack tip) due to the current loading conditions. A widely used criterion for the failure of material is the so-called von Mises stress which encodes the tensorial stress data at a point into a scalar quantity. The von Mises stress is computed from the Jacobian of the displacement field.

In all examples considered here, the PPUM simulations employed linear splines as weight functions, given in (7). Yet the PPUM allows for the use of arbitrary non-negative weight functions, e.g., higher-order B-Splines. The weight function interface of our visualization framework provides this flexibility and quadratic B-Splines are implemented already in the framework. In all examples our visualization approach is highly interactive (Table 1).

5.1. Center Crack

First we consider a panel with a horizontal crack at the center of the panel that is fixed at the lower horizontal boundary and loaded in vertical direction on the upper boundary. Thus the crack will open in vertical direction and the characteristic singularities at the two crack tips will be clearly visible in the von Mises stress. This behavior can be observed from the visualizations shown in Figure 5. The PPUM simulation in this example employed additive enrichments only in the vicinity of the two crack tips to capture the singular behavior of the solution. Here, we used a coarse uniform subdivision on quadtree level 6 to define the PPUM cover and compare our visualization results with the traditional resampling approach which is likely to yield misleading results because the sampling points cannot capture the leading singularity (Figure 5 (a)). Moreover, even an extremely fine resampling cannot resolve the singular solution behavior at the crack tip (Figure 5 (c) and (e)). With our pixel-exact visualization approach we clearly capture the singular point. Comparable results by resampling would roughly require a tenfold sampling resolution which would lead to a dataset of more than 2 GB for this simple example. Moreover, our approach also allows us to detect very fine details of the simulation result which can be used to analyze the PPUM and its properties. For instance the visual artifacts observable on



(a) resampled (b) direct visualization (c) resampled (d) direct visualization (e) resampled (f) direct visualization

Figure 5: *Center crack with typical stress distribution at both crack tips. Comparison of our method (right column) to traditional resampling (left). (a) Resampled at the PPUM patch resolution of 64^2, misses features and shows incorrect features as in the middle of the crack. (c) Resampled on a regular 2048^2 grid and closeup (e). Direct visualization (b), (d), (f) in a pixel accurate manner provides clear advantages—and prevents misinterpretations, e.g., at the singularity at the crack tip (third row). Resampling would require at least 10 times higher resolution at this zoom level.*

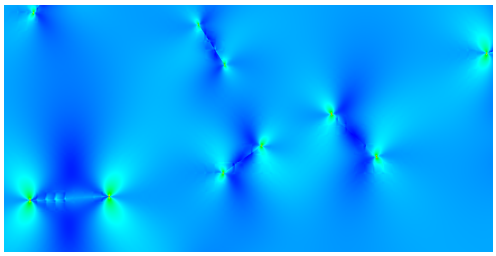


Figure 6: *Visualization of the von Mises stress in a steel plate that is precracked at multiple locations.*

Table 1: *Render time per frame. Center crack with 2nd order polynomials and with one quarter enriched patches. Intel Xeon X5570, NVIDIA GeForce GTX 470 @* 1024×768.

patches	16	64	256	1024	4096	16384
t in ms	2.33	2.38	2.48	2.89	4.76	11.76

the crack line in Figure 5 (b) are due to the fact that the additive enrichment functions employed here around the two tips meet in the center of the domain and cannot match. This leads to an (insubstantial) oscillation in the derivatives and stress field. Figure 6 shows an extension of the center-crack example involving multiple arbitrarily oriented cracks.

5.2. Crack Propagation

Next, we consider the propagation of a crack in a steel plate. Here we employ a quasi-static approach and predict the direction in which the crack will grow via the maximum hoop stress criterion. To this end, we extract the stress intensity factors from the computed solution via the contour integral method. From the visualizations depicted in Figure 7 we can clearly observe the expected behavior due to the employed loading conditions. The singularity at the crack tip moves through the simulation domain and the stress levels grow rapidly as the tip comes close to the domain boundary.

5.3. Adaptive Simulation

Finally, we visualize the results obtained with two adaptive PPUM simulations. Both are cell-size adaptive, one with fixed polynomial order $p = 1$ and the other with $p = 2$ on all patches. The overall quality of the results is comparable with respect to accuracy, see Figure 8. A direct comparison of the refinement patterns however clearly indicates the overall performance advantages of a higher-order method, especially in connection with the chosen enrichment scheme.

6. Conclusion

We have presented an efficient and accurate approach to the visualization of particle-partition of unity data. Combining

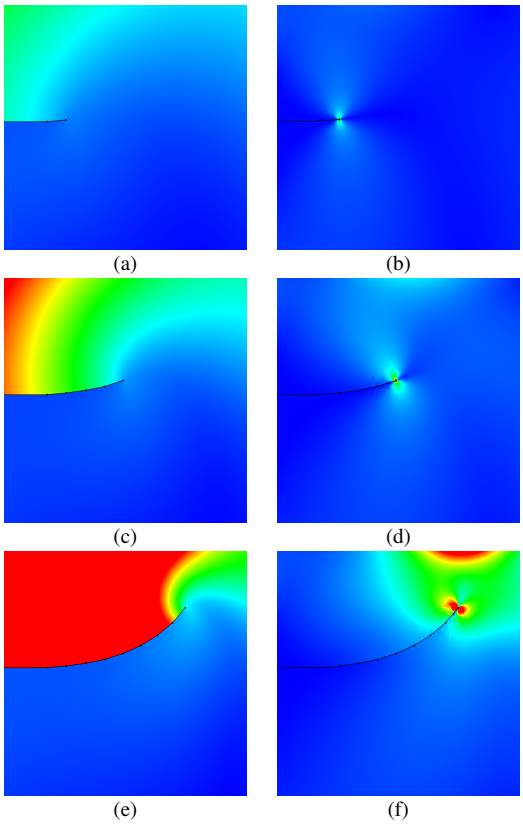


Figure 7: *Crack propagation visualized by displacement magnitude (left column) and von Mises field (right column), with an overlay (black) highlighting crack geometry.*

techniques from both scattered and cell-based higher-order data visualization we presented a method providing insight in this upcoming type of data. Accurate and efficient visualization of these data is of particular importance because, in contrast to low-order uniform techniques such as traditional FEM, the involved highly flexible simulation basis requires careful and thorough analysis both to support efficient simulation case design but also to support the research of this promising field of simulation techniques. Providing flexible evaluation schemes on GLSL runtime shaders has proven to be an elegant and viable approach to handle the manifold representation in terms of problem-specific enrichment functions together with scattered simulation bases. Alongside with planned extensions of the PPUM to the flow simulation domain with prospective enrichment functions accounting for, e.g., log-layer flow, we plan to account for these advances with respective visualization techniques. It is likely that the extension of the PPUM to 3D domains will not be amenable by GLSL shaders, leading to more involved GPU-based parallelization techniques for their visualization, but on the other hand, allowing for superior numerical accuracy using, e.g., double precision in CUDA.

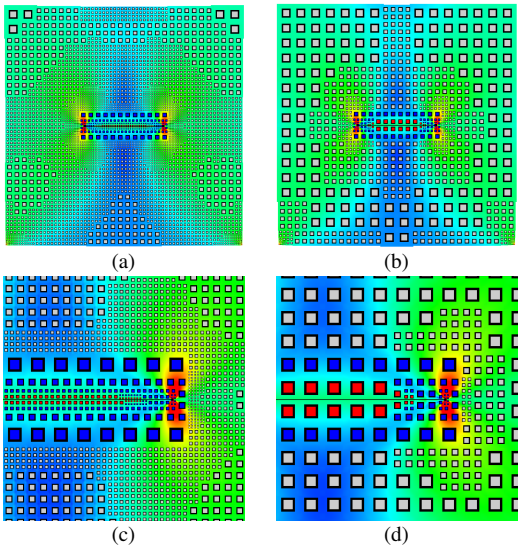


Figure 8: *Visual PPUM debugging. Results of two simulation runs with adaptive refinement (closeup in bottom row). Polynomials of order one (left) and two (right). Due to lower polynomial order the domain needs to be refined more extensively in the order one case. The colored boxes indicate the type of enrichment, multiplicative (red), and additive (blue).*

Acknowledgements

The authors would like to thank the German Research Foundation (DFG) for financial support of the project within the *Cluster of Excellence in Simulation Technology* (EXC 310/1) and the Collaborative Research Centre SFB-TRR 75 at Universität Stuttgart.

References

[BB99] BELYTSCHKO T., BLACK T.: Elastic crack growth in finite elements with minimal remeshing. *Int. J. Numer. Meth. Engrg. 45* (1999), 601–620. 2

[BCO94] BABUŠKA I., CALOZ G., OSBORN J. E.: Special finite element methods for a class of second order elliptic problems with rough coefficients. *SIAM J. N. Anal. 31* (1994), 945–981. 2

[BM96] BABUŠKA I., MELENK J. M.: The partition of unity finite element method: Basic theory and applications. *Comput. Meth. Appl. Mech. Engrg. 139* (1996), 289–314. 2

[DO96] DUARTE C. A. M., ODEN J. T.: hp Clouds – A meshless method to solve boundary value problems. *Numer. Meth. for PDE 12* (1996), 673–705. 2

[FB10] FRIES T.-P., BELYTSCHKO T.: The extended/generalized finite element method: An overview of the method and its applications. *International Journal for Numerical Methods in Engineering 84* (2010), 253–304. 2

[GN89] GALLAGHER R. S., NAGTEGAAL J. C.: An efficient 3-D visualization technique for finite element models and other coarse volumes. *Proceedings of the 16th annual conference on Computer graphics and interactive techniques - SIGGRAPH '89 23*, 3 (1989), 185–194. 2

[GS00] GRIEBEL M., SCHWEITZER M. A.: A particle-partition of unity method for the solution of elliptic, parabolic and hyperbolic PDE. *SIAM J. Sci. Comput. 22*, 3 (2000), 853–890. 1

[GS02] GRIEBEL M., SCHWEITZER M. A.: A particle-partition of unity method—part III: A multilevel solver. *SIAM J. Sci. Comput. 24*, 2 (2002), 377–409. 2

[HOR*03] HAASDONK B., OHLBERGER M., RUMPF M., SCHMIDT A., SIEBERT K. G.: Multiresolution visualization of higher order adaptive finite element simulations. *Computing 70*, 3 (2003), 181–204. 2

[MDB99] MOËS N., DOLBOW J., BELYTSCHKO T.: A finite element method for crack growth without remeshing. *Int. J. Numer. Meth. Engrg. 46* (1999), 131–150. 2

[MNKW07] MEYER M., NELSON B., KIRBY R., WHITAKER R.: Particle systems for efficient and accurate high-order finite element visualization. *IEEE Transactions on Visualization and Computer Graphics 13*, 5 (2007), 1015–26. 2

[NK06] NELSON B., KIRBY R. M.: Ray-tracing polymorphic multidomain spectral/hp elements for isosurface rendering. *IEEE Transactions on Visualization and Computer Graphics 12*, 1 (2006), 114–125. 2

[POS*11] PAGOT C., OSMARI D., SADLO F., WEISKOPF D., ERTL T., COMBA J.: Efficient parallel vectors feature extraction from higher-order data. *Computer Graphics Forum (CGF) 30*, 3 (2011), 751–760. 2

[PVS*10] PAGOT C., VOLLRATH J. E., SADLO F., WEISKOPF D., ERTL T., COMBA J. L. D.: Interactive isocontouring of high-order surfaces. In *Proceedings of Schloss Dagstuhl Scientific Visualization Workshop* (2010). 2

[RCMG07] REMACLE J.-F., CHEVAUGEON N., MARCHANDISE E., GEUZAINE C.: Efficient visualization of high-order finite elements. *Int. J. f. Num. Meth. in Eng. 69*, 4 (2007), 750–771. 1

[SBC00] STROUBOULIS T., BABUŠKA I., COPPS K.: The design and analysis of the generalized finite element method. *Comput. Meth. Appl. Mech. Engrg. 181*, I 3 (2000), 43–69. 2

[Sch03] SCHWEITZER M. A.: *A Parallel Multilevel Partition of Unity Method for Elliptic Partial Differential Equations*, vol. 29 of *Lecture Notes in Computational Science and Engineering*. Springer, 2003. 1

[Sch11a] SCHWEITZER M. A.: *Generalizations of the Finite Element Method*. Simtech preprint 2011-62, Universität Stuttgart, 2011. 2

[Sch11b] SCHWEITZER M. A.: Stable enrichment and local preconditioning in the particle-partition of unity method. *Numerische Mathematik 118* (2011), 307–328. 2

[SFBP09] SCHINDLER B., FUCHS R., BIDDISCOMBE J., PEIKERT R.: Predictor-corrector schemes for visualization of smoothed particle hydrodynamics data. *IEEE Trans. on Vis. and Computer Graphics 15*, 6 (2009), 1243–1250. 1, 2

[SUP*11] SADLO F., ÜFFINGER M., PAGOT C., OSMARI D., COMBA J. L. D., ERTL T., MUNZ C.-D., WEISKOPF D.: Visualization of cell-based higher-order fields. *Computing in Science & Engineering 13*, 3 (2011), 84–91. 1

[UFE10] ÜFFINGER M., FREY S., ERTL T.: Interactive high-quality visualization of higher-order finite elements. *Computer Graphics Forum (CGF) 29*, 2 (2010), 115–136. 1, 2, 5

[Zum94] ZUMBUSCH G. W.: *Visualizing Functions of the h-p-Version of Finite lements*. Tech. rep., Konrad-Zuse-Zentrum für Informationstechnik Berlin, Berlin, 1994. 2