Abstract

Surface approximation from unorganized points belongs to the state-of-art of computer graphics. In this work, we present approaches for surface reconstruction that are based on efficient numerical schemes for function approximation from scattered data and on sophisticated data structures. In addition, we develop a relevant surface reconstruction method to model moving interfaces, specifically, interfaces of numerically simulated multiphase fluid flow. Finally, from our accumulated experiences on surface reconstruction and on the development of surface reconstruction methods, we propose a matrix-free approach for rendering arbitrary volumetric scattered data, which presents interesting properties to be implemented on GPU.

1. Introduction

It is not difficult to find reasons for surface reconstruction methods being a strong tendency in computer graphics. First, despite significant advances, the manipulation of complicate point clouds originated from arbitrary geometries is an important issue that still requires improvements. Second, several research areas have make use of surface reconstruction methods, for instance, reverse engineering, optimization and medicine.

In this work, we present our contributions to surface reconstruction from unorganized points, whose developments were focused on novel moving-least-squares and partition of unity implicits methods. It will be noticed that our methods have common features of having low computational cost (mainly regarding matricial computations) and being robust. The use of efficient data structures to aid the surface reconstruction is also in the scope of our work [7, 9, 8, 10, 18, 6]. In addition, we address a surface reconstruction scheme for the front-tracking problem. An important contribution of such scheme is the absence of Lagrangian meshes. In fact, it is designed on the algebraic moving-least-squares method and on a approach for Lagrangian advection of particles. Finally, from our experiences on surface reconstruction and function approximation, we also developed a method for rendering scattered volumetric data based on a recently proposed matrix-free approach.

In the following sections, we describe relevant details of our approaches. Further information and comparisons can be found in [7, 9, 8, 10, 18, 6].

2. Twofold Adaptive Partition of Unity Implicits

We propose an approach based on partition of unity implicits [8, 9] which combines an adaptive and algebraic triangulation, named $J^a$ [3], and a recursively defined local approximation method based on multivariate orthogonal polynomials [2].

Similar to other implicit methods for surface reconstruction, the partition of unity implicits defines the reconstructed surface as the zero set of a function $F$, in a finite domain $\Omega$, from a linear combination of local approximations and weighting functions. For that purpose, we define a set of nonnegative weighting functions with compact support $\Phi = \{\omega_1, \ldots, \omega_n\}$, satisfying $\sum_{i=1}^{n} \omega_i(x) \equiv 1$, $x \in \Omega$ and we also consider a set of signed local approximations $F = \{f_1, \ldots, f_n\}$, which completely define the function $F: \mathbb{R}^3 \rightarrow \mathbb{R}$:

$$F(x) \equiv \sum_{i=0}^{n} f_i(x) \omega_i(x), \ x \in \Omega. \quad (1)$$

The partition of unity is then defined from a set of nonnegative functions with compact support $\theta$:

$$\omega_i(x) = \frac{\theta(||x - c_i||/R_i)}{\sum_{k=1}^{n} \theta(||x - c_k||/R_k)}, \quad (2)$$
where \( c_i \) and \( R_i \) are the center and the support radius of \( \omega_i \), respectively, and \( \theta(s) = (1 - s^2)^4 \) if \( s < 1 \); \( \theta(s) = 0 \) otherwise.

Thus, the implicit surface is given by \( \mathcal{S} = \{ x \in \mathbb{R}^3 : F(x) = 0 \} \). Figure 1 illustrates a CSG operation between Neptune model reconstructed from our method and a cylinder.

Figure 1. A CSG operation from Neptune dataset reconstructed using our partition of unity implicits method and a cylinder [9].

Briefly, the main contributions of this approach are:

**Adaptive isosurface extraction with topological guarantee:** in contrast to previous works, our method is able to extract the isosurface directly from the data structure which subdivides the space (the \( J_a^1 \) triangulation). In addition, since \( J_a^1 \) triangulation is defined by tetrahedra, the isosurface algorithm is not affected by ambiguous cases.

**Numerical stability:** despite their simplicity, canonical polynomial basis leads to ill-conditioned normal equations. In order to avoid such an issue, we make use of the multivariate orthogonal polynomial basis proposed by Bartels and Jezioranski [2], which improves the approximation quality without increasing the computational burden.

**Adaptiveness of the local fittings:** the use of orthogonal polynomials allows to increase the polynomial degree recursively. For that reason, our method is not only adaptive with respect to the spatial decomposition, but also with respect to the local fittings, defining a twofold adaptiveness. Figure 2 illustrates such a twofold adaptiveness.

**Robustness:** we propose not only computationally inexpensive, but also effective robustness criteria which allow our method to produce better results than previous works based on partition of unity implicits. In fact, our method is less susceptible to generate spurious surfaces.

**Mesh enhancement:** the aspect ratios of the triangles generated by the isosurface extractor based on the \( J_a^1 \) triangulation, in general, are poor. For that reason, we define a simple, but efficient procedure to improve the surface mesh quality. This procedure is based on the displacement of the \( J_a^1 \) vertices [4]. Figure 6 presents a result attesting that the mesh quality significantly increases. For quantitative analyses, see [9, 6].

**Interactive implicit function edition:** in order to satisfies the robustness criteria, details in some regions can be poorly approximate, since low degree polynomials are used. On the other hand, neglecting the robustness criteria is not a convenient choice, since spurious surface and artifacts can occur. In addition, increasing the restrictions on the robustness conditions may cause undesirable smoothing of
surface details without ensuring that all imperfections will disappear. Therefore, we developed an interactive implicit function edition which allows to locally change the local fittings without the need of re-computing completely the implicit function (Figures 3-(e)–(f) and 4).

**Sharp features modeling:** finally, we also properly adapt the scheme from Ohtake et al. [14] for detecting and modeling sharp features to our twofold partition of unity implicits method (Figure 5).

### 3. Front-tracking with moving-least-squares surfaces

We propose a meshfree front-tracking method for numerical simulation of fluid flows, where the interface is represented by an algebraic-moving-least-squares (AMLS) surface. Our method is interesting for being capable of preserving mass and geometry as well as mesh-based front-tracking methods [19] and being as flexible as level-set methods [15].

Let us again consider a finite set of points \( P = \{ p_1, \ldots, p_m \} \) almost regularly spaced at a distance \( h \) and \( \{ q_k \} \) an algebraic sphere basis [16], i.e.,

\[
q_1(x) = 1, \quad q_2(x) = x_1, \quad q_3(x) = x_2, \quad q_4(x) = x_3, \quad q_5(x) = x_1^2 + x_2^2 + x_3^2,
\]

where \( x = (x_1, x_2, x_3) \). Thus, we define a function \( f_\alpha(x) \) whose zero level approximates \( P \):

\[
f_\alpha(x) = \sum_{k=1}^{5} \alpha_k q_k(x) \tag{3}
\]

where \( \alpha = (\alpha_1, \ldots, \alpha_5) \) are the coefficients which completely defines \( f_\alpha \) and \( \alpha \) satisfies:

\[
\alpha = \arg\min_{\beta \in Q} \sum_{j=1}^{m} w_j |f_\beta(p_j)|^2, \tag{4}
\]

where \( w_j = w(x) = \theta \left( \frac{|x-p_j|}{3h} \right) \), defining non-negative weights \( \theta(s) = (1 - s^2)^2 \) if \( s < 1 \); \( \theta(s) = 0 \) otherwise and \( Q = \{ \beta \in \mathbb{R}^5 : \beta_1^2 + \beta_2^2 + \beta_3^2 - 4\beta_1\beta_5 = 1 \} \), being the restriction used to ensure that \( f_\alpha(x) \) approximates the signed distance of \( x \) to the surface \( f_\alpha \equiv 0 \).

The minimization (4) can be computed by a generalized eigenvalue problem in \( \mathbb{R}^5 \) [11]. In addition, for each point \( p_i \), the approximate normal is given by:

\[
N_i = \nabla f_\alpha(p_i)/\|\nabla f_\alpha(p_i)\|. \tag{5}
\]

The sphere-fitting problem, previously described, can be substantially improved by introducing the normals [11]. Furthermore, the introduction of normal in the minimization problem allows the splitting of the problem in two main steps. First, we compute the subset:

**Figure 3. Interactive function edition:** (a)-(b) present results from Ohtake et al. [14] with their default parameters and those one that we suggested, respectively; (c) our method but not using the coverage domain criterion [9]; (d) our method making use of the coverage domain criterion; (e) selecting surface artifacts; (f) the edited function eliminates imperfections [9].
Figure 4. Reconstruction enhancement by the interactive function edition: (a) original result with low degree approximations, (b) support selection for function edition (c) final result [9].

Figure 6. Mesh enhancement: Comparison between the mesh produced by the $f_1^a$ triangulation without the displacement vertices procedure (left) and the mesh produced by the displacement vertices procedure (right) [9].

\[
\hat{\gamma}(x) = (\gamma_2(x), \ldots, \gamma_5(x)) \quad \text{of} \quad \gamma = (\gamma_1, \ldots, \gamma_5) \quad \text{finding:}
\]

\[
\hat{\gamma}(x) = \arg\min_{\hat{\beta} \in \mathbb{R}^4} \sum_{i=1}^{m} w_i \| \nabla f_{\hat{\beta}}(p_i) - N_i \|^2.
\] (6)

where $N_i$ is computed as previously described. Notice that the solution $\hat{\gamma}(x)$ of (6) can be determined simply by solving a $4 \times 4$ linear system.

Finally, the first term $\gamma_1$ of $\gamma$ is computed by:

\[
\sum_{i=1}^{m} w_i F_{\hat{\beta}}(p_i)^2
\] (7)

where $\hat{\beta} = \hat{\gamma}(x)$. Thus:

\[
\gamma_1(x) = - \frac{\sum_{i=1}^{m} w_i \left( \sum_{k=2}^{5} \gamma_k(x) q_k(p_i) \right)}{\sum_{i=1}^{m} w_i}.
\] (8)

We name such an approximation **RAMLS-Robust Algebraic Moving Least-Squares**. It can be seen as an improvement of the Guennebaud and Gross method [11], since we eliminate the parameter that weighs the normal restriction into the minimization problem (in fact, we assume that the parameter $\beta$, in [11], as $\beta \to \infty$).
3.1. Point Advection

In order to update the RAMLS surface during the simulation, we move the points $\mathcal{P}$ according to the velocity field $\mathbf{v}$. Mathematically, we solve the following equation:

$$\frac{d\mathbf{p}_i}{dt} = \mathbf{v}(\mathbf{p}_i(t), t),$$

where $\mathbf{p}_i(0)$ are the initial points.

We denote by $\mathcal{P}(t) = \{\mathbf{p}_i(t)\}_{i=1,...,m}$ the set of points to be advected from $t$ to $t + \Delta t$ ($\Delta t$ is the time step). During the transport, the points in $\mathcal{P}(t)$ can become poorly distributed, which could compromise the surface $\mathcal{S}(t)$. In order to avoid such an issue, we propose to re-generate the point set periodically in a regularly spaced grid.

Re-generation points process: Let us consider a set of regularly spaced rays $R$ with spacing $h$ (Figure 7). The set of points $\mathcal{P}(t)$ is transported along the trajectories given by Equation 9. The point transportation is used to define the new RAMLS surface. The intersections between $R$ and the surface define the new set of points $\mathcal{P}(t + \Delta t)$. We depict such a re-generation process in Figure 7.

It is important to mention that Equation 9 is solved by the fourth-order Runge-Kutta method. A ray-tracing-like algorithm is used to detect the intersections between $R$ and the RAMLS surface. Since points can be re-generated very close, we remove points that are closer than $\frac{h}{2}$. Finally, the normals remain unchanged during the transportation in order to be used to orient the new normals in $\mathcal{P}(t + \Delta t)$.

Considering the convergence rates, for the two-dimensional case, we are able to show that RAMLS accomplishes order $O(h^3)$, $O(h^2)$ and $O(h)$ for geometrical, normal and curvature, respectively. Considering the three-dimensional case, RAMLS achieves rates of $O(h^2)$ and $O(h)$ for geometrical and normal curvature, respectively. During the transportation, the error is $O(h^r/\Delta t)$, where $r = 2, 3$ according to the geometrical error, previously argued.

For the Zalesak Sphere Test, we consider a sphere with radius 0.15, slope with length 0.10 and width 0.20, centered at $(0.5, 0.75, 0.75)$ in a unity cube. The velocity field is given by a rigid rotation:

$$\mathbf{v}(x_1, x_2, x_3) = \frac{\pi}{314}(0.5 - x_2, x_1 - 0.5, 0).$$

We assume $\Delta t = 1$ and a complete turn takes 628 time steps. We show the geometry of the Zalesak spheres at the times $t = 0, 79, 157, 236, 314, 393, 471, 550$ and 628 in Figure 8. We adopt the mesh size $h = 1/256$, and the number of particles as about 19000 for each time step. It can be noticed that the sharp features become smooth along the time, but the volume and shape remain similar [7, 6].

We also test our method to a three-dimensional deforming problem. We consider a sphere as $\mathcal{S}(0)$ with...
radius 0.15, centered at (0.35, 0.35, 0.35). The computational domain is the unity cube. The velocity field is given by:

\[
\begin{bmatrix}
2 \sin^2(\pi x_1) \sin(2\pi x_2) \sin(2\pi x_3) \\
- \sin(2\pi x_1) \sin^2(\pi x_2) \sin(2\pi x_3) \\
- \sin(2\pi x_1) \sin(2\pi x_2) \sin^2(\pi x_3)
\end{bmatrix}
\]

At \( t = T/2 = 1 \) time units, we reverse the velocity field in order to check the capability of the method in preserving topology and geometry. Figure 9 presents results of such a simulation, using \( h = 1/512, \Delta t = 0.0064 \) and 628 time steps.

4. Iterate Approximate Moving Least Squares Surfaces

One of the challenges to develop efficient surface reconstruction methods in modern GPU is the conformance of numerical computations to textures [18], represented by \( 4 \times 4 \) matrices.

When implementing a moving-least-squares surface method, in order to achieve high order approximations, the local approximations must be constructed by higher order polynomial basis, and consequently, a single \( 4 \times 4 \) matrix does not allow to achieve efficiently such a requirement. Studies to select a reduced polynomial basis [18, 10] have been done. However, the local approximation quality and high order approximations are not ensured.
To overcome the matrix reliance, we propose an iterative method based on the “iterate approximate moving least squares – iaMLS” [5]. This method is interesting for being able to enhance features simply by controlling the number of iterations and the single smooth parameter $\epsilon$. Formally, let us consider a function $F : \mathbb{R}^3 \rightarrow \mathbb{R}$, which its zero set approximates the surface. Let us also consider a set of points $x_i$, equipped with their normal vector $n_i$. Thus, we define:

$$F(x) = \sum_{i=1}^{m} \partial_i \Psi_i(x),$$

where $\partial_i = \langle x_i - x, n_i \rangle$, $\Psi_i(x) = \frac{\epsilon^3}{h^3/2} (r_i(x)) \exp(-r_i(x))$, $r_i = \frac{||x-x_i||}{h}$ and $h$ is the average spacing between the points [6]. The functions $\Psi_i$ are named second order generating functions [5].

From these definitions, and making use of the iterative process of the iaMLS [5], we define the *iaMLS surface* from the following iterative process:

$$F^{(0)}(x) = \sum_{i=1}^{m} \partial_i \Psi_i(x)$$

$$F^{(n+1)}(x) = F^{(n)}(x) + \sum_{i=1}^{m} \left[ \partial_i - F^{(n)}(x_i) \right] \Psi_i(x),$$

where $F \equiv 0$ gives the implicit surface reconstruction.
Figure 10. Effects caused by the number of iterations and the $\epsilon$ parameter. From left to right: 3, 4 and 5 iterations. From top to bottom: $\epsilon = 0.4$, $\epsilon = 0.8$, $\epsilon = 1.22$, $\epsilon = 2.0$. The greater the number of iterations or $\epsilon$ are, the more detailed the surface becomes.

5. Volumetric approximation from unorganized points

The last contribution of this work is related to unorganized volumetric data rendering. In fact, the goal is the definition of a method for rendering arbitrary meshes, i.e., a unified approach that explores the iaMLS. Firstly, in order to make the method more efficient for all mesh types, we make use of anisotropic spaces. Notice that the original iaMLS, to the best of our knowledge, does not make use of any anisotropy scheme [6, 17].

Our method is also able to render isosurfaces from
unorganized volumetric data. For that purpose, we also apply the iaMLS to estimate the gradient function [6, 17]. In fact, it is required to define, for each mesh vertex \( x_i \), a gradient vector estimate, using a least-squares approach. It is achieved by extending the approach by Mavriplis to the three-dimensional space [13]. It is important to notice that the use of such an approach remains our method independent of meshes.

Let us consider a mesh vertex \( x_i \), in which we evaluate the gradient, and its neighbors \( x_k = (x_k, y_k, z_k) \) in its star \( V_i \). Thus, we minimize the following equation with respect to \( \nabla f(x_i) = (f_x, f_y, f_z) \):

\[
\sum_{x_k \in V_i} \exp(-\langle x_k - x_i, x_k - x_i \rangle)(E_{ik})^2
\]

where:

\[
E_{ik} = (f_x)_{ik} \cdot dx_{ik} + (f_y)_{ik} \cdot dy_{ik} + (f_z)_{ik} \cdot dz_{ik} - df_{ik}
\]

and \( df_{ik} = f(x_k) - f(x_i) \). By the same token, \( dx_{ik} = x_k - x_i, dy_{ik} = y_k - y_i, dz_{ik} = z_k - z_i \).

The gradient vector, in an arbitrary point \( x \), indeed, is given by the iterative process:

\[
\nabla_{\psi f} f(x) = \sum_{x_i \in \psi} \nabla f(x_i) \Psi_i(x),
\]

\[
\frac{\partial^{(n+1)}}{\partial f} f^{(n)}(x) = \frac{\partial^{(n)}}{\partial f} f^{(n)}(x) + \sum_{x_i \in \psi} \left[ \nabla f(x_i) - \frac{\partial^{(n)}}{\partial f} f^{(n)}(x_i) \right] \Psi_i(x).
\]

Another interesting feature of our approach is its capability to be efficient for GPU implementation. We are able to provide pre-computations, which are stored in GPU textures. We perform two pre-computations based on the iterative process of iaMLS. They are related to the function evaluation and the gradient estimate.

Thus, we can rewrite the iterative process by Fasshauer [5] as:

\[
\frac{\partial^{(n+1)}}{\partial f} f^{(n)}(x) = \sum_{x_i \in \psi} \left[ f(x_i) + \sum_{j=0}^{n} \left( f(x_i) - \frac{\partial^{(j)}}{\partial f} f^{(j)}(x_i) \right) \right] \Psi_i(x),
\]

where, we can accumulate the results for each mesh vertex as:

\[
g(x_i) = f(x_i) + \sum_{j=0}^{n} \left( f(x_i) - \frac{\partial^{(j)}}{\partial f} f^{(j)}(x_i) \right),
\]

and store them in textures which accommodate the unorganized volumetric data. Then, during the rendering, the reconstructed function value is simply given by:

\[
\frac{\partial^{(n+1)}}{\partial f} f^{(n)}(x) = \sum_{x_i \in \psi} g(x_i) \Psi_i(x).
\]

Rendering results are presented in Figure 12 for different meshes. We use \( h = 0.25 \) and \( e = 0.9 \). Both parameters are chosen empirically after numerical tests.
6. Conclusion

In this work, we presented novel and important contributions towards the field of dynamic and static surface reconstruction methods from unorganized points. There are several ways to extend the techniques here presented. First, we intend to investigate theoretically our surface reconstruction method based on iaMLS.

With respect to the twofold adaptive partition of unity implicits, we intend further to improve the function edition process by incorporating new local approximating functions. In addition, we want to improve the mesh enhancement technique, making use of the whole scheme by Dietrisch et al. [4], but, in our case, considering the adaptiveness of $J^a$ triangulation.

Considering the front-tracking with AMLS surfaces approach, the definition of a good curvature from RAMLS is a mandatory future work. In addition, we aim at applying this surface representation for numerical simulation of fluid flows. We are also investigating schemes to perform topological changes efficiently.

Finally, we intend to investigate the possibility to estimate automatically the parameters $h$ and $e$ of the iaMLS defined in anisotropic domains for volumetric rendering.

Acknowledgments

This work was supported by FAPESP (Brazil) and DAAD (Germany). The methods in Sections 2 and 5 were developed also with Eduardo Tejada and Tomas Ertl (University of Stuttgart–Germany). The work in Section 4 was developed also with Gustavo C. Buscaglia and Anderson Nakano. The work on Section 3 was also developed together with Tiago Etiene, Valdecir Polizelli, Eduardo Tejada and Thomas Ertl. We gratefully thank Luis G. Nonato, an important collaborator for all of these works.

References