Pacific Graphics 2013
B. Levy, X. Tong, and K. Yin
(Guest Editors)

Volume 32 (2013), Number 7

A Semi-Lagrangian Closest Point Method for Deforming Surfaces

S. Auer[1,2] and R. Westermann[2]

[1]Metaio GmbH, Germany
[2]Computer Graphics & Visualization Group, Technische Universität München, Germany

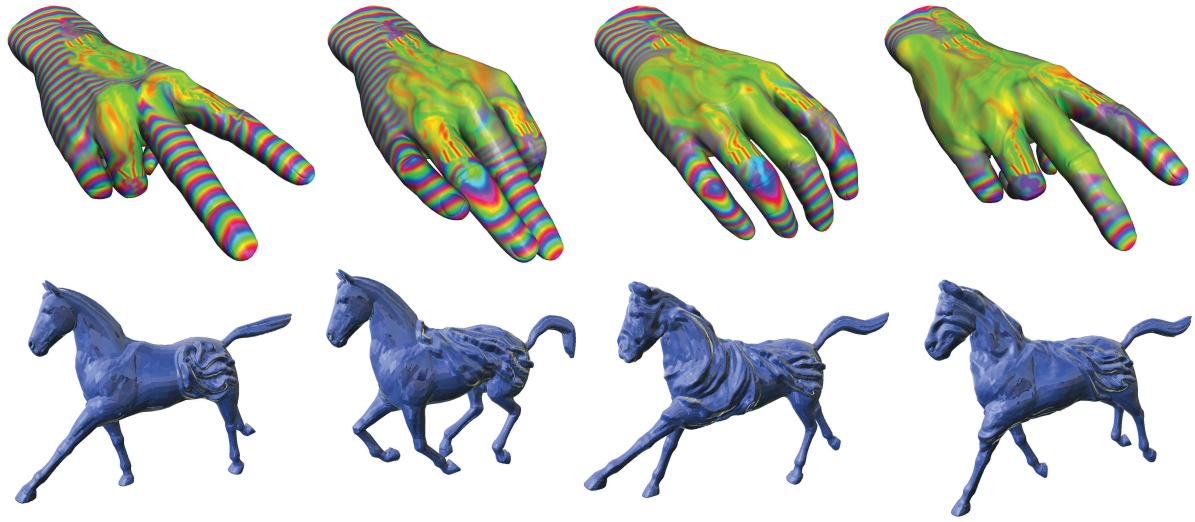

Figure 1: *Intrinsic flow simulations on deforming surfaces via a semi-Lagrangian closest point method. At a resolution corresponding to a* 320^3 *Cartesian grid, simulation and rendering takes less than 120 ms per time step.*

Abstract

We present an Eulerian method for the real-time simulation of intrinsic fluid dynamics effects on deforming surfaces. Our method is based on a novel semi-Lagrangian closest point method for the solution of partial differential equations on animated triangle meshes. We describe this method and demonstrate its use to compute and visualize flow and wave propagation along such meshes at high resolution and speed. Underlying our technique is the efficient conversion of an animated triangle mesh into a time-dependent implicit representation based on closest surface points. The proposed technique is unconditionally stable with respect to the surface deformation and, in contrast to comparable Lagrangian techniques, its precision does not depend on the level of detail of the surface triangulation.

Categories and Subject Descriptors (according to ACM CCS): Simulation and Modeling [I.6.8]: Types of Simulation——Parallel Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism——Raytracing

© 2013 The Author(s)
Computer Graphics Forum © 2013 The Eurographics Association and John Wiley & Sons Ltd. Published by John Wiley & Sons Ltd.

1. Introduction

Solving partial differential equations (PDEs) on surfaces embedded in three-dimensional space is an important ingredient in many computer graphics applications, including, for instance, texture synthesis [Tur91], fluid simulation [Sta03] or gradient-domain image processing [CLB*09]. While most existing techniques consider only static surfaces, in some recent works it was raised the importance of solving PDEs on deforming surfaces. A practical example can be found in the context of fluid simulation, where free boundary surfaces resulting from coarse simulations are enriched in a post-process by high-frequency details [KTT13]. These details are either to costly to be simulated in 3D or they cannot be captured by the underlying physical model. Up-resing techniques can considerably increase the apparent resolution, and they give a high degree of control over the resulting simulations because they preserve the coarse flow features. Since the post-process only requires keeping track of the free surface part of the fluid, this also has the beneficial side effect that the problem size is vastly reduced.

Considering a deforming simulation domain, it is worthwhile to classify the available algorithms for solving PDEs on such domains according to the time dependence of the employed discretization. On one side of the spectrum there are purely Eulerian methods, based on a sampling that remains fixed in the three-dimensional embedding space. These techniques often employ implicit surface representations and provide a very natural coupling to fully three-dimensional Eulerian simulations [KTT13]. On the other side there are purely Lagrangian methods, employing a dynamic sampling that follows the surface movement *and* the evolution of the simulated properties on the surface at the same time. Most existing techniques are hybrids, however, which employ, for example, an Eulerian fluid simulation on a Lagrangian grid [ATBG08, TWGT10].

The algorithm we propose utilizes semi-Lagrangian discretization schemes and a Lagrangian representation of the surface deformation. We still consider it an Eulerian technique, because it carries out the simulation on a static simulation grid. To the best of our knowledge, it is the first real-time Eulerian method for the solution of PDEs involving intrinsic differential operators on deforming surfaces. It is based on the closest point method (CPM) [RM08], a so-called embedding technique for the solution of PDEs on surfaces, which recently gained attention in the computer graphics community.

Building upon the work of Hong, Auer, Kim and coworkers [HZQW10, AMT*12, KTT13], we contribute a semi-Lagrangian closest point method (SLCPM) for the interactive simulation of fluid effects on *deforming* surfaces. This method has some beneficial properties for upresing. In particular, it works on a closest point representation into which the surface deformation can be embedded. As a consequence, the method can run independent of the specific fluid simulation used. We especially demonstrate this by simulating fluid effects on arbitrary animated surface meshes, which do not result from fluid simulations, and which motion is only known from a sequence of time-varying triangle meshes. Thus, our proposed technique supports a wide range of widely used animation techniques, including keyframing, skinning, rigid or soft body simulation, and fluid simulation with mesh-based surface tracking.

Our technique can overcome some major restrictions and limitations of previous techniques which have applied the CPM to deforming surfaces. While the method by Hong et al. [HZQW10] requires an additional level-set representation of the surface on which the 2D simulation is carried out, Kim et al. [KTT13] relies on the existence of an external velocity field in which MacCormack advection can be performed.

In addition to techniques using the CPM, Angst et al. [ATBG08] have performed interactive fluid effects on animated triangle meshes directly. In comparison, an Eulerian embedding grid and implicit surface representation, as required by our proposed SLCPM, may seem as an unnecessary overhead in the first place. The resulting advantage, however, is a simulation methodology which is decoupled from the surface topology, level of detail and other properties of the input triangulation, and which adapts automatically to deformations which alter the surface geometry and topology.

This work is structured as follows: In Section 2 we review previous work that is related to ours. The beginning of Section 3 motivates the integration of the semi-Lagrangian scheme into the CPM and gives a high-level overview of our approach. In Section 3.1 we summarize the basis techniques and detail the aspects relevant to their combination. Section 3.2 describes the implicit representation of the deforming surface. In Sections 4 and 5 we conclude our work with a presentation of the results and an outlook at future work.

2. Related Work

In a number of previous works, fluid-related phenomena have been simulated numerically on deforming surfaces. The majority of the presented techniques uses an unstructured tessellation of the surface as a Lagrangian spatial discretization. Bargteil et al. [BGOS06] developed a method for texturing fluid surfaces which contours and re-initializes a zero level set in each time step. They advect the triangle mesh in the velocity field of the background fluid simulation, which thereby transports per-vertex texture coordinates or attributes of a reaction-diffusion simulation [Tur91]. Their simulation is very sensitive to the

© 2013 The Author(s)
Computer Graphics Forum © 2013 The Eurographics Association and John Wiley & Sons Ltd

motion of the surface, so that even small perturbations can cause large changes in the resulting surface texture. Angst et al. [ATBG08] discretize the wave equation on an animated character mesh, based on a mixed finite volume / finite element method. Their interactive simulation is very robust because of the implicit time integration scheme and the fixed topology of the Lagrangian discretization grid. However, because the computation of an additional well conditioned triangulation at runtime is avoided, the discretization does not adapt to deformations affecting the triangle sizes. Thürey et al. [TWGT10] also discretize the wave equation on a Lagrangian mesh in order to simulate capillary waves driven by surface tension. Their simulation employs a finite element method similar to the one of Angst et al., yet it transforms the surface tessellation into a height field on a lower resolution grid in each time step of the background simulation. The numerical analysis of methods based on Lagrangian surface meshes is difficult, because unstructured deforming grids in combination with approximate differential surface characteristics lead to non-trivial discretizations of the intrinsic differential operators [BCOS01, RM08].

An alternative approach for simulating fluid effects on surfaces is to solve for the differential equations in the 3D embedding space in such a way that the restriction to the surface provides a solution to the intrinsic surface equations. In contrast to techniques working in a 2D surface parametrization or the surface directly, such embedding methods usually employ an Eulerian spatial discretization, often in the form of a regular 3D grid [BCOS01, Bur05, Gre06, RM08]. This simplifies the numerical analysis considerably, because the discretization is not affected by the deformation of the surface. As with all Eulerian techniques, the explicit handling of advection requires special care, especially when movements and deformations of the surface have to be considered.

Among the embedding techniques, the closest point method [RM08, MR08, MR09, MM12] stands out for its simplicity and generality. It employs the unmodified three-dimensional Cartesian counterparts of the intrinsic surface differential operators in the embedding PDE, and it restricts the calculations to a narrow band around the surface without enforcing any non-physical boundary conditions. Hong et al. [HZQW10] apply the CPM to the level set equation in order to simulate the spreading of fire on an animated surface. Since they also treat the movement of the surface itself as a level set evolution, they can only handle advections in directions normal to the surface. Kim et al. [KTT13] suggest to increase the apparent resolution of a simulated liquid surface by simulating additional wave propagation on this surface via the CPM in a post-process. They use a level-set to re-initialize the closest point function and an extended velocity field, generated by the low-resolution volumetric simulation, to advect the simulation attributes with the deforming surface.

Besides the fact that our proposed method works directly on the embedded surface and does not require knowledge about the 3D background simulation, another major difference to Kim et al. lies in the handling of numerical dissipation due to interpolation. Similar to both approaches is the use of WENO interpolation in the semi-Lagrangian surface advection step. In the closest point extension step, however, Kim et al. employ a vastly different strategy: For grid points in the vicinity of the surface they do not perform any extension at all, and for all other grid points they perform nearest-neighbor interpolation. Thus, the method trades numerical dissipation for loss of accuracy in the embedding step. Our method, in contrast, does not rely on a special treatment of near surface points, since it requires only one interpolation in a combined advection-extension step. Concerning the amount of dissipation, our approach is thus en par with the semi-Lagrangian approach for fluid advection in tetrahedral meshes by Feldmann et al. [FOKG05]. To reduce dissipation they trace backwards from the current mesh velocities, instead of first resampling the old velocities in the current mesh and tracing back using these velocities.

3. A Semi-Lagrangian Closest Point Method

Our proposed technique is based on an efficient CUDA implementation [AMT*12] of the explicit closest point method [RM08] for the numerical solution of initial value problems. It discretizes a narrow spatial band around the surface with a 3D Cartesian grid and runs two phases in each time iteration. In the evolution phase, it solves an embedding PDE, containing only Cartesian differentials using standard finite difference methods. In the extension phase, it replaces the solution at each grid vertex with the value obtained from the closest point on the surface. Through the equivalence principles of the CPM, it is ensured that the resulting volumetric solution solves the surface PDE for all points on the surface.

In existing applications of the explicit CPM to deforming surfaces, the advection of simulation attributes with the surface is handled in an additional, decoupled step. Either level-set advection [HZQW10] or MacCormack advection in an external velocity field [KTT13] have been employed. We instead propose to integrate the advection step directly into the CPM.

Our algorithm is based on the observation that the explicit CPM and the backward semi-Lagrangian method exhibit remarkable similarities. The former finds the closest point of a computational node on the surface, whereas the later integrates the position of a node backward in time to find the foot of a characteristic curve in a velocity field. Both methods obtain updated nodal values through

© 2013 The Author(s)
Computer Graphics Forum © 2013 The Eurographics Association and John Wiley & Sons Ltd

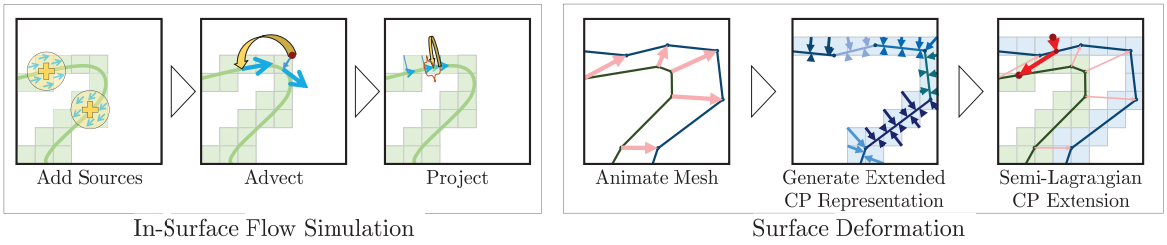

Figure 2: *SLCPM overview. The semi-Lagrangian CP extension step advects the flow field with the deforming surface.*

interpolation at the found position. We propose a combined method, which first finds the closest point and then integrates this position backward in time along the characteristics defined by the motion and deformation of the surface.

Integrating the closest point extension with a semi-Lagrangian backtrace in a single step has three main advantages: First, the approach is unconditionally stable with respect to the surface deformation, which is important when the input animation is not under our control and may therefore exhibit arbitrary deformation rates. Second, we reduce the number of spatial interpolations, which are computationally expensive and introduce numerical dissipation [AMT*12, KTT13]. Third, if the backtrace starts at a closest point on a triangulated surface, the foot of the trajectory can be found efficiently by considering only the movement and deformation of an individual triangle.

Figure 2 gives an overview of our algorithm for the interactive simulation of fluid flows on deforming surfaces based on these ideas. The steps in the left panel solve the incompressible Navier-Stokes equations

$$\frac{\partial \boldsymbol{v}}{\partial t} = -(\boldsymbol{v}\cdot\nabla_S)\boldsymbol{v} - \frac{1}{\rho}\nabla_S\, p + \nu\nabla_S^2\, \boldsymbol{v} + \boldsymbol{f} \quad \text{and}$$

$$\nabla_S \cdot \boldsymbol{v} = 0$$

on the surface, using a method similar to the one described by Auer et al. [AMT*12] for static surfaces. They combine an adaptive multiblock CPM with a standard splitting technique. In the first step, velocity sources are injected by applying Dirichlet boundary conditions in user-defined regions on the surface. User-defined boundary conditions can also be employed to simulate obstacles within the flow or to control the behavior at the borders of open surfaces. The second step addresses the convective acceleration term $(\boldsymbol{v}\cdot\nabla_S)\boldsymbol{v}$ by performing standard first-order semi-Lagrangian advection along the surface, followed by a tangential projection of the velocities. We skip the viscosity term $\nu\nabla_S^2\, \boldsymbol{v}$ because we do not intend to simulate highly viscous fluids and because a small amount of numerical dissipation is already introduced by the interpolation steps.

The artificial viscosity can be reduced considerably with high-order stability-preserving WENO interpolation [AMT*12, KTT13]. We also do not use the external forces term $\boldsymbol{f}$ in our current examples, in order to demonstrate the pure in-surface evolution of the intrinsic flow. If desired, viscous flows can be integrated explicitly and external forces can be used to integrate extrinsic effects such as inertia, as suggested by Angst et al. [ATBG08]. To enforce the continuity equation, we use the pressure term $\frac{1}{\rho}\nabla_S\, p$ of the momentum equation to project the velocity field onto a space of solenoidal functions. This third step utilizes a conjugate gradient method and an artificial Neumann boundary condition to solve a Poisson problem of the form $\nabla\cdot\nabla p(\boldsymbol{x}) = \nabla\cdot\tilde{\boldsymbol{v}}\big(cp(\boldsymbol{x})\big)$, in which the right-hand side is the closest point extension of the divergence.

After the pressure correction, the method of Auer et al. [AMT*12] obtains a closest point extension of the velocity field. We replace this final sub-step with a *semi-Lagrangian* closest point extension which additionally advects the velocity field together with the deforming surface, as detailed in Section 3.1.

As an alternative to the intrinsic flow simulation, we also consider the simulation of intrinsic waves which propagate along a deforming surface membrane. Therefore, we replace the velocity field with a height field, and instead of the Navier-Stokes equations we solve the surface PDE version of the classic wave equation

$$\frac{\partial^2 u}{\partial t^2} = c^2\nabla_S^2\, u.$$

An embedding PDE is obtained by replacing the Laplace-Beltrami operator on the right-hand side with the standard Cartesian Laplace operator and an explicit Verlet integration scheme is employed as discretization method. In contrast to Auer et al. [AMT*12], we again obtain a *semi-Lagrangian* closest point extension instead of a standard closest point extension at the end of each time integration step.

Before a semi-Lagrangian extension can be embedded

© 2013 The Author(s)
Computer Graphics Forum © 2013 The Eurographics Association and John Wiley & Sons Ltd

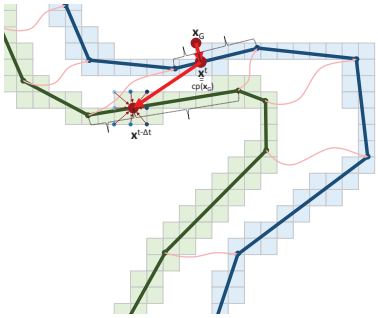

Figure 3: *The animation has deformed the triangulated surface and moved it to the right. For each grid vertex $\boldsymbol{x}_G$ in the narrow band at time t, we determine the closest point on the surface $\boldsymbol{x}^t$ and its previous position $\boldsymbol{x}^{t-\Delta t}$. To obtain the closest point extension at $\boldsymbol{x}_G$, we interpolate the values stored in the narrow band of the previous time step.*

into our simulation method, however, we must first deform the triangulated surface and update its implicit representation, as depicted in the right panel of Figure 2. Our algorithm considers the animation of the triangle mesh, i.e. the update of the vertex positions, to be an external input on which it does not impose any restrictions: Topological changes of the surface are supported without special treatment and the precision of the simulation is unaffected of strongly varying triangle sizes. The animation may be key-framed, come from an interactive technique such as skinning, or it may be the output of some real-time physics simulation. If the animation technique itself does not support access to arbitrary time steps, we keep the vertex positions of the last time step in a buffer. The implicit surface representation and its generation from the deformed triangle mesh are discussed in Section 3.2.

3.1. Semi-Lagrangian Closest Point Extension

The closest point extension is a core concept in the equivalence principles which form the basis of the CPM. For a surface S embedded in three-dimensional Euclidean space, the equivalence of gradients principle, for example, states that the intrinsic surface gradient $\nabla_s u$ and the Cartesian gradient ∇v agree if the volume function v is a closest point extension of the surface function u:

$$v(\boldsymbol{x}) = u\big(cp(\boldsymbol{x})\big) \qquad \Rightarrow \qquad \nabla_s u(\boldsymbol{x}) = \nabla v(\boldsymbol{x}), \quad \boldsymbol{x} \in S.$$

Like all embedding methods, the CPM further assumes that the surface function is a restriction of the volume function to the surface. In a discrete integration scheme, it therefore enforces the precondition of the equivalence principles by projecting the volumetric solution to the space of closet point extensions. In each integration step, the simulation attributes stored at the grid vertices within the narrow band are updated with values interpolated at the respective closest points. To ensure the consistency and accuracy of the method, high-order WENO schemes can be used to interpolate values in the three-dimensional grid.

Another crucial aspect in the realization of the SLCPM is the semi-Lagrangian scheme which is used to advect quantities through an Eulerian simulation grid. The classical scheme traces a pathline beginning at a grid vertex backwards in time trough the velocity field. At the foot of this characteristic curve, the attribute fields of the last time step are sampled using an interpolation technique to obtain the updated values for the grid vertex. The accuracy of this approach depends on the order of the time-integration employed for the backtrace and the order of the spatial interpolation. Raising only the accuracy of the time integrator can have negative effects, since typically multiple spatial interpolations are required in this case.

Our semi-Lagrangian closest point extension combines the explicit closest point extension with a semi-Lagrangian backtrace along the characteristics defined by the surface motion as depicted in Figure 3. In order to ensure a good performance and accuracy of our technique, we directly utilize the triangle animation for the backtrace and employ a stabilized WENO4 scheme [MR09, AMT*12] for interpolation. By this means the advection scheme can achieve up to third-order accuracy, depending on the smoothness of the surface and the accuracy of the input animation.

For each grid vertex, we first compute the barycentric coordinates of its closest point on the mesh with respect to the closest triangle. The foot of the characteristic is then obtained by applying barycentric interpolation to the vertex positions in the last time step. Since the semi-Lagrangian closest point extension typically needs to be performed several times for multiple closest points per triangle, we accelerate the process by pre-computing an implicit surface representation in each time-step as discussed in the next section.

3.2. Extended Closest Point Representation

To facilitate an efficient CUDA implementation of the semi-Lagrangian CPM, we convert the triangle mesh into an implicit representation after each deformation of the surface, i.e. usually once per time step of the simulation. The implicit surface representation employed in this work is an extension of the adaptive closest point grid presented by Auer et al. [AMT*12]. The embedding space is discretized with a uniform Cartesian grid and the closest points of the grid vertices within a narrow band around the surface are stored in a sparse multi-block grid. For the width of the narrow band we follow the suggestions of Macdonald and Ruuth [MR08]. In addition to the closest point, our extended implicit representation stores also

© 2013 The Author(s)
Computer Graphics Forum © 2013 The Eurographics Association and John Wiley & Sons Ltd

the index of the closest triangle to enable a coupling to the input animation.

For the grid generation and computation of closest points, we employ a sort-middle strategy to calculate the distances to the individual triangles directly [AMT*12]. Since all computations are restricted to a narrow band around the surface, we avoid approximative distance transforms, which could affect the precision of the CPM. We first determine the coarse-level blocks which overlap the narrow bands and allocate the resources in global GPU memory required to store the closest points, closest triangles, and the simulation attributes in the fine-level subgrids. Additionally we determine for each overlapped block the triangles within the narrow band radius. We then iterate over the fine-level vertices of each overlapped block, compute for each vertex the distances to the respective triangles and store the index of the closest triangle in global memory. Then we iterate over the fine-level vertices a second time, compute for each of them the closest point on the closest triangle and store it in global memory. While this approach adds another kernel call, a redundant calculation and an indirection over global memory, we found it nevertheless to work faster than the original algorithm [AMT*12], because it requires less shared memory to hold the index of the closest triangle (4 byte) instead of the closest point (12 byte).

Together with the time-dependent positions of the triangle vertices, given by the input animation, this implicit surface representation enables the efficient mapping between the grid vertices and the former positions of their closest points, which is required for the semi-Lagrangian closest point extension. As discussed in the section before, we need the barycentric coordinates of the closest points within their respective triangles whenever we perform this mapping. Since the semi-Lagrangian extension needs to be performed for a very large number of grid vertices, the computation of the barycentric coordinates can have a severe impact on the performance of our method. If multiple semi-Lagrangian extensions are required, for example for additional simulation attributes, we could pre-compute the required data already in the closest point calculation and store it in an additional buffer. This would considerably increase the memory and bandwidth requirements of our method, however. We make instead use of the fact that the number of triangles was always *considerably* below the number of grid vertices in all of our test cases. For each triangle we pre-compute a barycentric transformation whenever the surface is deformed. This transformation allows us to obtain the barycentric coordinates of multiple closest points on a triangle very efficiently.

To the best of our knowledge, the most efficient way to compute barycentric coordinates for multiple points $\boldsymbol{x}_{cp}$ on a triangle with vertices $\boldsymbol{x}_U, \boldsymbol{x}_V, \boldsymbol{x}_W$ was presented by Schneider [Sch09] (page 121). The idea is to first transform all vector components into a triangle coordinate system with basis vectors $\boldsymbol{t} := \boldsymbol{x}_V - \boldsymbol{x}_U$ and $\boldsymbol{b} := \boldsymbol{x}_W - \boldsymbol{x}_U$, and then to obtain the barycentric coordinates $\alpha_U, \alpha_V, \alpha_W$ by solving the linear 3×3 system

$$\begin{bmatrix} \boldsymbol{t}\cdot\boldsymbol{x}_U & \boldsymbol{t}\cdot\boldsymbol{x}_V & \boldsymbol{t}\cdot\boldsymbol{x}_W \\ \boldsymbol{b}\cdot\boldsymbol{x}_U & \boldsymbol{b}\cdot\boldsymbol{x}_V & \boldsymbol{b}\cdot\boldsymbol{x}_W \\ 1 & 1 & 1 \end{bmatrix} \cdot \begin{pmatrix} \alpha_U \\ \alpha_V \\ \alpha_W \end{pmatrix} = \begin{pmatrix} \boldsymbol{t}\cdot\boldsymbol{x}_{cp} \\ \boldsymbol{b}\cdot\boldsymbol{x}_{cp} \\ 1 \end{pmatrix}.$$

Note that the inner products in the above equation transform the covariant vector components from three-dimensional Cartesian coordinates into two-dimensional triangle coordinates.

By taking advantage of matrix inversion and homogeneous coordinates, we pre-compute two transformation vectors $\boldsymbol{T}_V, \boldsymbol{T}_W \in \mathbb{R}^4$ for each triangle which allow us to obtain the barycentric coordinates of a closest point, given in homogeneous Cartesian coordinates, with only 16 floating point operations as

$$\alpha_V = \boldsymbol{x}_{cp}\cdot\boldsymbol{T}_V, \quad \alpha_W = \boldsymbol{x}_{cp}\cdot\boldsymbol{T}_W, \quad \alpha_U = 1-\alpha_V-\alpha_W. \quad (1)$$

The computation of $\boldsymbol{T}_V$ and $\boldsymbol{T}_W$, which requires 61 floating point operations, is described in Algorithm 1.

Algorithm 1 Barycentric Transformation

Input
Triangle vertices $\boldsymbol{x}_U, \boldsymbol{x}_V, \boldsymbol{x}_W \in \mathbb{R}^3$.

Output
Transformation vectors $\boldsymbol{T}_V, \boldsymbol{T}_W \in \mathbb{R}^4$ for Equation 1.

Procedure

$\boldsymbol{t} \leftarrow \boldsymbol{x}_V - \boldsymbol{x}_U$, $\quad \boldsymbol{b} \leftarrow \boldsymbol{x}_W - \boldsymbol{x}_U$
$tu \leftarrow \boldsymbol{t}\cdot\boldsymbol{x}_U$, $\quad bu \leftarrow \boldsymbol{b}\cdot\boldsymbol{x}_U$

$tt \leftarrow \boldsymbol{t}\cdot\boldsymbol{t}$, $\quad bb \leftarrow \boldsymbol{b}\cdot\boldsymbol{b}$, $\quad tb \leftarrow \boldsymbol{t}\cdot\boldsymbol{b}$
$D \leftarrow tt\cdot bb - tb\cdot tb$
$tt \leftarrow tt/D$, $\quad bb \leftarrow bb/D$, $\quad tb \leftarrow tb/D$

$\boldsymbol{T}_V.xyz \leftarrow \boldsymbol{t}\cdot bb - \boldsymbol{b}\cdot tb$, $\quad \boldsymbol{T}_V.w \leftarrow bu\cdot tb - tu\cdot bb$
$\boldsymbol{T}_W.xyz \leftarrow \boldsymbol{b}\cdot tt - \boldsymbol{t}\cdot tb$, $\quad \boldsymbol{T}_W.w \leftarrow tu\cdot tb - bu\cdot tt$

4. Results and Performance Analysis

The semi-Lagrangian CPM is able to produce high-quality fluid effects on deforming surfaces as depicted in Figure 1. Here, the flow is driven by several attached sources in the form of Neumann boundary conditions, and the appearance and shape of the surface is modulated to visualize the solution. In Figure 4 a high-resolution simulation was interactively steered by a user, who painted waves onto a flexing hand and thereby created an visual effect in an exploratory way. The experiment in Figure 5 investigates the accuracy of the semi-Lagrangian closest point extension.

© 2013 The Author(s)
Computer Graphics Forum © 2013 The Eurographics Association and John Wiley & Sons Ltd

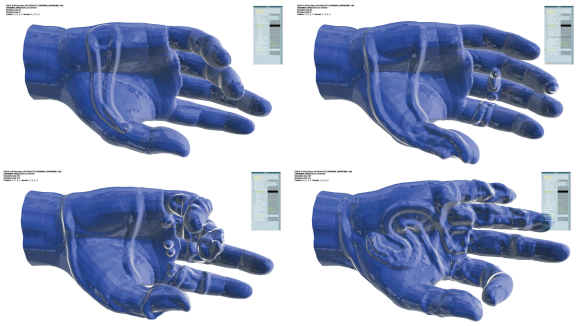

Figure 4: *Snapshots of an interactive simulation session, in which the user painted waves on a deforming surface. At a resolution corresponding to a* 320^3 *Cartesian grid and a* 1920×1177 *viewport, simulation and rendering ran at 8 frames per second on a Geforce GTX 480 graphics card.*

Figure 5: *The accuracy of the semi-Lagrangian closest point extension depends only on the interpolation scheme. One full animation cycle with disabled PDE solver is shown from left to right. A semi-Lagrangian closest point extension of the height field was computed for each of the 44 animation key-frames. We used linear interpolation in the top panel and stabilized WENO4 interpolation in the bottom panel.*

The backtrace along the characteristic is exact up to floating point precision in our scenario, because of the barycentric interpolation of vertex positions on the piecewise-linear surface. The discretization error depends therefore on the spatial interpolation technique. The example clearly shows that higher-order WENO interpolation introduces significantly less numerical dissipation than tri-linear interpolation.

To validate the efficiency of the semi-Lagrangian closest point method, we have performed a number of experiments using simulation grids at different resolutions. In all of our experiments, a key-frame animation of the Laurent hand model was used, consisting of 15855 triangles and 44 animation steps. Note that the surface movements between two consecutive key frames are often an order of magnitude larger than the grid spacing. This could be a problem for advection algorithms with a stability criterion dependent on the Courant-Friedrichs-Lewy condition. All measurements were performed on a 2.4 GHz Core 2 Duo processor and an NVIDIA GeForce GTX 480 graphics card with 1536 MiB local video memory. A detailed memory and performance statistic of the GPU SLCPM for the simulation of fluid flow and waves is given in Table 1. Here, numbers separated by a slash refer to the simulation using linear, respectively WENO4 interpolation exclusively. All timings are given in milliseconds and rendering was always performed on a 1280×720 viewport, using a raycasting technique for smooth surface displacements similar to the one presented by Auer et al. [AMT*12].

The first line gives the resolution of the uniform Cartesian grid to which the simulation resolution corresponds. The second line lists the number of closest points within the computational band. The differences are due to differently sized computational bands that are dictated by the numerical stencils of the respective discretization methods. The third line gives the GPU memory requirements of SLCPM. The Navier-Stokes simulation requires more memory due to its larger computational band and the additional buffers required to store the simulation attributes. For comparison, the fourth line lists the simulation times on the static surface, using linear and WENO4 interpolation in the closest point extension. The fifth line gives the respective timings of the SLCPM on the animated surface. As expected, the increase in simulation time between 15 and 50 milliseconds contains a constant factor due to the generation of the extended closest point representation, which scales mainly in the number of triangles [AMT*12]. The WENO4 interpolation therefore remains the most expensive operation at higher resolutions. The last two lines show the rendering times and the total time for simulation and rendering.

5. Conclusion and Further Work

We have presented a real-time Eulerian method for the simulation of fluid flow and wave propagation on deforming triangulated surfaces. Our approach successfully combines the semi-Lagrangian method and the closest point method, and rigorously exploits the synergies between the two. The method can simulate intrinsic fluid effects at a high, constant and uniform resolution, and its precision does not depend on the input geometry.

The Eulerian discretization of the surface sidesteps all problems with dynamic parameterizations in Lagrangian methods. The embedding of the surface into a three-dimensional Euclidean space also results in an extrinsic notion of the surface topology, however. This especially means that the simulation treats a deformation leading to a self-intersection as a connection of the respective surface parts. Depending on the goals of the user, this may be seen as an advantage or a limitation.

The proposed method opens a number of future re-

© 2013 The Author(s)
Computer Graphics Forum © 2013 The Eurographics Association and John Wiley & Sons Ltd

Table 1: *Performance statistics for fluid simulation and rendering on the GPU.*

	Wave Equation		Navier-Stokes	
Resolution	128^3	320^3	128^3	320^3
# Closest Points	79k / 124k	524k / 860k	128k / 162k	884k / 1.2M
GPU Memory	4MiB / 5MiB	32MiB / 39MiB	13MiB / 15MiB	99MiB / 115MiB
CPM Solver	1.1ms / 3.9ms	2.4ms / 24ms	8.7ms / 41ms	54ms / 294ms
SLCPM Solver	15ms / 30ms	24ms / 75ms	27ms / 64ms	84ms / 312ms
Raycasting	14ms / 15ms	28ms / 30ms	18ms / 19ms	31ms / 32ms
SLCPM Total	29ms / 45ms	52ms / 105ms	45ms / 83ms	115ms / 344ms

search directions. By enabling support for level-of-detail changes of both the input triangulation and the Eulerian computational grid, the algorithm could run with a fixed time budget, a requirement for the application in computer games. We plan also to combine our method with triangulation-based surface tracking [BGOS06, BHLW12] in order to employ it for the up-resing of interactive, fully three-dimensional fluid simulations, e.g. [CLT07, CM11]. The integration with techniques for free-surface flow would also be an important step towards the real-time simulation of surface tension [TWGT10] and the interactive artistic control of fluid behavior [SY05].

Acknowledgment

This work was funded by the Munich Centre of Advanced Computing at the Technische Universität München.

References

[AMT*12] AUER S., MACDONALD C. B., TREIB M., SCHNEIDER J., WESTERMANN R.: Real-time fluid effects on surfaces using the closest point method. *Computer Graphics Forum 31*, 6 (2012), 1909–1923. 2, 3, 4, 5, 6, 7

[ATBG08] ANGST R., THÜREY N., BOTSCH M., GROSS M.: Robust and efficient wave simulations on deforming meshes. *Computer Graphics Forum (Proc. Pacific Graphics) 27* (2008), 1895–1900. 2, 3, 4

[BCOS01] BERTALMÍO M., CHENG L.-T., OSHER S., SAPIRO G.: Variational problems and partial differential equations on implicit surfaces. *J. Comput. Physics 174*, 2 (2001), 759–780. 3

[BGOS06] BARGTEIL A. W., GOKTEKIN T. G., O'BRIEN J. F., STRAIN J. A.: A semi-lagrangian contouring method for fluid simulation. *ACM Trans. Graph. 25*, 1 (Jan. 2006), 19–38. 2, 8

[BHLW12] BOJSEN-HANSEN M., LI H., WOJTAN C.: Tracking surfaces with evolving topology. *ACM Trans. Graph. 31*, 4 (July 2012), 53:1–53:10. 8

[Bur05] BURGER M.: *Finite element approximation of elliptic partial differential equations on implicit surfaces.* Cam report 05-46, University of California, Los Angeles, 2005. 3

[CLB*09] CHUANG M., LUO L., BROWN B. J., RUSINKIEWICZ S., KAZHDAN M.: Estimating the Laplace-Beltrami operator by restricting 3D functions. In *Proceedings of the Eurographics Symposium on Geometry Processing* (2009), pp. 1475–1484. 2

[CLT07] CRANE K., LLAMAS I., TARIQ S.: *Real Time Simulation and Rendering of 3D Fluids.* Addison-Wesley, 2007, ch. 30. 8

[CM11] CHENTANEZ N., MÜLLER M.: Real-time eulerian water simulation using a restricted tall cell grid. In *ACM SIGGRAPH 2011 papers* (New York, NY, USA, 2011), SIGGRAPH '11, ACM, pp. 82:1–82:10. 8

[FOKG05] FELDMAN B. E., O'BRIEN J. F., KLINGNER B. M., GOKTEKIN T. G.: Fluids in deforming meshes. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2005), SCA '05, ACM, pp. 255–259. 3

[Gre06] GREER J. B.: An improvement of a recent Eulerian method for solving PDEs on general geometries. *J. Sci. Comput. 29* (December 2006), 321–352. 3

[HZQW10] HONG Y., ZHU D., QIU X., WANG Z.: Geometry-based control of fire simulation. *The Visual Computer 26* (September 2010), 1217–1228. 2, 3

[KTT13] KIM T., TESSENDORF J., THUEREY N.: Closest-Point Turbulence for Liquid Surfaces. *ACM Transactions on Graphics (in press)* (August 2013), 10. 2, 3, 4

[MM12] MÄRZ T., MACDONALD C. B.: Calculus on surfaces with general closest point functions. *SIAM J. Numerical Analysis 50*, 6 (2012), 3303–3328. 3

[MR08] MACDONALD C. B., RUUTH S. J.: Level set equations on surfaces via the Closest Point Method. *J. Sci. Comput. 35*, 2–3 (June 2008), 219–240. doi:10.1007/s10915-008-9196-6. 3, 5

[MR09] MACDONALD C. B., RUUTH S. J.: The implicit Closest Point Method for the numerical solution of partial differential equations on surfaces. *SIAM J. Sci. Comput. 31*, 6 (2009), 4330–4350. doi:10.1137/080740003. 3, 5

[RM08] RUUTH S. J., MERRIMAN B.: A simple embedding method for solving partial differential equations on surfaces. *J. Comput. Phys. 227* (January 2008), 1943–1961. 2, 3

[Sch09] SCHNEIDER J.: *Efficient Methods for the Display of highly-detailed Models in Computer Graphics.* PhD thesis, Technise Universität München, May 2009. 6

[Sta03] STAM J.: Flows on surfaces of arbitrary topology. *ACM Trans. Graph. 22* (July 2003), 724–731. 2

[SY05] SHI L., YU Y.: Taming liquids for rapidly changing targets. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2005), SCA '05, ACM, pp. 229–236. 8

[Tur91] TURK G.: Generating textures on arbitrary surfaces using reaction-diffusion. In *Proceedings of the 18th annual conference on Computer graphics and interactive techniques* (1991), SIGGRAPH '91, ACM, pp. 289–298. 2

[TWGT10] THÜREY N., WOJTAN C., GROSS M., TURK G.: A multiscale approach to mesh-based surface tension flows. In *ACM SIGGRAPH 2010 papers* (New York, NY, USA, 2010), SIGGRAPH '10, ACM, pp. 48:1–48:10. 2, 3, 8

© 2013 The Author(s)
Computer Graphics Forum © 2013 The Eurographics Association and John Wiley & Sons Ltd